2019 级本科

软件工程

Software Engineering

张昕

zhangxin@cust.edu.cn

计算机科学技术学院软件工程系

Software Process Structure

Please consider

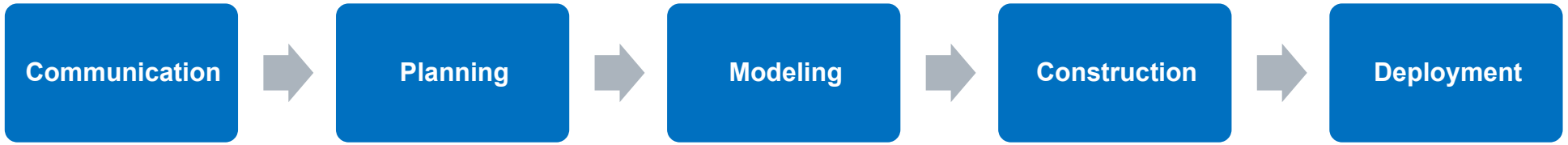
- When you work to build a product or system, it's important to go through a series of predictable steps—a road map that helps you create a timely, high-quality result. The road map that you follow is called a "software process."
- Because it provides stability, control, and organization to an activity that can, if left uncontrolled, become quite chaotic.
  - a modern software engineering approach must be "agile." It must demand only those activities, controls, and work products that are appropriate for the project team and the product that is to be produced.
- At a detailed level, the process that you adopt depends on the software that you're building. One process might be appropriate for creating software for an aircraft avionics system, while an entirely different process would be indicated for the creation of a website.

Please consider

- From the point of view of a software engineer, the work products are the programs, documents, and data that are produced as a consequence of the activities and tasks defined by the process.
- There are a number of software process assessment mechanisms that enable organizations to determine the “maturity” of their software process. However, the quality, timeliness, and long-term viability of the product you build are the best indicators of the efficacy of the process that you use.

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 resides within a framework or model that defines their relationship with the process and with one another.

