





PHÁT TRIỂN VẬN HÀNH BẢO TRÌ PHẦN MỀM

ThS. NGUYỄN THỊ THANH TRÚC

Nội dung (Chương 3)

- 
-  **QUI TRÌNH BẢO TRÌ PHẦN MỀM**
 -  **CÁC MÔ HÌNH BẢO TRÌ PHẦN MỀM**
 -  **KHI THỰC HIỆN THAY ĐỔI**
 -  **Thảo luận và làm bài tập**
 -  **Q&A**

Chương 3:

QUI TRÌNH VÀ MÔ HÌNH BẢO TRÌ PHẦN MỀM

3.1 QUI TRÌNH BẢO TRÌ PHẦN MỀM

3.2 CÁC MÔ HÌNH BẢO TRÌ PHẦN MỀM

3.3 KHI THỰC HIỆN THAY ĐỔI

Chương 3:

QUI TRÌNH VÀ MÔ HÌNH BẢO TRÌ PHẦN MỀM

1. QUI TRÌNH BẢO TRÌ PHẦN MỀM

- Định nghĩa
- Quy trình sản phẩm phần mềm
- Đánh giá phê bình quy trình mô hình truyền thống
 - ✓ Code-and-Fix Model
 - ✓ Waterfall Model
 - ✓ Spiral Model

2. CÁC MÔ HÌNH BẢO TRÌ PHẦN MỀM

- Mô hình Quick-Fix
- Mô hình Boehm
- Mô hình Osborne
- Iterative Enhancement Model
- Mô hình Reuse-Oriented

3. KHI THỰC HIỆN THAY ĐỔI

- Tăng trưởng quy trình
- Mô hình tăng trưởng CMM (Capability Maturity Model) cho phần mềm
- Cơ sở kinh nghiệm phần mềm

3.1 QUI TRÌNH BẢO TRÌ PHẦN MỀM

- ❑ Định nghĩa
- ❑ Quy trình sản phẩm phần mềm
- ❑ Đánh giá phê bình quy trình mô hình truyền thống
 - Code-and-Fix Model
 - Waterfall Model
 - Spiral Model

Software Process

Fundamental Assumption:

Good processes lead to **good software**

Good processes reduce **risk**

Good processes enhance **visibility**

Basic Process Steps in all Software Development

- **Feasibility** and planning
 - **Requirements**
 - System and program **design**
 - **Implementation** and testing
 - **Acceptance** testing and release
-
- Operation and **maintenance**

It is essential to distinguish among these process steps and to be clear which you are doing at any given moment.

Do not confuse requirements and design

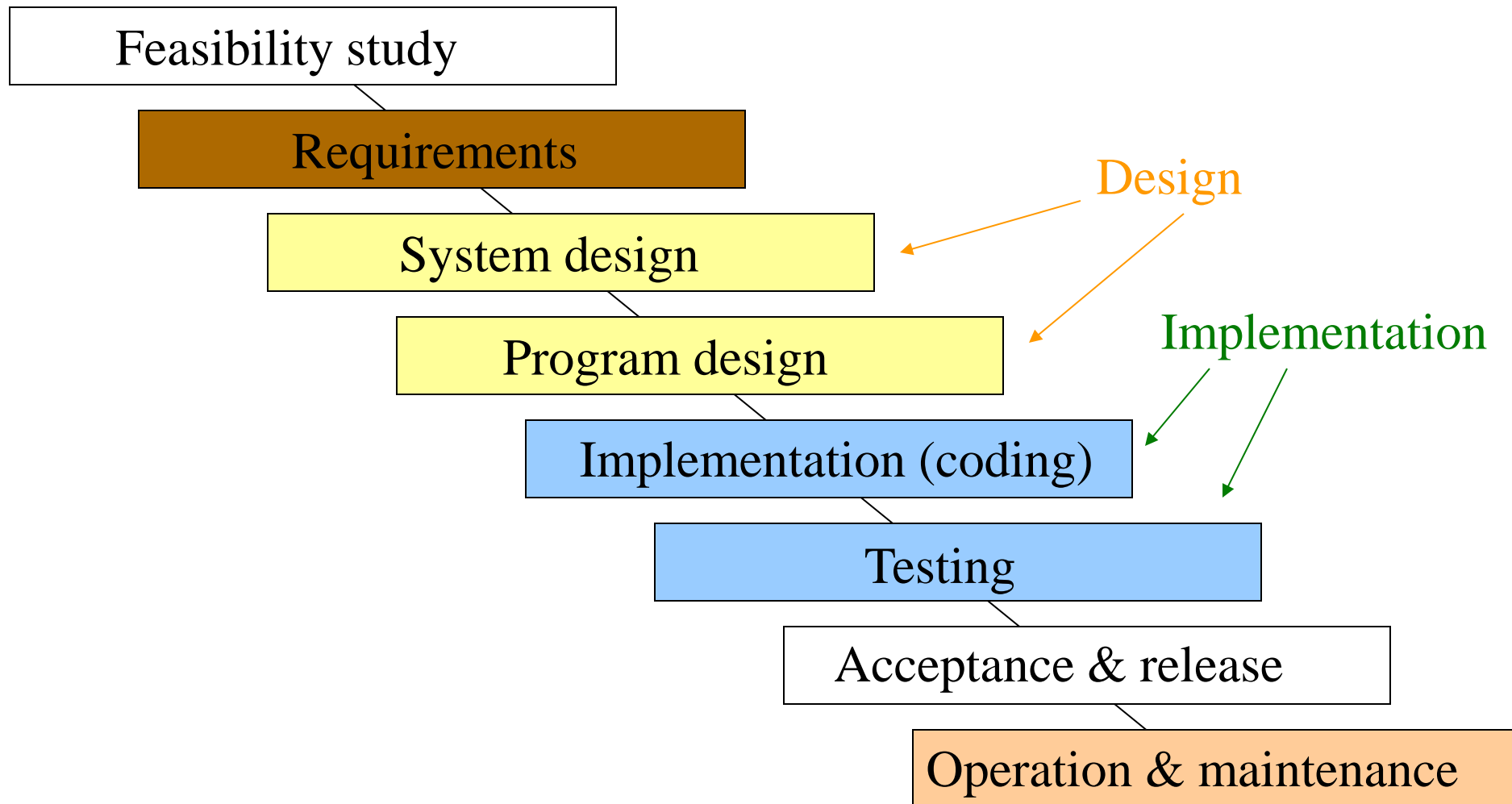
Sequence of Processes (software lifecycle)

