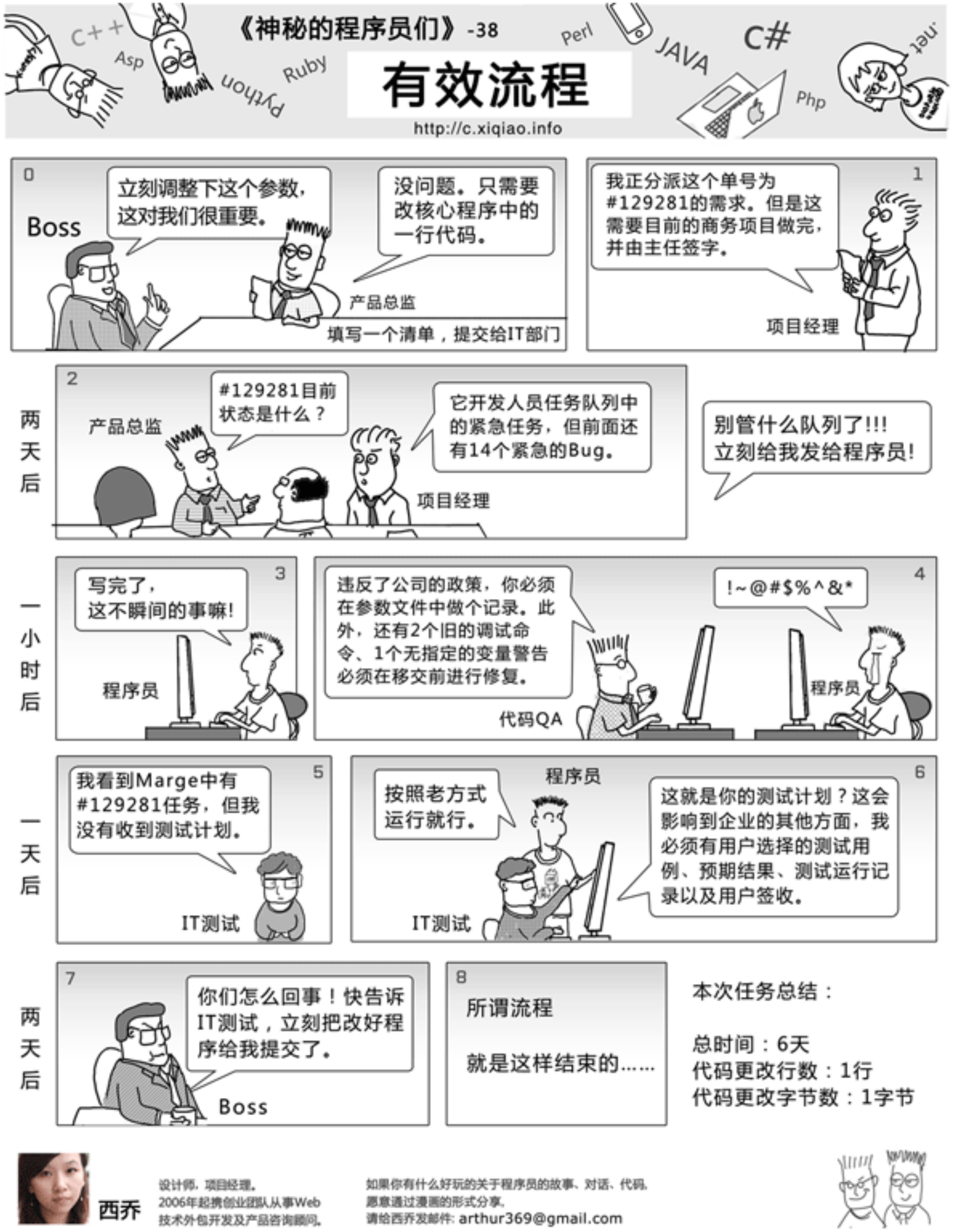
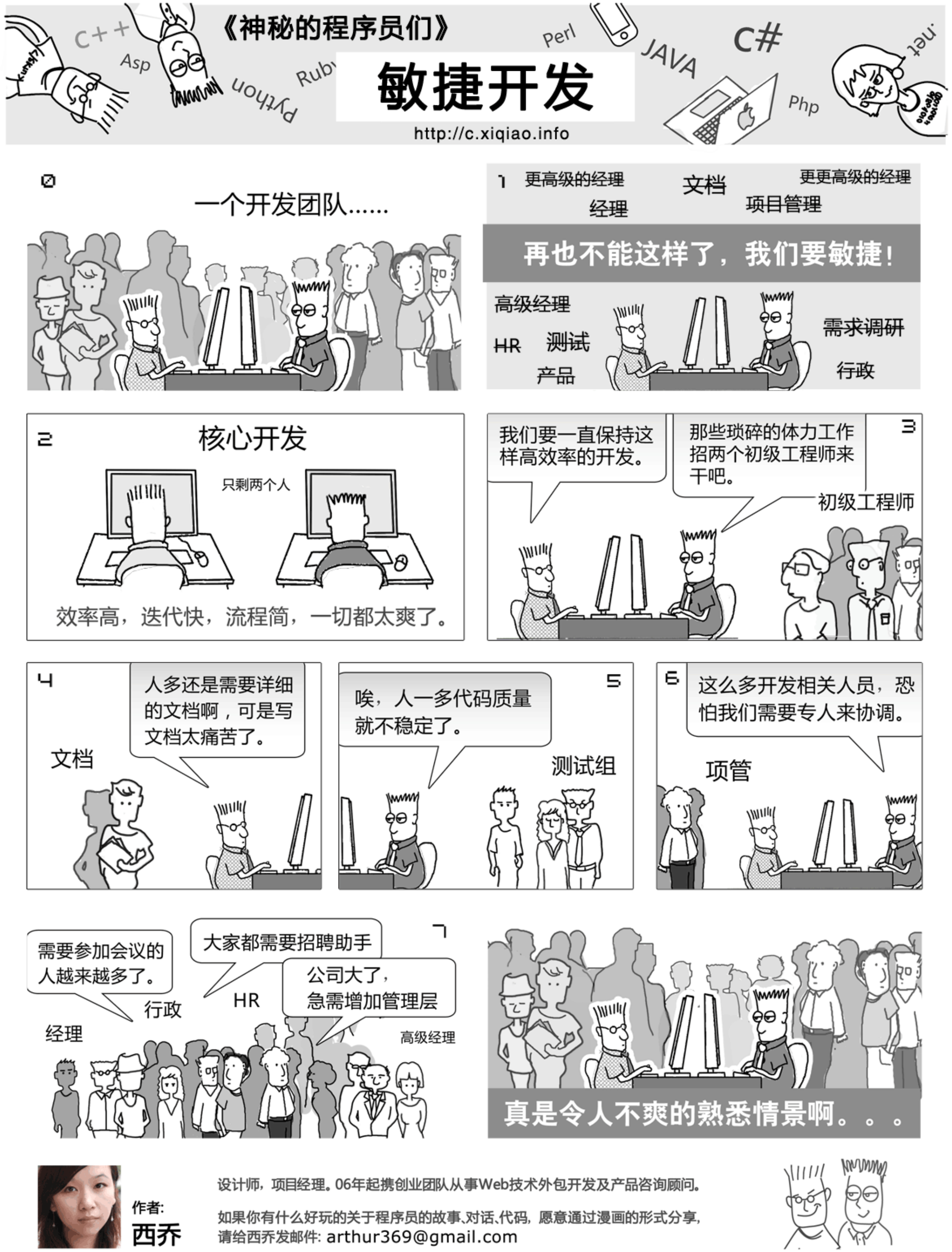
SOFTWARE ENGINEERING

CHAPTER-2
PROCESS MODELS

Software Engineering: A Practitioner's Approach, 7th edition
Originated by Roger S. Pressman

Software School, Fudan University
Spring Semester, 2016


《神秘的程序员们》-38
有效流程
http://c.xiqiao.info
C++
Asp
Ruby
Python
Perl
JAVA
C#
Php
.net
0
Boss
立刻调整下这个参数，这对我们很重要。
没问题。只需要改核心程序中的一行代码。
产品总监
填写一个清单，提交给IT部门
1
我正分派这个单号为#129281的需求。但是这需要目前的商务项目做完，并由主任签字。
项目经理
两天后
2
产品总监
#129281目前状态是什么？
它开发人员任务队列中的紧急任务，但前面还有14个紧急的Bug。
项目经理
别管什么队列了!!!立刻给我发给程序员!
一小时后
3
写完了，这不瞬间的事嘛!
程序员
4
违反了公司的政策，你必须在参数文件中做个记录。此外，还有2个旧的调试命令、1个无指定的变量警告必须在移交前进行修复。
代码QA
!~@#$%^&*
程序员
一天后
5
我看到Marge中有#129281任务，但我没有收到测试计划。
IT测试
6
按照老方式运行就行。
程序员
这就是你的测试计划？这会影响到企业的其他方面，我必须有用户选择的测试用例、预期结果、测试运行记录以及用户签收。
IT测试
两天后
7
你们怎么回事！快告诉IT测试，立刻把改好程序给我提交了。
Boss
8
所谓流程
就是这样结束的……
本次任务总结：
总时间：6天
代码更改行数：1行
代码更改字节数：1字节
西乔
设计师，项目经理。
2006年起携创业团队从事Web技术外包开发及产品咨询顾问。
如果你有什么好玩的关于程序员的故事、对话、代码，愿意通过漫画的形式分享，请给西乔发邮件: arthur369@gmail.com


《神秘的程序员们》
敏捷开发
http://c.xiqiao.info
C++
Asp
Python
Ruby
Perl
JAVA
C#
Php
.net
0
一个开发团队……
1
~~更高级的经理~~
~~经理~~
~~文档~~
~~更更高级的经理~~
~~项目管理~~
再也不能这样了，我们要敏捷！
~~高级经理~~
~~HR~~
~~测试~~
~~产品~~
~~需求调研~~
~~行政~~
2
核心开发
只剩两个人
效率高，迭代快，流程简，一切都太爽了。
我们要一直保持这样高效率的开发。
那些琐碎的体力工作招两个初级工程师来干吧。
3
初级工程师
4
人多还是需要详细的文档啊，可是写文档太痛苦了。
文档
唉，人一多代码质量就不稳定了。
5
测试组
6
这么多开发相关人员，恐怕我们需要专人来协调。
项管
需要参加会议的人越来越多了。
大家都需要招聘助手
7
公司大了，急需增加管理层
行政
HR
经理
高级经理
真是令人不爽的熟悉情景啊。。。
作者:
西乔
设计师，项目经理。06年起携创业团队从事Web技术外包开发及产品咨询顾问。
如果你有什么好玩的关于程序员的故事、对话、代码，愿意通过漫画的形式分享，
请给西乔发邮件: arthur369@gmail.com

How to Ensure Quality

- Test and Fix
- Check/Inspection and Rework/Reject
- Production Process
  eliminate problems from the beginning

Process is especially important to software development due to the lack of dependable test/inspection measures

THE PRIMARY GOAL OF ANY PROCESS: *HIGH QUALITY*

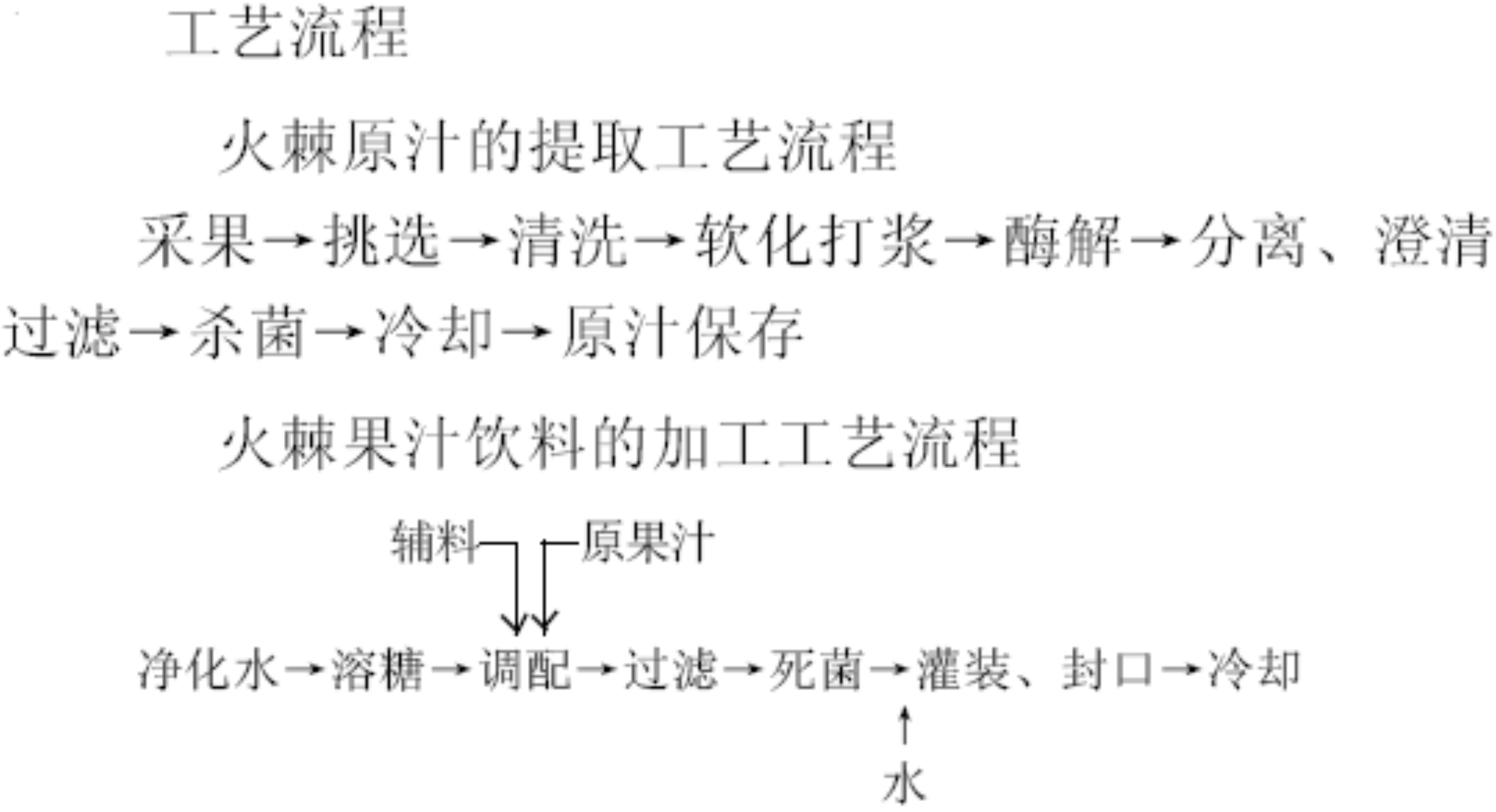
Remember:

High quality ⇒ project timeliness

Why?