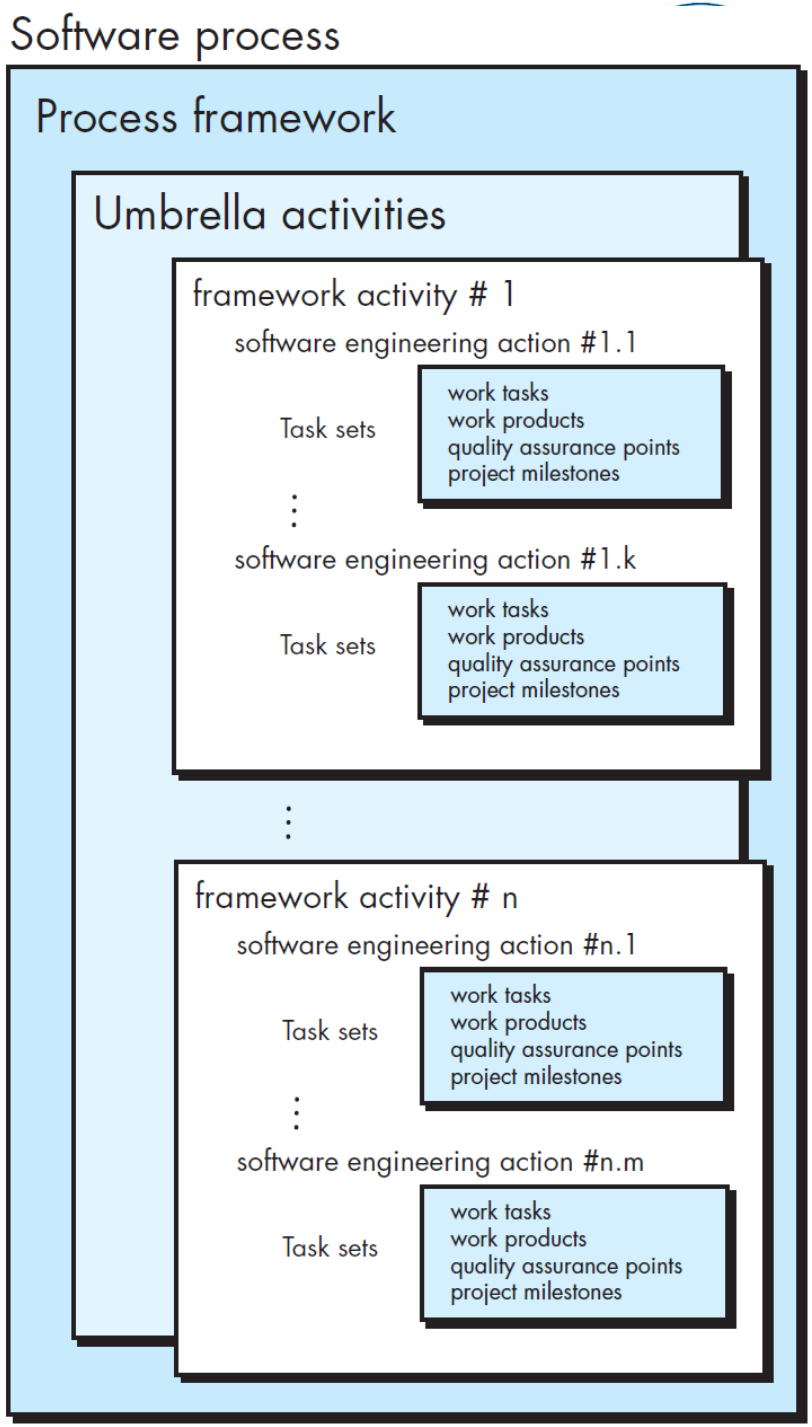


  - A set of umbrella activities—project tracking and control, risk management, quality assurance, configuration management, technical reviews, and others—are applied throughout the process.

- A linear process fl ow executes each of the five framework activities in sequence, beginning with communication and culminating with deployment.

A generic process model

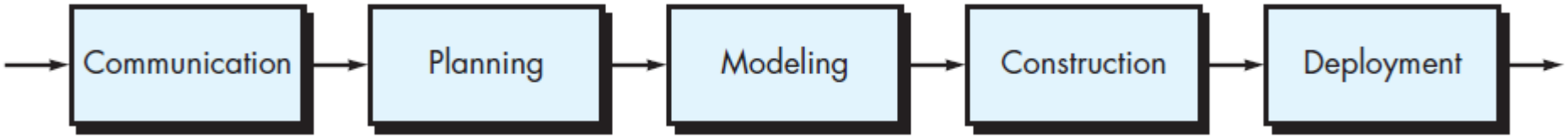
- Each framework activity is populated by a set of software engineering actions.
  - Each software engineering action is defined by a task set that identifies the work tasks that are to be completed, the work products that will be produced, the quality assurance points that will be required, and the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



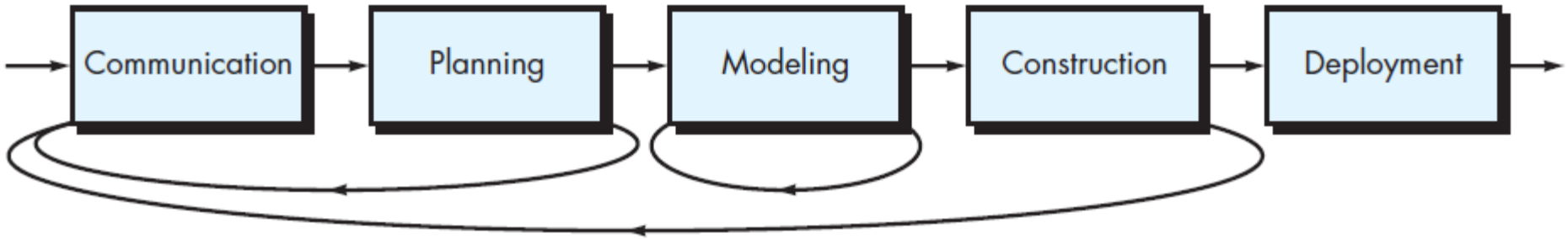
A generic process model

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
- A linear process flow executes each of the five framework activities in sequence, beginning with communication and culminating with deployment

