共计:56人

Grouping List

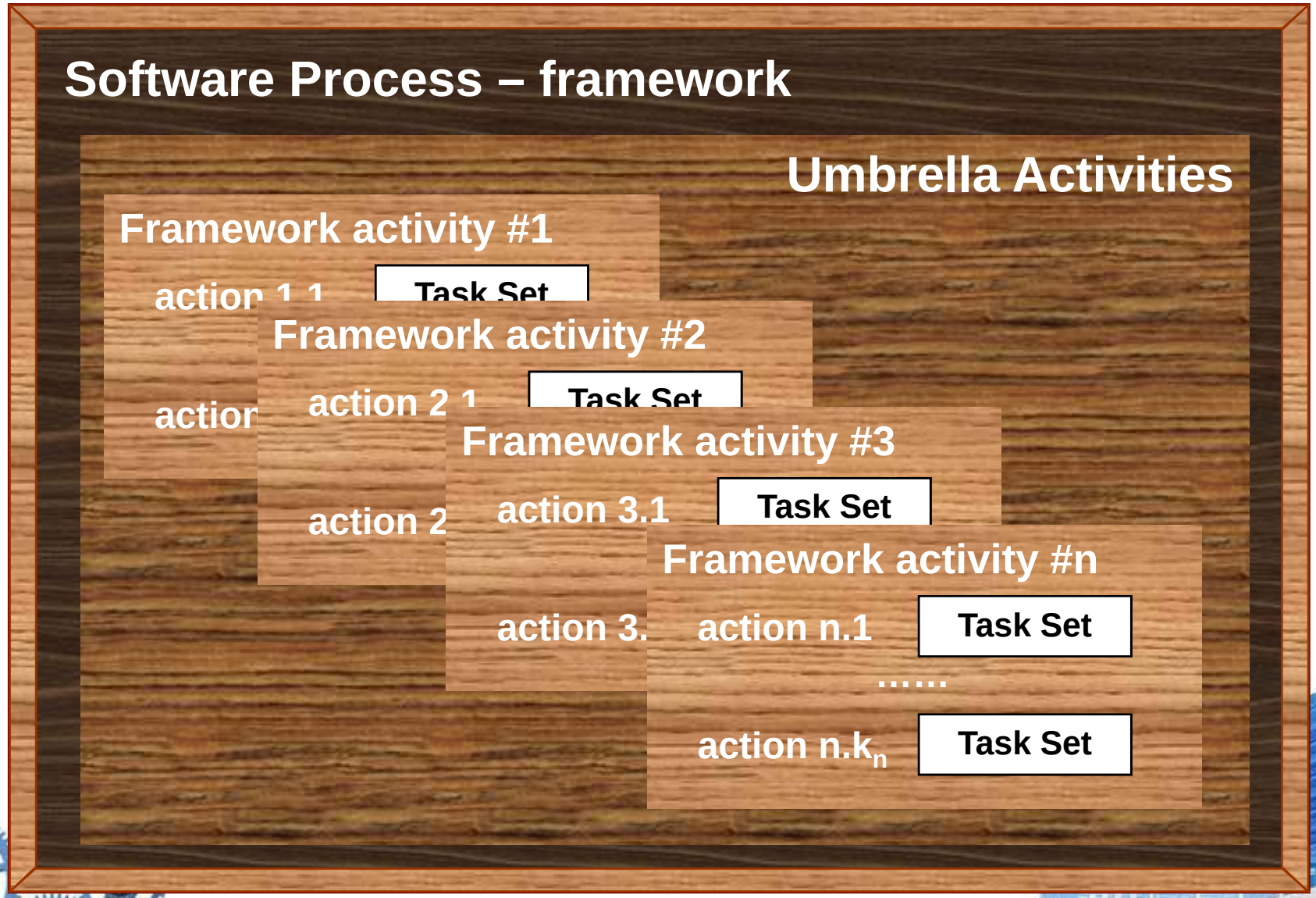
队别	组别	组长	成员1	成员2	成员3	成员4	分配模块
A队 大队长: 张宇昊	A1	葛现隆	范元瑞	李经纶	吴忆杰	林立文	基础信息管理
	A2	宋志平	魏煜欣	张孝舟	徐桦林	郑一村	自动排课
	A3	张　闻	周宇恒	杨梦晗	胡译心		选课
	A4	弓得力	张宇昊	李逸婷	胡冯欣	高　涛	资源共享
	A5	吕锴燮	刘俊灏	曾泽栋	田逸飞		在线测试
	A6	项王盟	俞佳炳	陈梦静	唐思远	李　昊	成绩管理
B队 大队长: 胡滨	B1	张永航	林初剑	徐嘉伟	谭　啸		基础信息管理
	B2	王天露	姜兴华	王　涛	陈炯坚	胡春望	自动排课
	B3	李书楠	周　天	庞罕天	张鹏程	刘耕铭	选课
	B4	徐可添	万　博	辛　浩	王禹杰		资源共享
	B5	胡　滨	于音之	傅益芳	陆　洲	沈　赟	在线测试
	B6	李思捷	陈　爽	邓永辉	何天杨		成绩管理

