Software Engineering

How To Create Software Products Everyone Loves!

Module 1
Fundamentals

Ali Samanipour

Shiraz University of Technology
September. 2025

Ali Samanipour
linkedin.com/in/Samanipour

Session Roadmap

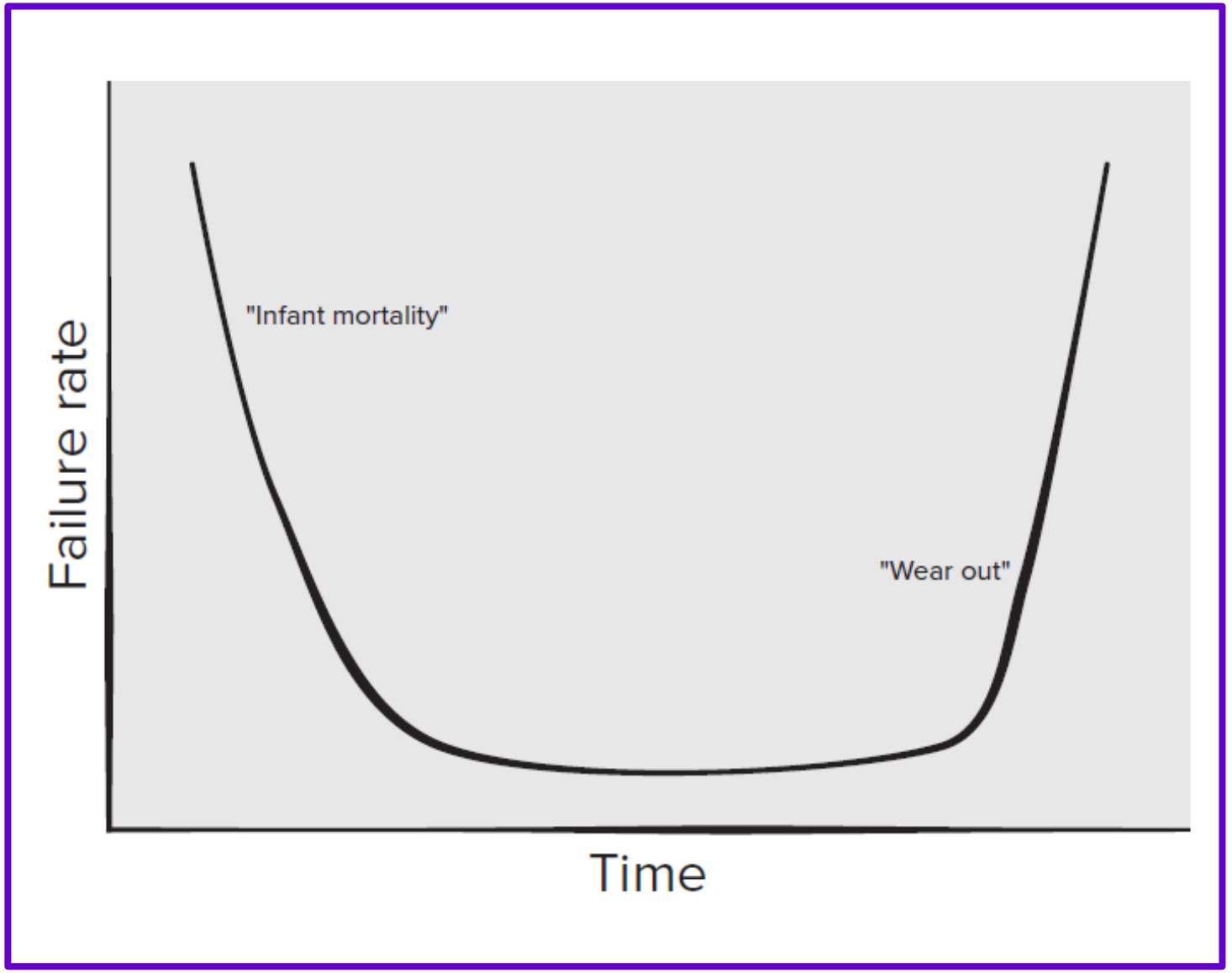
1. Nature of Software
2. Defining the Discipline
3. The Software Process
4. Software Engineering Practice
5. How It All Starts

Defining 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.

