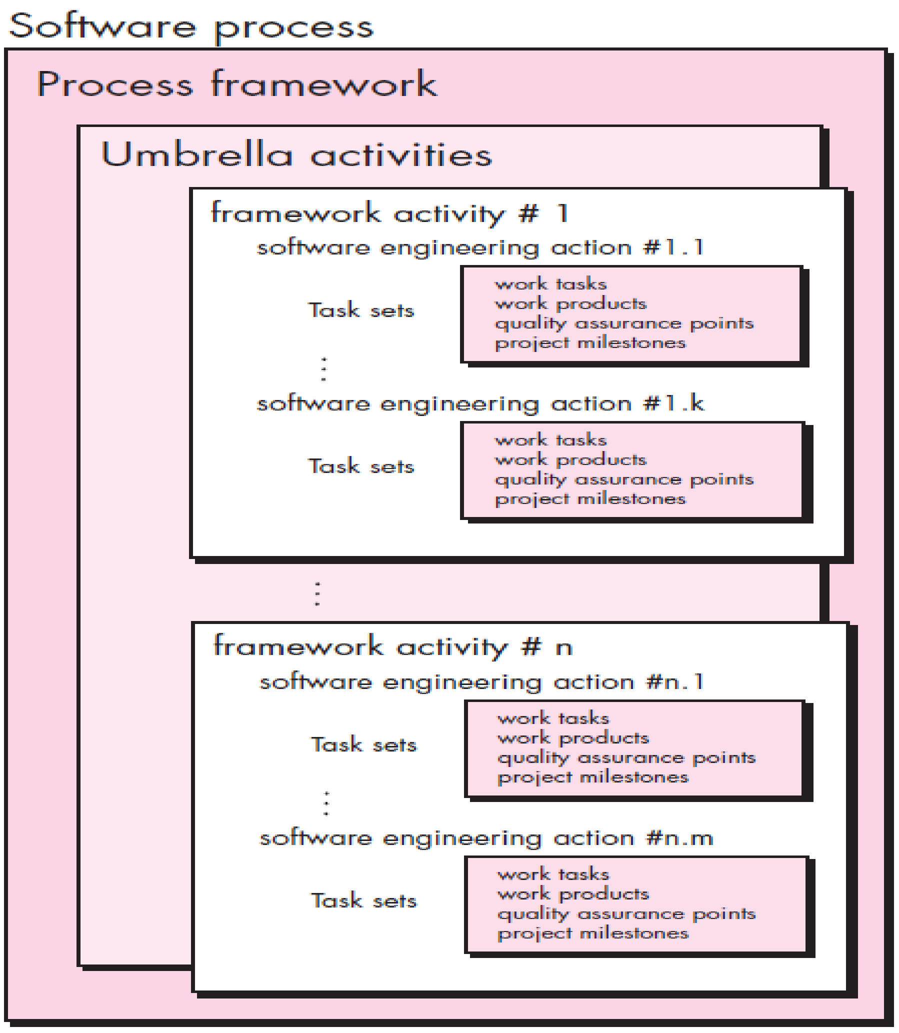
PROCESS MODELS

A GENERIC PROCESS MODEL

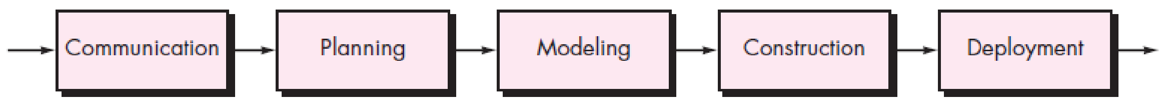
The software process is represented schematically in following figure. Each framework activity is populated by a set of software engineering actions. Each software engineering action is defined by a *task set* that identifies the work tasks that are to be completed, the work products that will be produced, the quality assurance points that will be required, and the milestones that will be used to indicate progress.



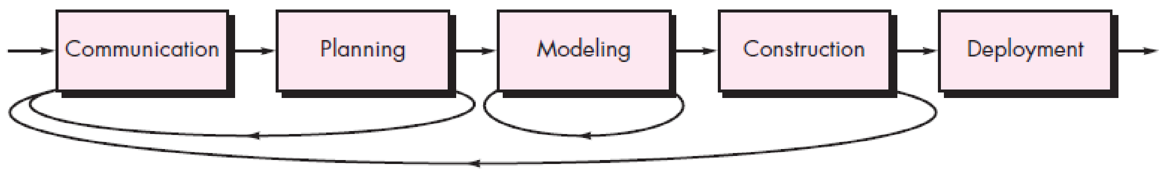
A generic process framework defines **five** framework activities—**communication, planning, modeling, construction,** and **deployment.**

In addition, a set of umbrella activities **project tracking and control, risk management, quality assurance, configuration management, technical reviews, and others** are applied throughout the process.