共计：57人

Ch.3 Software Process Structure

March 22, 2015

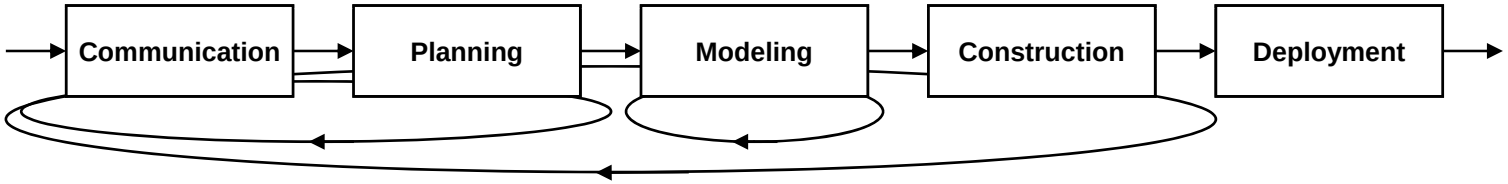
3.1 A Generic Process Model



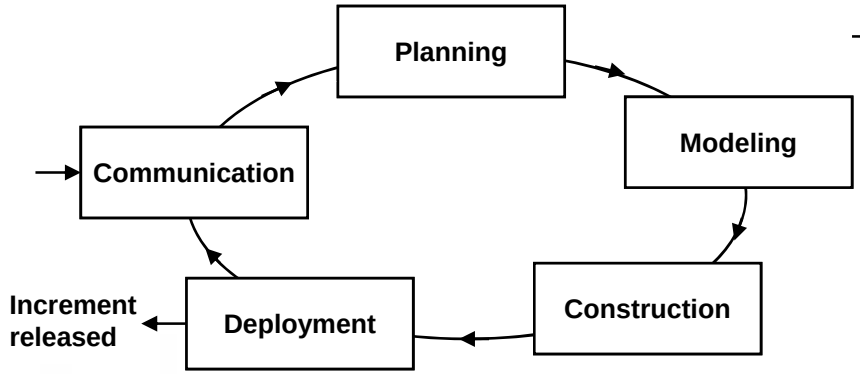
3.1 A Generic Process Model

- **Process flow**

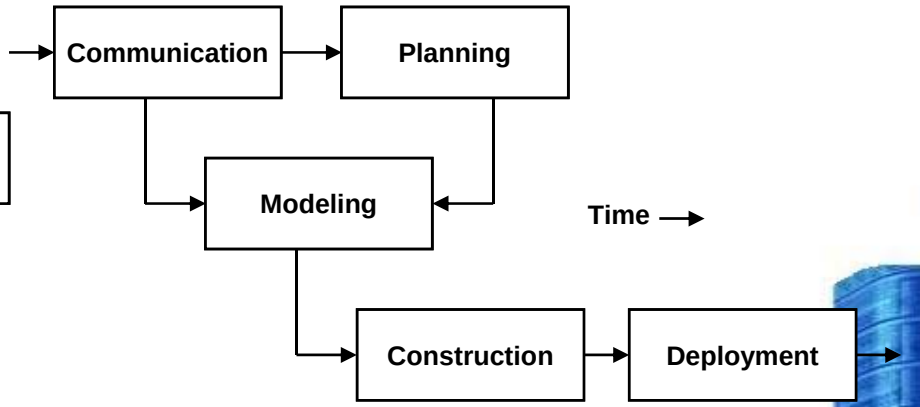
(a) Linear process flow



(b) Iterative process flow



(c) Evolutionary process flow



(d) Parallel process flow

3.4 Process Patterns

- **Process patterns** define a set of activities, actions, work tasks, work products and/or related behaviors
- A **template** is used to define a pattern
- **Generic software pattern elements**

•**Generic software pattern elements**

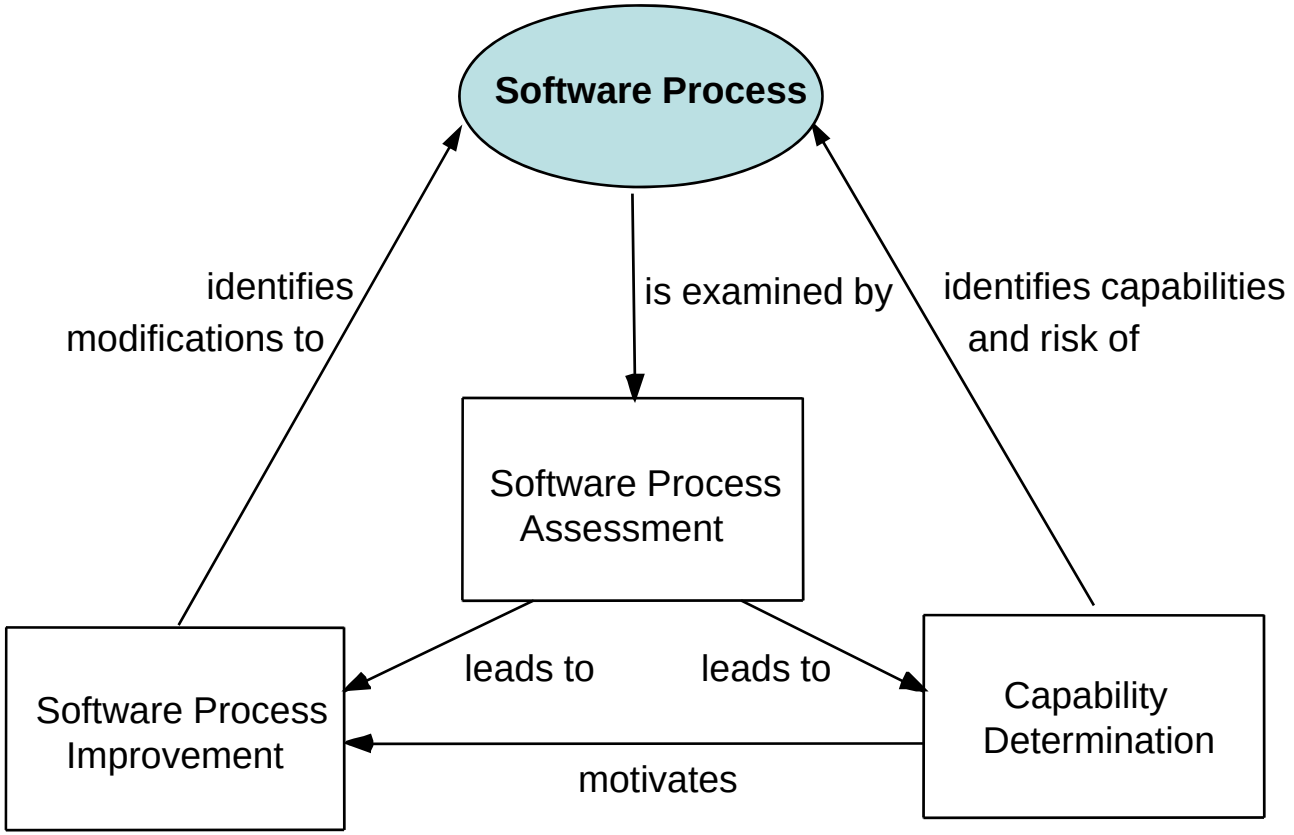
– **Meaningful pattern name (e.g. Customer Com**

– **Intent (objective of pattern)**

– **Type**

- **Task pattern (defines engineering action (e.g. Requirement Gathering，ZJU Watch)**
- **Stage pattern (defines framework activity for the process) (e.g. Communication)**
- **Phase pattern (defines sequence or flow of framework activities that occur within process) (e.g. Spiral model or Prototyping<--原型)**

3.5 Process Assessment



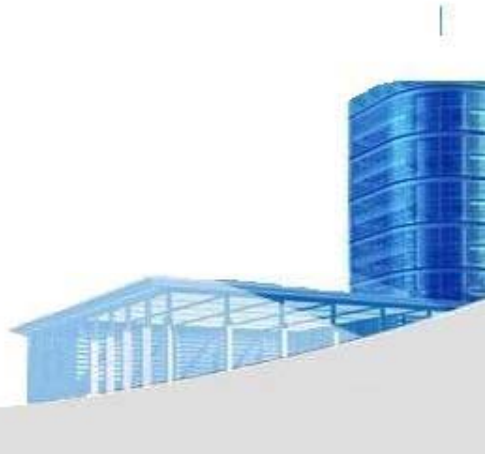
- 📖 **SCAMPI**
- 📖 **CBA IPI**
- 📖 **SPICE (ISO/IEC15504)**
- 📖 **ISO 9001:2000 for Software**