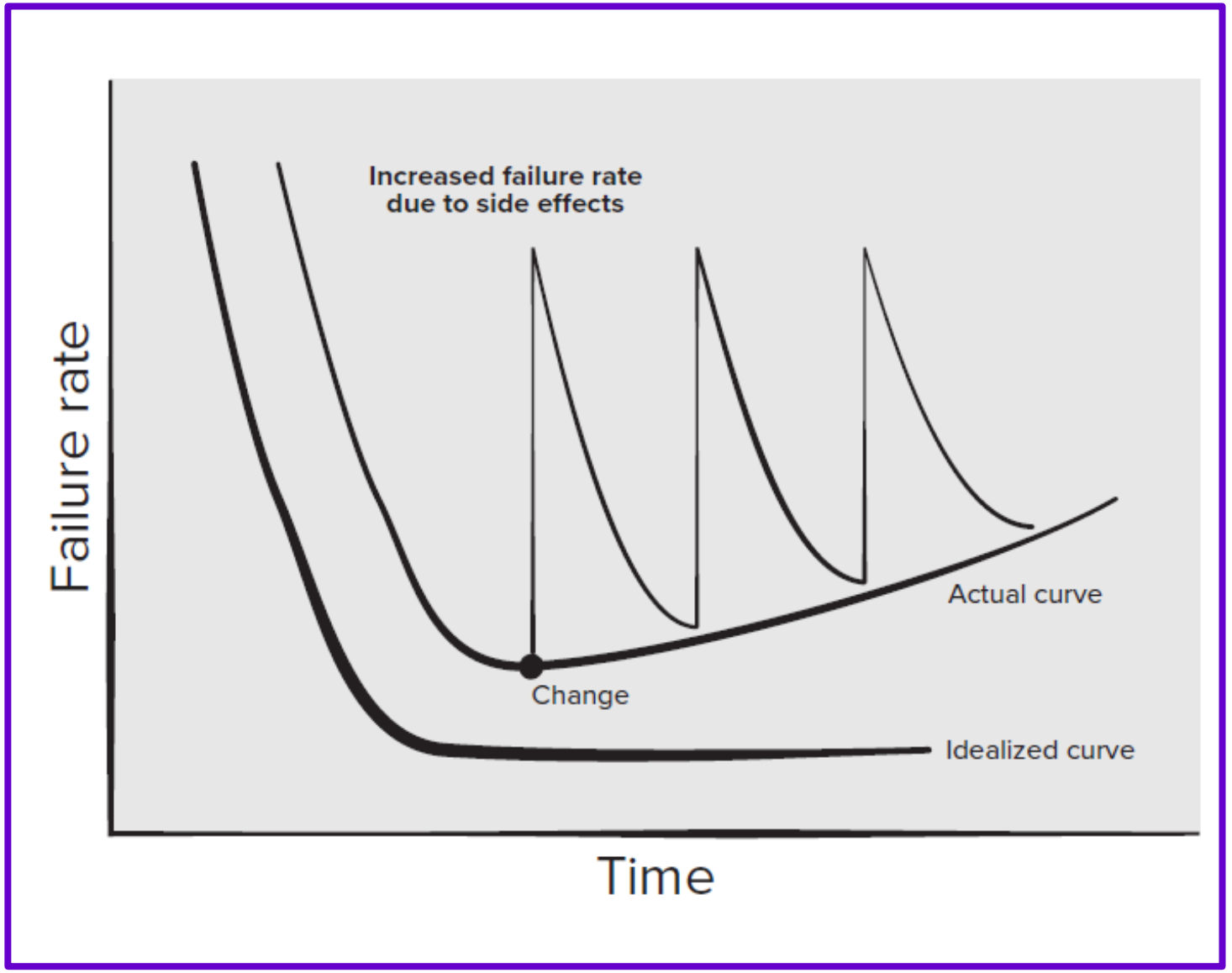
Failure curve for hardware

hardware exhibits high failure rates early in its life, As time passes, however, the failure rate rises again as hardware components suffer from the cumulative effects



Failure curve for hardware

software doesn’t wear out. But it does deteriorate!



Legacy Software

Legacy software systems . . . were developed decades ago and have been continually modified to meet changes in business requirements and computing platforms. The proliferation of such systems is causing headaches for large organizations who find them costly to maintain and risky to evolve.

