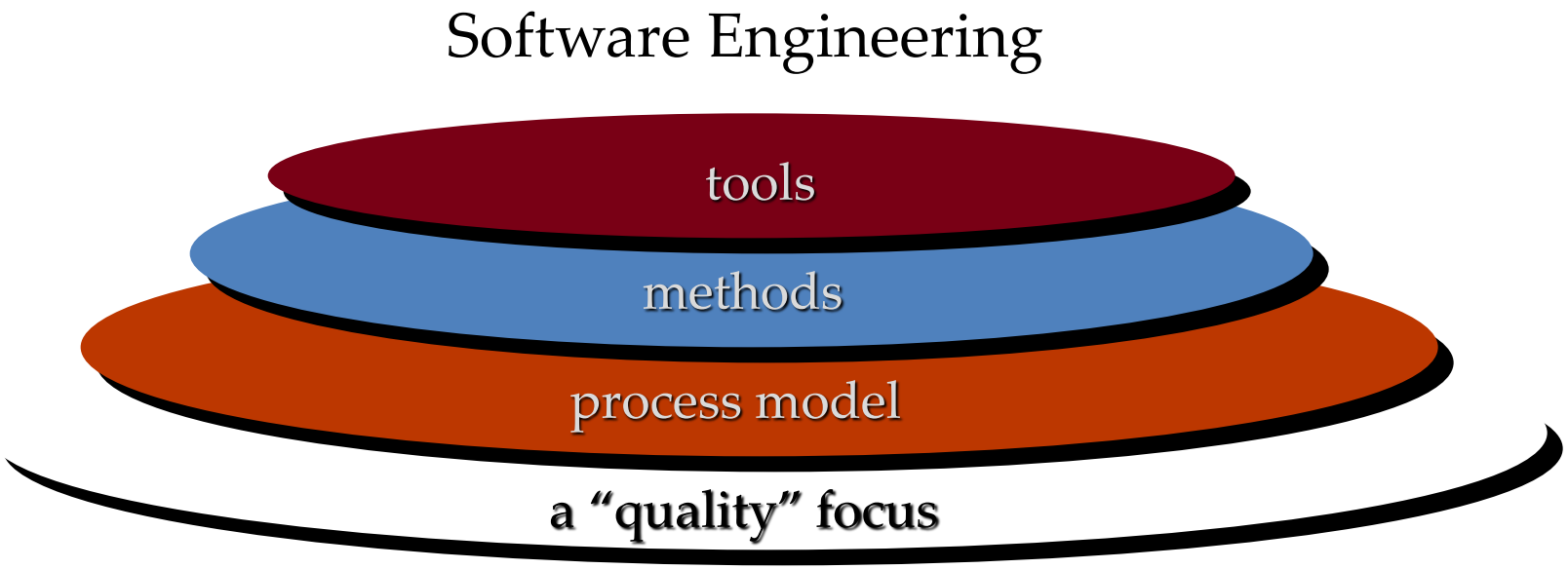
Lecture-2
Process: A Generic View

-Nafisa Tabassum

A Layered Technology



A Layered Technology

a)Quality Focus:

Software engineering is a layered technology.

Referring to Figure 1.3, any engineering approach (including software engineering) must rest on an organizational commitment to quality. The bedrock that supports software engineering is a quality focus.

A Layered Technology

b)Process layer:

- The foundation for software engineering is the *process* layer. The software engineering process is the glue that holds the technology layers together and enables rational and timely development of computer software.
- Process defines a framework that must be established for effective delivery of software engineering technology.
- The software process forms the basis for management control of software projects and establishes the context in which technical methods are applied, work products(models, documents, data, reports, forms, etc.) are produced, milestones are established, quality is ensured, and change is properly managed.

A Layered Technology

c)Method:

- Software engineering *methods* provide the technical how-to's for building software.
- Methods encompass a broad array of tasks that include communication, requirements analysis, design modeling, program construction, testing, and support.
- Software engineering methods rely on a set of basic principles that govern each area of the technology and include modeling activities and other descriptive techniques.

A Layered Technology

d)Tools:

- Software engineering *tools* provide automated or semi-automated support for the process and the methods. When tools are integrated so that information created by one tool can be used by another, a system for the support of software development,called *computer-aided software engineering*, is established.