The Capability Maturity Model Integration

— by Software Engineering Institute (SEI) of Carnegie Mellon University (CMU)

- **Level 0: Incomplete** (process is not performed or does not achieve all goals defined for this level)
- **Level 1: Performed** (work tasks required to produce required work products are being conducted)
- **Level 2: Managed** (people doing work have access to adequate resources to get job done, stakeholders are actively involved, work tasks and products are monitored, reviewed, and evaluated for conformance to process description)
- **Level 3: Defined** (management and engineering processes documented, standardized, and integrated into organization-wide software process)
- **Level 4: Quantitatively Managed** (software process and products are quantitatively understood and controlled using detailed measures)
- **Level 5: Optimizing** (continuous process improvement is enabled by quantitative feedback from the process and testing innovative ideas)

Ch.4 Process Models



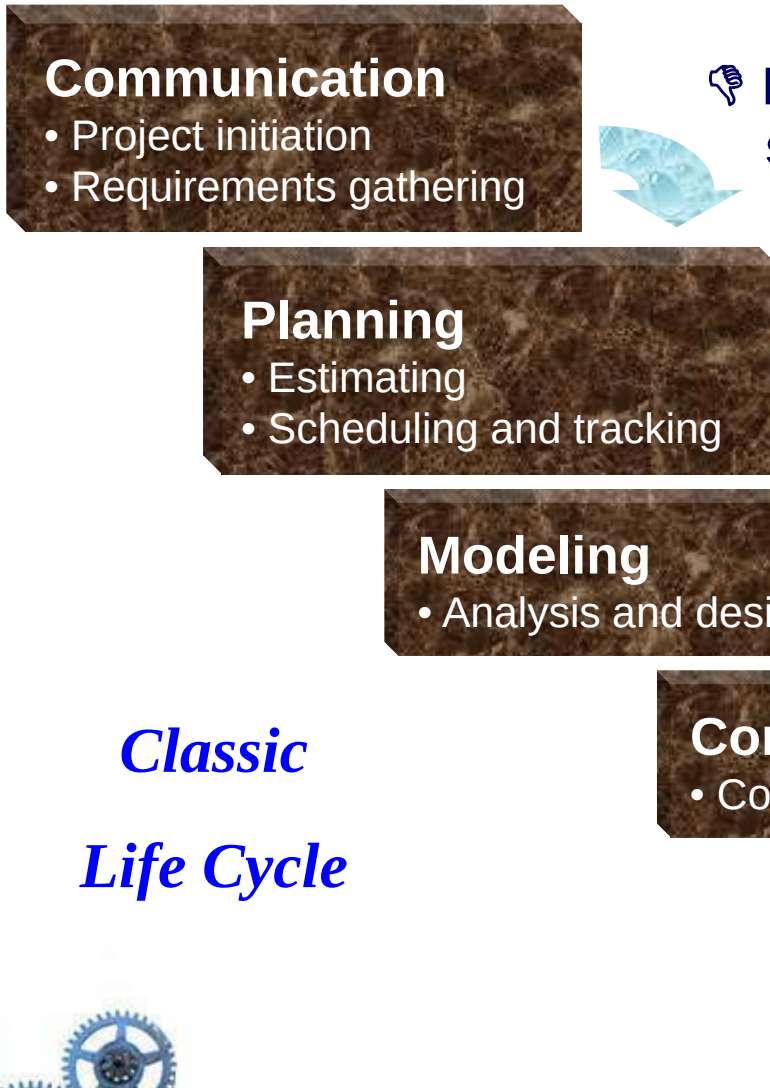
4.1 Prescriptive Models

- **Prescriptive （惯例） process models advocate an orderly approach to software engineering**

- **Questions:**

1. If prescriptive process models strive （兴盛 / 持续） for structure and order, are they inappropriate for a software world that thrives on change?
2. Yet, if we reject traditional process models (and the order they imply) and replace them with something less structured（e.g. Agile）, do we make it impossible to achieve coordination （协调） and coherence （一致） in software work?

