CS 361 – Software Engineering

Software Process

Kamonphop Srisopha

Churee Techawut

Adapted from: Ian Sommerville, 2016, Software engineering, Addison Wesley.
Mark Sherriff and Will McBurney, 2020, Advanced Software Development

Faculty of Science, Chiang Mai University
คณะวิทยาศาสตร์ มหาวิทยาลัยเชียงใหม่

Agenda

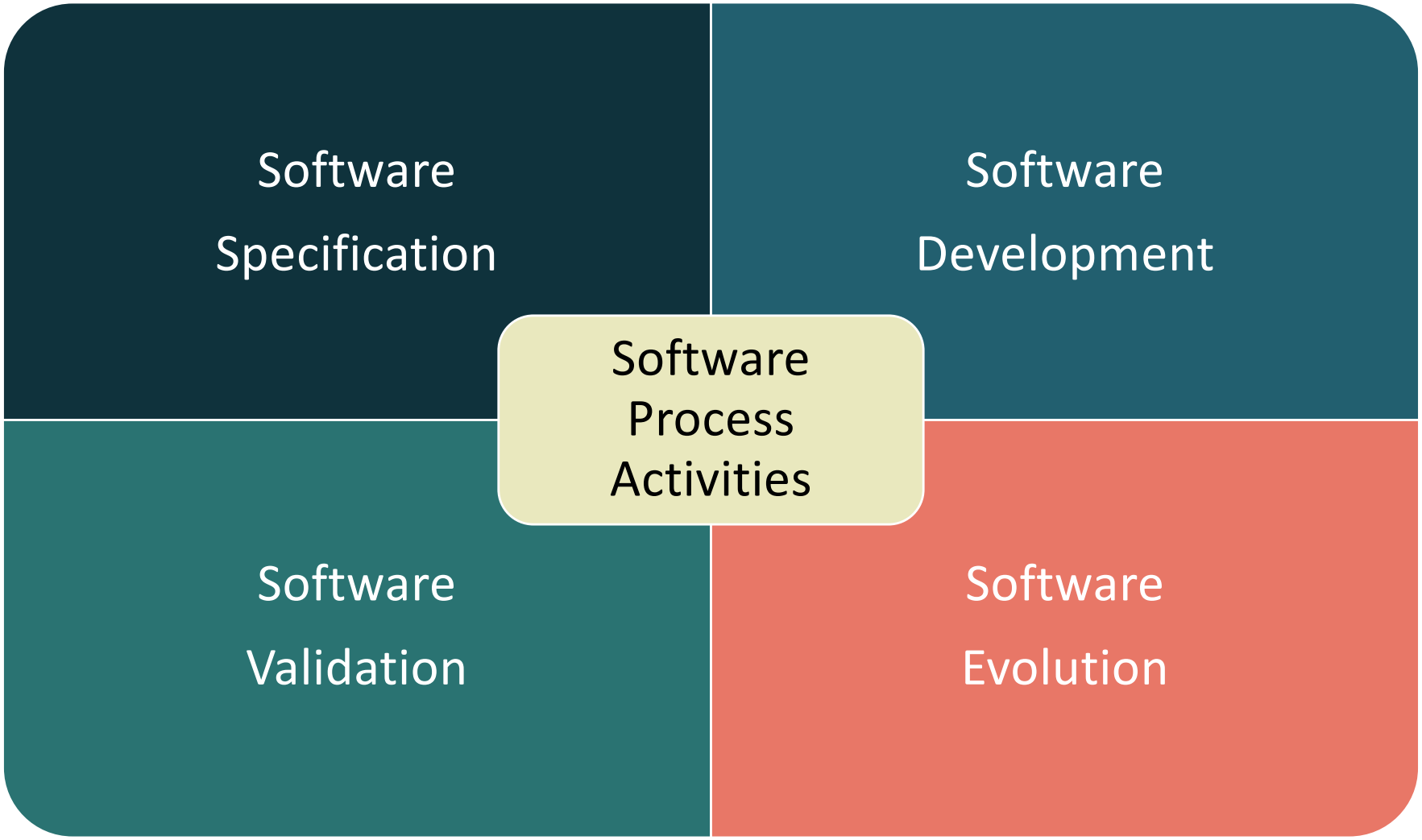
- What is software process?
- Software process models
  - Waterfall model
  - Spiral model
  - Iterative and Incremental model
  - Integration and configuration
- Process activities
  - Specification
  - Design and Implementation
  - Validation
  - Evolution
- Choosing a software process

Engineering Good Software

- Writing Code is EASY
- Engineering good software is HARD
  - Humans are a BIG problem. As soon as you introduce more than one group of people into the process, complications arises (communication, understanding, etc).
    - Clients/Customers
    - Developers
    - Managers
  - Large software/system – complexity, quality, etc.

Following a process is key to making it work

What is software process?



What is software process?

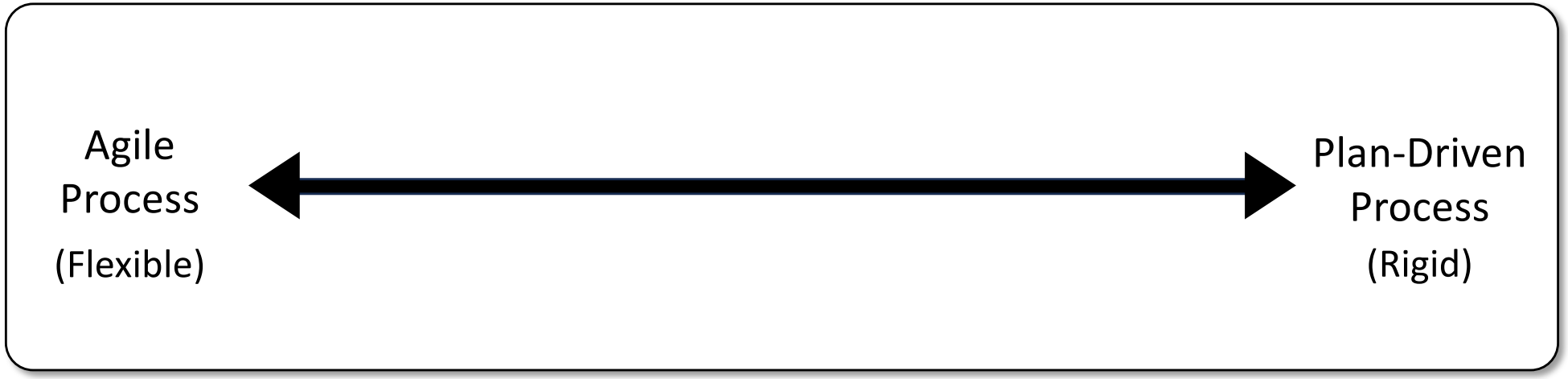
- คือเซ็ตของกิจกรรม (a set of activities) ที่เป็นลำดับเพื่อใช้ในการพัฒนาซอฟต์แวร์

- เซ็ตของกิจกรรมของทุก process จะประกอบไปด้วย (กว้างๆ)
  - **Requirements Elictation and Specification** การหาความต้องการและกำหนดข้อมูลเฉพาะ; กำหนดว่าระบบควรทำอะไร
  - **Design and implementation** การออกแบบและการพัฒนา; การกำหนดองค์ประกอบและรูปแบบของระบบ รวมถึงการพัฒนาระบบ
  - **Validation** การตรวจสอบความถูกต้อง; การตรวจสอบว่าเป็นไปตามที่ลูกค้าต้องการหรือไม่
  - **Evolution (aka Maintenance)** วิวัฒนาการ;. การเปลี่ยนแปลงระบบเพื่อตอบสนองการเปลี่ยนแปลงความต้องการของลูกค้า