Less rework!

AN EXAMPLE PROCESS OF DRINK PRODUCTION -1



工艺流程

火棘原汁的提取工艺流程

采果→挑选→清洗→软化打浆→酶解→分离、澄清过滤→杀菌→冷却→原汁保存

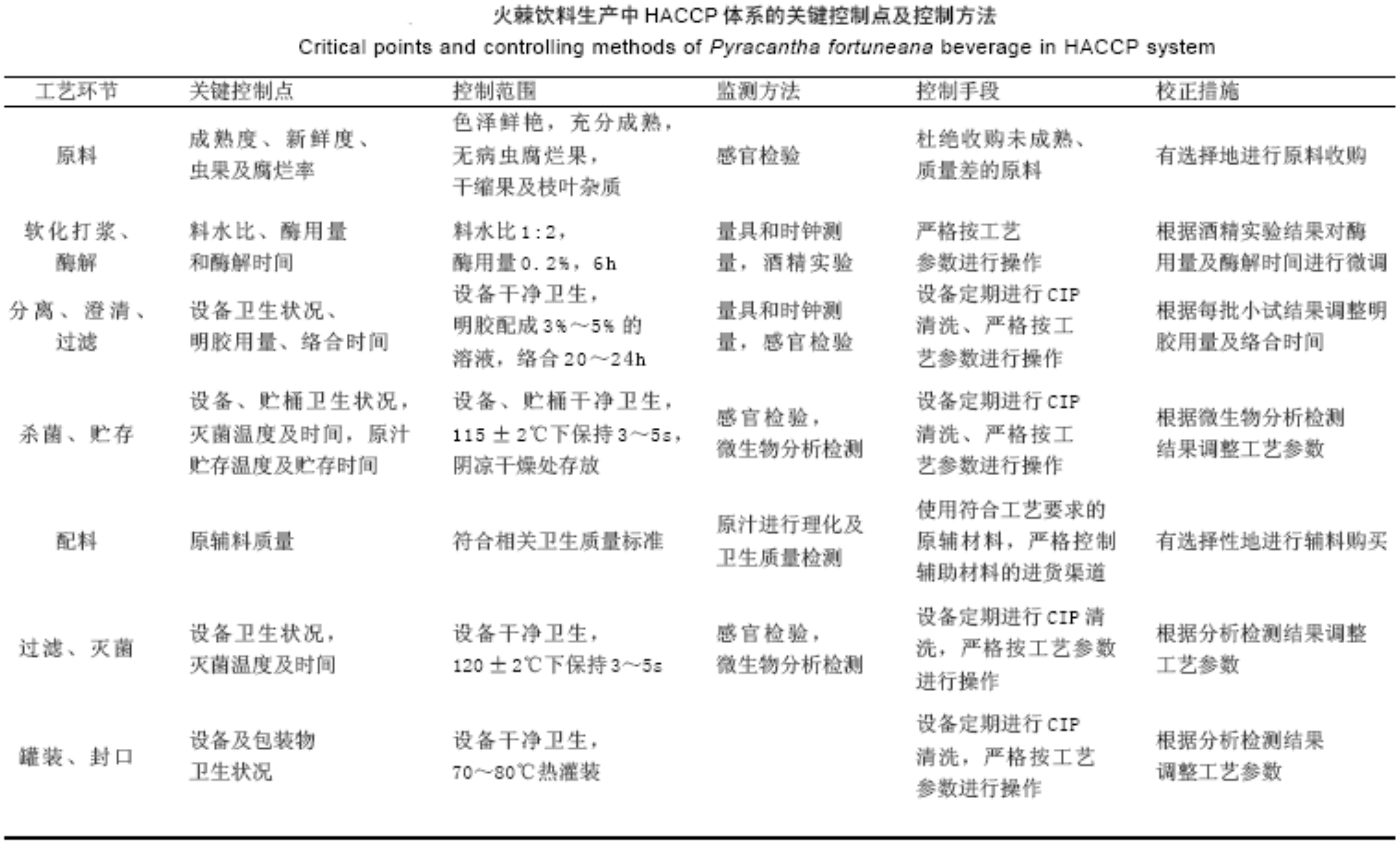
火棘果汁饮料的加工工艺流程

辅料 原果汁

净化水→溶糖→调配→过滤→死菌→灌装、封口→冷却

水

An Example Process of Drink Production -2



火棘饮料生产中 HACCP 体系的关键控制点及控制方法

Critical points and controlling methods of *Pyracantha fortuneana* beverage in HACCP system

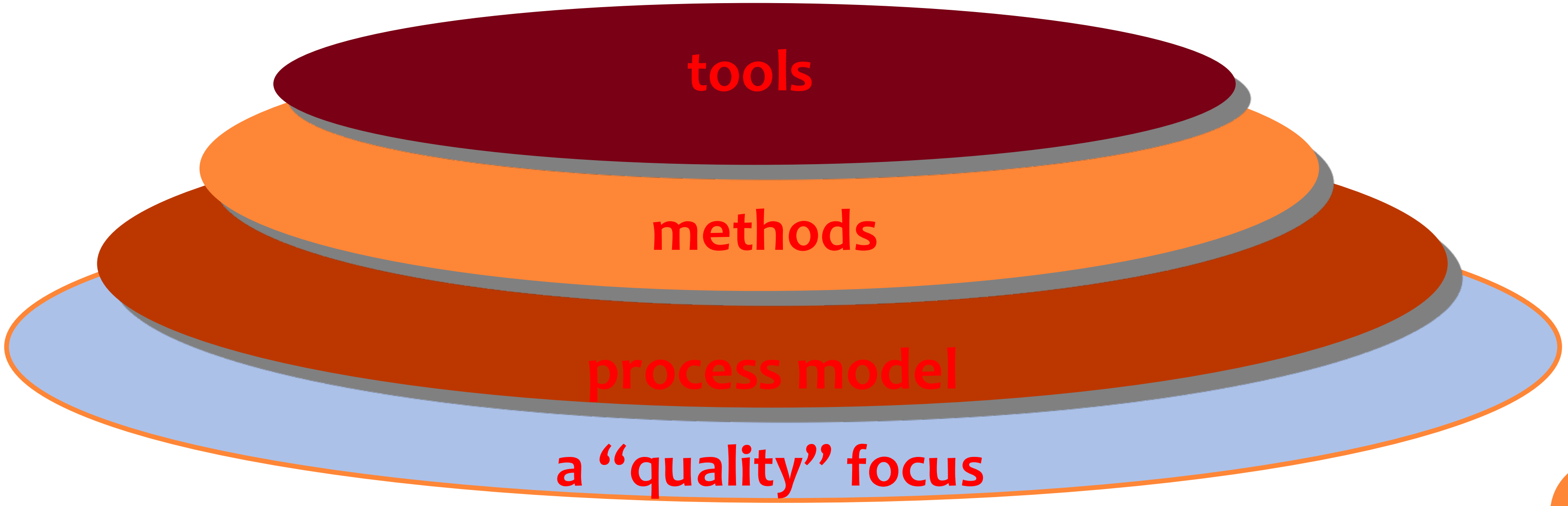
工艺环节	关键控制点	控制范围	监测方法	控制手段	校正措施
原料	成熟度、新鲜度、虫果及腐烂率	色泽鲜艳，充分成熟，无病虫腐烂果，干缩果及枝叶杂质	感官检验	杜绝收购未成熟、质量差的原料	有选择地进行原料收购
软化打浆、酶解	料水比、酶用量和酶解时间	料水比 1:2，酶用量 0.2%，6h	量具和时钟测量，酒精实验	严格按工艺参数进行操作	根据酒精实验结果对酶用量及酶解时间进行微调
分离、澄清、过滤	设备卫生状况、明胶用量、络合时间	设备干净卫生，明胶配成 3%～5% 的溶液，络合 20～24h	量具和时钟测量，感官检验	设备定期进行 CIP 清洗、严格按工艺参数进行操作	根据每批小试结果调整明胶用量及络合时间
杀菌、贮存	设备、贮桶卫生状况，灭菌温度及时间，原汁贮存温度及贮存时间	设备、贮桶干净卫生，115 ± 2℃下保持 3～5s，阴凉干燥处存放	感官检验，微生物分析检测	设备定期进行 CIP 清洗、严格按工艺参数进行操作	根据微生物分析检测结果调整工艺参数
配料	原辅料质量	符合相关卫生质量标准	原汁进行理化及卫生质量检测	使用符合工艺要求的原辅材料，严格控制辅助材料的进货渠道	有选择性地进行辅料购买
过滤、灭菌	设备卫生状况，灭菌温度及时间	设备干净卫生，120 ± 2℃下保持 3～5s	感官检验，微生物分析检测	设备定期进行 CIP 清洗，严格按工艺参数进行操作	根据分析检测结果调整工艺参数
罐装、封口	设备及包装物卫生状况	设备干净卫生，70～80℃热灌装		设备定期进行 CIP 清洗，严格按工艺参数进行操作	根据分析检测结果调整工艺参数

HACCP(hazard analysis and critical control point)

A LAYERED TECHNOLOGY

Software Engineering

The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.

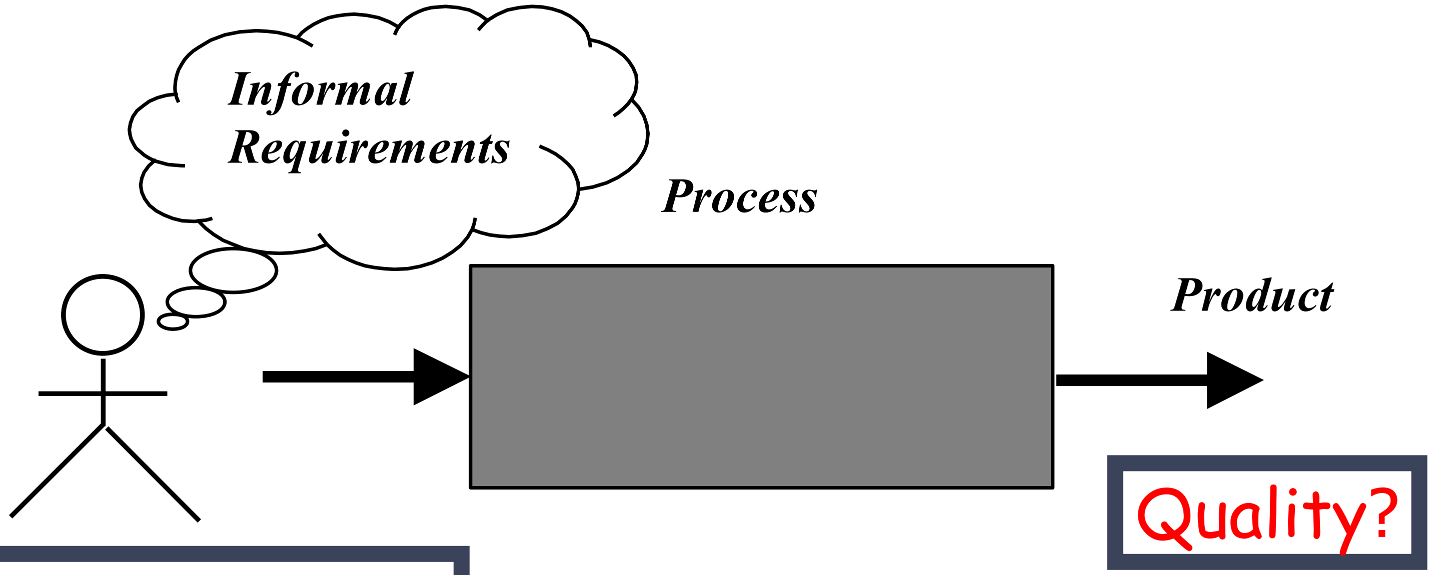


[illegible]

Chapter Overview

- What? A software process - a series of predictable steps that leads to a timely, high-quality product.
- Who? Managers, software engineers, and customers.
- Why? Provides stability, control, and organization to an otherwise chaotic activity.
- Steps? A handful of activities are common to all software processes, details vary.
- Work product? Programs, documents, and data.
- Correct process? Assessment, quality deliverable.