4.1.1 The Waterfall Model

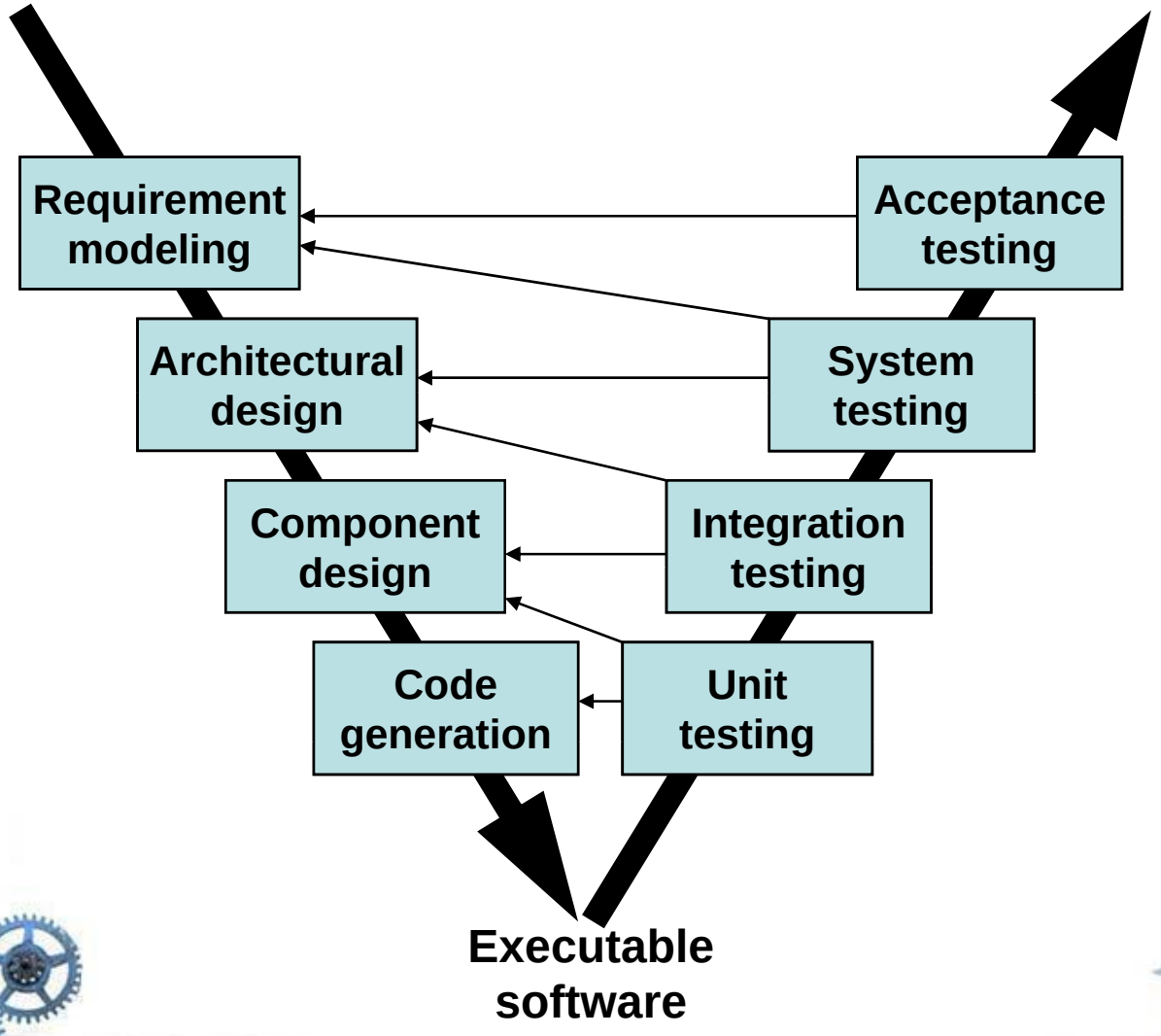


- Real projects rarely follow the sequential flow.
- Customers usually can't state all requirements explicitly.
- A working version will not be available until late in the project time-span.