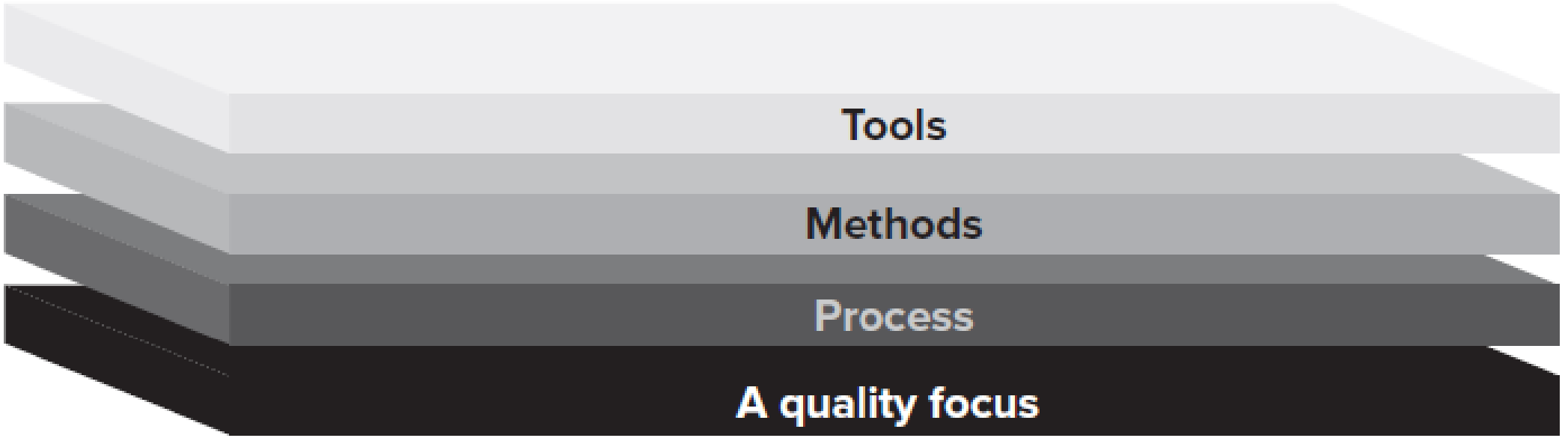
Session Roadmap

1. Nature of Software
2. Defining the Discipline
3. The Software Process
4. Software Engineering Practice
5. How It All Starts

Definition for software engineering

Software Engineering: The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.

Software engineering layers



Session Roadmap

1. Nature of Software
2. Defining the Discipline
3. The Software Process
4. Software Engineering Practice
5. How It All Starts

The Software Process

A Process is a collection of activities, actions, and tasks that are performed when some work product is to be created.

An Activity strives to achieve a broad objective (e.g., communication with stakeholders) and is applied regardless of the application domain, size of the project, complexity of the effort, or degree of rigor with which software engineering is to be applied.

The Software Process …

An action (e.g., architectural design) encompasses a set of tasks that produce a major work product (e.g., an architectural model).

A task focuses on a small, but well-defined objective (e.g., conducting a unit test) that produces a tangible outcome.

The Process Framework

A process framework establishes the foundation for a complete software engineering process by identifying a small number of framework activities that are applicable to all software projects, regardless of their size or complexity

In addition, the process framework encompasses a set of umbrella activities that are applicable across the entire software process.

Generic Process Framework Activities

Before any technical work can commence, it is critically important to **communicate** and collaborate with the customer

Software project is a complicated journey, and the planning activity creates a “map” that helps guide the team as it makes the journey. The map—called a software project **plan**

A software engineer does the same thing by creating **models** to better understand software requirements and the design that will achieve those requirements.

Generic Process Framework Activities …

Construction. What you design must be built. This activity combines code generation(either manual or automated) and the testing that is required to uncover errors in the code.

Deployment. The software (as a complete entity or as a partially completed increment) is delivered to the customer who evaluates the delivered product and provides feedback based on the evaluation.

Umbrella Activities

Software engineering process framework activities are complemented by a number of umbrella activities.

In general, umbrella activities are applied throughout a software project and help a software team **manage** and **control progress**, **quality**, **change**, and **risk**.

The Importance of a Process

A process organizes the work on a software product into distinct **phases**.

In general, the work that is undertaken for a project will transition from one phase to the next. Each phase has specific activities and tasks that must be performed.

The Importance of a Process

So, Why do we need to break a Process into district phases?

By creating distinct phases with **clear steps and deliverables**, the use of processes structures the software development in a manner that provides clarity for everyone involved.

Life Cycles and Processes

A **software life cycle process** refers to a process that covers the entire spectrum of software development, from the product's initial conception to its eventual retirement.

how to get the product from an idea in the client's mind to working software that an end-user experiences ?

The Answer is : Processes

Processes and Phases

A process is divided into multiple **phases**. For example, the phases of a process to create a software product may be:

- Specification
- Design and Implementation
- Verification and Validation

Phases—or **milestones**—within a process need only be identified in a logical manner and organized so that the phases contribute to the completion of the process

Phases, Activities, and Tasks

For example, consider the Verification and Validation phase identified previously; one of the activities within might be Executing Tests. Tasks related to the activity of executing tests could include:

- Writing test framework code
- Designing tests
- Writing tests
- Running tests

Tasks are where the real work gets done, no matter what process is used. Tasks are the small, manageable units of work for the project.