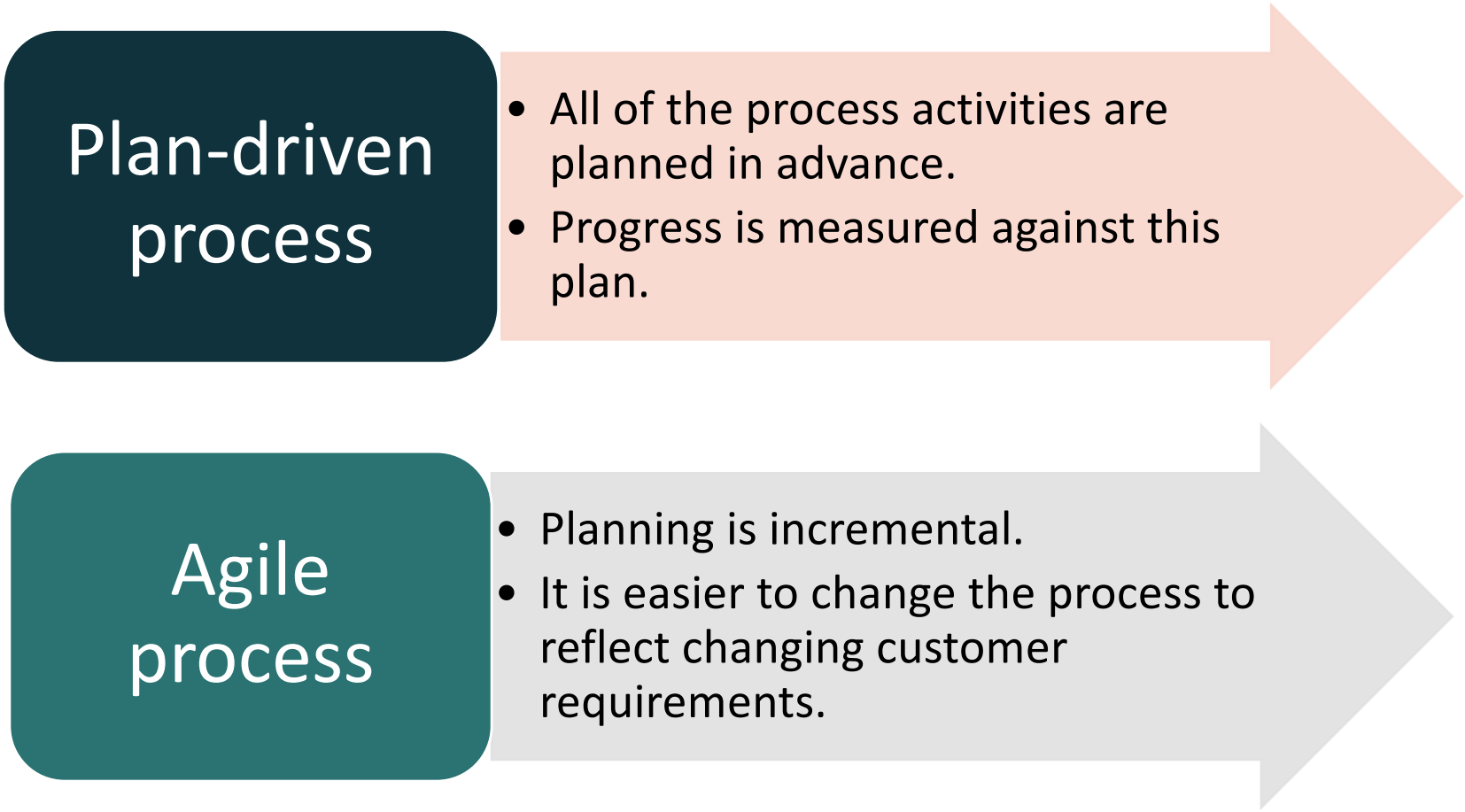
ลำดับของกิจกรรมเหล่านี้, การทำซ้ำ, เรื่องของเวลา, เรื่องของ deliverables/ milestones และ อื่นๆ เป็นตัวกำหนดแบบแผนของ process หรือ methodology ชนิดต่างๆ

Software Process Models

- โมเดลกระบวนการพัฒนา software ส่วนใหญ่จะสามารถวางอยู่สักที่บนเส้น (“continuum”) ระหว่าง Agile Process และ Plan-driven Process



Plan-driven Process vs Agile Process



- In practice, most practical processes include elements of both plan-driven and agile approaches. (อยู่บน “continuum”)
- There are no right or wrong software processes (no one-size-fits-all approach).

Plan-driven and Agile

- **Plan-driven Methodologies**
  - Rational Unified Process (RUP)
  - Personal Software Process (PSP)
  - Team Software Process (TSP)

- **Agile Methodologies**
  - Scrum
  - Extreme Programming (XP)
  - Kanban

Note: A methodology ("how") is an instance of a process ("what")

Software Process Models

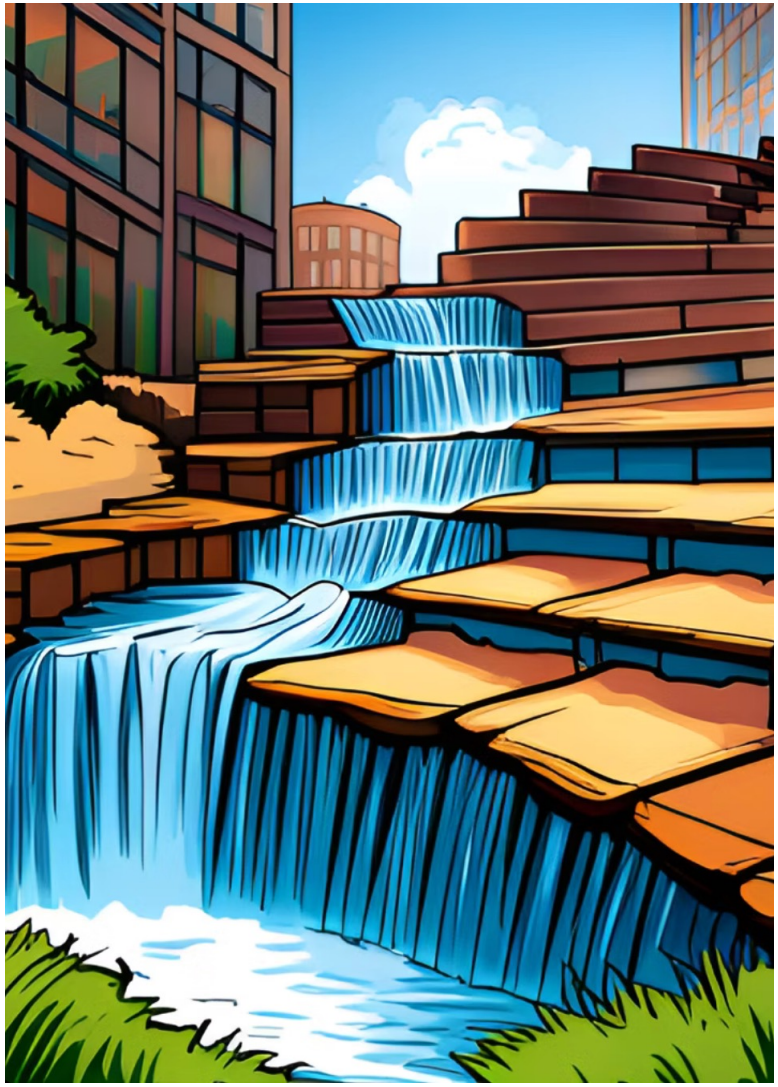
The Basis of Software Process

- Waterfall model
- Spiral model
- Iterative and Incremental model
- Integration and Configuration

ส่วนใหญ่ในทางปฏิบัติระบบขนาดใหญ่จะถูกพัฒนาโดยการใช้ process ที่รวมส่วนประกอบต่างๆของ model เหล่านี้

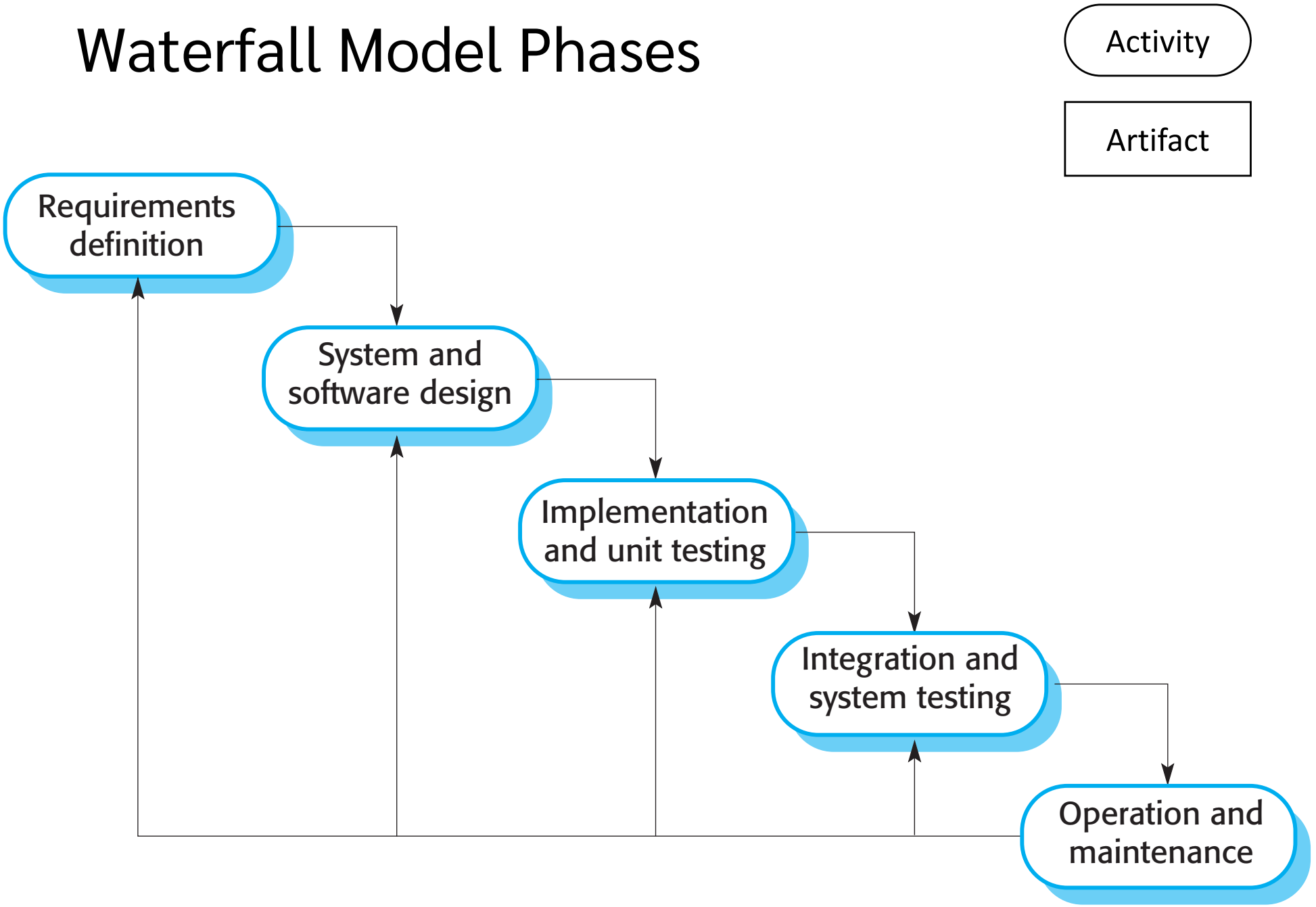
The Waterfall Model

Waterfall Model



- Separate and distinct phases
- There are separate identified phases in the waterfall model:
    1. Requirements analysis and definition
    2. System and software design
    3. Implementation and unit testing
    4. Integration and system testing
    5. Operation and maintenance
- Produce a document at each phase and a document that defines what to do at the start of each phase.
- Complete one phase before moving on to the next.