Classic Life Cycle

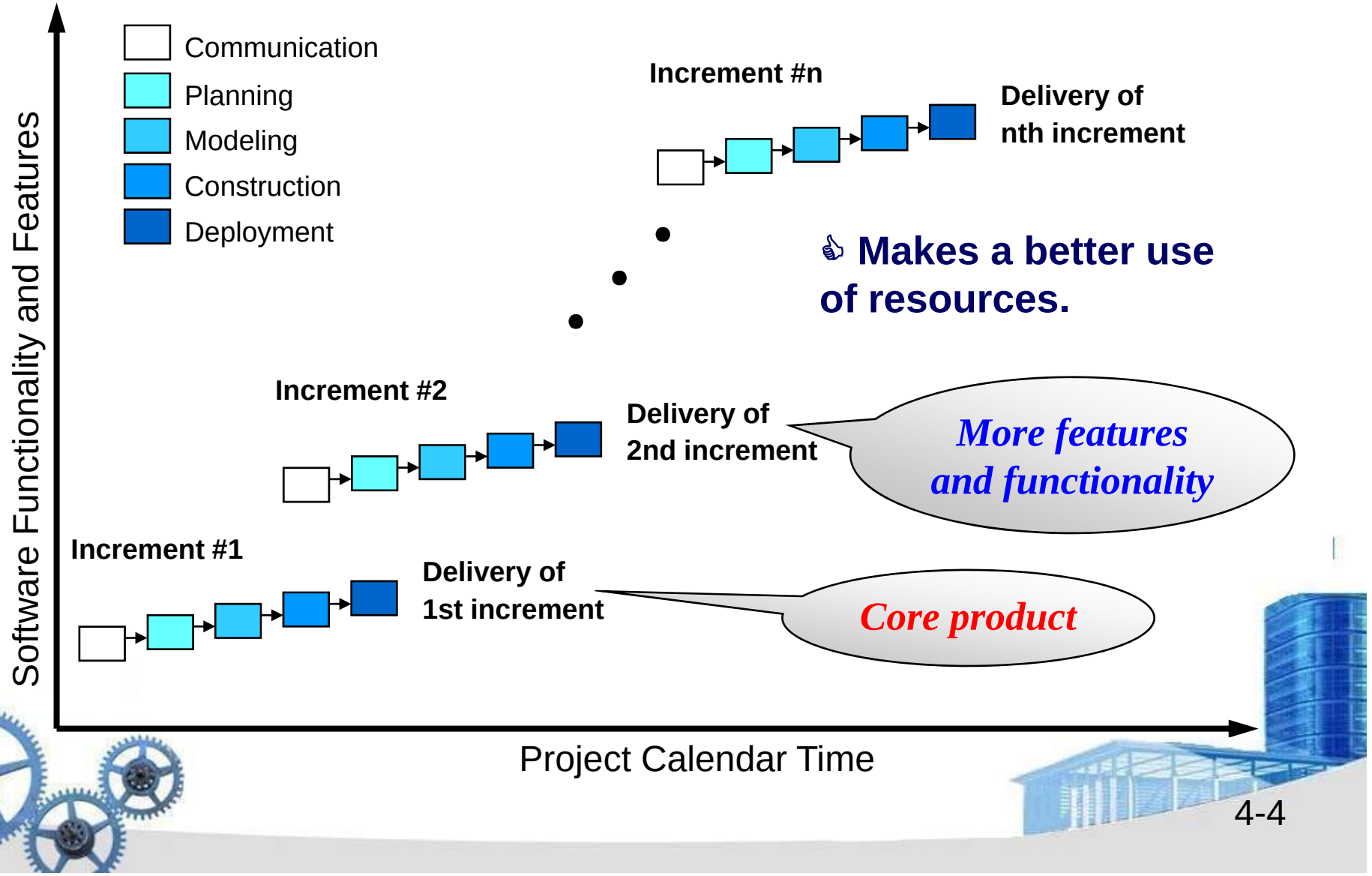
4.1.1 The Waterfall Model

- The V-model



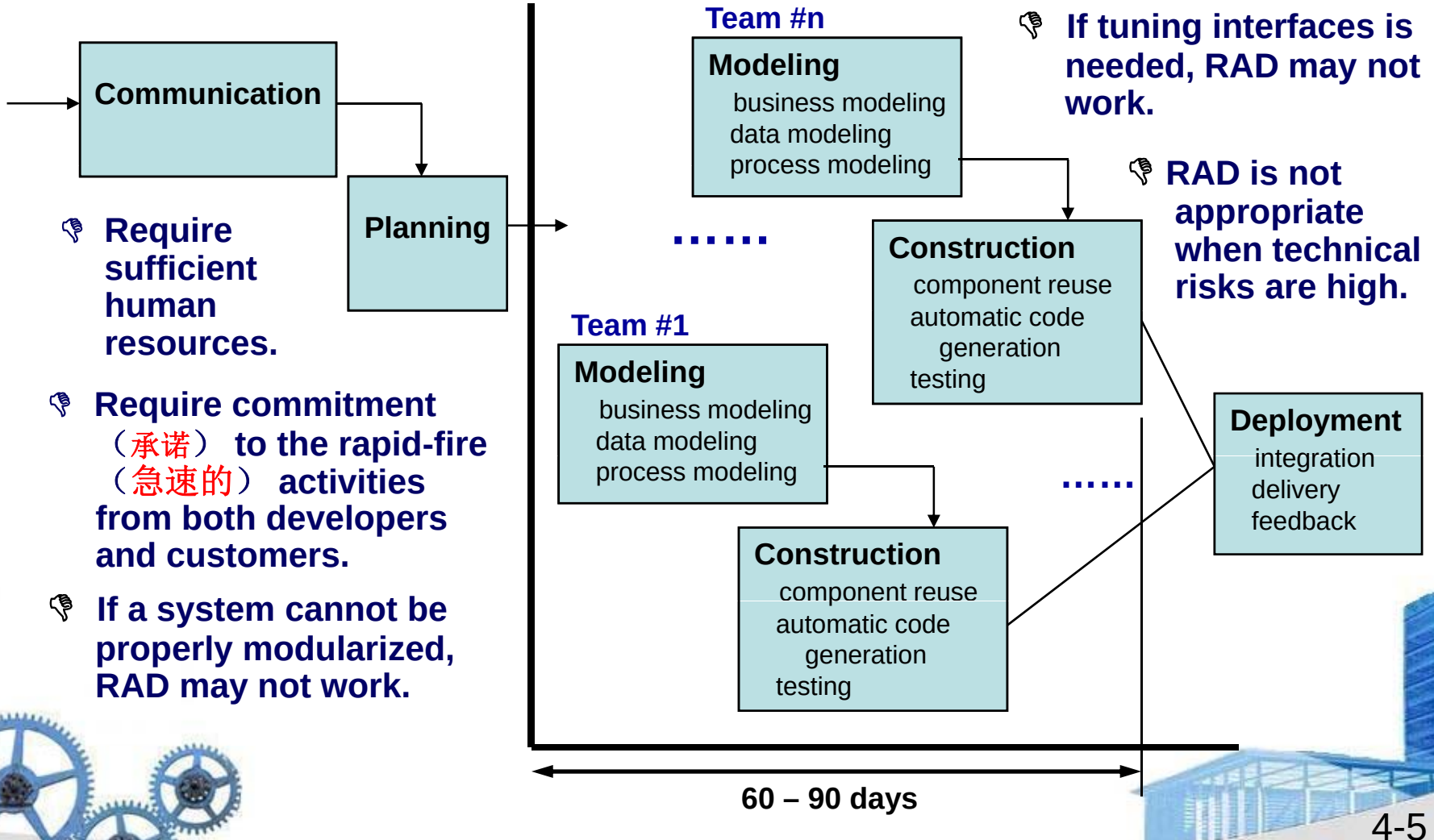
4.1.2 Incremental (增量) Process Models

- **The Incremental Model**



4.1.2 Incremental Process Models

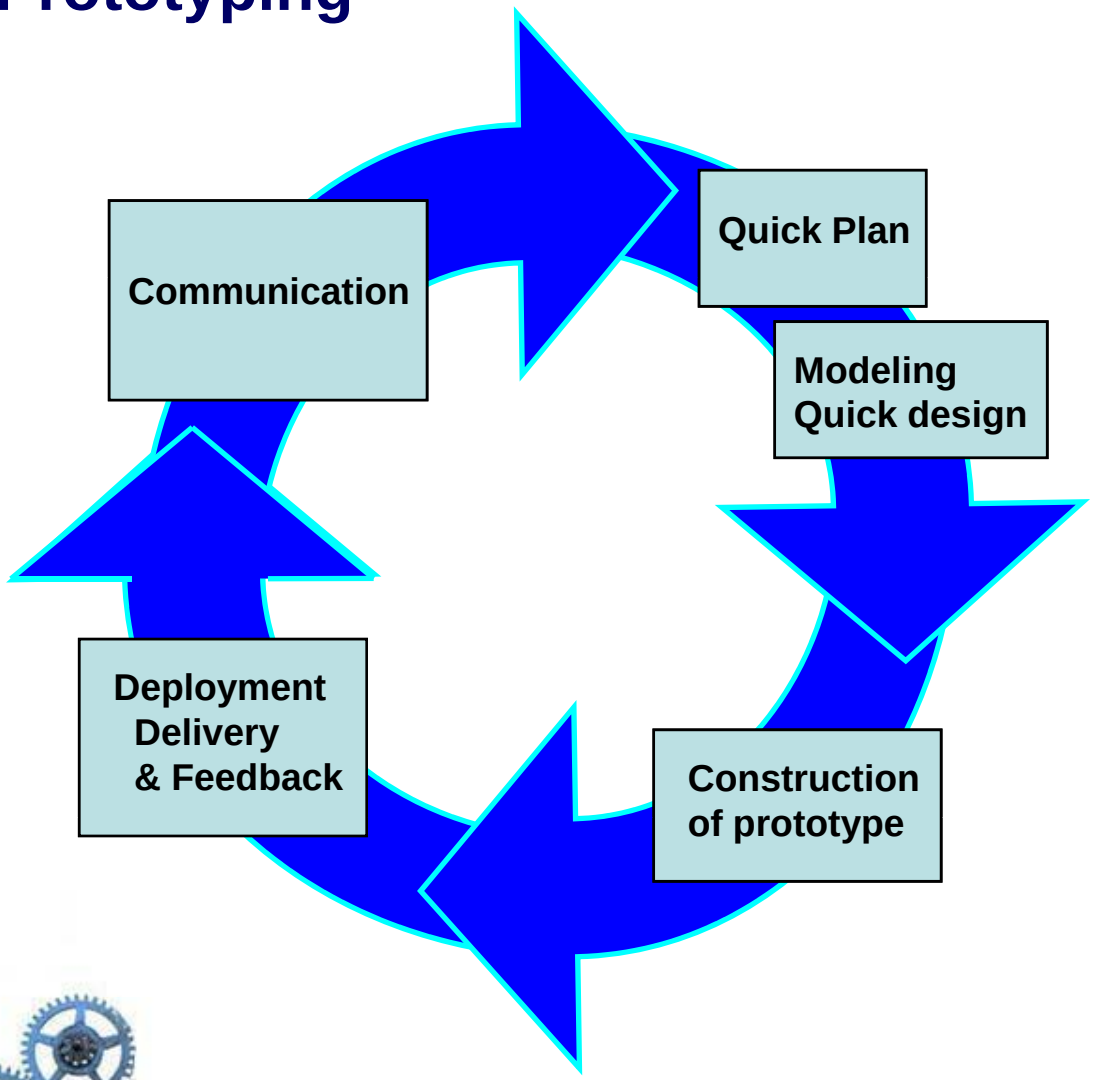
• The Rapid Application Development (RAD) Model



- Require sufficient human resources.
- Require commitment （承诺） to the rapid-fire （急速的） activities from both developers and customers.
- If a system cannot be properly modularized, RAD may not work.
- If tuning interfaces is needed, RAD may not work.
- RAD is not appropriate when technical risks are high.