Last Word on Phases

There are major differences between Process models in terms of what activities constitute their phases and the sequencing of the phases:

- **Linear process models** – phases that happen sequentially, one after another
- **Iterative process models** – phases that are repeated in cycles
- **Parallel process models** – activities that occur concurrently

Designing a process can start from scratch or may be based on one of the many process models that have already been used successfully

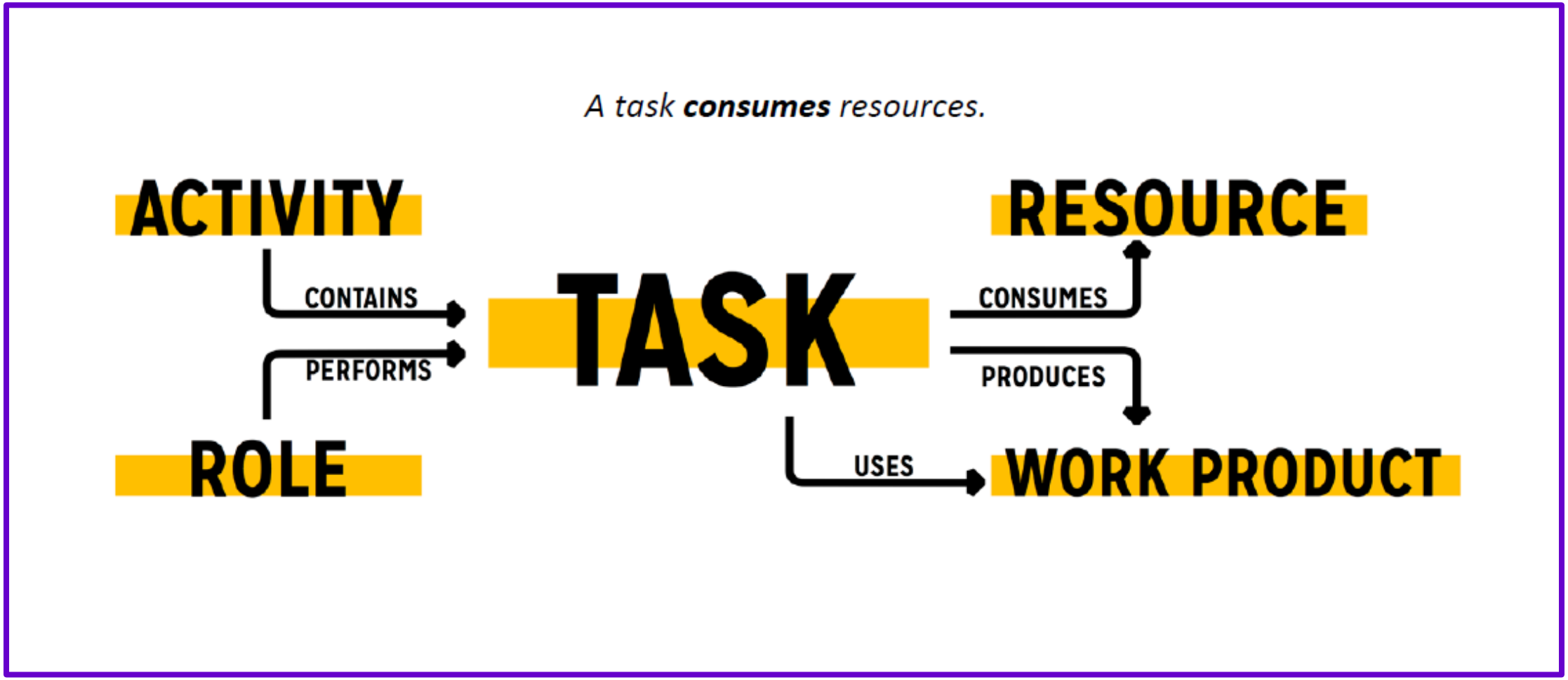
Task Dependencies, Roles, Work Product

A **role** is job-related activity assigned to a person. As mentioned above, a team member named Sam may have a job title "Software Engineer"; however, one of his roles is to *program*

A **work product** is an output produced by completing a specific task. Work products, of course, do include the final product.

Task **dependencies** will impose a specific order for tasks to be completed.

Tasks and Resources



Session Roadmap

1. Nature of Software
2. Defining the Discipline
3. The Software Process
4. Software Engineering Practice
5. How It All Starts

Practices

Practices are strategies used to execute processes smoothly. Practices contribute to the effectiveness of a development process.