Waterfall Model Phases

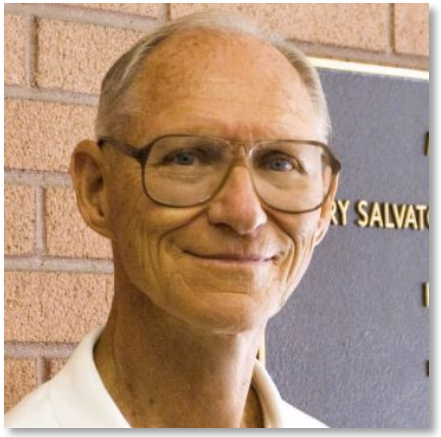


Waterfall Model Problems

- It’s very rigid so it is difficult to respond to changing customer requirements.
  - only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.
  - Few business systems have stable requirements.
- Does not emphasize important issues
  - Risk Management
  - Feedback and iteration
  - Things can change (especially requirements)
  - Quality of software post release

The Spiral Model

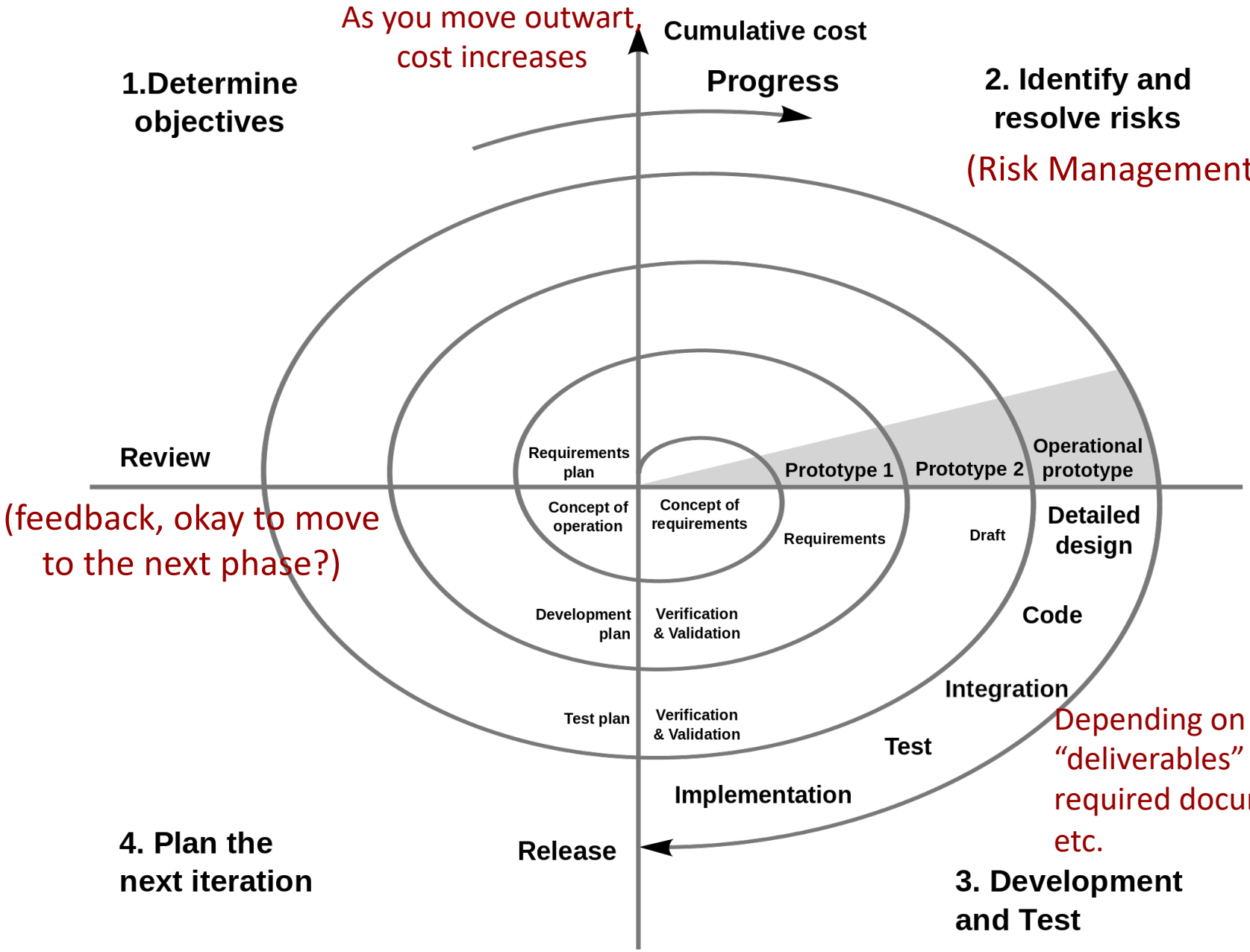
Spiral Model



- Introduced by Barry Boehm (1986)
- An *improved* version of the Waterfall model
- It's technically waterfall, with the ability to cycle back on phases
- **Important features:**
  - Risk analysis and other managements are explicitly shown in the model at each stage
  - Prototyping and iterative development "fit" in this model
  - Reptition of activities is clearly shown in the model
    - Suggests that alternatives can be explored

Spiral Model

Each cycle around the spiral can be like a phase and has 4 stages



Is the Spiral Model the Answer?

- For some situations, yes. (**Again, there are no one-size-fits-all models**)

- Original study that analyzed the Spiral model says:
  - Intended for internal development of large systems
  - Intended for large-scale systems where cost of no risk management is high

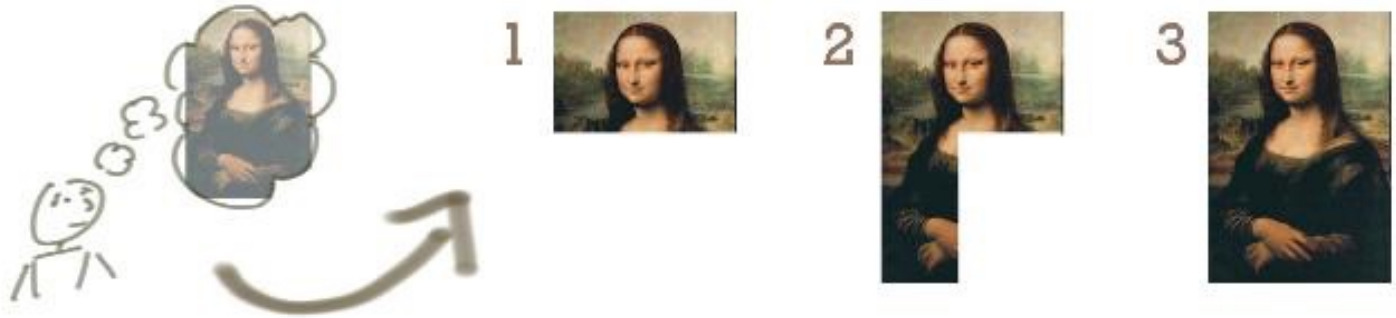
- It’s important
  - Historically … and many software development processes **are based on** the spiral model one way or another.