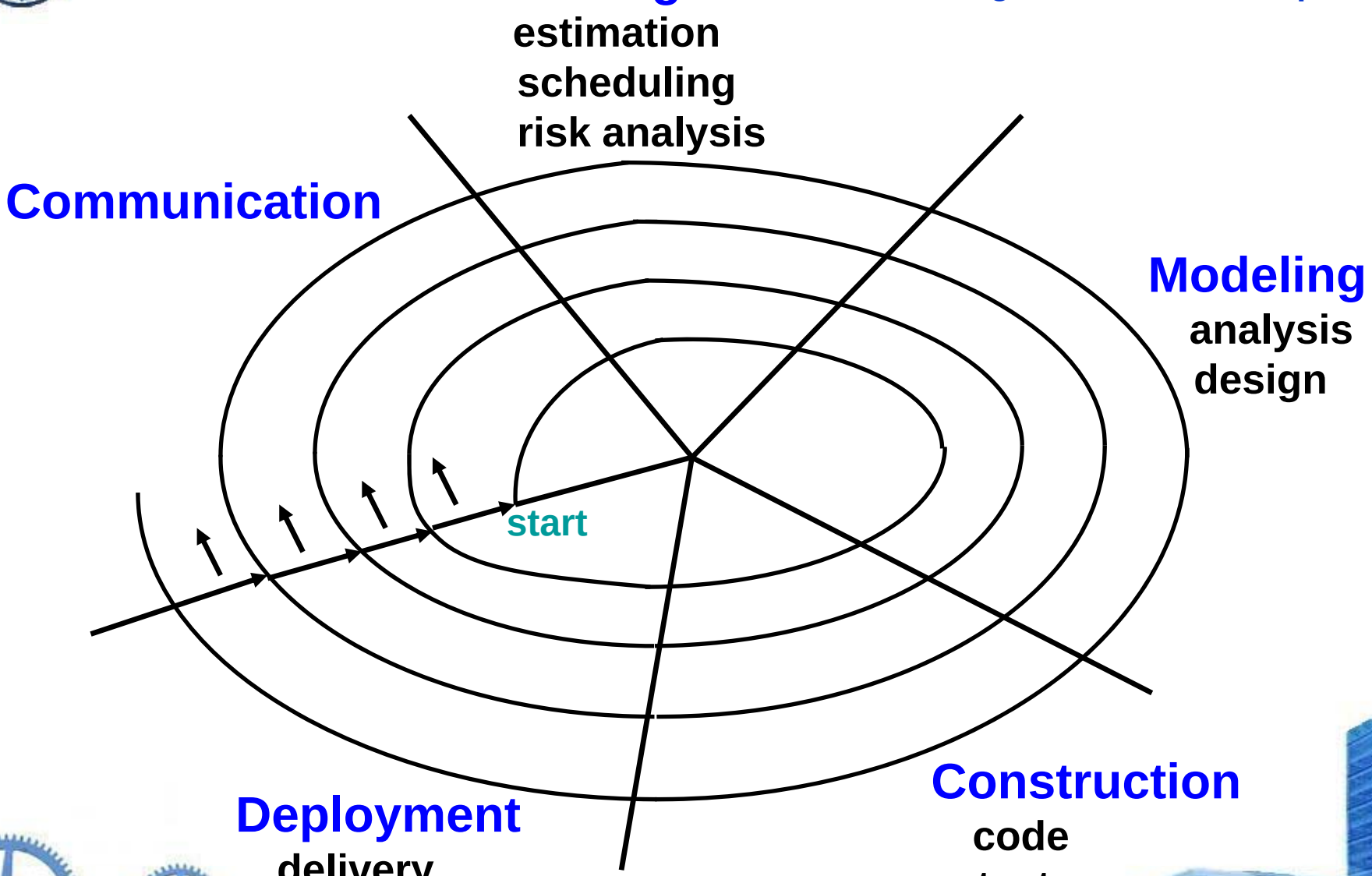
4.1.3 Evolutionary Process Models

- **Prototyping**

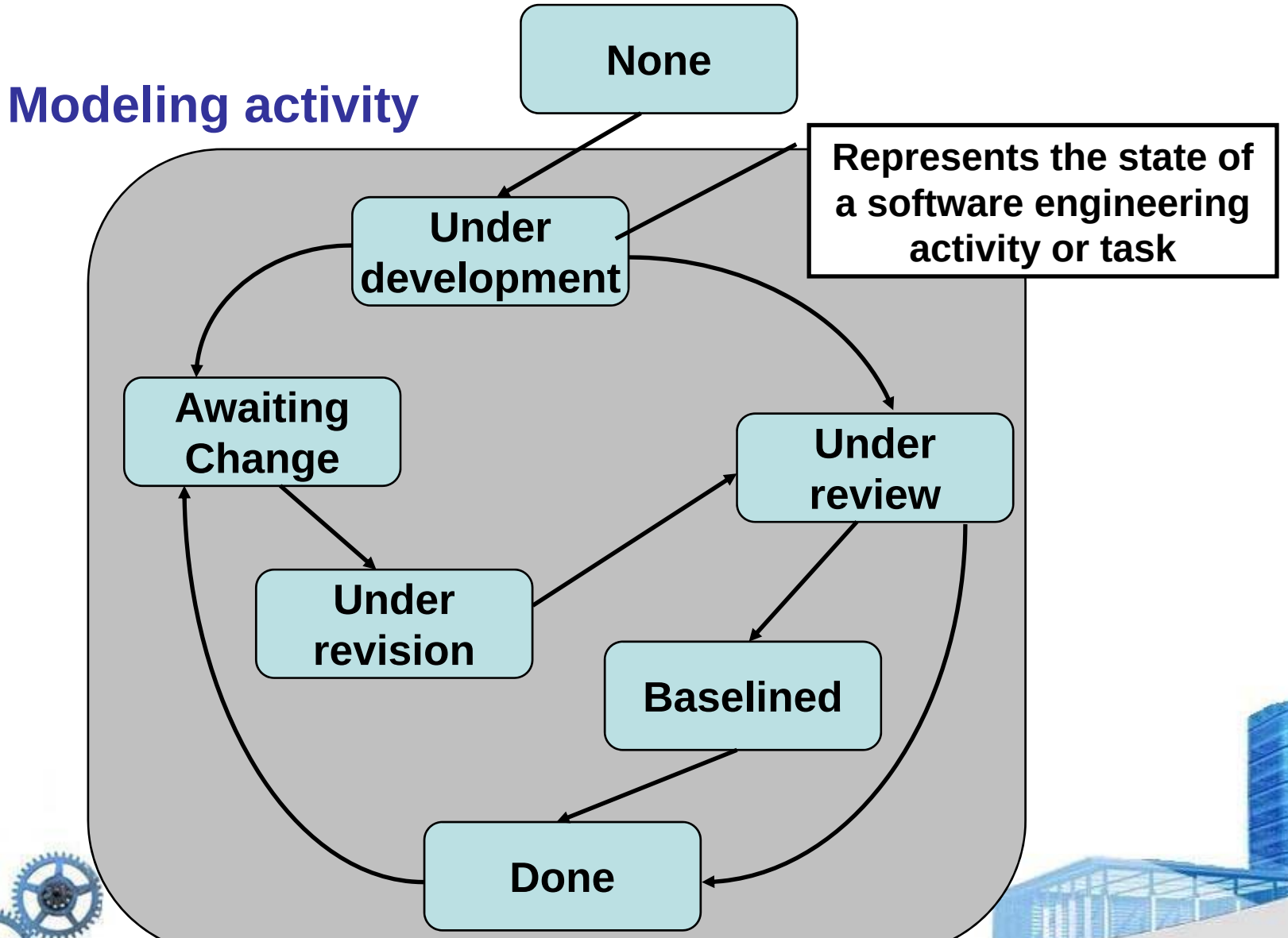


- Good first step when customer has a legitimate need, but is clueless about the details
- The prototype must be thrown away

4.1.3 Evolutionary Process Models---Spiral Model



4.1.4 Evolutionary Process Models
Concurrent（协同）Development Model



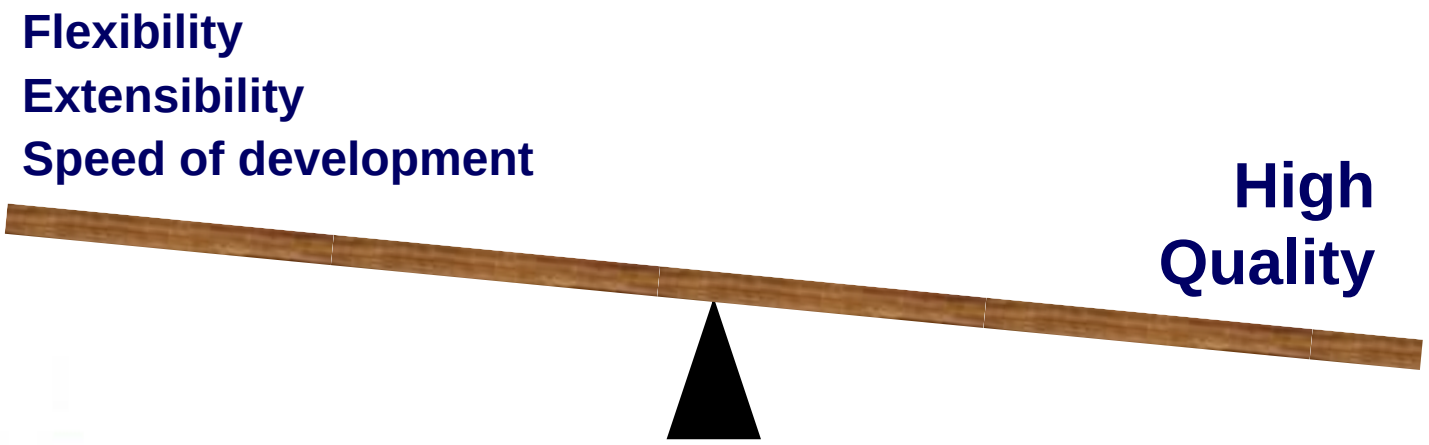
4.1.4 Evolutionary Process Models

- **The Concurrent Development Model**

➢ Defines a series of events that will trigger transitions from state to state for each of the activities, actions or tasks.

➢ Especially good for client/server applications.

➢ Defines a network of activities instead of linear sequence of events.