Some important terms

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

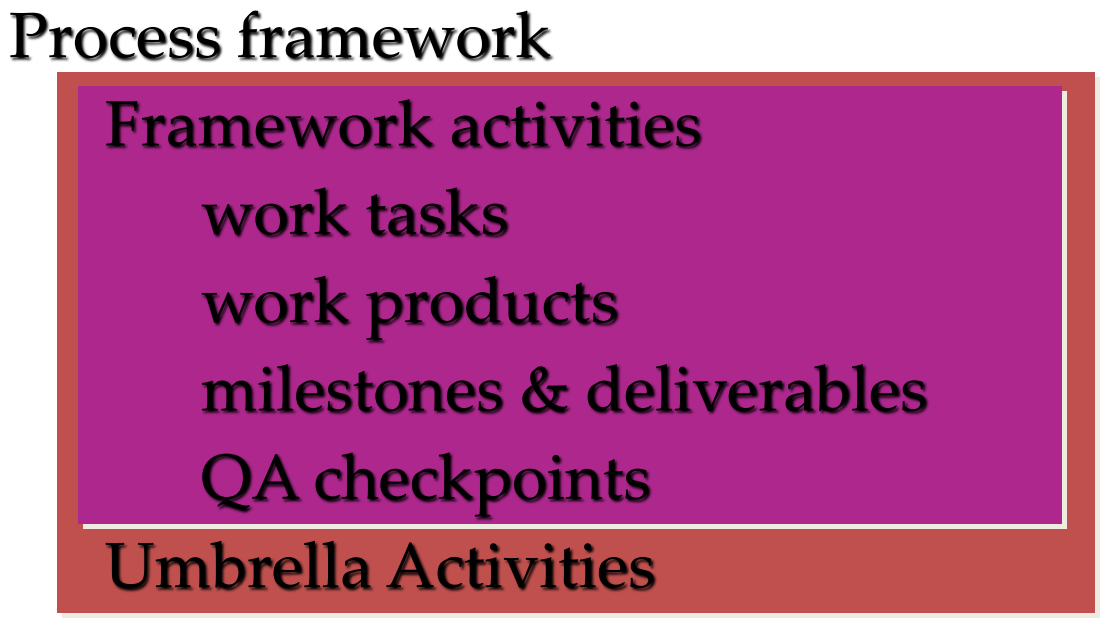
Some important terms

- **Action:** An *action* (e.g., architectural design) encompasses a set of tasks that produce a major work product (e.g., an architectural design model).
- **Task**: A *task* focuses on a small, but well-defined objective (e.g., conducting a unit test) that produces a tangible outcome.

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.

A Process Framework



Framework Activities

- Communication
- Planning
- Modeling
  - Analysis of requirements
  - Design
- Construction
  - Code generation
  - Testing
- Deployment

Communication. Before any technical work can commence, it is critically important to communicate and collaborate with the customer (and other Stakeholders). The intent is to understand stakeholders' objectives for the project and to gather requirements that help define software features and functions.

Planning. Any complicated journey can be simplified if a map exists. A 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*—defines the software engineering work by describing the technical tasks to be conducted, the risks that are likely, the resources that will be required, the work products to be produced, and a work schedule.

Modeling. Whether you're a landscaper, a bridge builder, an aeronautical engineer, a carpenter, or an architect, you work with models every day. You create a "sketch" of the thing so that you'll understand the big picture—what it will look like architecturally, how the constituent parts fit together, and many other characteristics. If required, you refine the sketch into greater and greater detail in an effort to better understand the problem and how you're going to solve it. A software engineer does the same thing by creating models to better understand software requirements and the design that will achieve those requirements.

Construction. 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 project tracking and control—allows the software team to assess progress against the project plan and take any necessary action to
maintain the schedule.

Risk management—assesses risks that may affect the outcome of the project or the quality of the product.

Software quality assurance—defines and conducts the activities required
to ensure software quality.

Technical reviews—assesses software engineering work products in an effort
to uncover and remove errors before they are propagated to the next activity.

Measurement—defines and collects process, project, and product measures that assist the team in delivering software that meets stakeholders' needs; can be used in conjunction with all other framework and umbrella activities.

Software configuration management—manages the effects of change throughout the software process.

Reusability management—defines criteria for work product reuse (including software components) and establishes mechanisms to achieve reusable components.

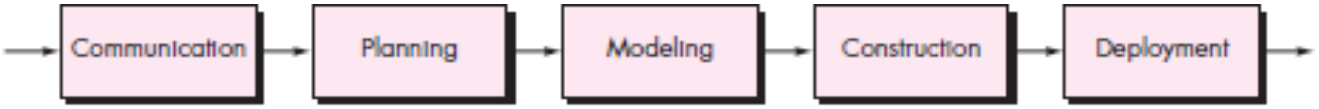
Work product preparation and production—encompasses the activities required to create work products such as models, documents, logs, forms, and lists.