The Iterative and Incremental Model

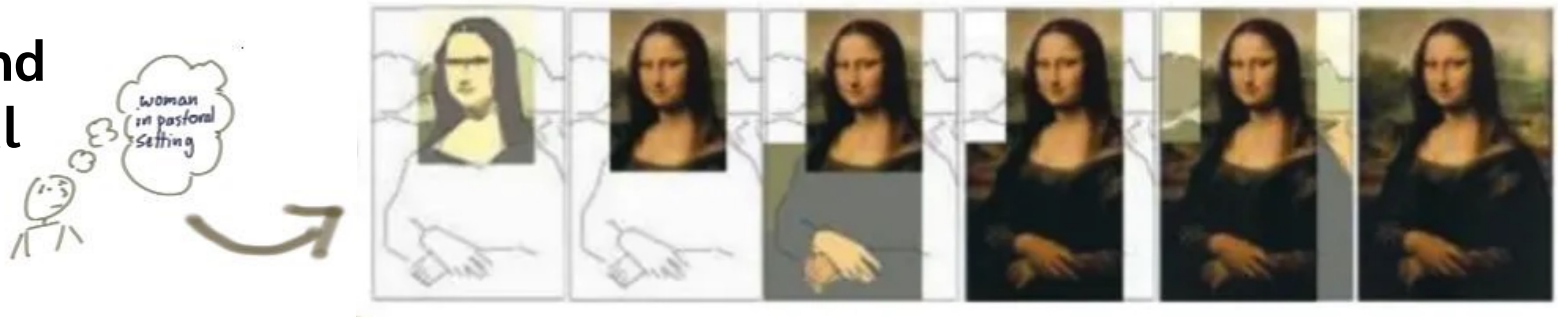
Iterative and Incremental

Iterative
(refining)

Increment
(piece by piece)



Interative and Incremental



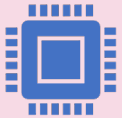
Images from Jeff Patton (2008): https://www.jpattonassociates.com/dont_know_what_i_want/

Iterative and Incremental

ปรกติแล้วเราไม่ได้ทำอาหารโดยการใส่วัตถุดิบทุกอย่างลงไปในครั้งเดียวและหวังว่ามันจะออกมาอร่อยหรือถูกเมนู

เราต้องมีการชิม เพื่อปรับปรุงรสชาติ (Iterative) แล้วก็เพิ่มส่วนต่างๆทีละอย่าง (incremental) จนกลายเป็นอาหารที่เราอยากจะให้มันเป็นและได้รสชาติที่ต้องการ

The Iterative and Incremental Model



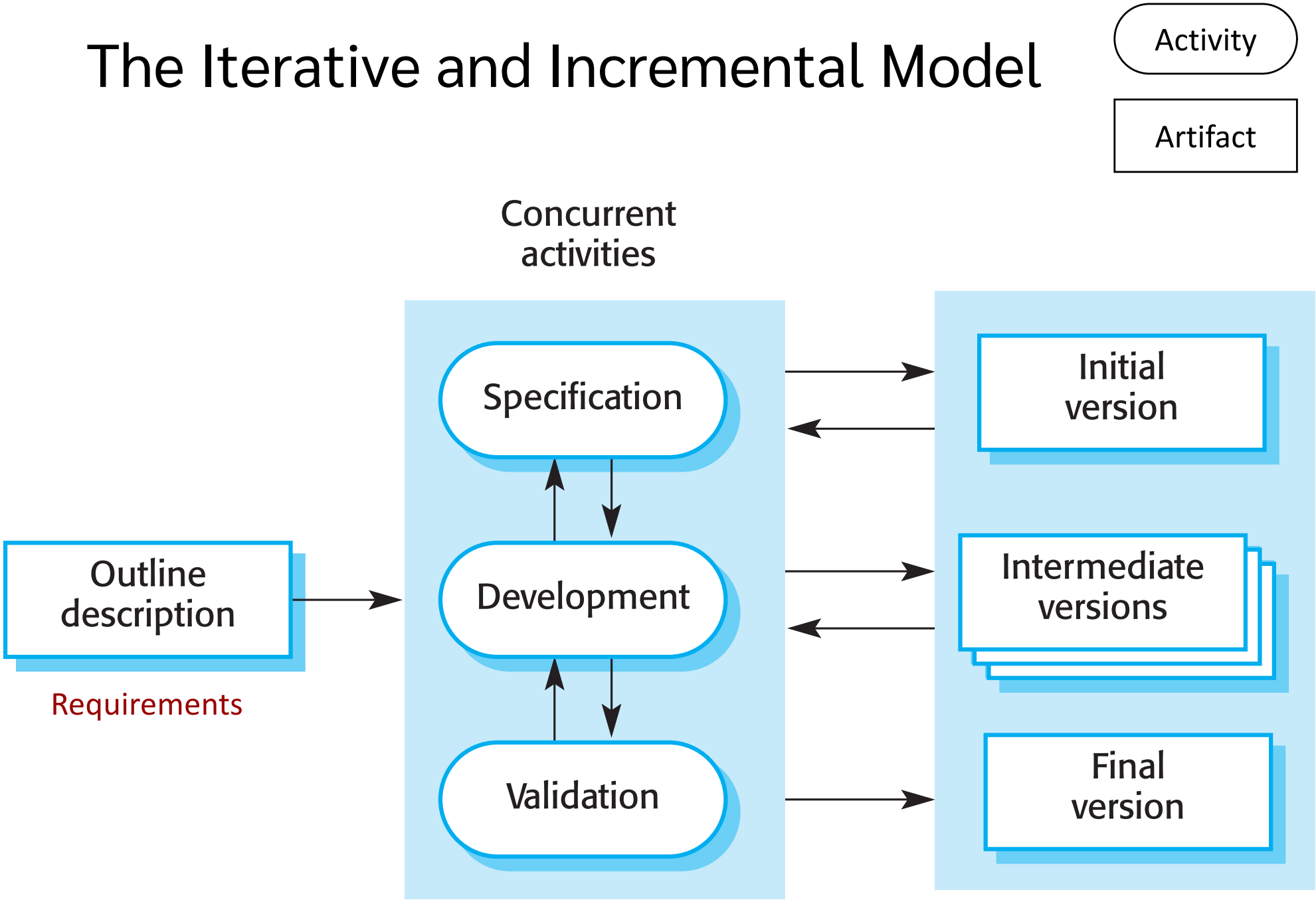
Specification, development and validation are interleaved. May be plan-driven or agile.

Adapting to customer changes is more cost-efficient, with less rework compared to the waterfall model.

It is easier to get customer feedback on the development work that has been done.

Speedier delivery and deployment of useful software to the customer is possible.

The Iterative and Incremental Model



Iterative and Incremental Problems

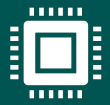
- The process is not visible.
  - **Managers need regular deliverables to measure progress**. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.

- System structure tends to degrade as new increments are added/integrated.
  - Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure. (*regular change leads to messy code*) **Incorporating further software changes becomes increasingly difficult and costly.**

The Integration and Configuration Model

Integration and Configuration

The system is assembled from existing configurable components. May be plan-driven or agile.



Based on software reuse where systems are integrated from existing components or application systems (sometimes called COTS -Commercial-off-the-shelf) systems).

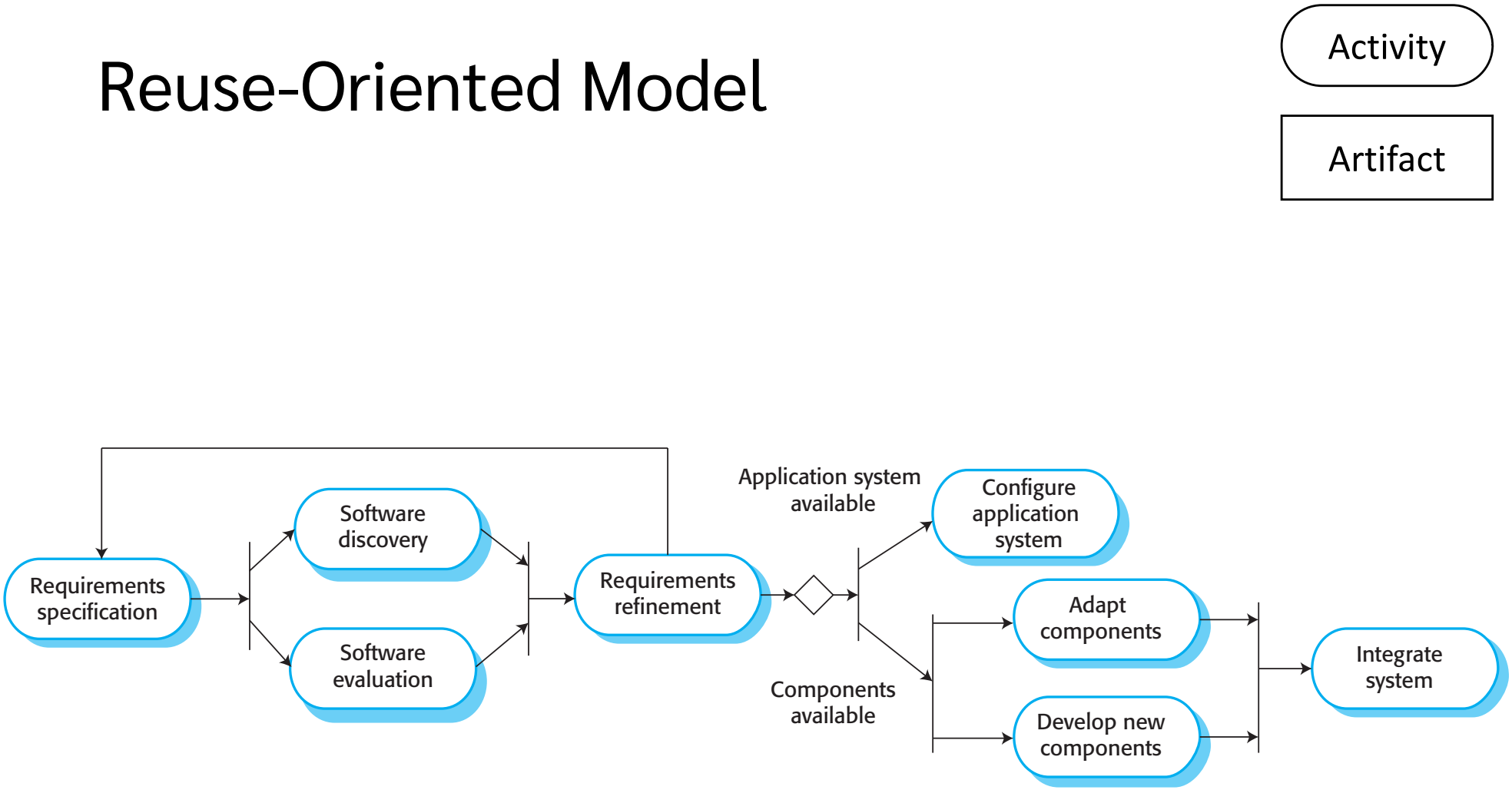
Reused elements may be configured to adapt their behaviour and functionality to a user's requirements