Every software project will include these basic processes, **in some shape or form**, but they may be carried out in various sequences

Major alternatives

- **Sequential:** Complete each process step before beginning the next (but see the next few slides). *Waterfall model.*
- **Iterative:** Go quickly through all process steps to create a rough system, then repeat them to improve the system. *Iterative refinement.*

Sequential Development: The Waterfall Model



Thảo luận Waterfall Model

Thuận lợi:

- qui trình rõ ràng
- công việc tách biệt
- kiểm soát chất lượng mỗi bước
- Kiểm soát chi phí ở mỗi bước

Không thuận lợi:

Mỗi giai đoạn trong qui trình thể hiện hiểu biết mới của giai đoạn trước đó mà thường đòi hỏi giai đoạn sớm hơn được xét duyệt lại.

The Waterfall Model is not enough!

Tính tuần tự của các qui trình

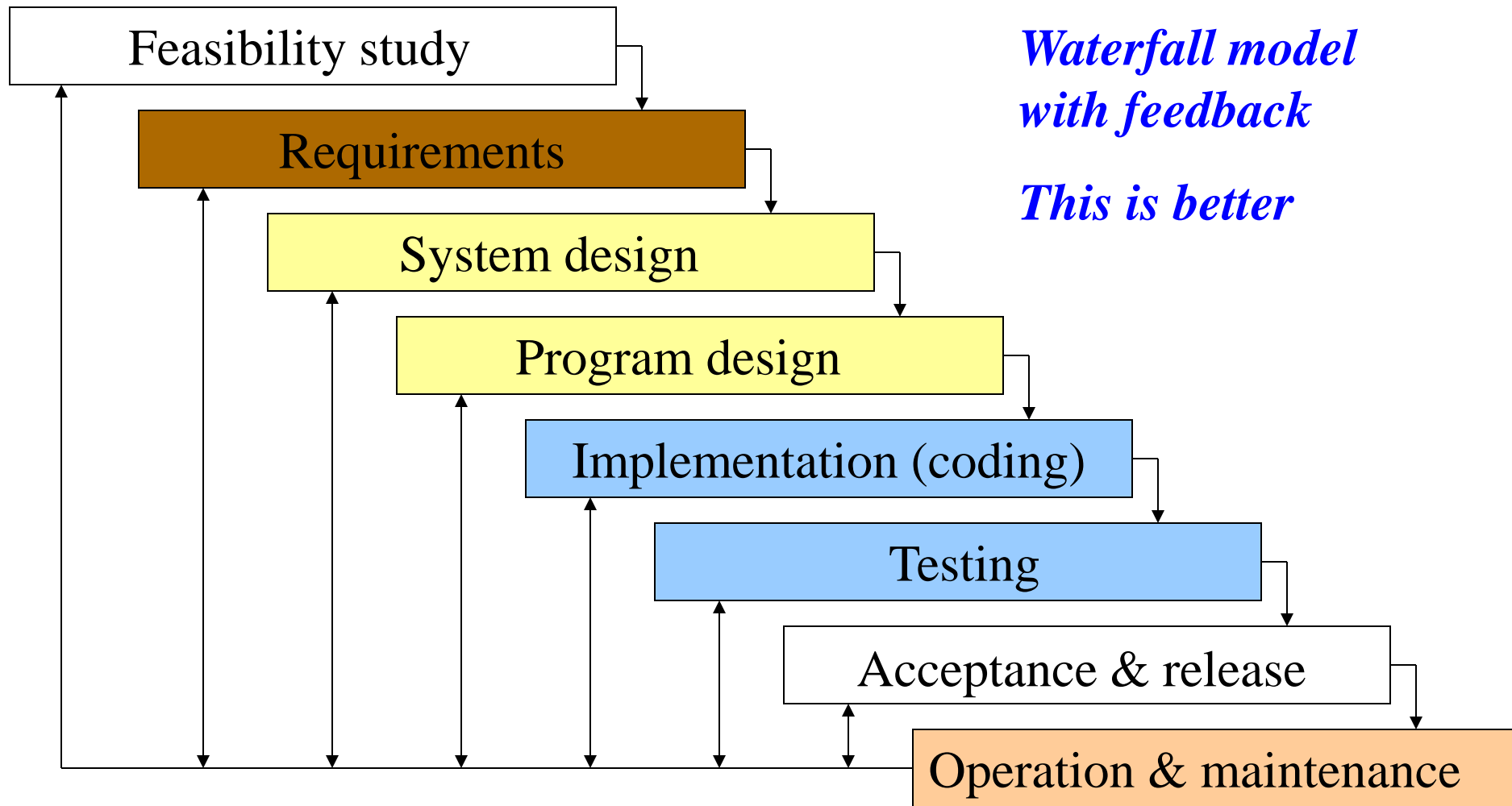
Mô hình thuần tuần tự thì không thể

Ví dụ:

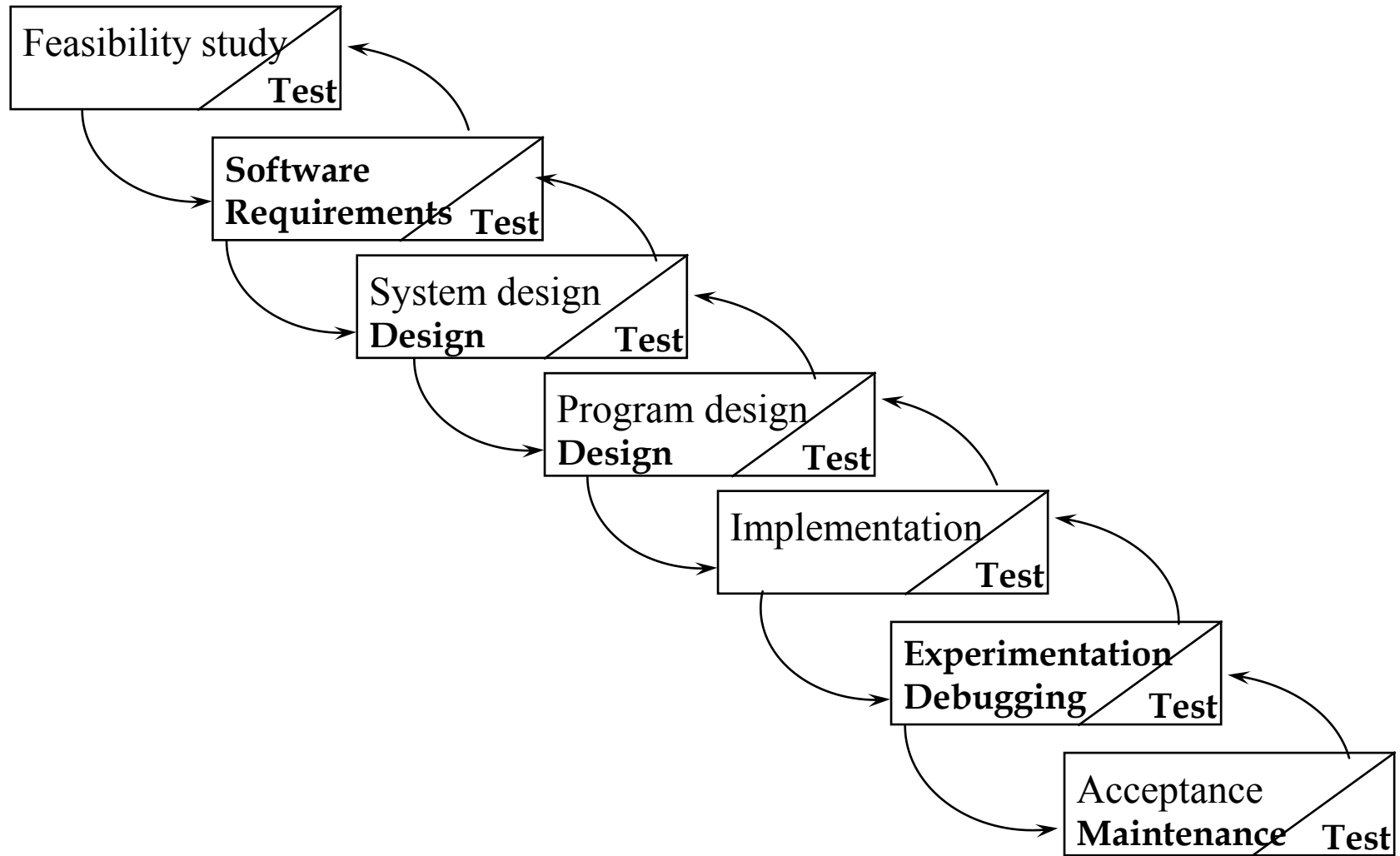
- Nghiên cứu khả thi không thể tạo ngân sách dự trù và lịch biểu mà không có nghiên cứu sơ bộ những yêu cầu và thiết kế thăm dò
- Thiết kế chi tiết hay thực thi thường bộc lộ kẽ hở trong đặc tả yêu cầu.

***Kế hoạch phải được cho phép cho những hình thành từ bước
lập.***

Modified Waterfall Model-1



Modified Waterfall Model-2

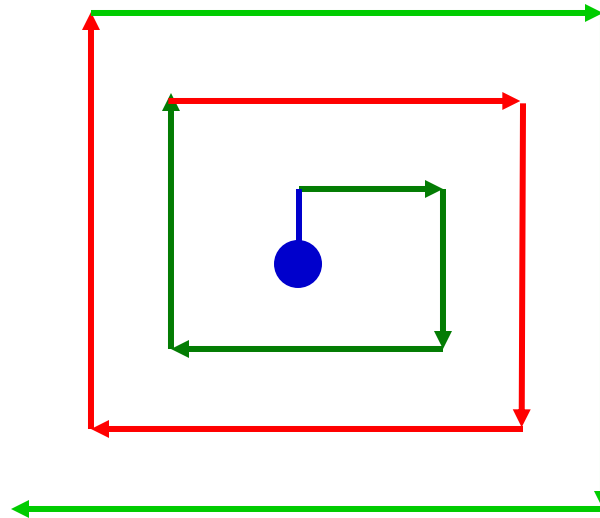


Iterative/spiral Refinement

Concept: Initial implementation for client and user comment, followed by refinement until system is complete