PROCESS AS A "BLACK BOX"



Quality?

Uncertain / Incomplete requirement In the beginning

CODE & FIX

The earliest approach

- Write code
- Fix it to eliminate any errors that have been detected, to enhance existing functionality, or to add new features
- Source of difficulties and deficiencies
  - impossible to predict
  - impossible to manage
  - late feedback of problems, high cost of reworking

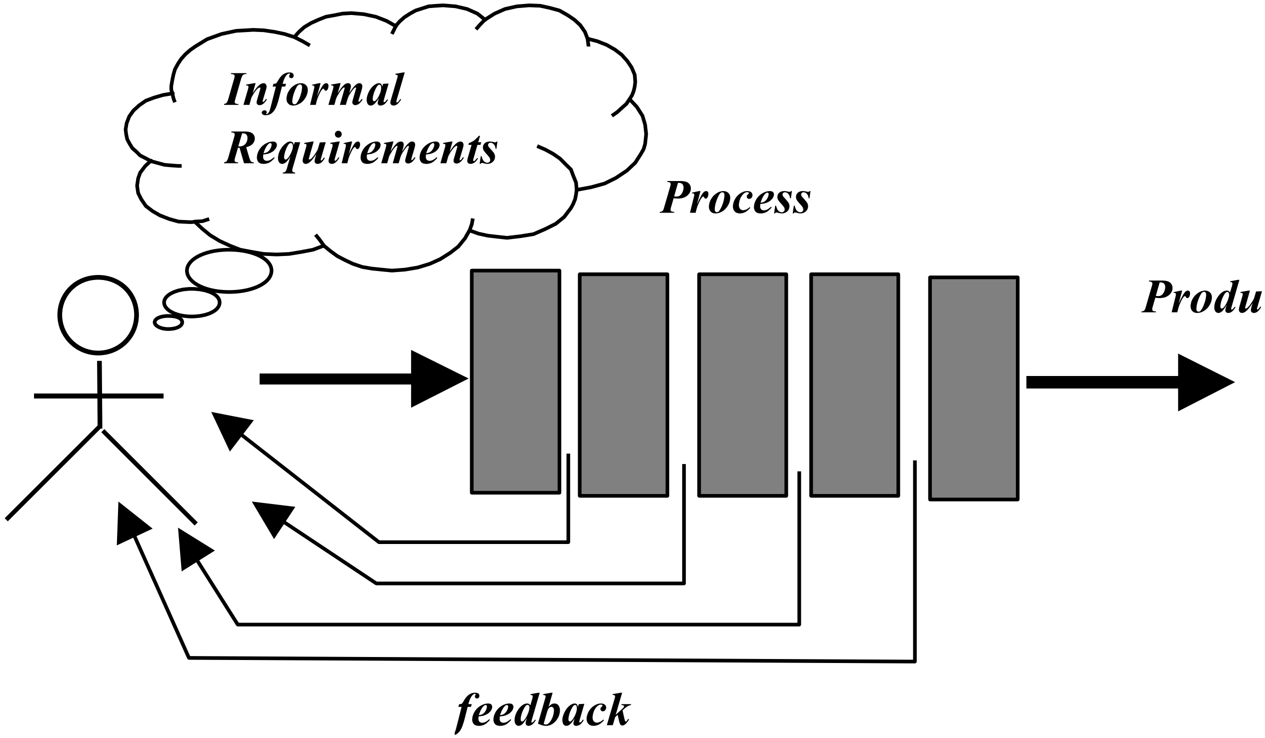
CODE & FIX: PROBLEMS

- The assumption is that requirements can be fully understood prior to development
- Interaction with the customer occurs only at the beginning (requirements) and end (after delivery)
- Unfortunately the assumption almost never holds

PROCESSES ARE NEEDED

- Symptoms of inadequacy: the software crisis
  - scheduled time and cost exceeded
  - user expectations not met
  - poor quality
- The size and economic value of software applications required appropriate "process models"

Process As a "White Box"



Software Process: Advantages

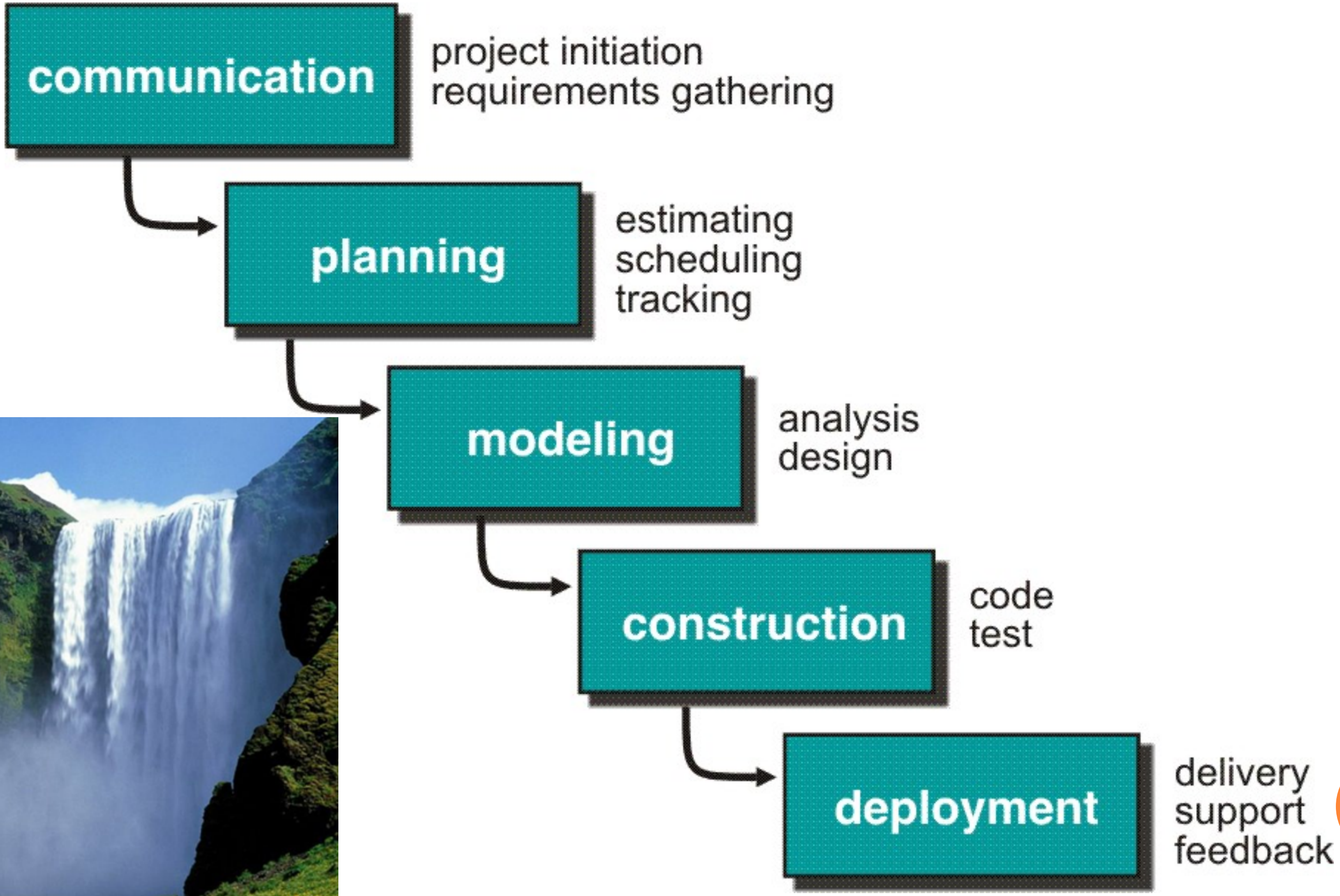
- Reduce risks by improving visibility
- Allow project changes as the project progresses based on feedback from the customer
- Main Activities of Software Production
  - The main activities of software production must be performed independently of the model
  - The model simply affects the flow among activities

SOFTWARE PROCESS MODEL

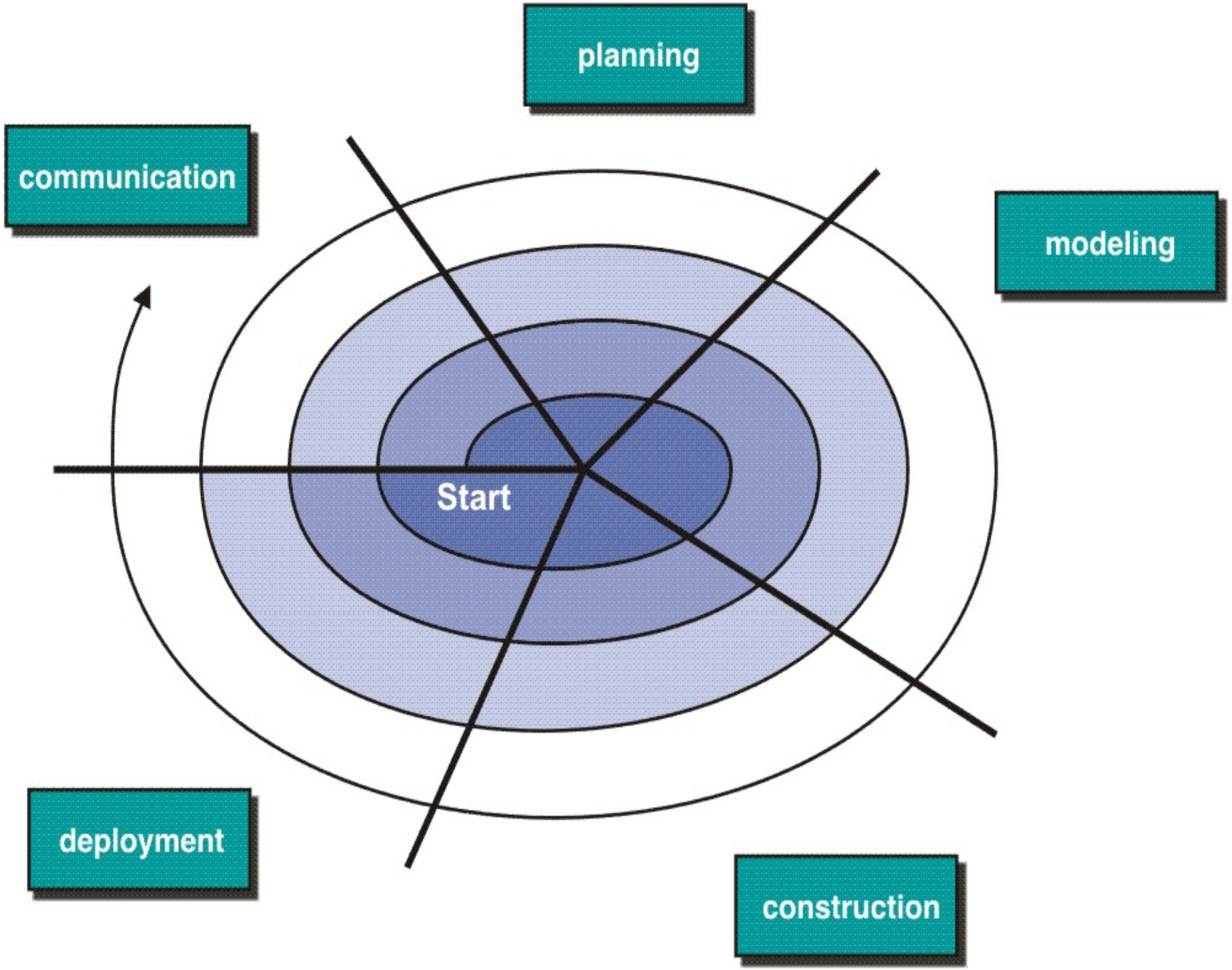
- Attempt to organize the software life cycle by
  - defining activities involved in software production
  - order of activities and their relationships
- Goals of a software process
  - standardization
  - predictability
  - productivity
  - high product quality
  - ability to plan time and budget requirements
  - ……

to manage, to predict

THE WATERFALL MODEL



EVOLUTIONARY MODELS: THE SPIRAL



Process Model Goals (B. Boehm 1988)

determine the order of stages involved in software development and evolution,

and to establish the transition criteria for progressing from one stage to the next. These include:

completion criteria for the current stage plus

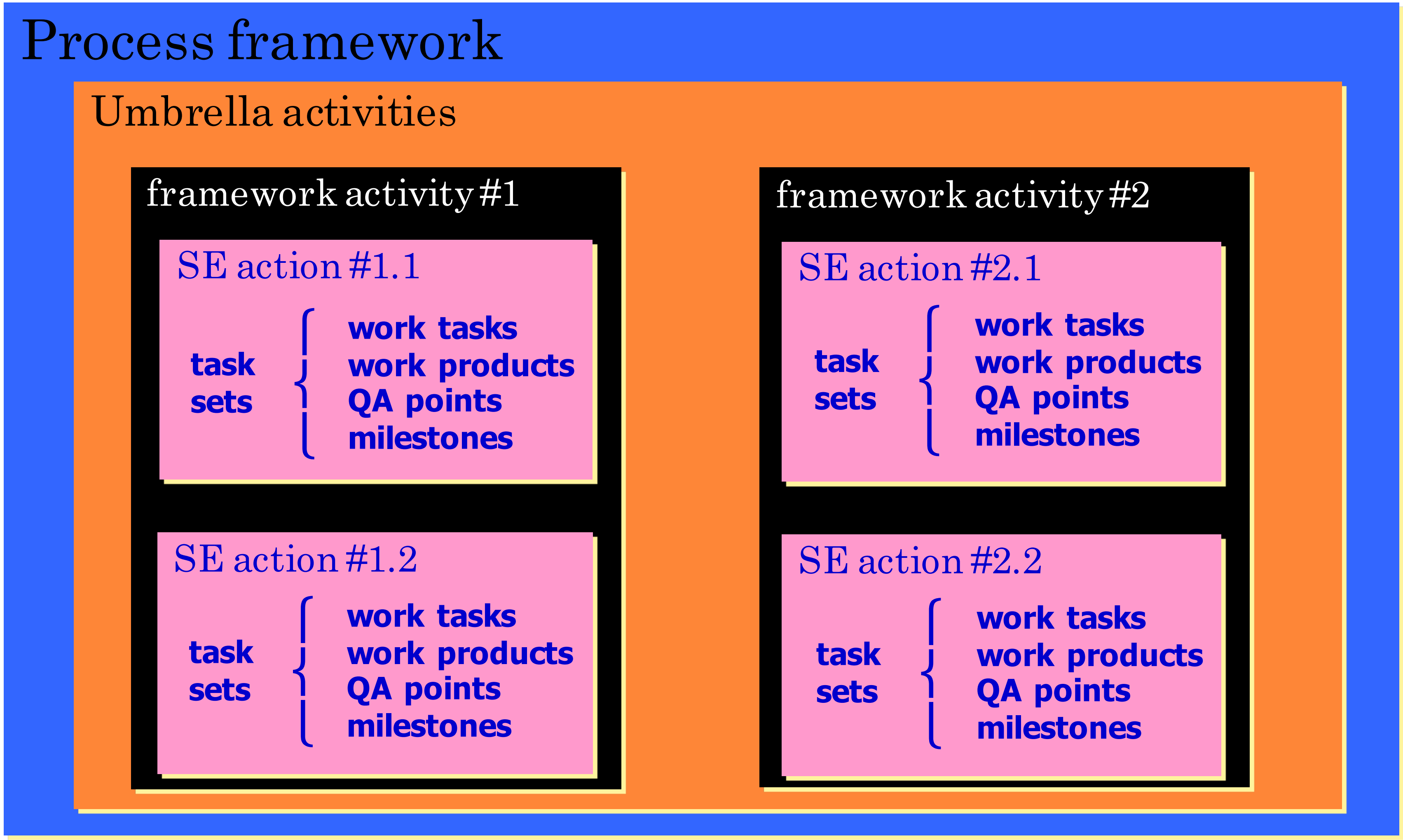
choice criteria and entrance criteria for the next stage.

Thus a process model addresses the following software project questions:

- What shall we do next?
- How long shall we continue to do it?

A Process Framework

Software process



Process framework

Umbrella activities

framework activity #1

SE action #1.1

task sets

- work tasks
- work products
- QA points
- milestones

SE action #1.2

task sets

- work tasks
- work products
- QA points
- milestones

framework activity #2

SE action #2.1

task sets

- work tasks
- work products
- QA points
- milestones

SE action #2.2

task sets

- work tasks
- work products
- QA points
- milestones

Framework Activities

- Communication
- Planning
- Modeling
  - Action 1: Analysis of requirements
  - Action 2: Design
- Construction
  - Code generation
  - Testing
- Deployment

UMBRELLA ACTIVITIES

- Software project management
- Formal technical reviews
- Software quality assurance
- Software configuration management
- Work product preparation and production
- Reusability management
- Measurement
- Risk management

The Process Model: Adaptability

- The framework activities will always be applied on every project …

BUT…

- The tasks (and degree of rigor) for each activity will vary based on:
  - the type of project
  - characteristics of the project
  - common sense judgment
  - concurrence of the project team
  - ……

Example: the Communication Activity in a Small Project

- Project Situation
  - requested by one person at a remote location
  - simple, straightforward requirements
- Only one action: *phone conversation*
  - Task 1: make contact with the customer via telephone
  - Task 2: discuss requirements and take notes
  - Task 3: organize notes into a brief written statement requirements
  - Task 4: e-mail to the customer for review and approval

Example: the Communication Activity in a Complex Project

- Project Situation
  - requested by many stakeholders
  - each has a different set of (sometimes conflicting) requirements
- Six distinct actions
  - inception
  - elicitation
  - elaboration
  - negotiation
  - specification
  - validation

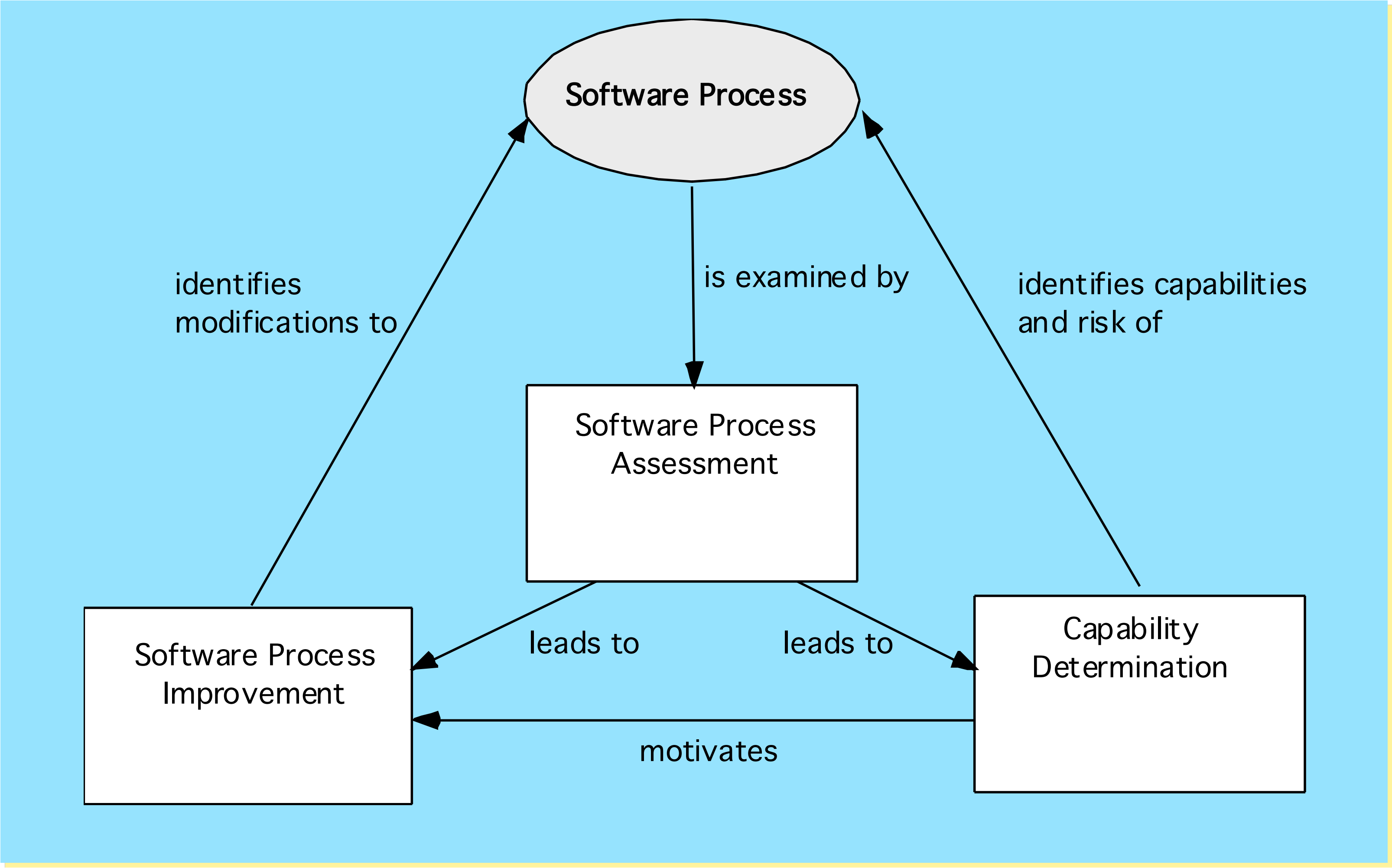
THE CMMI

- CMMI: Capability Maturity Model Integration
- A process improvement approach
  - provides organizations with the essential elements of effective processes
  - ultimately improve their performance
  - can be used to guide process improvement across a project, a division, or an entire organization
- Software Engineering Institute (SEI, 2008)
  - CMMI helps "integrate traditionally separate organizational functions, set process improvement goals and priorities, provide guidance for quality processes, and provide a point of reference for appraising current processes."

PROCESS ASSESSMENT

- The process should be assessed to ensure that it meets a set of basic process criteria that have been shown to be essential for a successful software engineering.
- Many different assessment options are available:
  - SCAMPI: Standard CMMI Assessment Method for Process Improvement
  - CBA IPI: CMM-Based Appraisal for Internal Process Improvement
  - SPICE: ISO/IEC 15504
  - ISO 9001:2000

Assessment and Improvement


Software Process
is examined by
identifies modifications to
identifies capabilities and risk of
Software Process Assessment
leads to
leads to
Software Process Improvement
Capability Determination
motivates

CMMI KPA

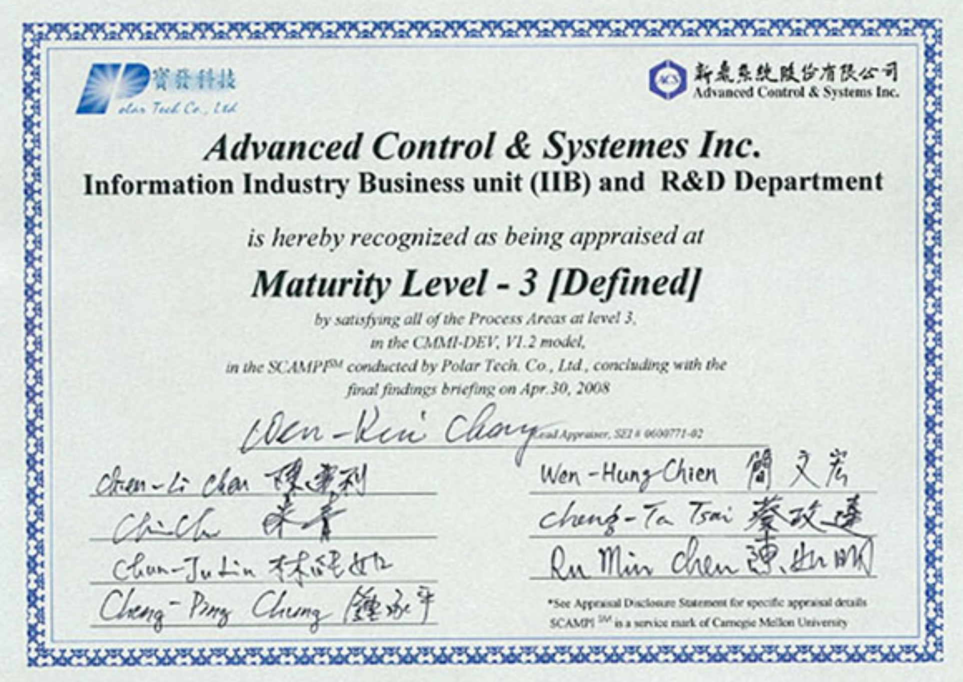
- Key Process Area (KPA): a KPA identifies a cluster of related activities that, when performed collectively, achieve a set of goals considered important
- The CMMI defines each process area in terms of “specific goals” and the “specific practices” required to achieve these goals
  - *Specific goals (SG)* establish the characteristics that must exist if the activities implied by a process area are to be effective
  - *Specific practices (SP)* refine a goal into a set of process-related activities

KPA: An Example

- Project Planning-one of the eight KPAs defined for the "project management" category in CMMI
  - SG 1 Establish estimates
    - SP 1.1-1 Estimate the scope of the project
    - SP 1.2-1 Establish estimates of work product and task attributes
    - SP 1.3-1 Define project life cycle
    - SP 1.4-1 Determine estimates of effort and cost
  - SG 2 Develop a Project Plan
    - SP 2.1-1 Establish the budget and schedule
    - SP 2.2-1 Identify project risks
    - SP 2.3-1 Plan for data management
    - ……
  - …..