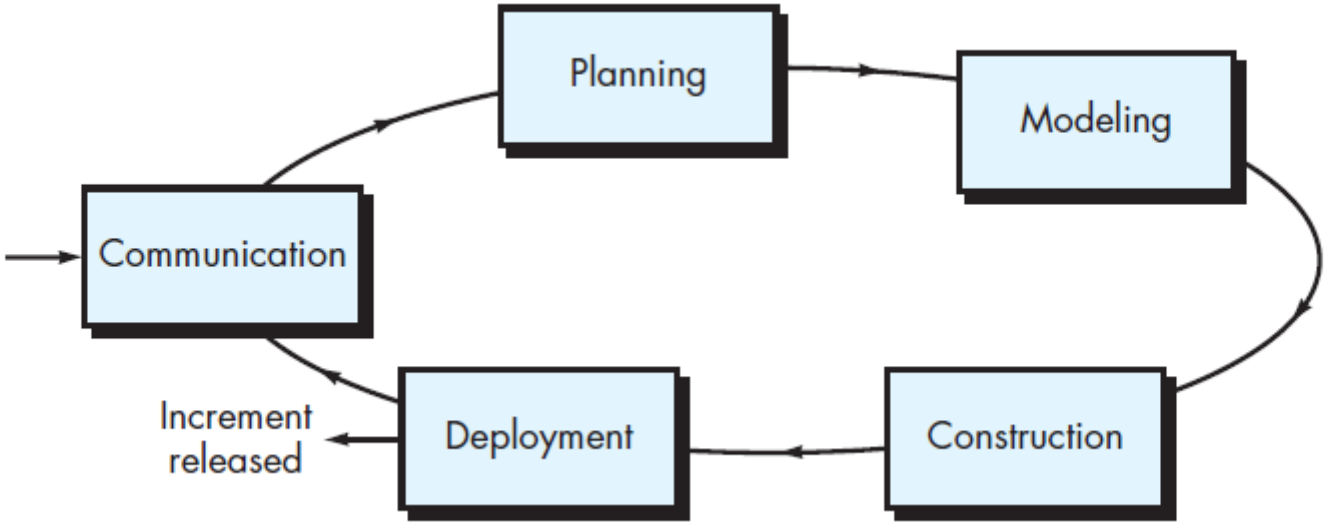


- An iterative process flow repeats one or more of the activities before proceeding to the next



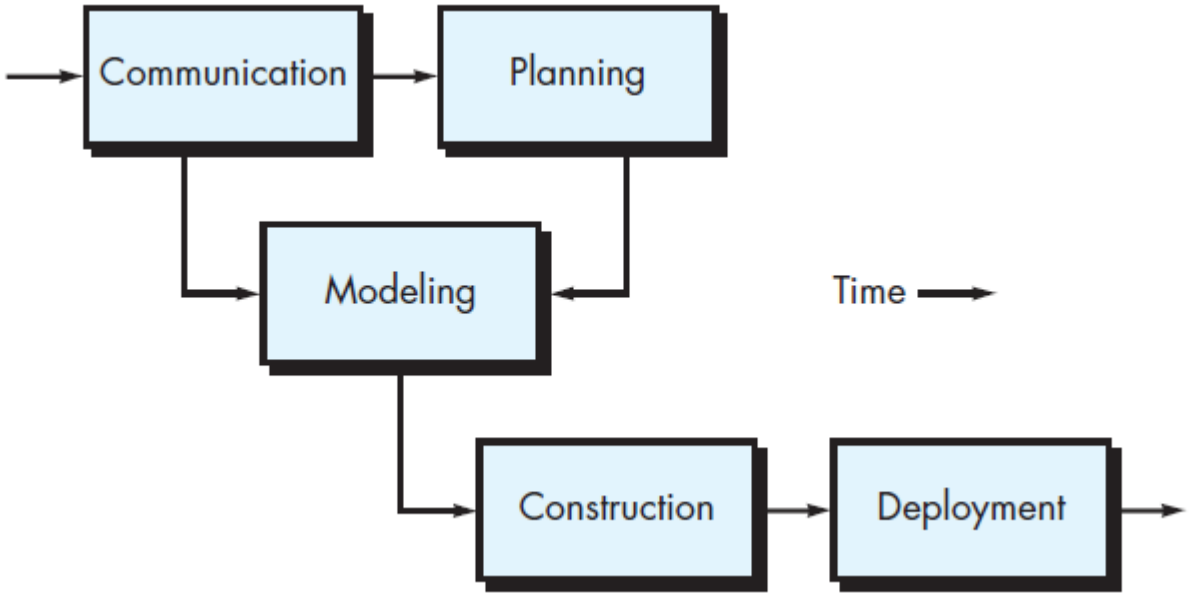
A generic process model

- 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
  - 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 generic process model

- 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
  - 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).



A generic process model

- A software team would need significantly more information before it could properly execute any one of these activities as part of the software process.
- Key questions:
  - 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

- A task set defines the actual work to be done to accomplish the objectives of a software engineering action.
  - For example
    - Elicitation: requirements gathering to understand what various stakeholders want from the software that is to be built.