Evaluation

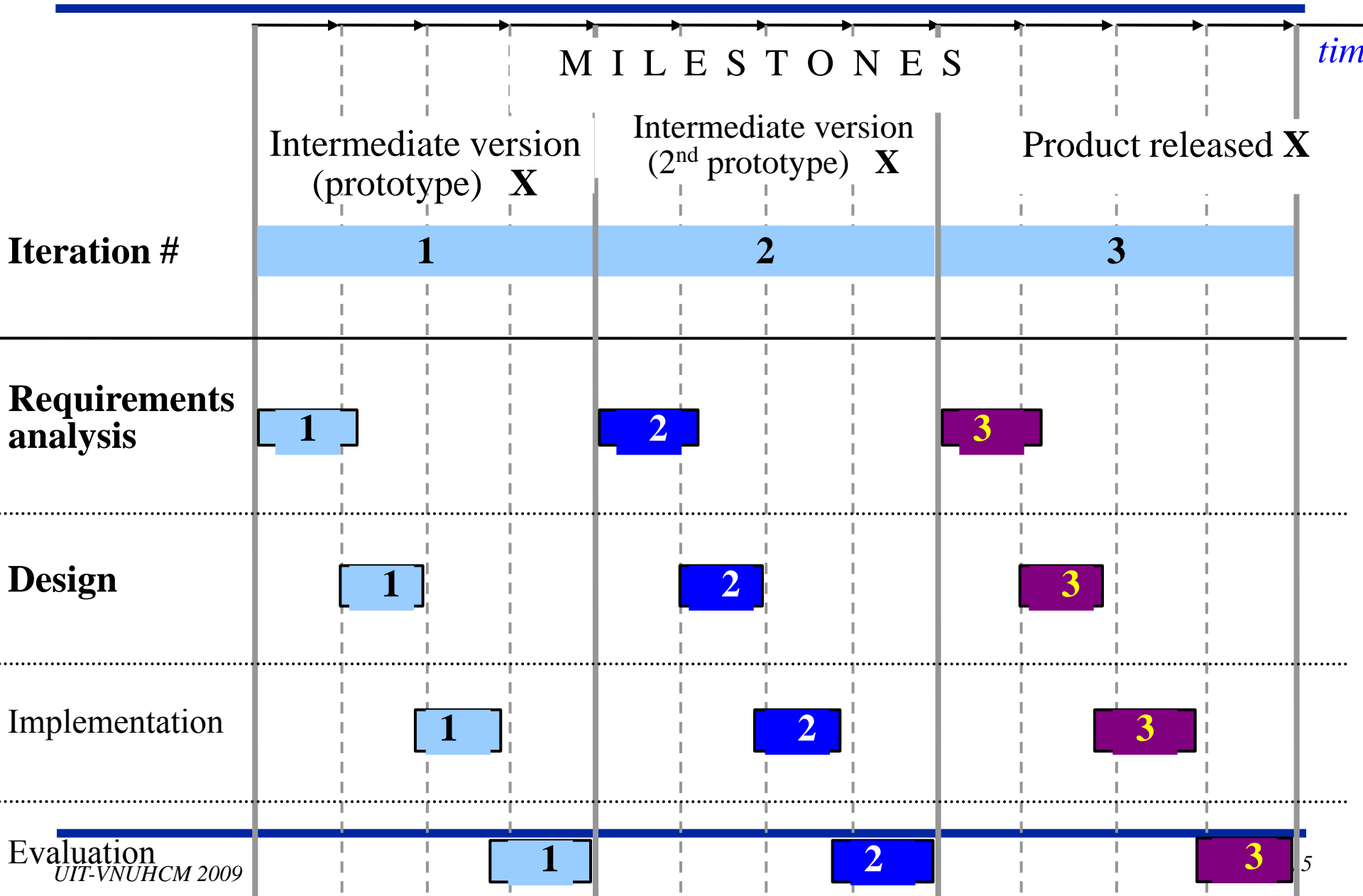
Requirements



Implementation

Design

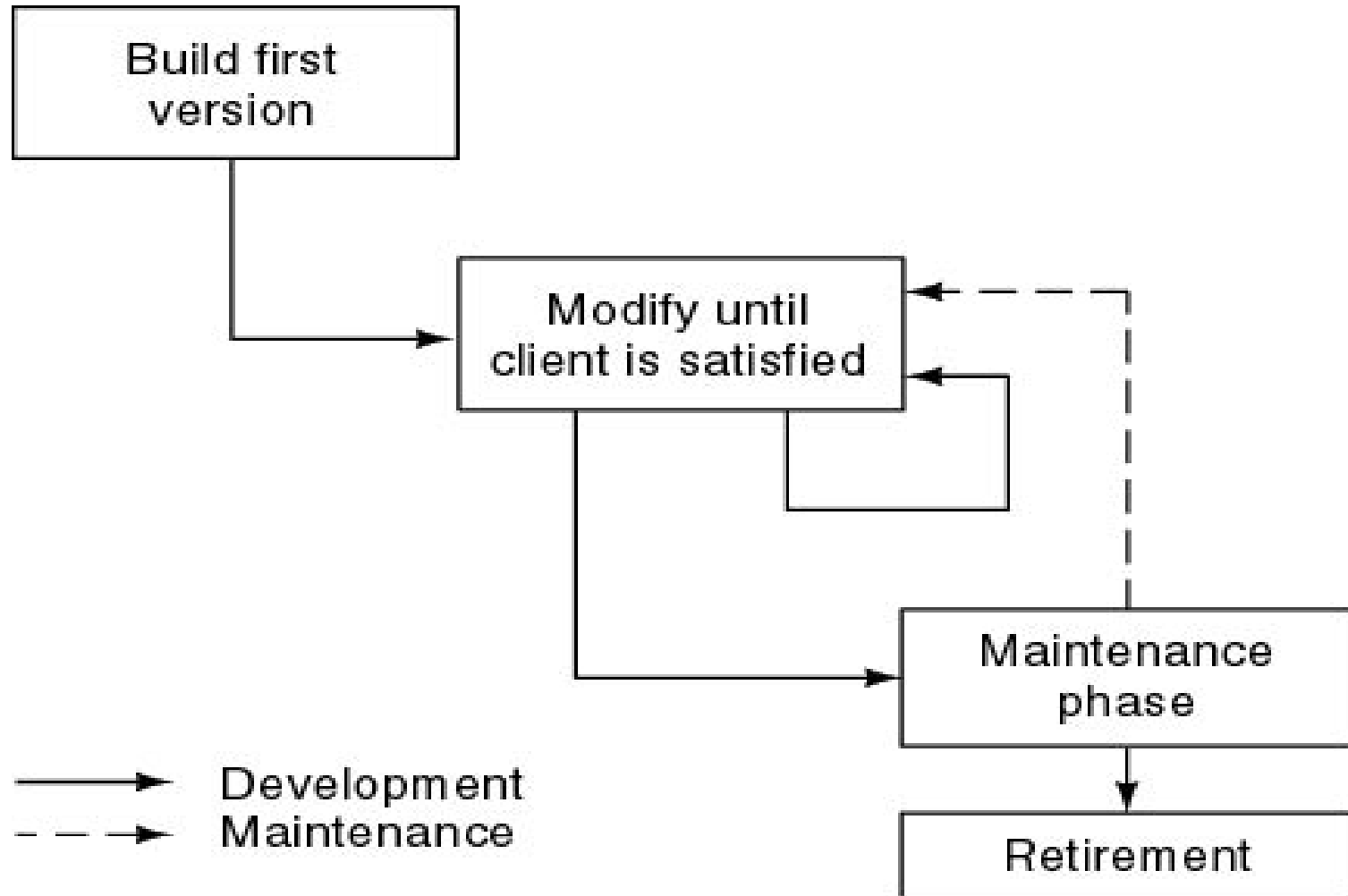
The Spiral Process



Mô hình Life-Cycle khác

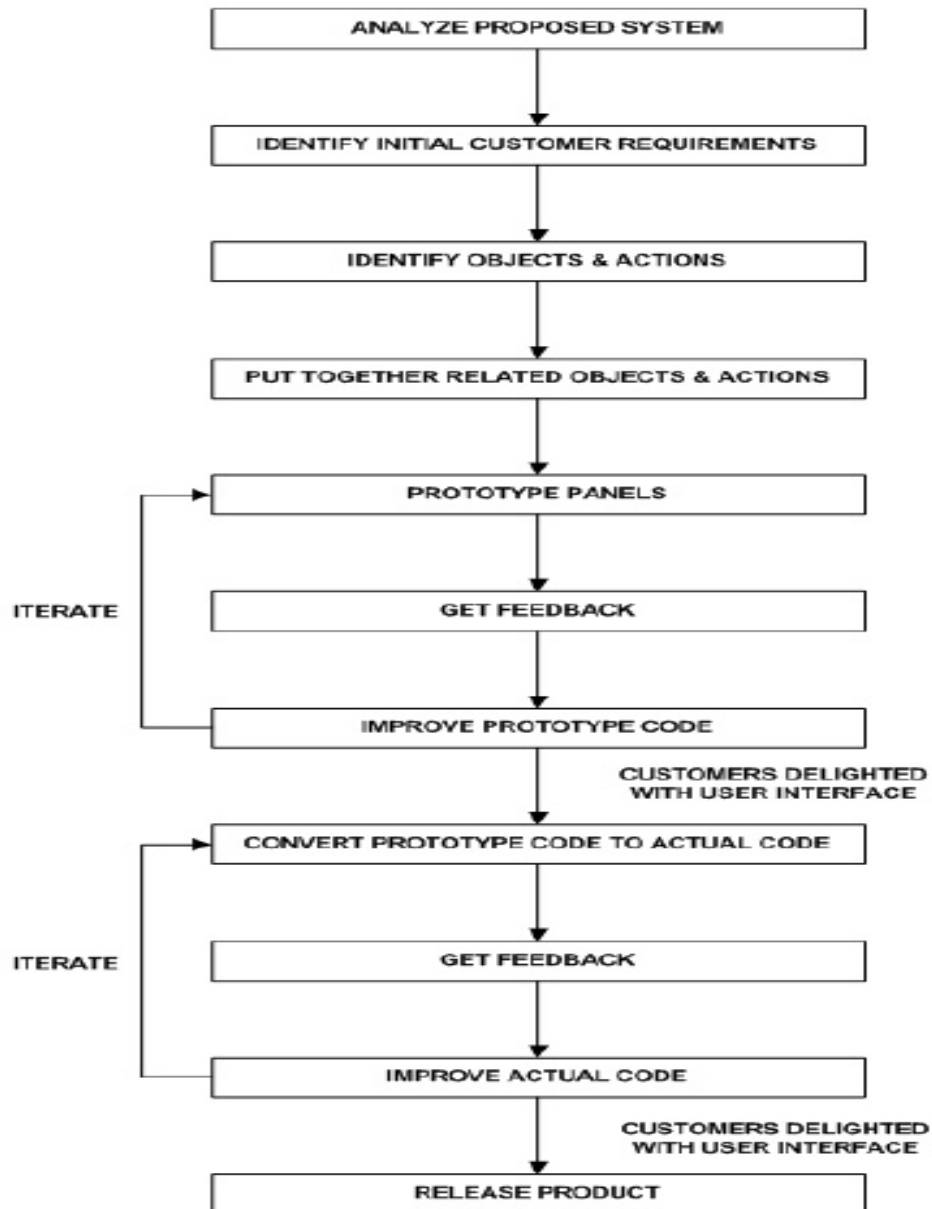
- ☐ **Build-and-fix model**
- ☐ **Rapid prototyping model**
- ☐ **Incremental model**
- ☐ **Extreme programming**
- ☐ **Component-based software engineering**
- ☐ **Mô hình đồng bộ hoá và ổn định
(Synchronize-and-stabilize) model**
- ☐ **Object-oriented life-cycle models**

1-Build and Fix Model



- ❑ Hầu hết phần mềm được phát triển dùng mô hình build-and-fix model. Cơ bản là không có mô hình.
 - Không đặc tả
 - Không thiết kế
- ❑ Mô hình này hoàn toàn không thoả mãn và không nên được chấp nhận.