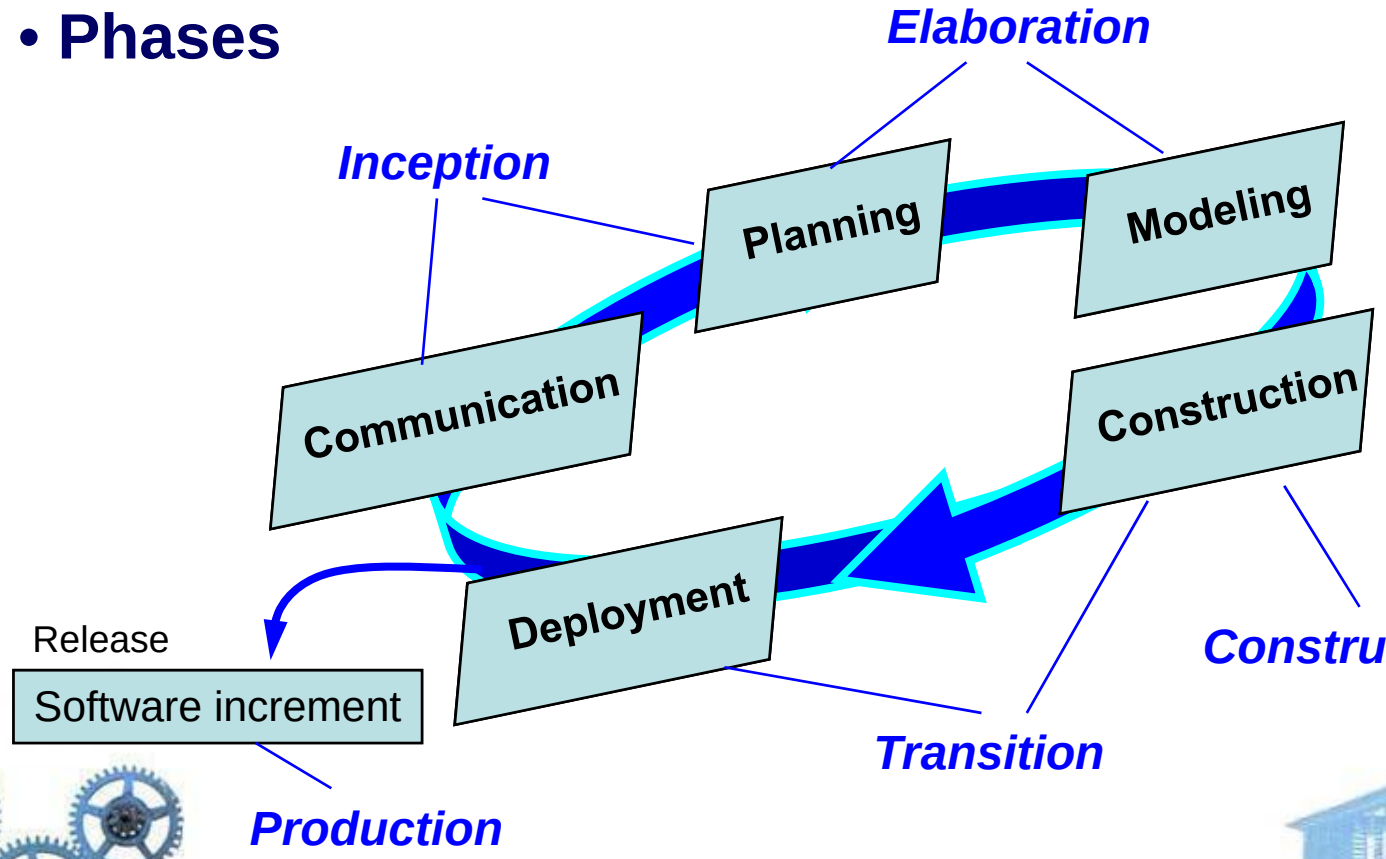
4.2 Specialized Process Models

- **Component based development** — the process to apply when reuse is a development objective
- **Formal methods** — emphasizes the mathematical specification of requirements
- **Aspect-Oriented Software Development** — provides a process and methodological approach for defining, specifying, designing, and constructing aspects

4.3 The Unified Process

- A "**use-case** driven, **architecture**-centric, **iterative** and **incremental**" software process closely aligned with the **U**nified **M**odeling **L**anguage (**UML**)

- **Phases**



4.3 The Unified Process

• Work Products

Inception phase

- Vision document
- Initial use-case model
- Initial project glossary
- Initial business case
- Initial risk assessment
- Project plan
  phases and iterations
- Business model
- Prototypes

Elaboration phase

- Use-case model
- Functional and non-functional requirements
- Analysis model
- Software architecture description
- Executable architectural prototype
- Preliminary design model
- Revise risk list
- Project plan
  iteration plan, workflow, milestones
- Preliminary user manual

Construction phase

- Design model
- Software components
- Integrated software increment
- Test plan
- Test cases
- Support documentation
  user
  installation
  increment

Transition phase

- Delivered software increment
- Beta test reports
- User feedback

4.4 Personal and Team Process Models

- **Personal Software Process (PSP)) Ex. WPS(求伯君)**

➢ Recommends five framework activities:

1. Planning
2. High-level design
3. High-level design review
4. Development
5. Postmortem (后验)

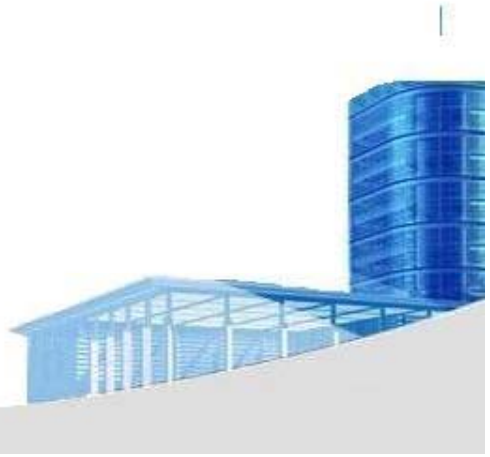
➢ Stresses the need for each software engineer to identify errors early and as important, to understand the types of errors

4.4 Personal and Team Process Models

- **Team Software Process (TSP)**

  - Each project is "launched" using a "script" that defines the tasks to be accomplished
  - Teams are self-directed
  - Measurement is encouraged
  - Measures are analyzed with the intent of improving the team process

Ch.31 Project Management Concepts



- **The 4P's**
  - *People* — the most important element of a successful project
  - *Product* — the software to be built
  - *Process* — the set of framework activities and software engineering tasks to get the job done
  - *Project* — all work required to make the product a reality

- **Stakeholders**

  - *Senior managers* who define the business issues that often have significant influence on the project.
  - *Project (technical) managers* who must plan, motivate, organize, and control the practitioners who do software work.
  - *Practitioners* who deliver the technical skills that are necessary to engineer a product or application.
  - *Customers* who specify the requirements for the software to be engineered and other stakeholders who have a peripheral interest in the outcome.
  - *End-users* who interact with the software once it is released for production use.

- **Software Teams**

How to lead?

How to organize?

How to collaborate?

How to motivate?

How to create good ideas?

- **Team Leader**
  - *The MOI Model*
    - *Motivation.* The ability to encourage (by "push or pull") technical people to produce to their best ability.
    - *Organization.* The ability to mold existing processes (or invent new ones) that will enable the initial concept to be translated into a final product.
    - *Ideas or innovation.* The ability to encourage people to create and feel creative even when they must work within bounds established for a particular software product or application.

- **Software Teams**
  - *The following factors must be considered when selecting a software project team structure ...*
    - the *difficulty of the problem* to be solved
    - the *size of the resultant program(s)* in lines of code or function points
    - the *time that the team will stay together* (team lifetime)
    - the *degree to which the problem can be modularized*
    - the *required quality and reliability* of the system to be built
    - the *rigidity of the delivery date*
    - the *degree of sociability (communication)* required for the project

- **Organizational Paradigms→范式，范型**
  - ***closed paradigm*** —structures a team along a traditional hierarchy of authority
  - ***random paradigm***—structures a team loosely and depends on individual initiative of the team members
  - ***open paradigm***—attempts to structure a team in a manner that achieves some of the controls associated with the closed paradigm but also much of the innovation that occurs when using the random paradigm
  - ***synchronous (同步的)paradigm***—relies on the natural compartmentalization of a problem and organizes team members to work on pieces of the problem with little active communication among themselves

*suggested by **Constantine** [Con93]*

破碎的

协调的

- **Avoid Team "Toxicity"**
  - A frenzied work atmosphere in which team members waste energy and lose focus on the objectives of the work to be performed.
  - High frustration caused by personal, business, or technological factors that cause friction among team members.
  - "Fragmented or poorly coordinated procedures" or a poorly defined or improperly chosen process model that becomes a roadblock to accomplishment.
  - Unclear definition of roles resulting in a lack of accountability and resultant **finger-pointing**.
  - "Continuous and repeated exposure to failure" that leads to a loss of confidence and a lowering of morale.

责任

士气，斗志

- **Agile Teams**
  - Team members must have trust in one another.
  - The distribution of skills must be appropriate to the problem.
  - Mavericks may have to be excluded from the team, if team cohesiveness is to be maintained.
  - Team is “self-organizing”
    - An adaptive team structure
    - Uses elements of Constantine’s random, open, and synchronous paradigms
    - Significant autonomy

不服从的人，害群之马

Task

- **Review** Chapter 3，4，31
- **Finish** “Problems and points to ponder” in Ch. 3, 4, 31
- **Prepare for** Requirement Report**!**
- **Preview** Ch.5 (Agile), 6, 7, 8, 9