A generic process model

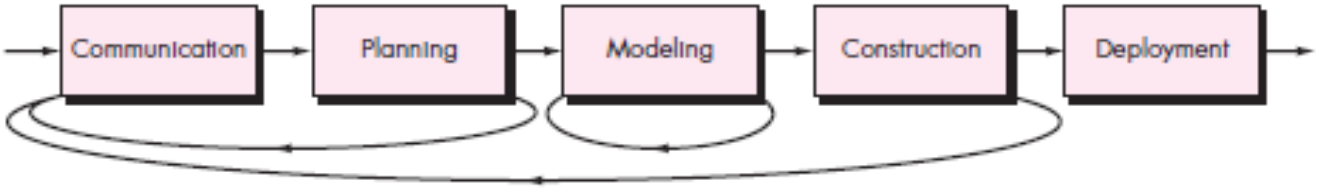
- A process was defined as a collection of work activities, actions, and tasks that are performed when some work product is to be created. Each of these activities, actions, and tasks reside within a framework or model that defines their relationship with the process and with one another.

Process Flow

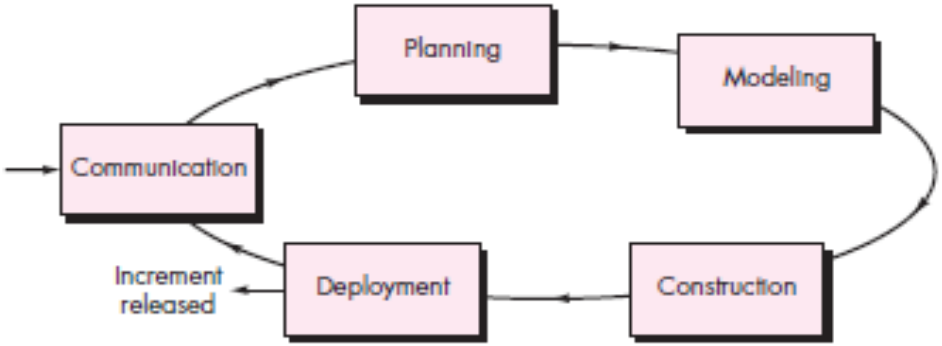
- *process flow*—describes how the framework activities and the actions and tasks that occur within each framework activity are organized with respect to sequence and time and is illustrated in Figure 2.2.



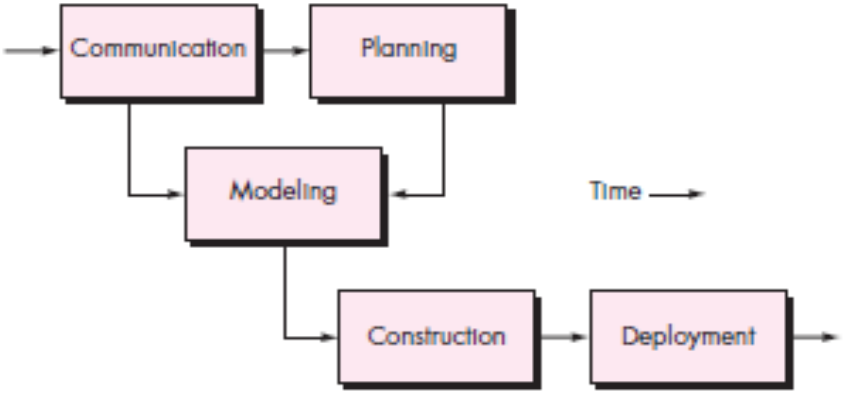
(a) Linear process flow



(b) Iterative process flow



(c) Evolutionary process flow



(d) Parallel process flow

Process Flow

- **A *linear process flow*** executes each of the five framework activities in sequence, beginning with communication and culminating with deployment (Figure 2.2a).

- **An *iterative process flow*** repeats one or more of the activities before proceeding to the next (Figure 2.2b).

- **An *evolutionary process flow*** executes the activities in a “circular” manner. Each circuit through the five activities leads to a more complete version of the software (Figure 2.2c).

- **A *parallel process flow*** (Figure 2.2d) executes one or more activities in parallel with other activities (e.g., modeling for one aspect of the software might be executed in parallel with construction of another aspect of the software).