Reuse is now the standard approach for building many types of business system

Examples of Reusable Software

- **Stand-alone application systems** (sometimes called COTS – **C**ommercial-**O**ff-**T**he-**S**helf) that are configured for use in a particular environment.
- **Software packages** or collections of objects that are developed to be integrated with a component framework such as .NET or J2EE.
- **Web services** that are developed according to service standards and which are available for remote invocation.
- **Software Libraries**

Reuse-Oriented Model



Advantages and Disadvantages

- Reduced costs and risks as less software is developed from scratch
- Faster delivery and deployment of system
- But requirements compromises are inevitable so system may not meet real needs of users
- Loss of control over evolution of reused system elements
- Security and trust
- License

Software Process Activities
(กิจกรรมใน Software Process)

Process Activities

- Real software processes are inter-leaved sequences of technical, collaborative and managerial activities.
- The four basic process activities are organized differently in different development processes.
    1. specification
    2. development
    3. validation
    4. evolution
- For example, in the waterfall model, they are organized in sequence, whereas in incremental development they are interleaved.

ลำดับของกิจกรรมเหล่านี้, การทำซ้ำ, เรื่องของเวลา, เรื่องของ deliverables/ milestones และ อื่นๆ เป็นตัวกำหนดแบบแผนของ process หรือ methodology ชนิดต่างๆ

Software Specification

- The process of establishing what services are required and the constraints on the system's operation and development.

- Requirements engineering process:

Requirements elicitation and analysis

- What do the system stakeholders require or expect from the system?