For instance, a manager can follow proven management practices for: scheduling and organizing tasks, minimizing resource waste, and tracking completed work.

Practices are often gathered into a **methodology**.

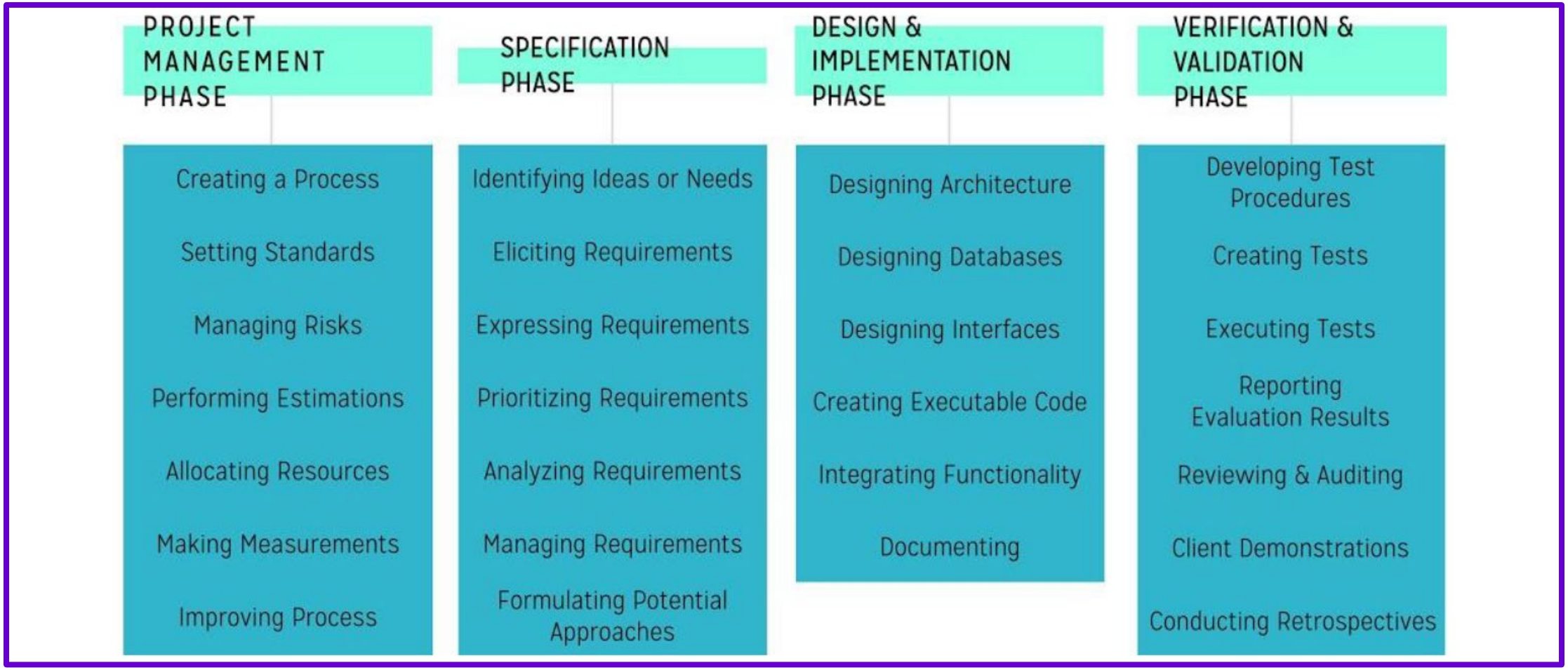
Agile Manifesto vs Agile Methodology

The Agile Manifesto provides the philosophy, and an Agile methodology provides a set of day-to-day practices that align with that philosophy.

From Process to Task

A process is made up of phases. A phase is made up of activities. An activity is made up of tasks.

Software Engineering Activities Organized In Phases



Session Roadmap

1. Nature of Software
2. Defining the Discipline
3. The Software Process
4. Software Engineering Practice
5. How It All Starts

How It All Starts?

Every software project is precipitated by some **business need**—the need **to correct a defect** in an existing application; the need **to adapt a "legacy system"** to a changing business environment; the need **to extend the functions and features** of an existing application; or the need **to create a new product**, service, or system.

Appendix:
Verification and Validation Activities

- **Validation:** Are we building the right product?
- **Verification:** Are we building the product right?

- **Verification** typically consists of internal testing and technical review activities that ensure that the product does what it is intended to do, according to the documented design and requirements.
- **Validation** activities such as demonstrating to clients help to check that the product meets the needs of the client and/or the end users.

Course References