THE CMMI (STAGED MODEL)

Maturity Level	Focus	Process Areas	Quality
5 Optimizing	Continuous Process Improvement	Organizational Innovation and Deployment Causal Analysis and Resolution	
4 Quantitatively Managed	Quantitative Management	Organizational Process Performance Quantitative Project Management	
3 Defined		Requirements Development Technical Solution Product Integration Verification Validation Organizational Process Focus	
2 Managed			
1 Initial			Risk & Rework



寶發科技 Polar Tech Co., Ltd

新鼎系統股份有限公司 Advanced Control & Systems Inc.

Advanced Control & Systemes Inc.

Information Industry Business unit (IIB) and R&D Department

is hereby recognized as being appraised at

Maturity Level - 3 [Defined]

by satisfying all of the Process Areas at level 3,
in the CMMI-DEV, V1.2 model,
in the SCAMPI℠ conducted by Polar Tech. Co., Ltd., concluding with the
final findings briefing on Apr.30, 2008

Wen-Kui Chang *Lead Appraiser, SEI # 0600771-02*

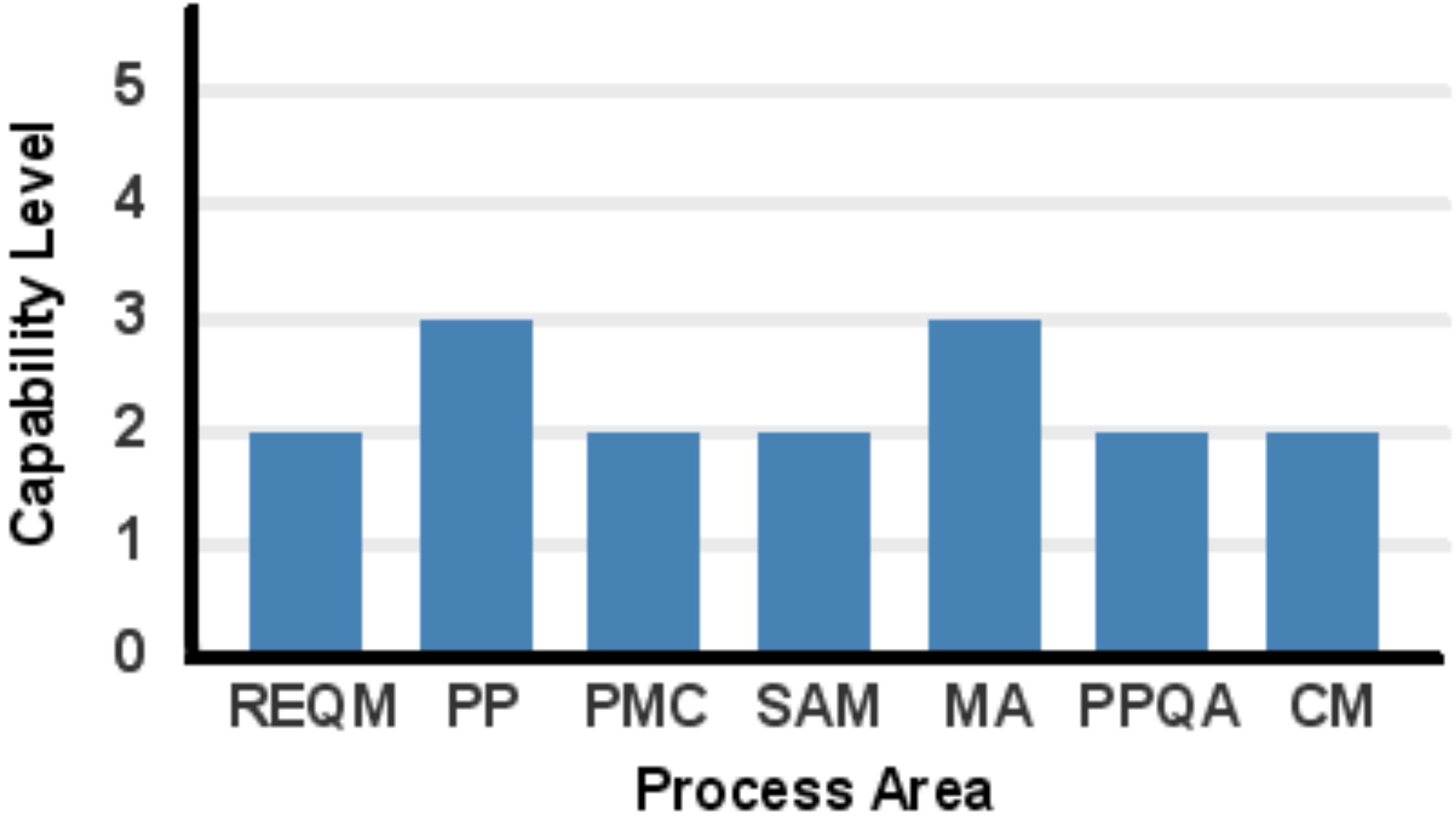
Chen-Li Chan — Wen-Hung Chien

Chun-Ju Lin — Cheng-Ta Tsai

Cheng-Ping Chung — Ru Min Chen

*See Appraisal Disclosure Statement for specific appraisal details
SCAMPI ℠ is a service mark of Carnegie Mellon University

THE CMMI (CONTINUOUS MODEL)



Personal Software Process (PSP)

- Recommends five framework activities:
  - Planning
  - High-level design
  - High-level design review
  - Development
  - Postmortem
- Stresses the need for each software engineer to identify errors early and as important, to understand the types of errors

TEAM SOFTWARE PROCESS (TSP)

- Each project is "launched" using a "script" that defines the tasks to be accomplished
- Teams (of 2 to 20 engineers) are self-directed:
  - Plan and track work, set goals, own processes and plans
- Measurement is encouraged
- Measures are analyzed with the intent of improving the team process (through coaching, motivation, …)

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
- Typical examples:
  - Customer communication (a process activity)
  - Analysis (an action)
  - Requirements gathering (a process task)
  - Reviewing a work product (a process task)
  - Design model (a work product)

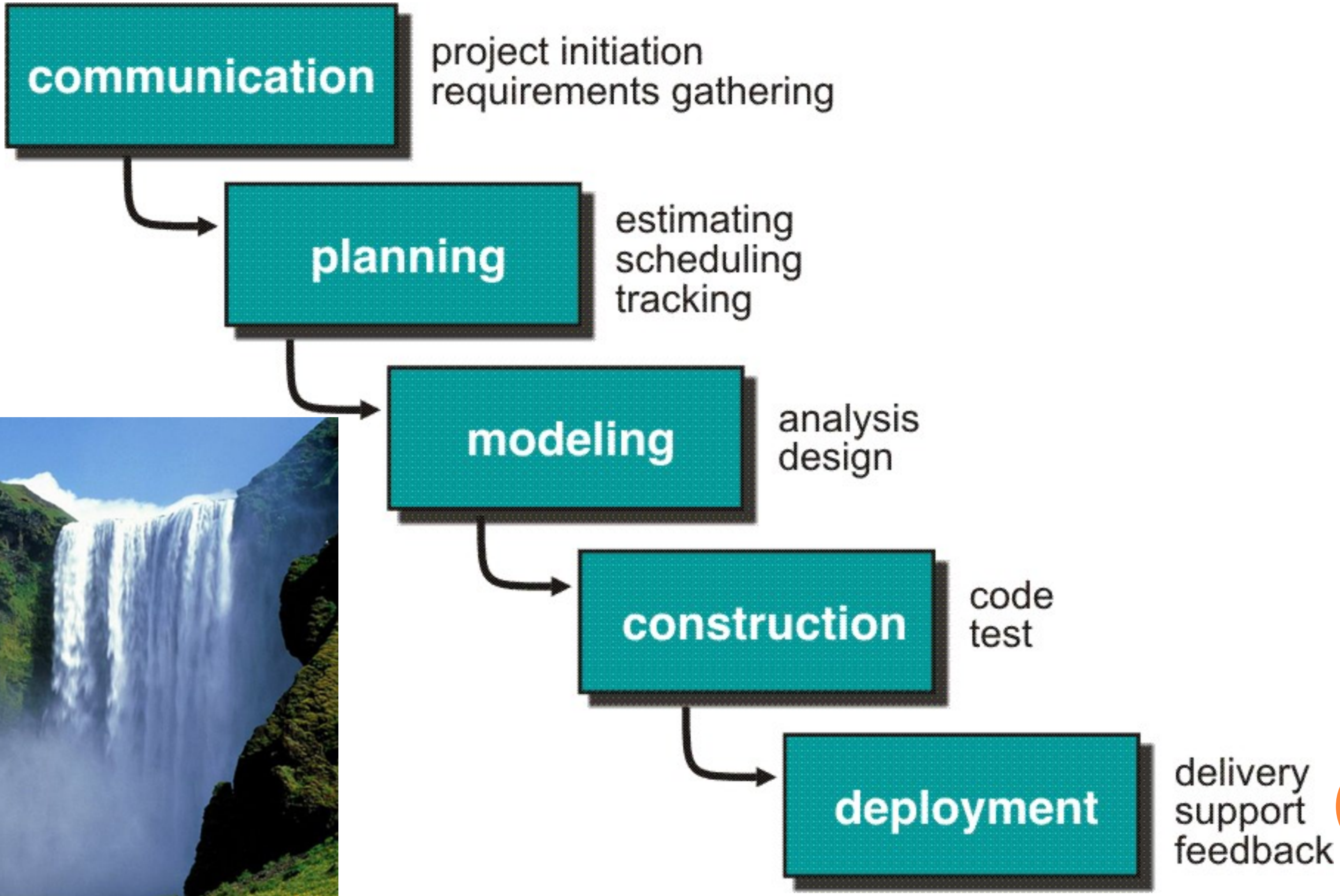
PRESCRIPTIVE MODELS

Prescriptive process models advocate an orderly approach to software engineering

That leads to a few questions …

- If prescriptive process models strive for structure and order, are they inappropriate for a software world that thrives on change?
- Yet, if we reject traditional process models (and the order they imply) and replace them with something less structured, do we make it impossible to achieve coordination and coherence in software work?