2-Rapid Prototyping Process



2-Rapid Prototyping Process

- ✈ After you **select a tool and form a team**, you can begin the rapid prototyping process.
 - ✈ **Analyze proposed system** – First, marketing and planning identify a customer need and determine whether the company can develop a product that will profitably meet the need.
 - ✈ **Identify initial customer requirements** – Marketing and planning identify general requirements for the product.
-
- Sau khi chọn công cụ và hình thành nhóm, có bắt đầu qui trình bản mẫu nhanh
 - Phân tích hệ thống đề nghị - Trước tiên, nhận diện thị trường và kế hoạch nhu cầu khách hàng và xác định những gì công ty phát triển phần đạt đến nhu cầu lợi nhuận
 - Nhận diện những yêu cầu khởi động của khách hàng – Tìm hiểu thị trường & kế hoạch nhận diện yêu cầu tổng thể cho sản phẩm

2-Rapid Prototyping Process

□ Tại sao:

Rapid Prototyping decreases development time by allowing corrections to a product to be made early in the process.

Advantages :

- ✈ Increasing number of variants of products.
- ✈ Increasing product complexity.
- ✈ Decreasing product lifetime before obsolescence.
- ✈ Decreasing delivery time.
- ✈ To increase effective communication.
- ✈ To decrease development time.
- ✈ To decrease costly mistakes.
- ✈ To minimize sustaining engineering changes.
- ✈ To extend product lifetime by adding necessary features and eliminating redundant features early in the design.

2-Rapid Prototyping Process

Disadvantages

Some people are of the opinion that rapid prototyping is not effective because, in actual, it fails in replication of the real product or system. It could so happen that some important developmental steps could be omitted to get a quick and cheap working model. This can be one of the greatest disadvantages of rapid prototyping. Another disadvantage of rapid prototyping is one in which many problems are overlooked resulting in endless rectifications and revisions. One more disadvantage of rapid prototyping is that it may not be suitable for large sized applications. Read more at

2-Rapid Prototyping Process

- ✈ **Identify objects and actions** – Next, your prototyping team identifies specific objects (nouns) and actions (verbs) to be used in the product. To perform this step, your team refines the initial, general requirements into specific objects and actions. Your team can also add other objects and actions that are needed.
- ✈ **Put together related objects and actions** – After the team has identified most of the application objects and actions, the next step is to organize the objects and actions in a logical, easy-to-understand way.
- ✈ **Prototype panels** – In this step, your prototyping team works together to prototype a portion of the proposed application user interface. Ideas are discussed, prototyped, commented on, improved, and prototyped again in quick, informal steps.

2-Rapid Prototyping Process

- ✚ **Get feedback** – Once your prototyping team is reasonably satisfied with the prototype, show it to other domain experts, information developers, marketers, planners, usability representatives or anyone else who has knowledge and interest in the product.
- ✚ **Improve prototype** – Use the feedback to improve the prototype. The feedback may trigger new ideas among the prototyping team.
- ✚ **Iterate** – Repeat the “prototype-feedback-improve prototype” cycle as quickly and as frequently as you can. Keep iterating until customers are *delighted* with your prototype user interface. The customers’ evaluations can be measured with questionnaires. portion.
- ✚ **Convert prototype code to actual code** – The next step is to ask product programmers to code the prototype user interface. During this step, they build the actual product. The prototype serves as the product functional specification.
- ✚ **Get feedback** – Since the actual code includes elements of the user interface that you did not prototype, get feedback again. Use informal customer walkthroughs and formal usability tests to get this feedback.

2-Rapid Prototyping Process

- ✍ **Improve actual code** – Based on the customer feedback, the programmers refine the code to make the product easier to learn and use.
- ✍ **Iterate** – Repeat the “actual code-feedback-improve actual code” cycle as quickly and as frequently as you can. Remember, good software is easy to use and hard to design.
- ✍ **Release the product** – Finally, when customers are delighted with the actual user interface, release the product.
 - Cải tiến code thực sự - Dựa trên phản hồi khách hàng, người lập trình viết lại code làm cho sản phẩm dễ học và sử dụng
 - Lặp – Lặp lại “code thật sự- phản hồi – cải tiến code” nhanh và liên tục có thể. Nhớ, phần mềm tốt là dễ dùng và khó để thiết kế
 - Phiên bản sản phẩm – Cuối cùng, khi khách hàng hài lòng với giao diện người dùng thực sự, phiên bản mới cho sản phẩm