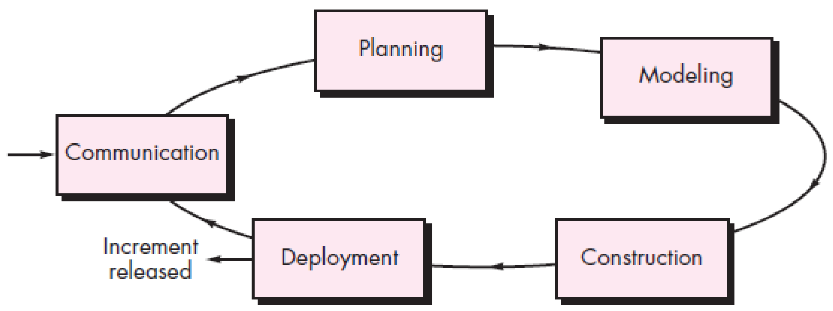
This aspect is called ***process flow.*** It describes how the framework activities and the actions and tasks that occur within each framework activity are organized with respect to sequence and time and is illustrated in following figure



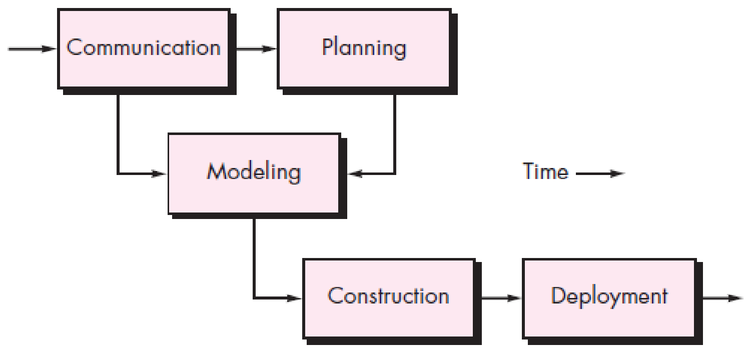
(a) Linear process flow



(b) Iterative process flow



(c) Evolutionary process flow



(d) Parallel process flow

A generic process framework for software engineering A ***linear process flow*** executes each of the **five** framework activities in sequence, beginning with communication and culminating with deployment.

An ***iterative process flow*** repeats one or more of the activities before proceeding to the next. 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. A *parallel process flow* 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).

Defining a Framework Activity

A software team would need significantly more information before it could properly execute any one of these activities as part of the software process. Therefore, you are faced with a key question: What actions are appropriate for a framework activity, given the nature of the problem to be solved, the characteristics of the people doing the work, and the stakeholders who are sponsoring the project?

Identifying a Task Set

Different projects demand different task sets. The software team chooses the task set based on problem and project characteristics. A task set defines the actual work to be done to accomplish the objectives of a software engineering action.

Process Patterns

A ***process pattern*** describes a process-related problem that is encountered during software engineering work, identifies the environment in which the problem has been encountered, and suggests one or more proven solutions to the problem. Stated in more general terms, a process pattern provides you with a template —**a consistent method for describing problem solutions within the context of the software process**.

Patterns can be defined at any level of abstraction. a pattern might be used to describe a **problem (and solution)** associated with a complete **process model** (e.g., prototyping). In other situations, patterns can be used to describe a problem (and solution) associated with a **framework activity** (e.g., **planning**) or an **action** within a framework activity (e.g., project estimating).

Ambler has proposed a template for describing a process pattern:

Pattern Name. The pattern is given a meaningful name describing it within the context of the software process (e.g., **TechnicalReviews**).

Forces. The environment in which the pattern is encountered and the issues that make the problem visible and may affect its solution.

Type. The pattern type is specified. Ambler suggests **three** types:

1. ***Stage pattern***—defines a problem associated with a framework activity for the process. Since a framework activity encompasses multiple actions and work tasks, a stage pattern incorporates multiple task patterns (see the following) that are relevant to the stage (framework activity). An example of a stage pattern might be **Establishing Communication.** This pattern would incorporate the task pattern **Requirements Gathering** and others.
2. ***Task pattern***—defines a problem associated with a software engineering action or work task and relevant to successful software engineering practice (e.g., Requirements Gathering is a task pattern).
3. ***Phase pattern***—define the sequence of framework activities that occurs within the process, even when the overall flow of activities is iterative in nature. An example of a phase pattern might be **Spira lModel** or **Prototyping.**

Initial context. Describes the conditions under which the pattern applies. Prior to the initiation of the pattern:

(1) What organizational or team-related activities have already occurred?

(2) What is the entry state for the process?

(3) What software engineering information or project information already exists?

Problem. The specific problem to be solved by the pattern.

Solution. Describes how to implement the pattern successfully. It also describes how software engineering information or project information that is available before the initiation of the pattern is transformed as a consequence of the successful execution of the pattern.

Resulting Context. Describes the conditions that will result once the pattern has been successfully implemented. Upon completion of the pattern:

(1) What organizational or team-related activities must have occurred?

(2) What is the exit state for the process?

(3) What software engineering information or project information has been developed?

Related Patterns. Provide a list of all process patterns that are directly related to this one. This may be represented as a hierarchy or in some other diagrammatic form.

Known Uses and Examples. Indicate the specific instances in which the pattern is applicable.

Process patterns provide an effective mechanism for addressing problems associated with any software process. The patterns enable you to develop a hierarchical process description that begins at a high level of abstraction (a phase pattern).

PROCESS ASSESSMENT AND IMPROVEMENT

Assessment attempts to understand the current state of the software process with the intent of improving it.

A number of different approaches to **software process assessment and improvement** have been proposed over the past few decades.

Standard CMMI Assessment Method for Process Improvement (SCAMPI)—provides a **five** step process assessment model that incorporates **five** phases: **initiating, diagnosing, establishing, acting, and learning.** The SCAMPI method uses the SEI CMMI as the basis for assessment.

CMM-Based Appraisal for Internal Process Improvement (CBA IPI)**—** provides a diagnostic technique for assessing the relative maturity of a software organization; uses the SEI CMM as the basis for the assessment.

SPICE (ISO/IEC15504)—a standard that defines a set of requirements for software process assessment. The intent of the standard is to assist organizations in developing an objective evaluation of the efficacy of any defined software process.

ISO 9001:2000 for Software—a generic standard that applies to any organization that wants to improve the overall quality of the products, systems, or services that it provides. Therefore, the standard is directly applicable to software organizations and companies.