The Waterfall Model



communication

project initiation
requirements gathering

planning

estimating
scheduling
tracking

modeling

analysis
design

construction

code
test

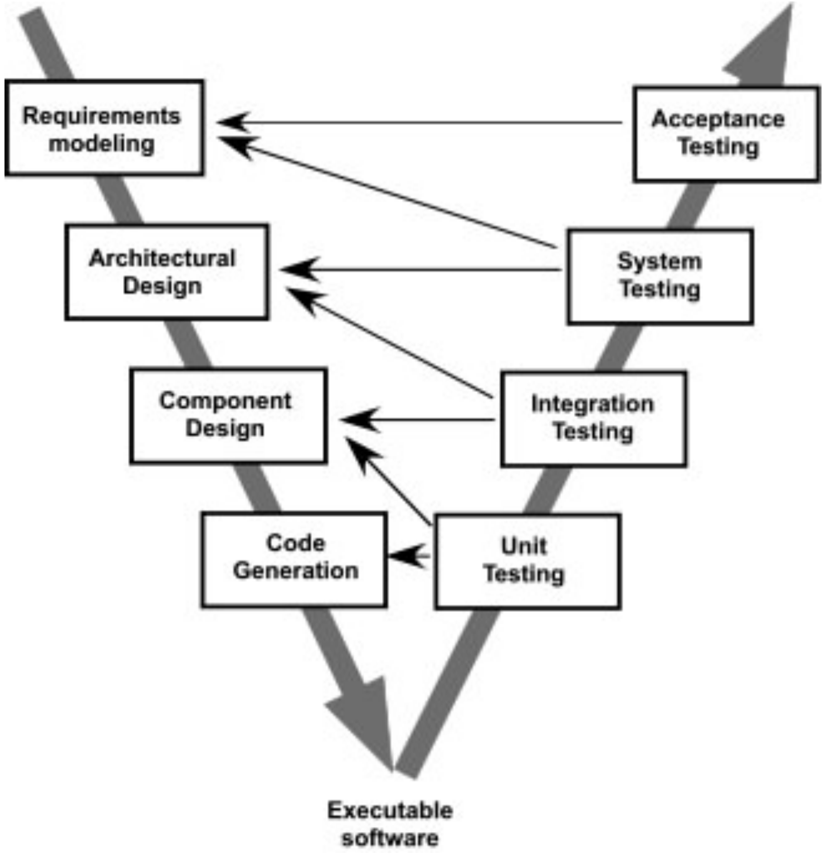
deployment

delivery
support
feedback

Waterfall Model Assumptions

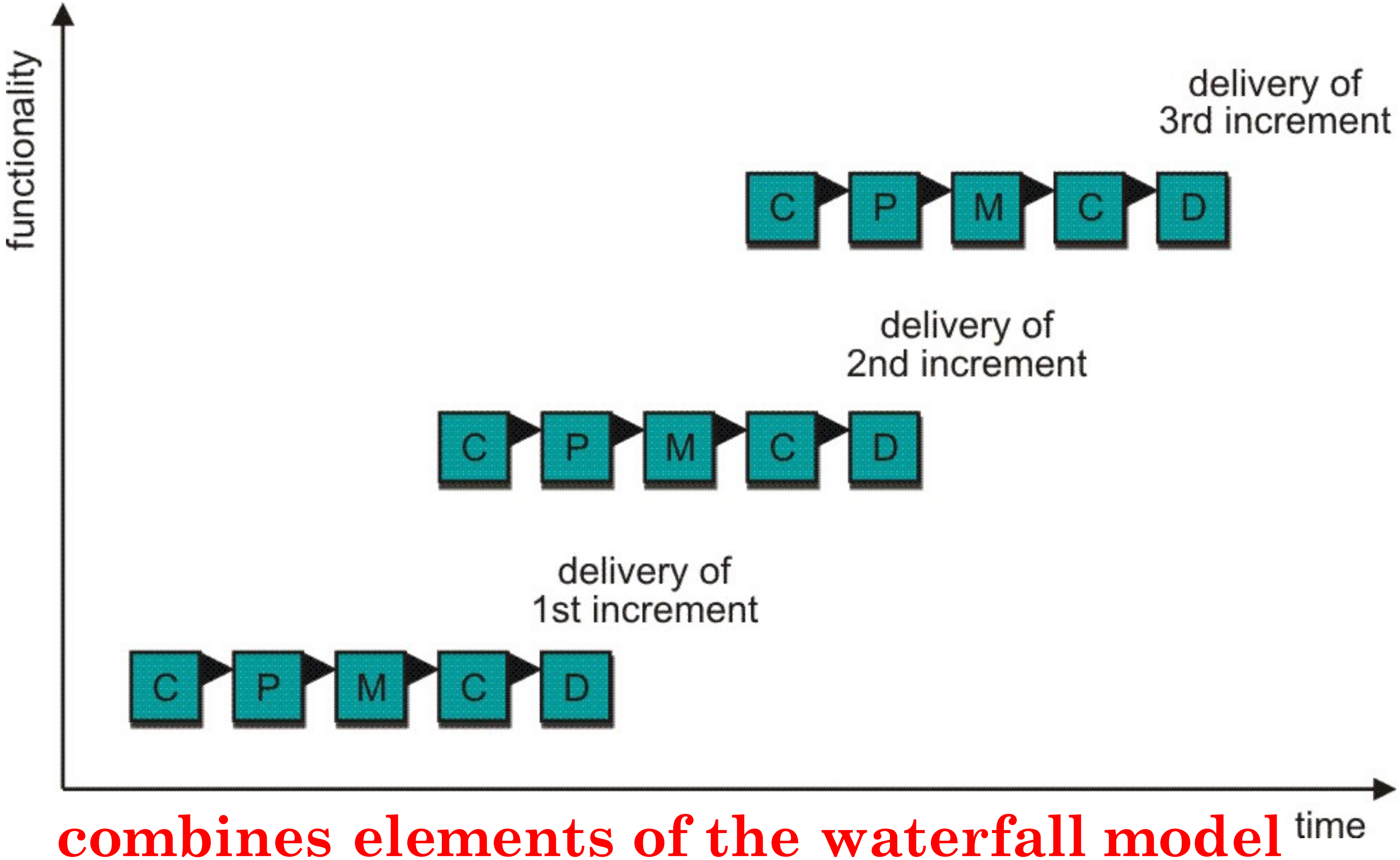
1. The requirements are knowable in advance of implementation.
2. The requirements have no unresolved, high-risk implications
   - e.g., risks due to COTS choices, cost, schedule, performance, safety, security, user interfaces, organizational impacts
3. The nature of the requirements will not change very much
   - During development; during evolution
4. The requirements are compatible with all the key system stakeholders' expectations
   - e.g., users, customer, developers, maintainers, investors
5. The right architecture for implementing the requirements is well understood.
6. There is enough calendar time to proceed sequentially.

The V Model



If we rely on testing alone, defects created first are detected last

Incremental Models: Incremental



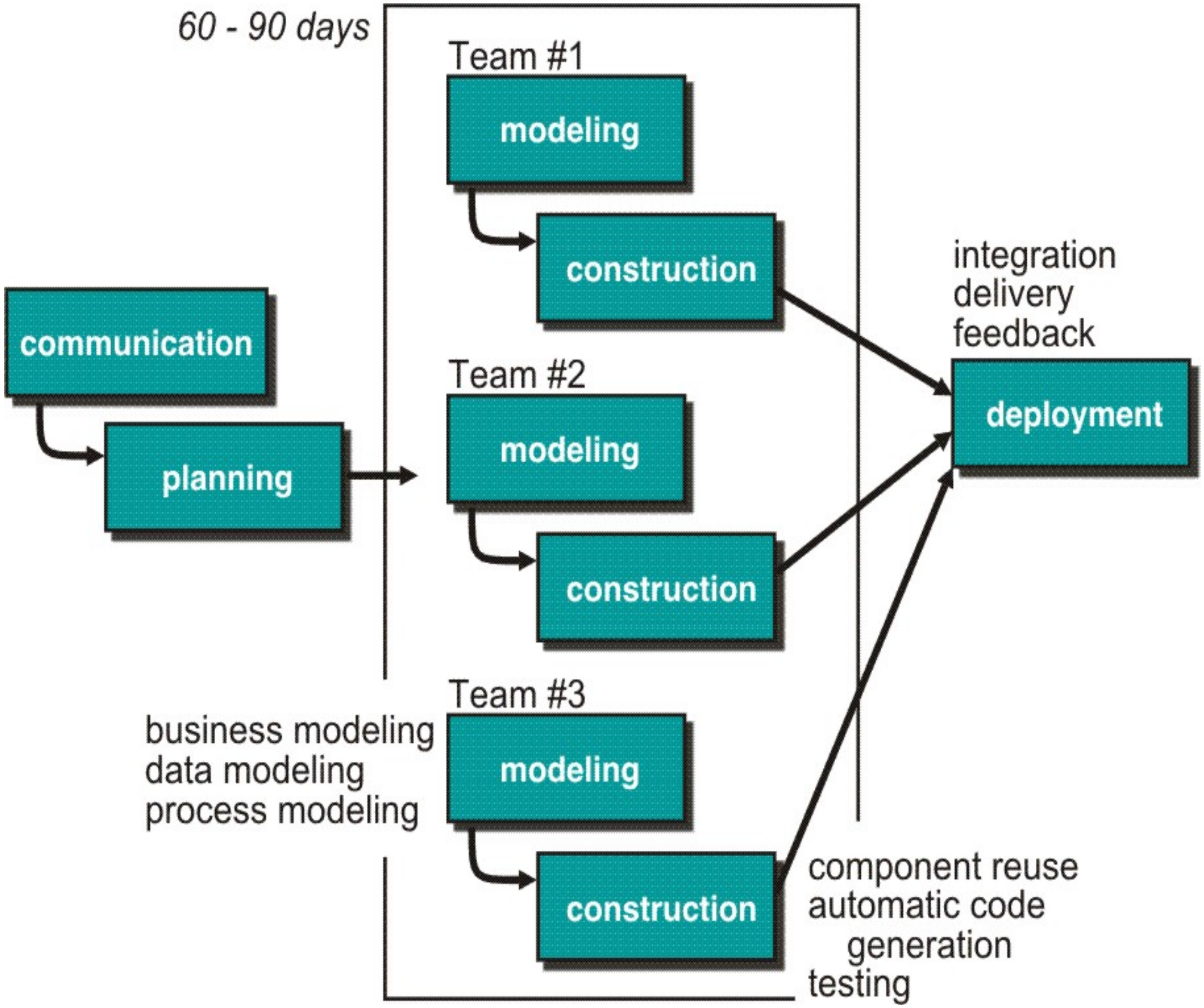
combines elements of the waterfall model applied in an iterative fashion

INCREMENTAL MODEL: AN EXAMPLE OF WORD-PROCESSING SOFTWARE

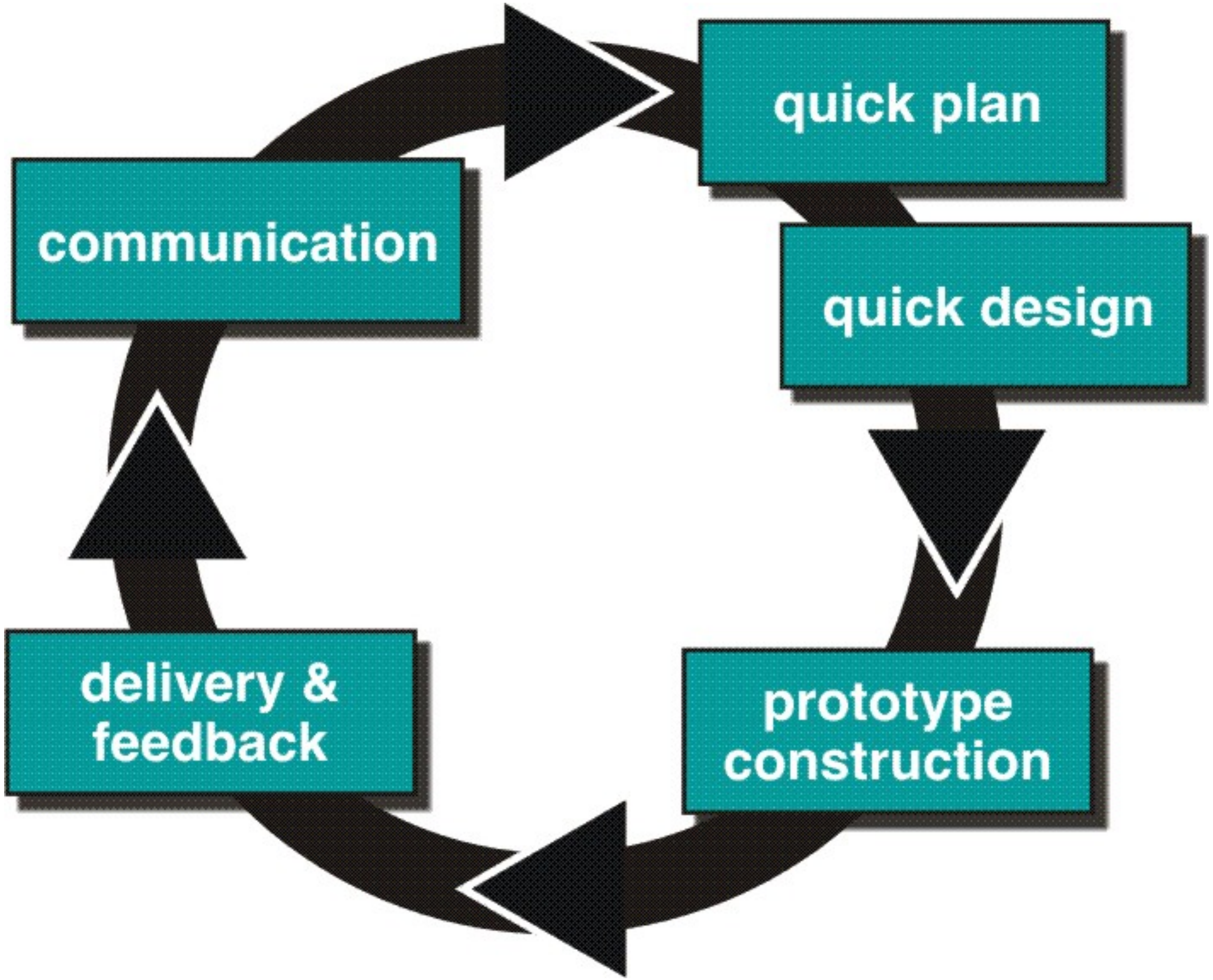
- 1st increment (*core product*)
  basic file management, basic editing (input), and document production functions
- 2nd increment
  sophisticated editing (font, size, copy/paste…)
- 3rd increment
  spelling and grammar checking
- 4th increment
  advanced page layout
- ……

- each increment produces a deliverable product
- well-known core requirements delivered first
- latter increments often depends on former ones