Software Process Framework – Software Engineering

Last Updated : 20 Jun, 2024

A **Software Process Framework** is a structured approach that defines the steps, tasks, and activities involved in software development. This framework serves as a **foundation for software engineering**, guiding the development team through various stages to ensure a **systematic and efficient process**. A Software Process Framework helps in project planning, risk management, and quality assurance by detailing the chronological order of actions.

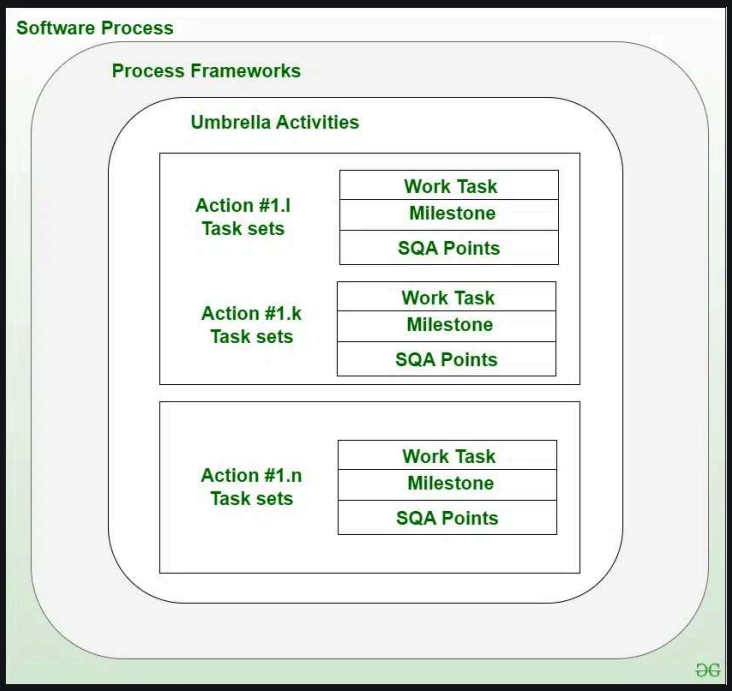
Table of Content

It includes task sets, umbrella activities, and process framework activities, all essential for a **successful software development lifecycle**. Utilizing a well-defined Software Process Framework enhances productivity, consistency, and the overall quality of the software product

What is a Software Process Framework?

Software Process Framework details the steps and chronological order of a process. Since it serves as a foundation for them, it is utilized in most applications. Task sets, umbrella activities, and process framework activities all define the characteristics of the software development process. Software Process includes:

1. **Tasks:** They focus on a small, specific objective.
2. **Action:** It is a set of tasks that produce a major work product.
3. **Activities:** Activities are groups of related tasks and actions for a major objective.



Software Process Framework

What Is a Software Development Framework?

A software development framework is a structured set of tools, libraries, best practices, and guidelines that help developers build software applications. Think of it as a template or foundation that provides the basic structure and components needed for a software project.

Key Points

1. **Foundation**: It gives a basic structure or template for developing software, so developers don't have to start from

scratch.
2. **Components and Tools**: It includes pre-built components and tools that make development faster and easier.
3. **Best Practices and Guidelines**: It offers best practices and guidelines to ensure the software is built in an organized and efficient way.
4. **Customization**: Developers can modify and add new functions to customize the framework to their specific needs.

Advantages of Software Development Framework

A **Software Development Framework** offers numerous benefits that streamline the **software development process** and enhance the quality and efficiency of the final product. Here are some key advantages:

1. **Increased Productivity**: Frameworks provide pre-built components and tools, allowing developers to focus on specific application logic rather than reinventing the wheel.
2. **Consistent Quality**: By following best practices and standardized processes, frameworks help ensure consistent code quality and structure across the project.
3. **Reduced Development Time**: With ready-to-use templates and libraries, developers can significantly cut down on the time needed to build applications from scratch.
4. **Better Maintainability**: A structured framework makes the codebase more organized and easier to understand, which simplifies maintenance and updates.
5. **Enhanced Security**: Frameworks often include built-in security features and follow industry best practices, reducing the risk of vulnerabilities.
6. **Scalability**: Frameworks are designed to handle growth, making it easier to scale applications as user demand increases.