Requirements specification

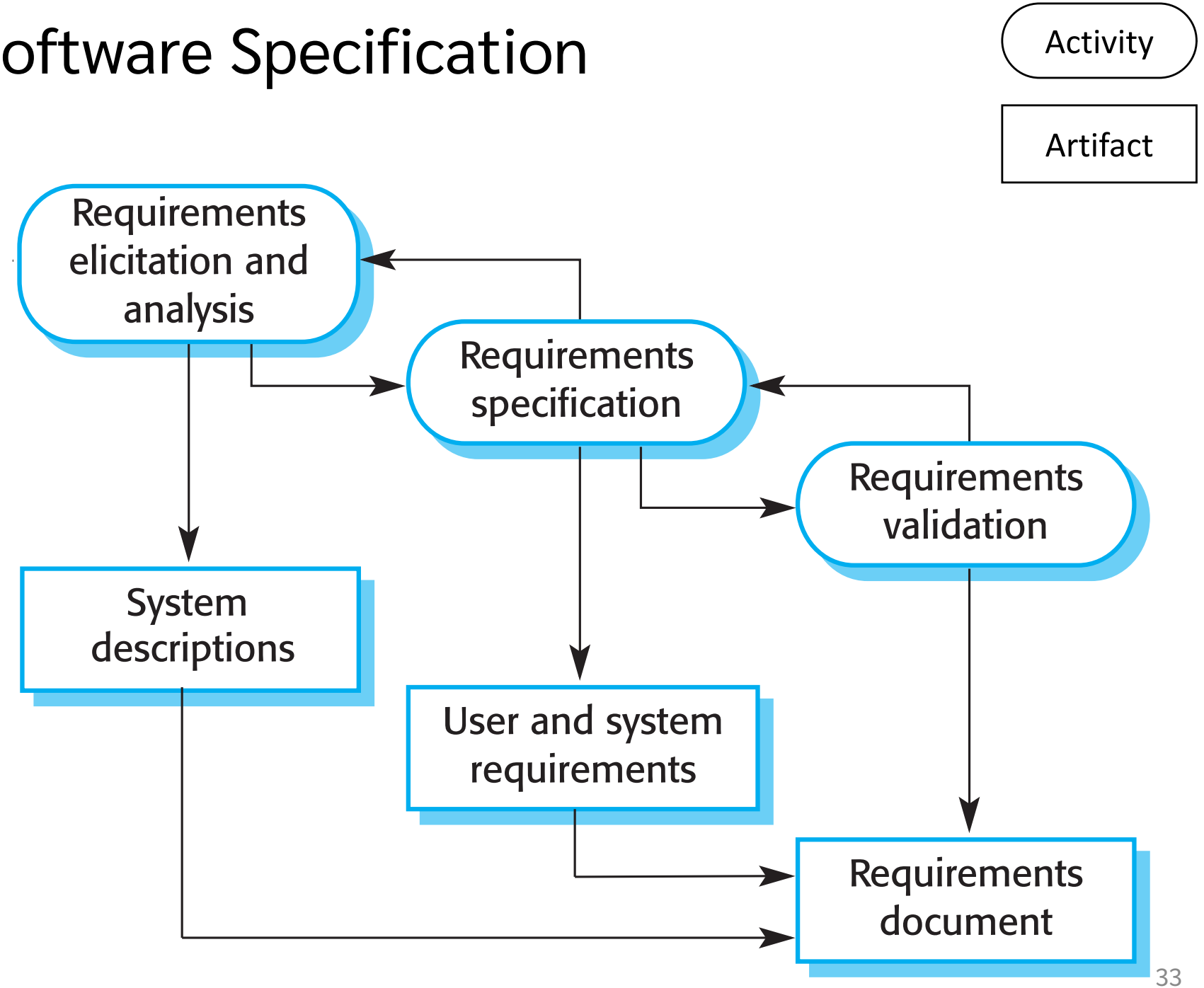
- Defining the requirements in detail

Requirements validation

- Checking the validity of the requirements

การแปลงความต้องการของผู้ใช้ให้เป็น Requirements

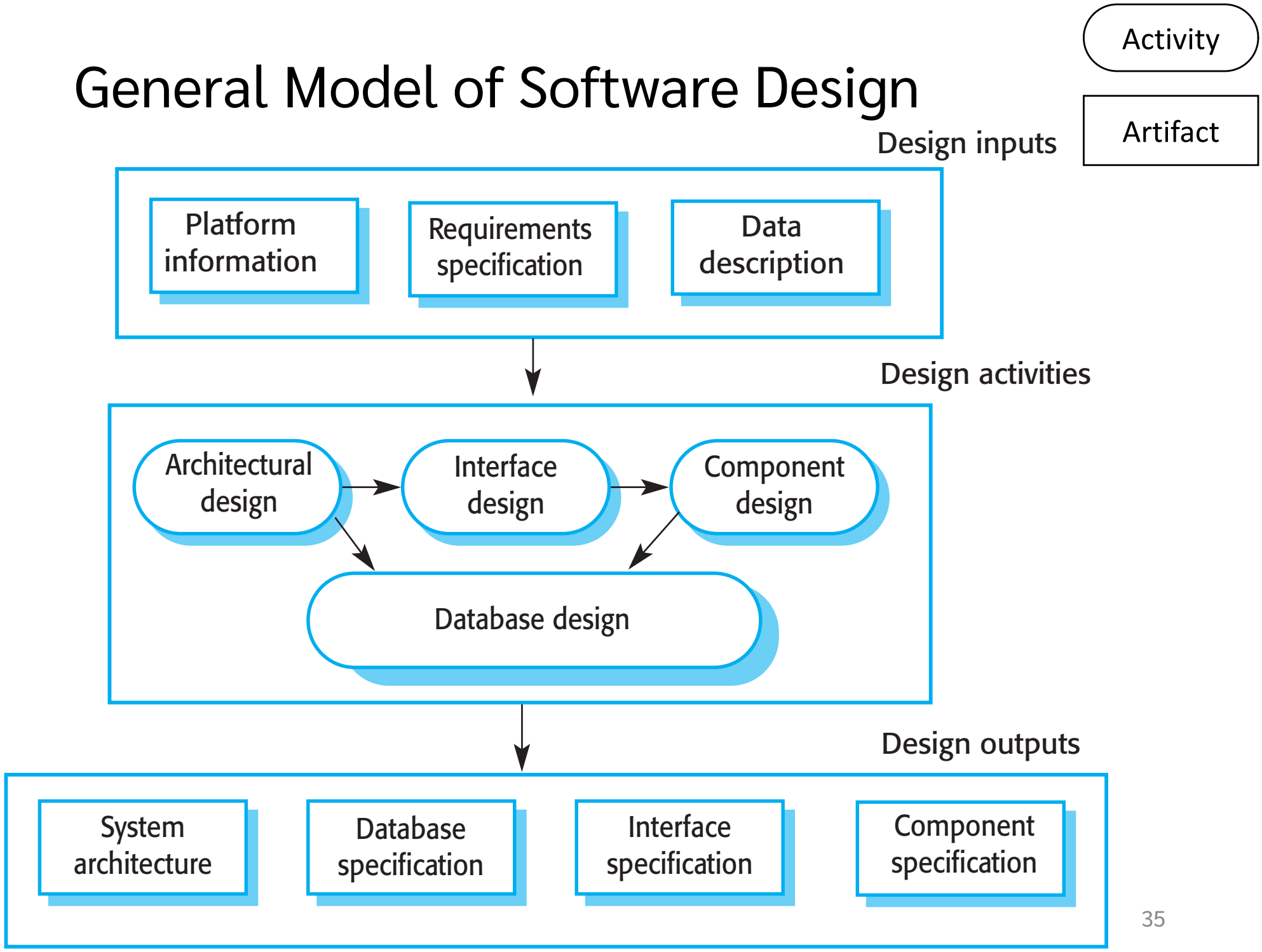
Software Specification



Software Design and Implementation

- The process of converting the system specification into an executable system.
- Software design
  - Design a software structure that realises the specification;
- Implementation
  - Translate this structure into an executable program;
- The activities of design and implementation are closely related and may be interleaved.

General Model of Software Design



Design activities

- **Architectural design**, where you identify the overall structure of the system, the principal components (subsystems or modules), their relationships and how they are distributed.
- **Database design**, where you design the system data structures and how these are to be represented in a database.
- **Interface design**, where you define the interfaces between system components.
- **Component selection and design**, where you search for reusable components. If unavailable, you design how it will operate.

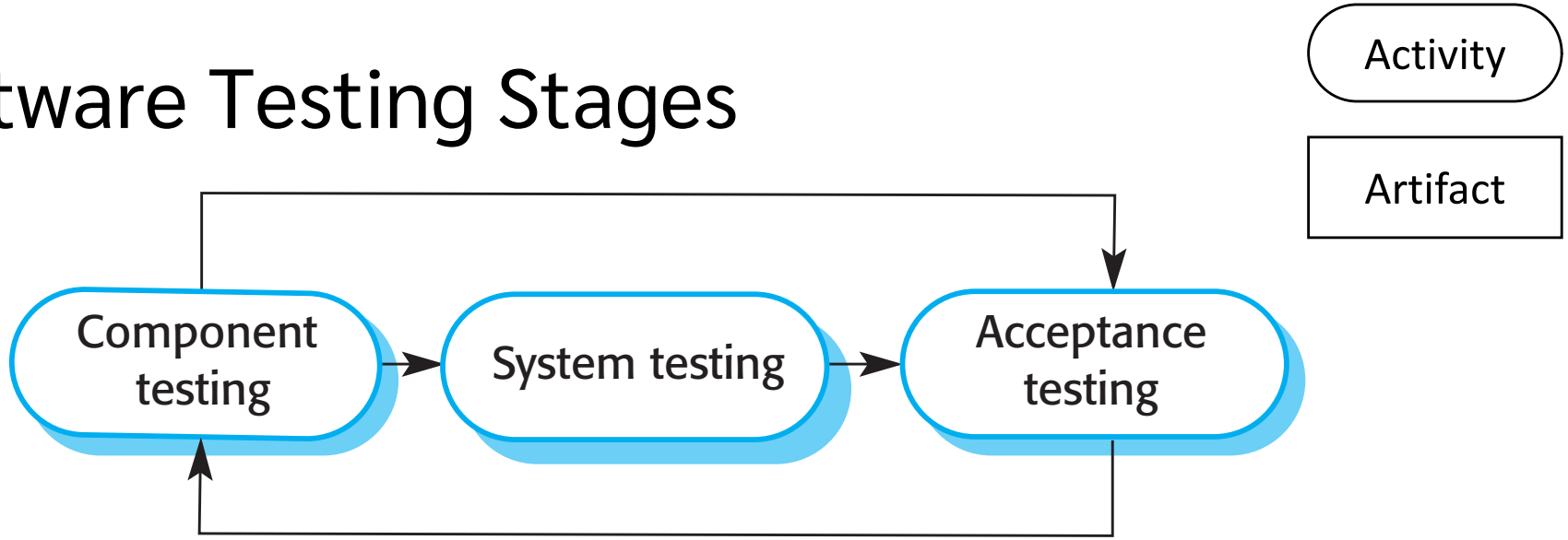
System Implementation

- The software is implemented either by developing a program or programs or by configuring an application system.
- Design and implementation are interleaved activities for most types of software system.
- Programming is an individual activity with no standard process.
- Debugging is the activity of finding program faults and correcting these faults.

Software Validation

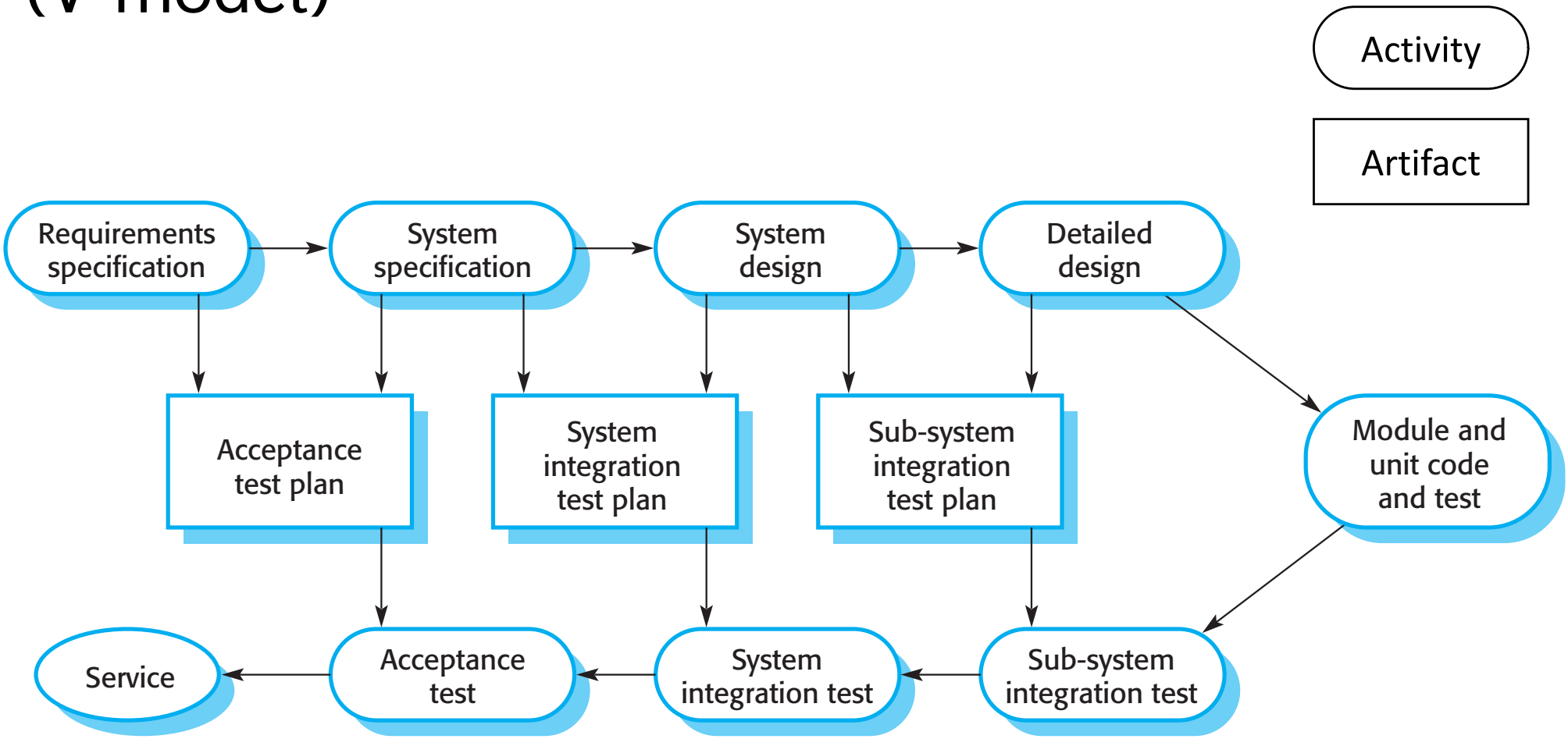
- Verification and validation (V & V) is intended to show that a system conforms to its specification and meets the requirements of the system customer.
- Involves checking and review processes and system testing.
- System testing involves executing the system with test cases that are derived from the specification of the real data to be processed by the system.
- **Testing** is the most commonly used V & V activity.

Software Testing Stages



- **Component testing**
  - Individual components are tested independently;
  - Components may be functions or objects or coherent groupings of these entities.
- **System testing**
  - Testing of the system as a whole. Testing of emergent properties is particularly important.
- **Acceptance (Customer) testing**
  - Testing with customer data to check that the system meets the customer's needs.