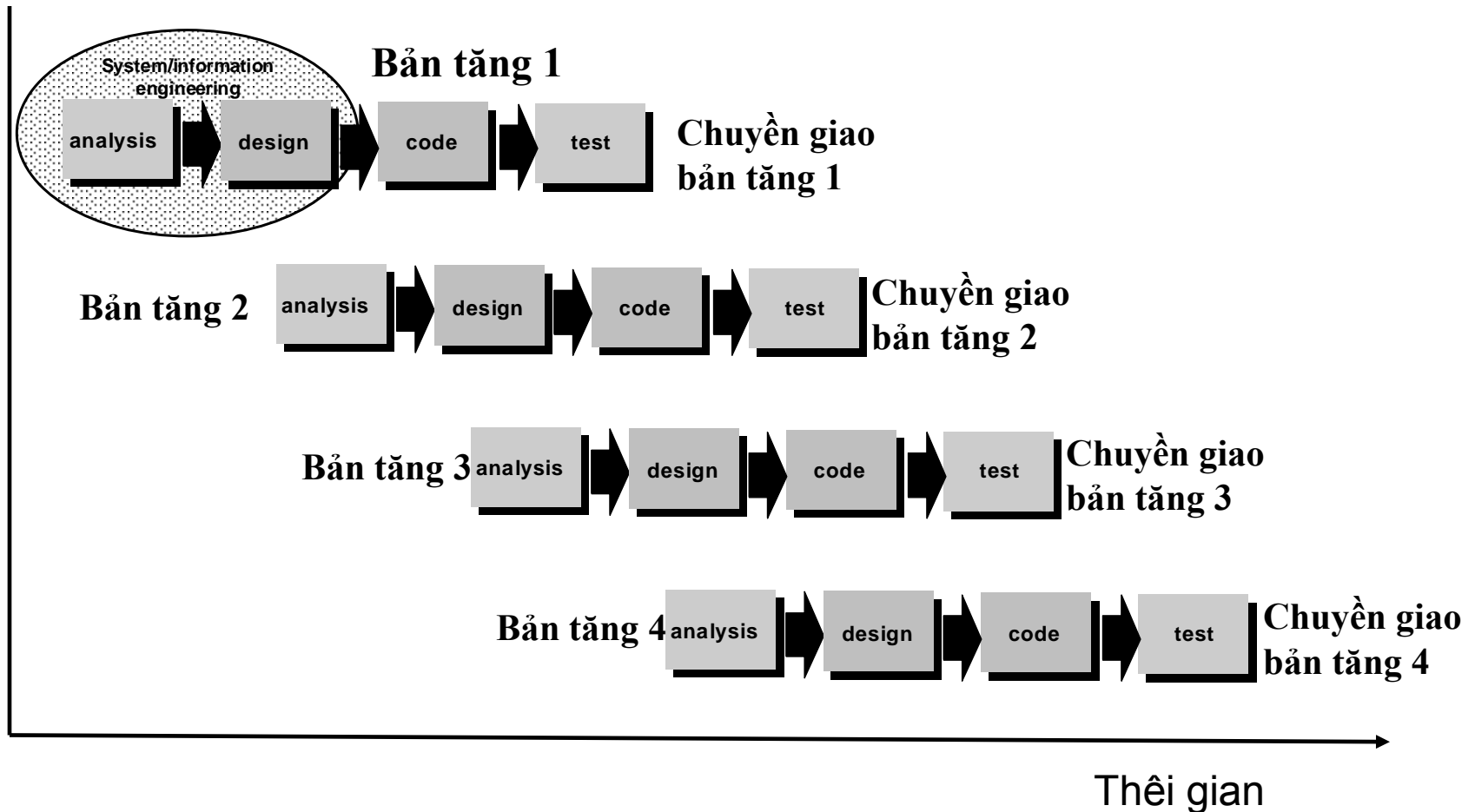
3-Incremental development advantages

- ❑ Customer value can be delivered with each increment so system functionality is available earlier.
- ❑ Early increments act as a prototype to help elicit requirements for later increments.
- ❑ Lower risk of overall project failure.
- ❑ The highest priority system services tend to receive the most testing.

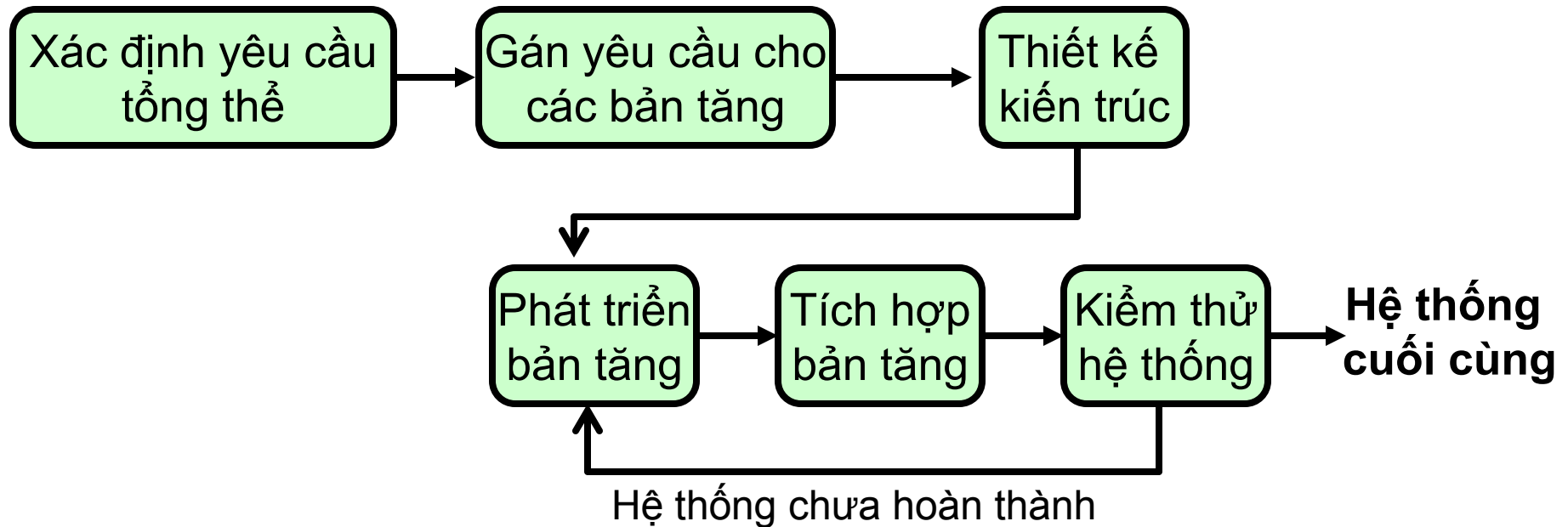
Example – Word processing software

- Basic file management, editing, and document production
- More sophisticated editing features
- Spelling and grammar
- Advanced page layout

MÔ HÌNH PHÁT TRIỂN TĂNG TRƯỞNG



HOẠT ĐỘNG PHÁT TRIỂN TĂNG TRƯỞNG



4-Extreme Programming (XP)