Dis-advantages of Software Development Framework

While **Software Development Frameworks** offer several advantages, they also come with certain drawbacks that developers and organizations should consider:

1. **Learning Curve**: Frameworks often have a steep learning curve, requiring developers to invest time and effort in understanding the framework's architecture, conventions, and best practices.
2. **Restrictions**: Some frameworks impose constraints and limitations on how developers can design and implement certain features, potentially limiting flexibility and creativity.
3. **Complexity Overhead**: In some cases, frameworks introduce unnecessary complexity, especially for smaller or simpler projects, which can lead to over-engineering.
4. **Performance Overhead**: Using a framework may introduce additional layers of abstraction and overhead, which can impact the performance of the application, particularly in resource-intensive environments.
5. **Vendor Lock-in**: Depending heavily on a specific framework can lead to vendor lock-in, making it challenging to switch to alternative technologies or frameworks in the future.

How to Choose a Suitable Development Framework

Choosing a suitable development framework is crucial for the success of a software project. Here are key steps to help you make an informed decision. Here is a simple and effective strategy to help you select the most suitable framework for your project

1. Consider the Framework's Language

- **Popular Languages**: Start with frameworks in popular programming languages like Java, Python, or Ruby if you have no preference. These languages often have robust frameworks with strong community support.

2. Open-Source vs. Paid Frameworks

- **Open-Source**: Generally have a large user base, frequent updates, and community contributions.
- **Paid**: Often more reliable with better support but may lack customization and timely updates.

3. Community and Support

- **Community Size**: A large, active community means better support, more tutorials, and a more mature framework. Look for frameworks with extensive community resources and engagement.

4. Review Case Studies and Example Applications

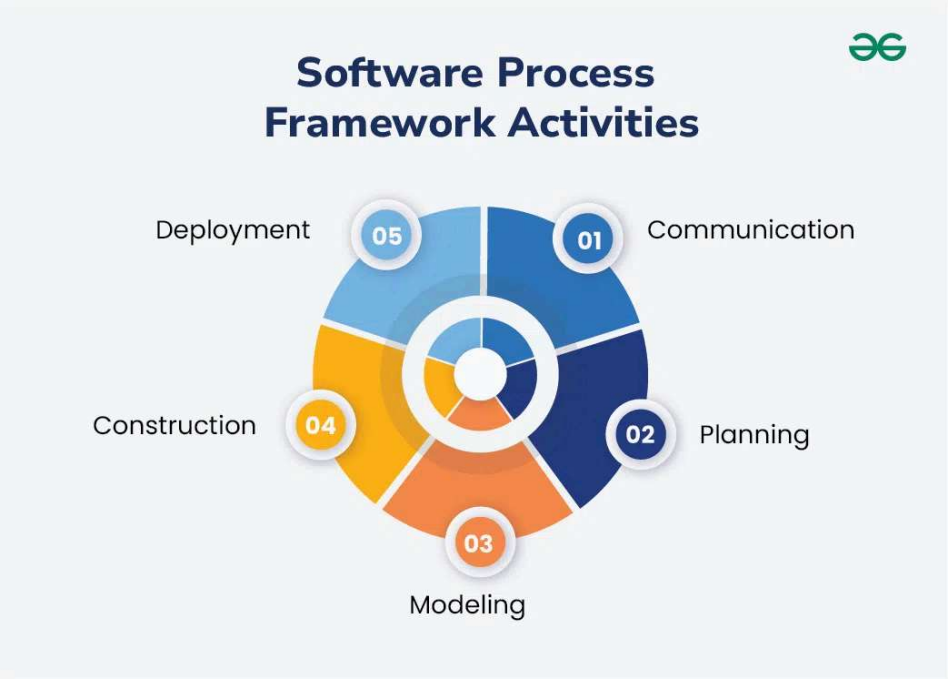
- **Practical Insights**: Check the framework's website or repositories for case studies or example applications. These can provide insights into development processes and methods that work well with the framework.

5. Test the Framework Yourself

- **Hands-On Experience**: Try out the framework in your own project to see how it fits your needs. Testing helps you understand the framework's functionality and whether it suits your development scenario.

Software Process Framework Activities

The Software process framework is required for representing common process activities. Five framework activities are described in a process framework for software engineering. Communication, planning, modeling, construction, and deployment are all examples of framework activities. Each engineering action defined by a framework activity comprises a list of needed work outputs, project milestones, and software quality assurance (SQA) points. Let's explain each:



Software Process Framework Activities

1. Communication

Definition: Communication involves gathering requirements from customers and stakeholders to determine the system's objectives and the software's requirements.

Activities:

- **Requirement Gathering**: Engaging with consumers and stakeholders through meetings, interviews, and surveys to understand their needs and expectations.
- **Objective Setting**: Clearly defining what the system should achieve based on the gathered requirements.