Testing Phases in A Plan-driven Software Process (V-model)

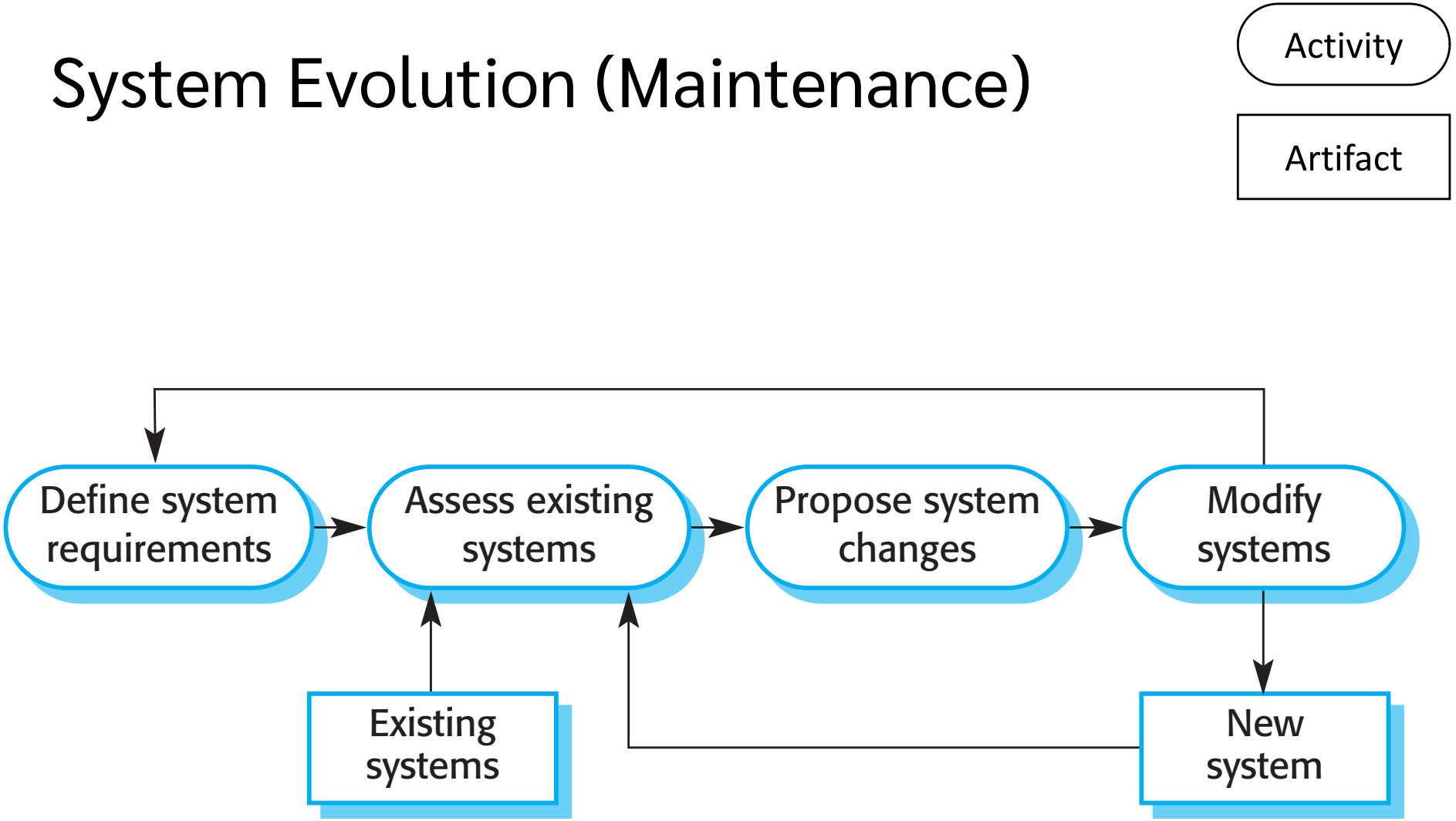


Key idea: At each stage, there has to be some sort of testing that accompoanies it

Software Evolution (Maintenance)

- Software is inherently flexible and can change.
- As requirements change through changing business circumstances, the software that supports the business must also evolve and change.
- Although there has been a dividing line between development and evolution (maintenance) this is increasingly irrelevant as fewer and fewer systems are completely new.

System Evolution (Maintenance)



Choosing A Software Process

Some of the Factors Contributing to the Decision

Organizational factors

- How is the team structured? co-located or global? Small or large? Culture?

Technology factors

- Does the team like to work using whiteboards, sticky notes, or online? Does the team have access to tools or work with any hardware?

Domain/ Sector factors

- How quickly is time-to-market of the product? How rapid is state-of-art changing? Need to keep up? Any changes in the requirements?

Regulartory factors

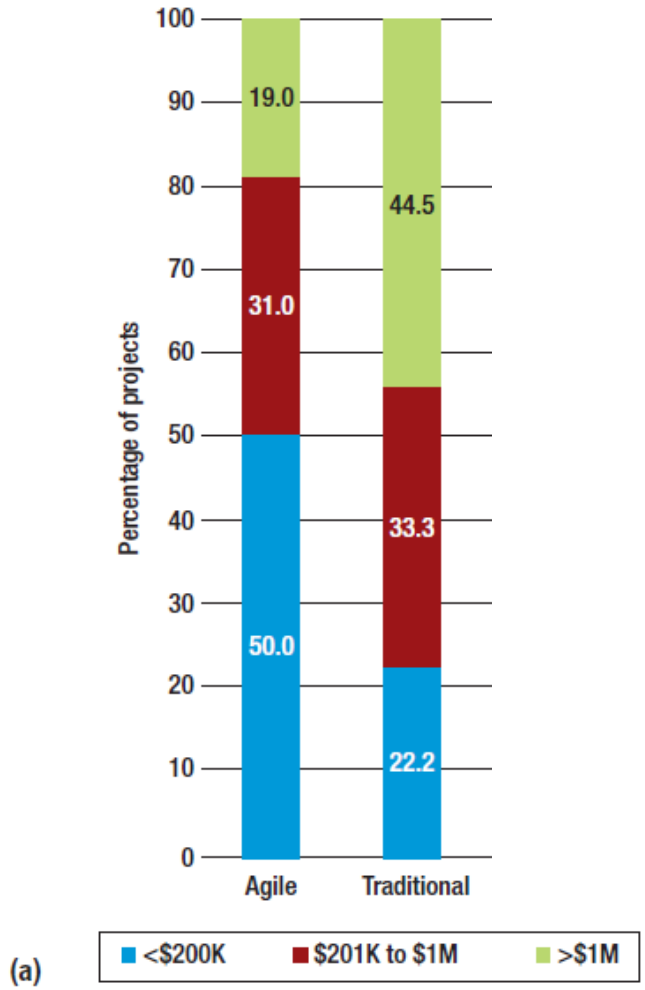
- Are you building something that requires government/third-party oversights? Life-critical software? Any legal documents you need to make?

Human factors

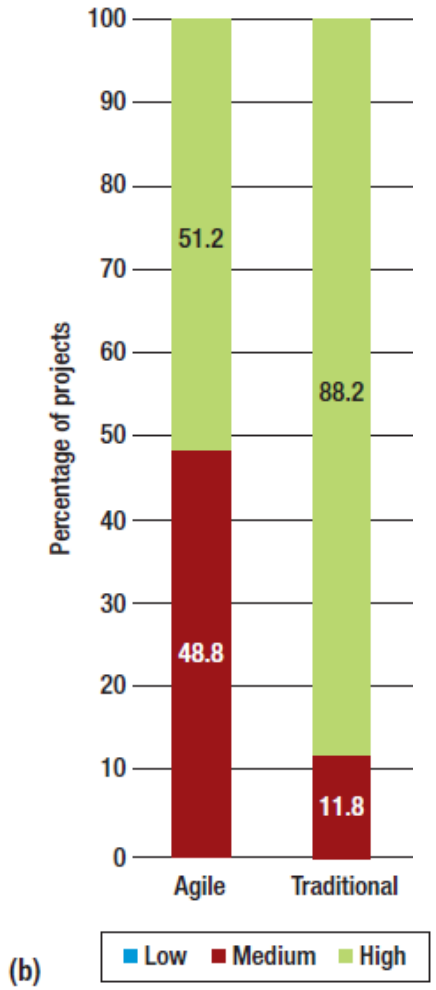
- What is your team make up? People coming and going? Experience level?