[small, relatively simple project]

1. Make a list of stakeholders for the project.
2. Invite all stakeholders to an informal meeting.
3. Ask each stakeholder to make a list of features and functions required.
4. Discuss requirements and build a final list.
5. Prioritize requirements.
6. Note areas of uncertainty.

[larger, more complex software project]

1. Make a list of stakeholders for the project.
2. Interview each stakeholder separately to determine overall wants and needs.
3. Build a preliminary list of functions and features based on stakeholder input.
4. Schedule a series of facilitated application specification meetings.
5. Conduct meetings.
6. Produce informal user scenarios as part of each meeting.
7. Refine user scenarios based on stakeholder feedback.
8. Build a revised list of stakeholder requirements.
9. Use quality function deployment techniques to prioritize requirements.
10. Package requirements so that they can be delivered incrementally.
11. Note constraints and restrictions that will be placed on the system.
12. Discuss methods for validating the system.

- a software engineering action can be adapted to the specific needs of the software project and the characteristics of the project team.

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.
- A process pattern provides you with a template — a consistent method for describing problem solutions within the context of the software process.
  - By combining patterns, a software team can solve problems and construct a process that best meets the needs of a project.
- Process patterns provide an effective mechanism for addressing problems associated with any software process.
- Patterns can be defined at any level of abstraction.
  - In some cases, 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).

Pattern types

- 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 EstablishingCommunication. This pattern would incorporate the task pattern RequirementsGathering and others.
- Task pattern
  - defines a problem associated with a software engineering action or work task and relevant to successful software engineering practice (e.g., RequirementsGathering is a task pattern).
- 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 SpiralModel or Prototyping .

Pattern template

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?
- For example, the Planning pattern (a stage pattern) requires that
  - (1) customers and software engineers have established a collaborative communication;
  - (2) successful completion of a number of task patterns [specified] for the Communication pattern has occurred; and
  - (3) the project scope, basic business requirements, and project constraints are known.

Pattern template

Problem

- The specific problem to be solved by the pattern.

Solution

- Describes how to implement the pattern successfully.
- This section describes how the initial state of the process (that exists before the pattern is implemented) is modified as a consequence of the initiation of the pattern.
- 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?

Pattern template

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.
- For example, the stage pattern Communication encompasses the task patterns: ProjectTeam, CollaborativeGuidelines, ScopeIsolation, RequirementsGathering, ConstraintDescription, and ScenarioCreation.

Known Uses and Examples

- Indicate the specific instances in which the pattern is applicable.
- For example, Communication is mandatory at the beginning of every software project, is recommended throughout the software project, and is mandatory once the Deployment activity is under way.

- The patterns enable you to develop a hierarchical process description that begins at a high level of abstraction (a phase pattern).
- The description is then refined into a set of stage patterns that describe framework activities and are further refined in a hierarchical fashion into more detailed task patterns for each stage pattern.
- Once process patterns have been developed, they can be reused for the definition of process variants.

Example

Pattern Name. **RequirementsUnclear**

Intent. This pattern describes an approach for building a model (a prototype) that can be assessed iteratively by stakeholders in an effort to identify or solidify software requirements.

Type. Phase pattern.

Initial Context.

The following conditions must be met prior to the initiation of this pattern:

(1) stakeholders have been identified;

(2) a mode of communication between stakeholders and the software team has been established;

(3) the overriding software problem to be solved has been identified by stakeholders;

(4) an initial understanding of project scope, basic business requirements, and project constraints has been developed.

Problem. Requirements are hazy or nonexistent, yet there is clear recognition that there is a problem to be solved, and the problem must be addressed with a software solution. Stakeholders are unsure of what they want; that is, they cannot describe software requirements in any detail.

Solution. A description of the prototyping process would be presented.

Example

Resulting Context.

A software prototype that identifies basic requirements (e.g., modes of interaction, computational features, processing functions) is approved by stakeholders.

Following this, (1) the prototype may evolve through a series of increments to become the production software or (2) the prototype may be discarded and the production software built using some other process pattern.

Related Patterns. The following patterns are related to this pattern: **CustomerCommunication, IterativeDesign, IterativeDevelopment, CustomerAssessment, RequirementExtraction.**

Known Uses and Examples. Prototyping is recommended when requirements are uncertain.