Explanation: Effective communication is essential to understand what the users need from the software. This phase ensures that all stakeholders are on the same page regarding the goals and requirements of the system.

2. Planning

Definition: Planning involves establishing an engineering work plan, describing technical risks, listing resource requirements, and defining a work schedule.

Activities:

- **Work Plan**: Creating a detailed plan that outlines the tasks and activities needed to develop the software.
- **Risk Assessment**: Identifying potential technical risks and planning how to mitigate them.
- **Resource Allocation**: Determining the resources (time, personnel, tools) required for the project.
- **Schedule Definition**: Setting a timeline for completing different phases of the project.

Explanation: Planning helps in organizing the project and setting clear expectations. It ensures that the development team has a roadmap to follow and that potential challenges are anticipated and managed.

3. Modeling

Definition: Modeling involves creating architectural models and designs to better understand the problem and work towards the best solution.

Activities:

- **Analysis of Requirements**: Breaking down the gathered requirements to understand what the system needs to do.

- **Design**: Creating architectural and detailed designs that outline how the software will be structured and how it will function.

Explanation: Modeling translates requirements into a visual and structured representation of the system. It helps in identifying the best design approach and serves as a blueprint for development.

4. Construction

Definition: Construction involves creating code, testing the system, fixing bugs, and confirming that all criteria are met.

Activities:

- **Code Generation**: Writing the actual code based on the design models.
- **Testing**: Running tests to ensure the software works as intended, identifying and fixing bugs.

Explanation: This phase is where the actual software is built. Testing is crucial to ensure that the code is error-free and that the software meets all specified requirements.

5. Deployment

Definition: Deployment involves presenting the completed or partially completed product to customers for evaluation and feedback, then making necessary modifications based on their input.

Activities:

- **Product Release**: Delivering the software to users, either as a full release or in stages.
- **Feedback Collection**: Gathering feedback from users about their experience with the software.
- **Product Improvement**: Making changes and improvements based on user feedback to enhance the product.

Popular Software Development Frameworks

- **Angular**
- **React**
- **Vue.js**
- **Django**
- **Flask**
- **Ruby on Rails**
- **Spring**
- **Express**
- **Laravel**
- **ASP.NET Core**

Umbrella Activities

Umbrella Activities that take place during a software development process for improved project management and tracking.

1. **Software project tracking and control:** This is an activity in which the team can assess progress and take corrective action to maintain the schedule. Take action to keep the project on time by comparing the project's progress against the plan.
2. **Risk management:** The risks that may affect project outcomes or quality can be analyzed. Analyze potential risks that may have an impact on the software product's quality and outcome.
3. **Software quality assurance:** These are activities required to maintain software quality. Perform actions to ensure the product's quality.
4. **Formal technical reviews:** It is required to assess engineering work products to uncover and remove errors before they propagate to the next activity. At each level of the process, errors are evaluated and fixed.
5. **Software configuration management:** Managing of configuration process when any change in the software occurs.
6. **Work product preparation and production:** The activities to create models, documents, logs, forms, and lists are carried

out.

7. **Reusability management:** It defines criteria for work product reuse. Reusable work items should be backed up, and reusable software components should be achieved.
8. **Measurement:** In this activity, the process can be defined and collected. Also, project and product measures are used to assist the software team in delivering the required software.

Conclusion

The Software Process Framework outlines a structured approach to software development, detailing key activities like communication, planning, modeling, construction, and deployment. It ensures systematic progress from gathering requirements to delivering the final product. Supporting activities like quality assurance and risk management enhance project efficiency and product quality. This framework helps teams build reliable, high-quality software efficiently.

Frequently Asked Questions Related on Software Process Framework

What are the 5 software process activities?

Five Software Process Activities are:

1. *Communication*
2. *Planning*
3. *Modeling*
4. *Construction*

5. Deployment

What are the types of software processes?

Some common Software Processes are

- *Waterfall Model*
- *Spiral Model*
- *Agile Model*
- *Incremental Model*

Want to learn **Software Testing** and **Automation** to help give a kickstart to your career? Any student or professional looking to excel in **Quality Assurance** should enroll in our course, ***Complete Guide to Software Testing and Automation***, only on GeeksforGeeks. Get hands-on learning experience with the latest testing methodologies, automation tools, and industry best practices through practical projects and real-life scenarios. Whether you are a beginner or just looking to build on existing skills, this course will give you the competence necessary to ensure the quality and reliability of software products. Ready to be a **Pro in Software Testing**? Enroll now and Take Your Career to a Whole New Level!