Choosing A Software Process

By Budget



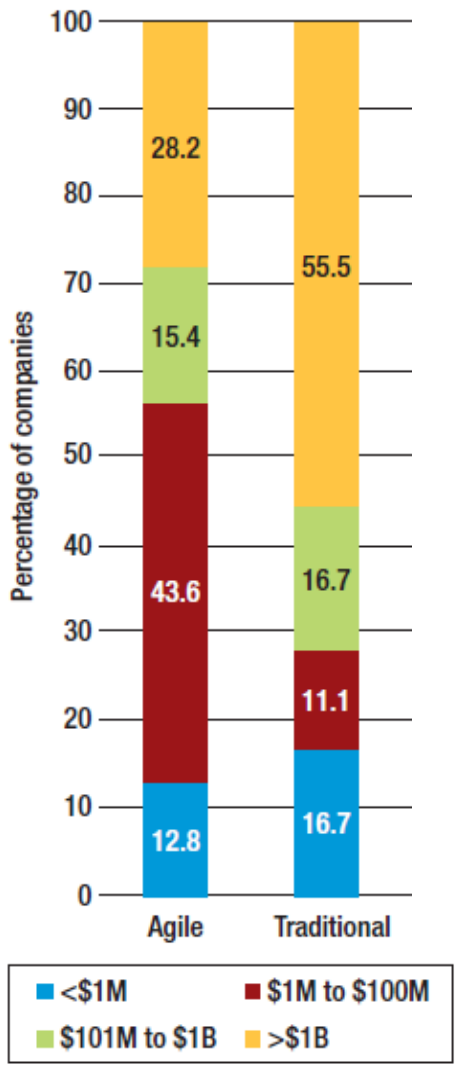
By Project Criticality to the Company



Leo R Vijayasarathy ad Charles Butler, “Choice of Software Development Methodologies”, IEEE Software, September/October, 2016

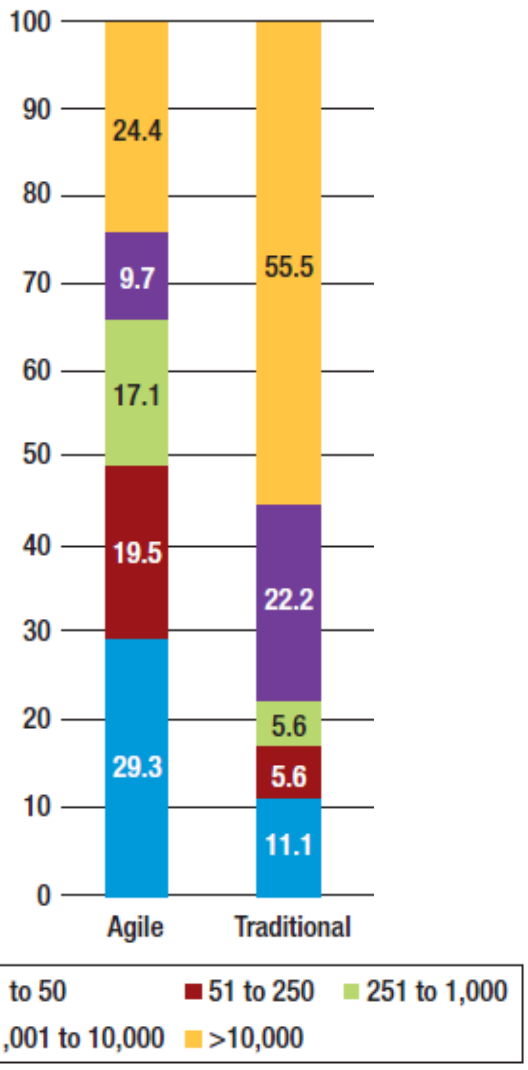
Choosing A Software Process

By Annual Revenue



(a)

By Number of Employees

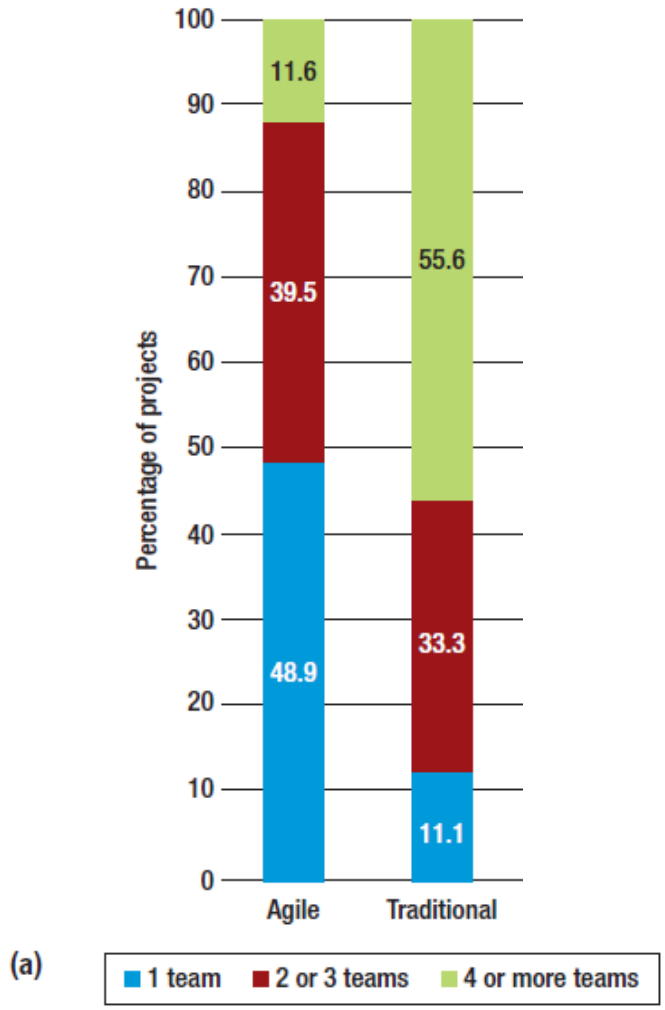


(b)

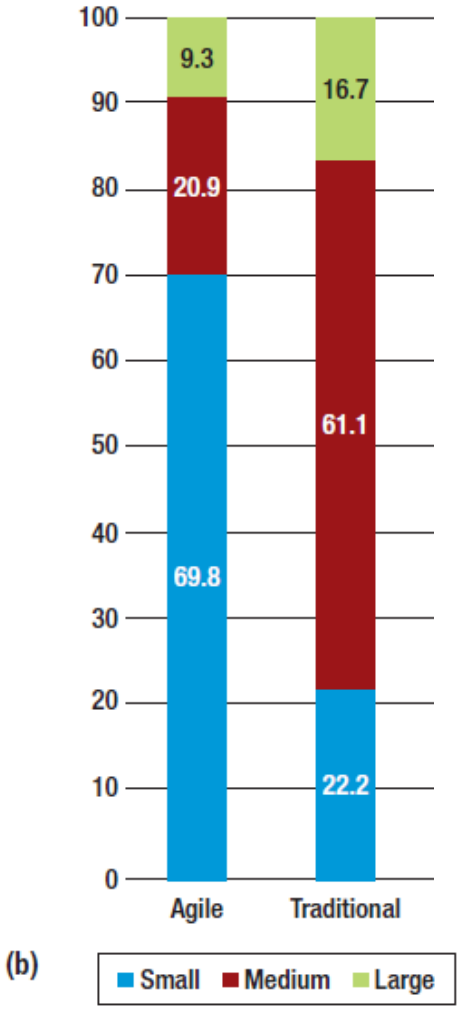
Leo R Vijayasarathy ad Charles Butler, “Choice of Software Development Methodologies”, IEEE Software, September/October, 2016

Choosing A Software Process

By Number of Teams



By Team Size



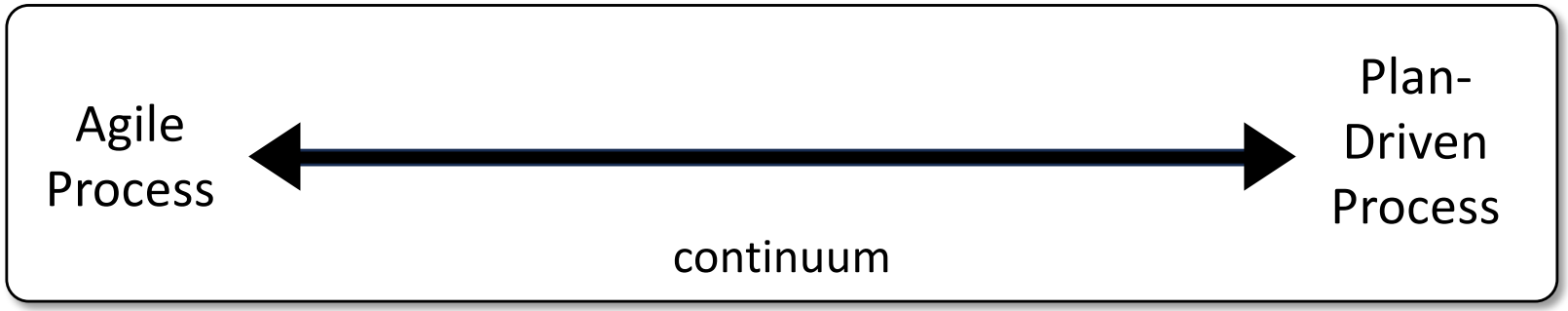
 # Members

Small	≤ 10
Medium	11 to 30
Large	> 30

Leo R Vijayasarathy ad Charles Butler, "Choice of Software Development Methodologies", IEEE Software, September/October, 2016

Bottom Line

- There is no one good “catch” all process (**no one-size-fits-all**)
- Most companies **don’t** use a stock process/methodology
- Everything **is customized** to the particular organization
- Once you determine where on the “**continuum**” your organization falls for process, then start looking at the aspects of various methodologies that fall in the same area.



Questions ?