Incremental Models: RAD Model

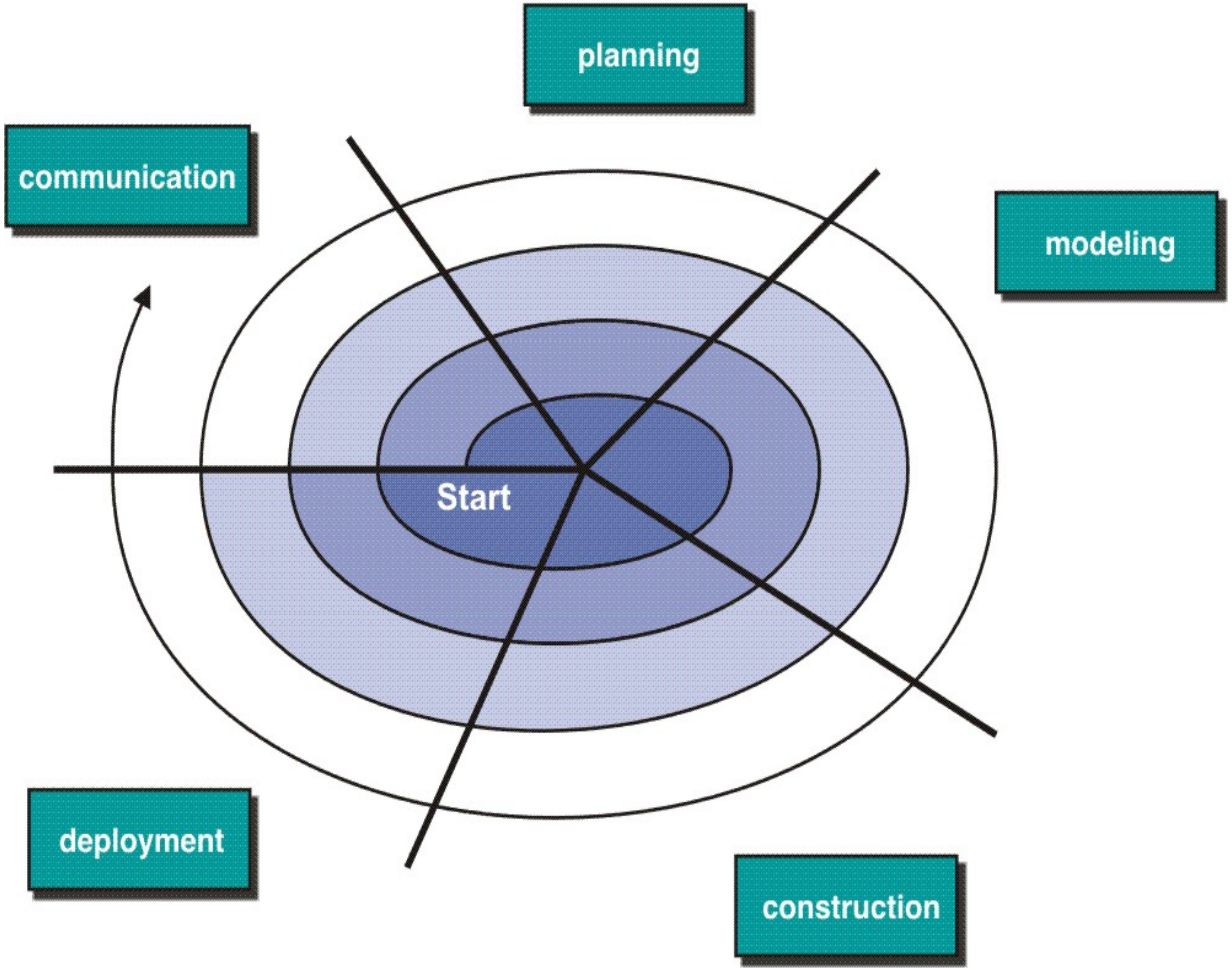


Evolutionary Models: Prototyping



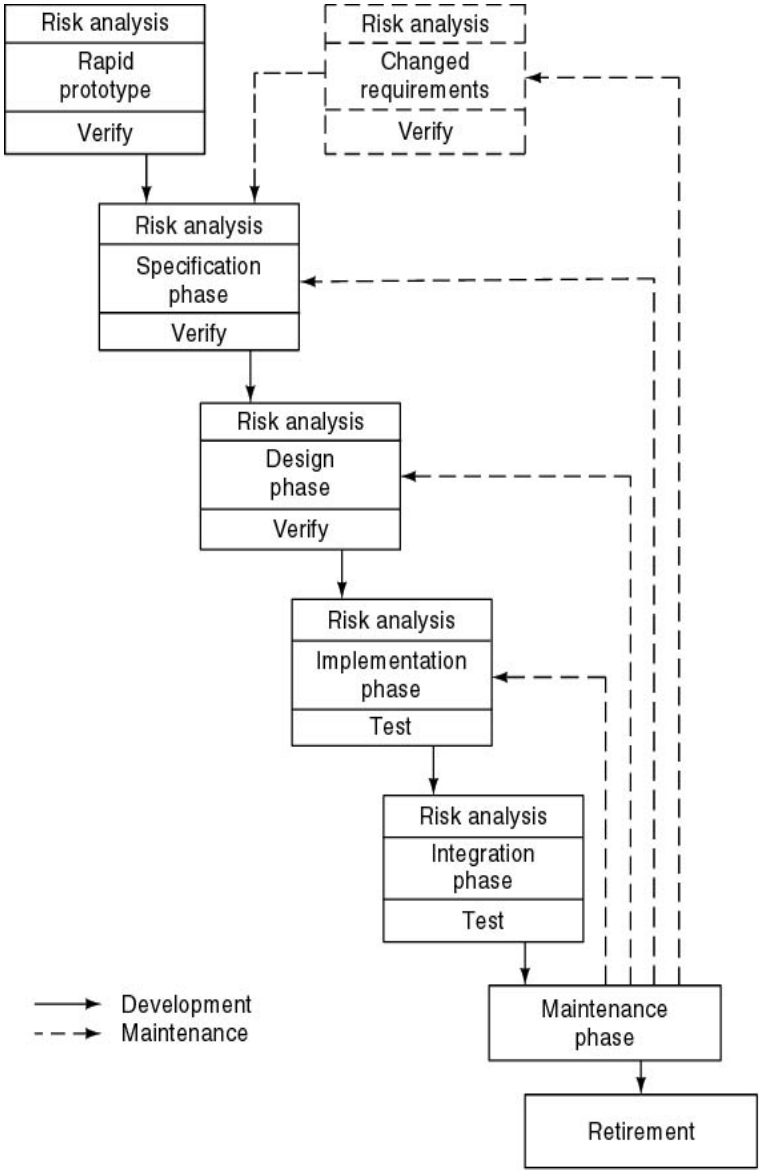
more commonly used as a technique than a standalone process model

EVOLUTIONARY MODELS: THE SPIRAL



SPIRAL MODEL

- Simplified form
  - Waterfall model plus risk analysis
- Precede each phase by
  - Alternatives
  - Risk analysis
- Follow each phase by
  - Evaluation
  - Planning of next phase



Analysis of Spiral Model

- Strengths
  - Iterative refinement to the product
  - No distinction between development and maintenance
  - Realistic approach to large-scale software
- Weaknesses
  - Difficult to convince customers that evolutionary approach is controllable
  - Demands considerable risk assessment expertise
  - Rely on risk assessment for success

OTHER PROCESS MODELS

- Component based development—the process to apply when reuse is a development objective
- Formal methods—emphasizes the mathematical specification of requirements
- AOSD—provides a process and methodological approach for defining, specifying, designing, and constructing *aspects*
- Unified Process—a "use-case driven, architecture-centric, iterative and incremental" software process closely aligned with the Unified Modeling Language (UML)

UNIFIED PROCESS (UP)

A software process that is:

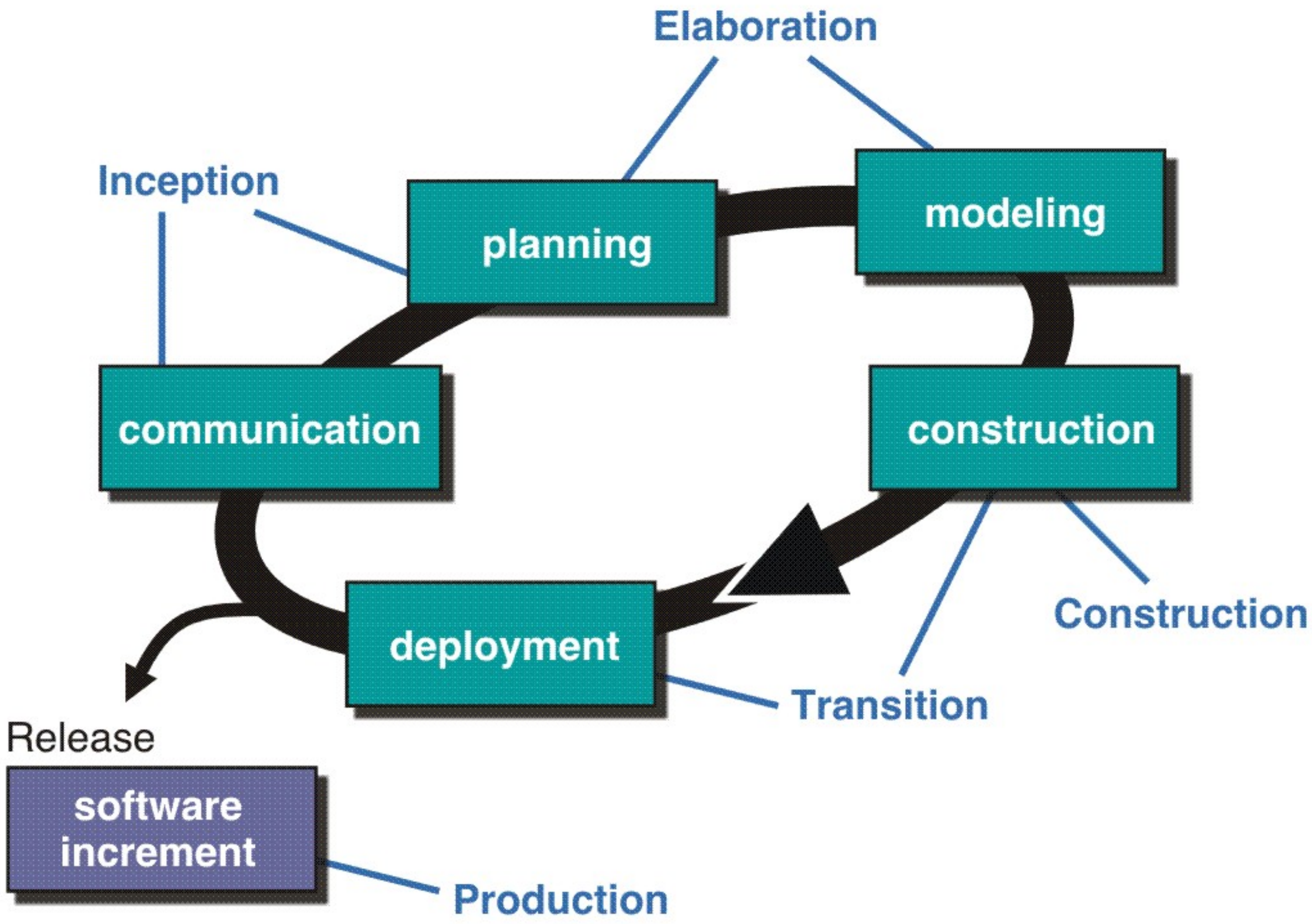
- use-case driven
- architecture-centric
- iterative and incremental

Closely aligned with the
Unified Modeling Language (UML)

UP: A BRIEF HISTORY

- 1980s-1990s: object-oriented methods and languages bloom
- Early 1990s: work on "unified method"
  - James Rumbaugh, Grady Booch, Ivar Jacobson
  - To combine the best features of their individual methods with additional features
  - UML (unified modeling language) proposed and became an industry standard by 1997
  - Rational Corporation developed automated tools for UML
- Over the next few years: UP proposed as a unified OO process framework using UML

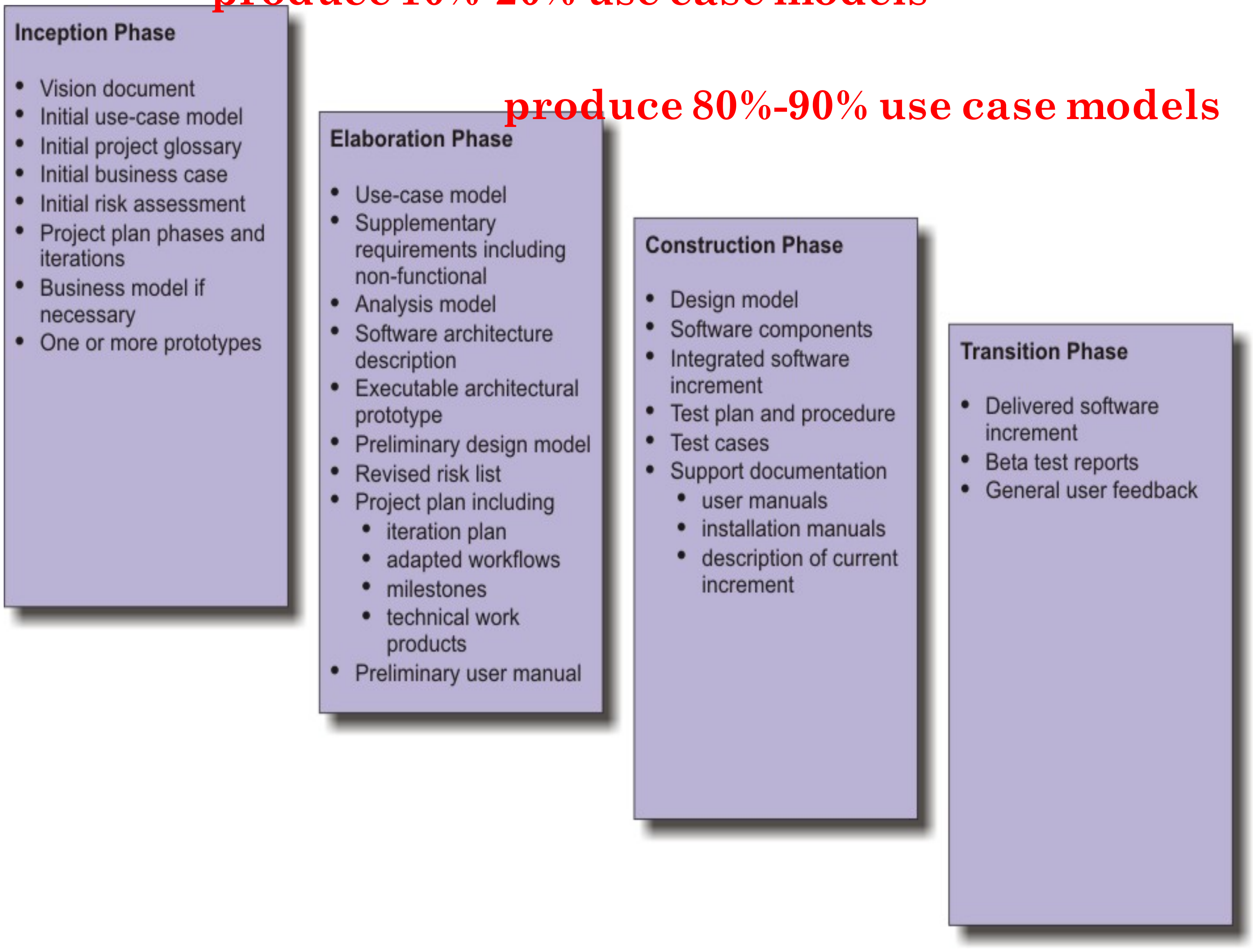
UNIFIED PROCESS MODEL



UP Work Products

produce 10%-20% use case models

produce 80%-90% use case models



Inception Phase

- Vision document
- Initial use-case model
- Initial project glossary
- Initial business case
- Initial risk assessment
- Project plan phases and iterations
- Business model if necessary
- One or more prototypes

Elaboration Phase

- Use-case model
- Supplementary requirements including non-functional
- Analysis model
- Software architecture description
- Executable architectural prototype
- Preliminary design model
- Revised risk list
- Project plan including
  - iteration plan
  - adapted workflows
  - milestones
  - technical work products
- Preliminary user manual

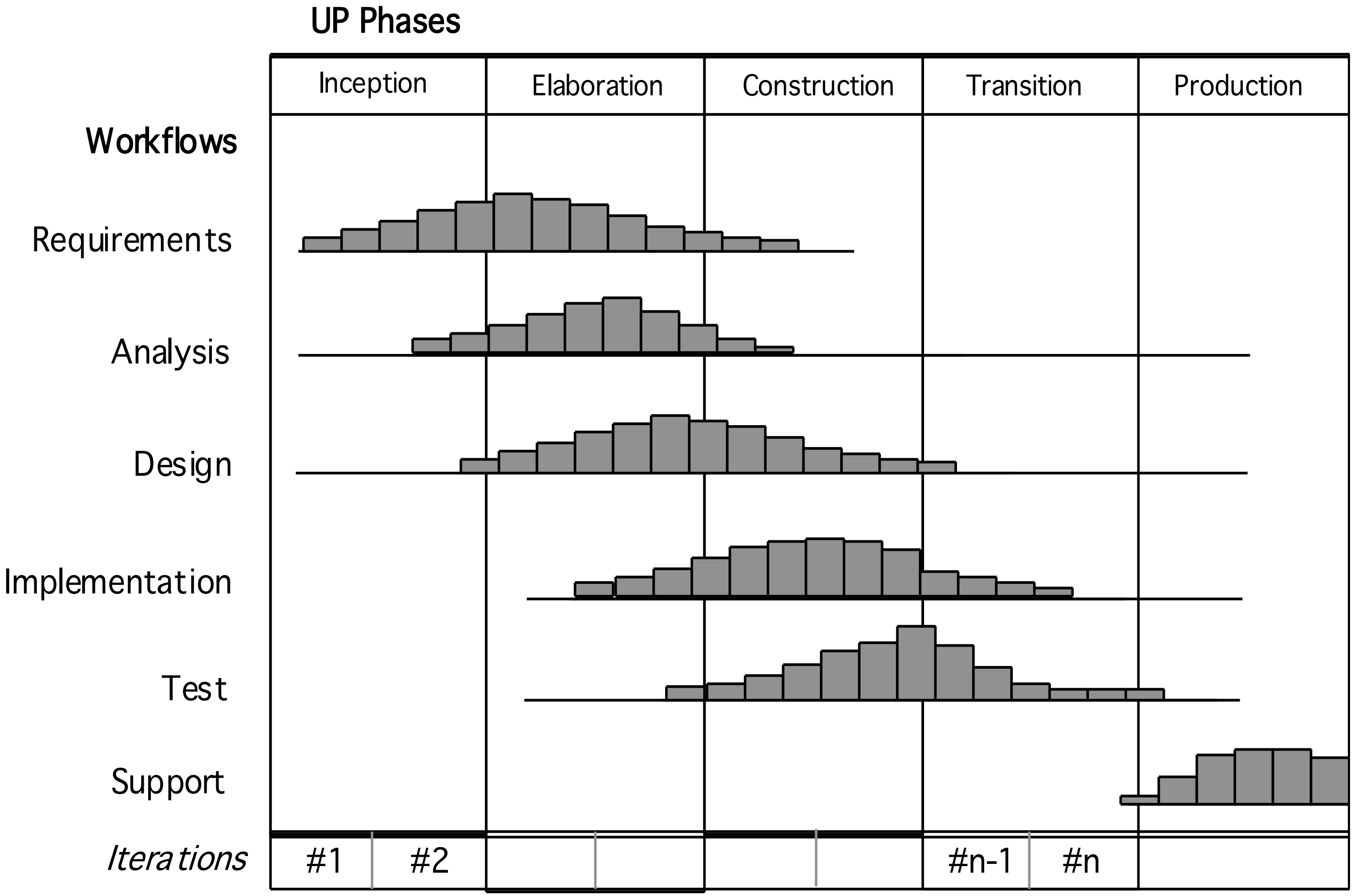
Construction Phase

- Design model
- Software components
- Integrated software increment
- Test plan and procedure
- Test cases
- Support documentation
  - user manuals
  - installation manuals
  - description of current increment

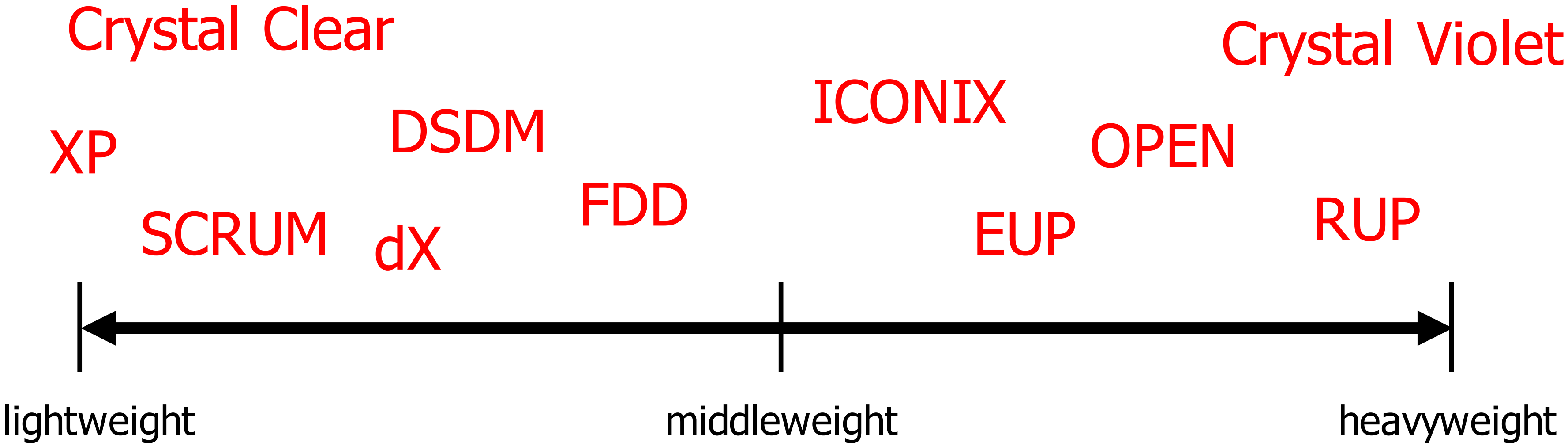
Transition Phase

- Delivered software increment
- Beta test reports
- General user feedback

UP PHASES


UP Phases
Inception
Elaboration
Construction
Transition
Production
Workflows
Requirements
Analysis
Design
Implementation
Test
Support
Iterations
#1
#2
#n-1
#n

Software Process Spectrum



Crystal Clear

XP

DSDM

SCRUM

dX

FDD

ICONIX

EUP

OPEN

Crystal Violet

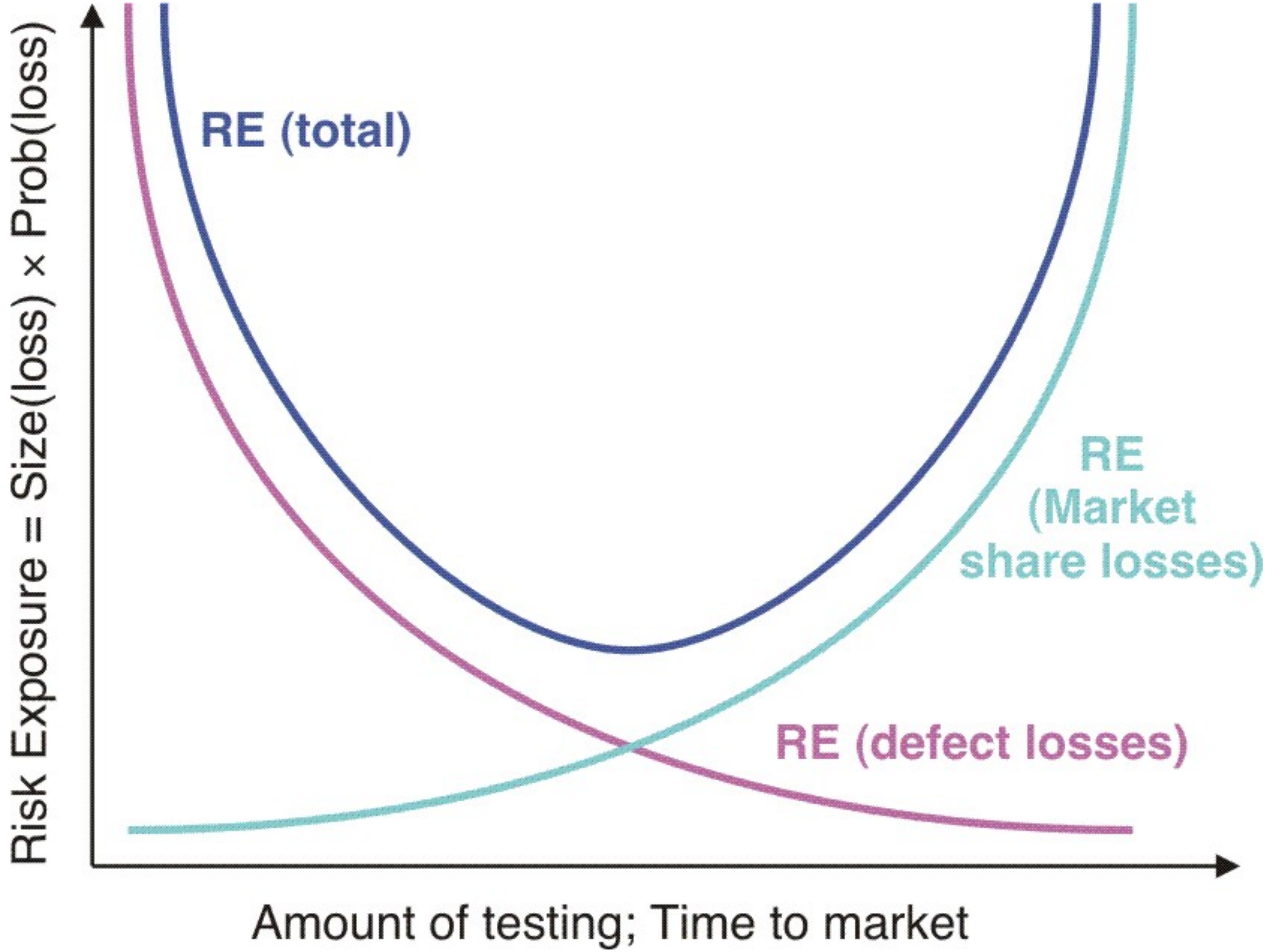
RUP

lightweight

middleweight

heavyweight

Balance between Quality and Time-to-Market



CONCLUSIONS

- Different life-cycle models
- Each with own strengths and weaknesses
- Criteria for deciding on a model include
  - The organization
  - Its management
  - Skills of the employees
  - The nature of the project
- Best suggestion
  - "Mix-and-match" life-cycle model

End of Chapter 2