- ❑ Là một điển hình qui trình Agile
- ❑ Phù hợp với môi trường:
 - Nhóm nhỏ
 - Yêu cầu thay đổi nhanh
- ❑ Một số nguyên lý XP đặc nền tảng trên:
 - **Small Releases** – Phần mềm đã phát triển trong những giai đoạn đã được cập nhật thường xuyên
 - **Simple Design** – Hiện thực code cần đạt kết quả khách hàng mong đợi không nhấn mạnh đến version tương lai
 - **Testing** – Hoàn tất qua toàn bộ qui trình phát triển. Kiểm thử là thiết kế đầu tiên trước khi viết phần mềm

5-Component-based software engineering

- ❑ Dựa vào tái dụng có hệ thống khi những hệ thống này được tích hợp từ cấu phần có sẵn hay COTS (Commercial-off-the-shelf) systems.
- ❑ Các giai đoạn qui trình
 - Phân tích thành phần (Component)
 - Cập nhật yêu cầu
 - Thiết kế hệ thống với tái sử dụng
 - Phát triển và tích hợp.
- ❑ Tiếp cận này ngày càng tăng được dùng khi tiêu chuẩn cấu phần hợp trội.

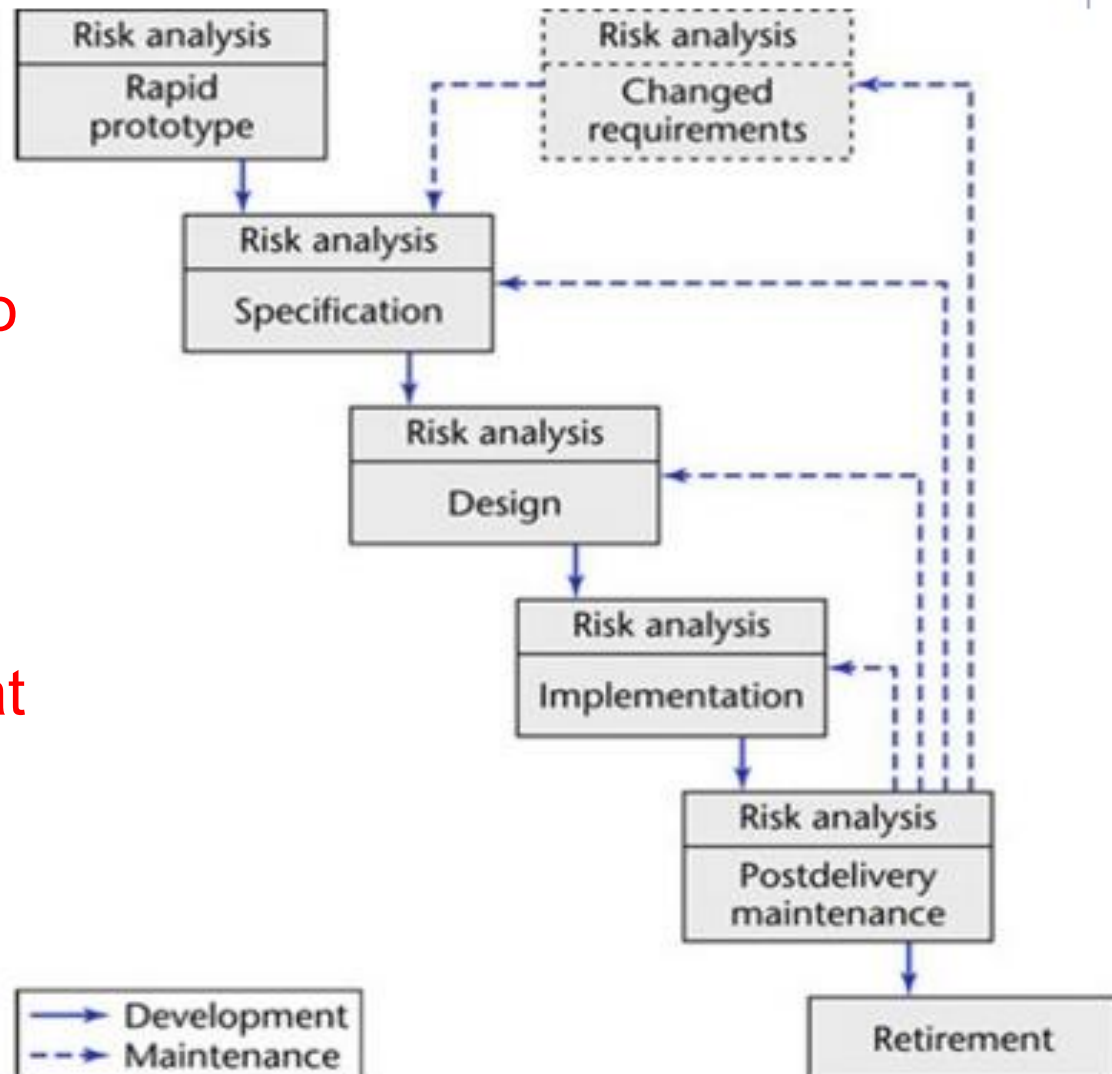
6-Mô hình đồng bộ hoá và ổn định

(Synchronize-and-stabilize) model

- ❑ Microsoft's life-cycle model
- ❑ Phân tích yêu cầu – phỏng vấn khách hàng tiềm năng
- ❑ Viết đặc tả
- ❑ Phân chia project (dự án) thành 3-4 builds
- ❑ Mỗi build thực hiện bởi nhóm nhỏ làm việc song song
- ❑ Cuối mỗi ngày –synchronize (test và debug
- ❑ Cuối mỗi build – stablize (freeze the build)
- ❑ Thành phần luôn làm việc cùng nhau: sớm tham gia quá trình vận hành sản phẩm

Water fall + Phân tích Rủi ro

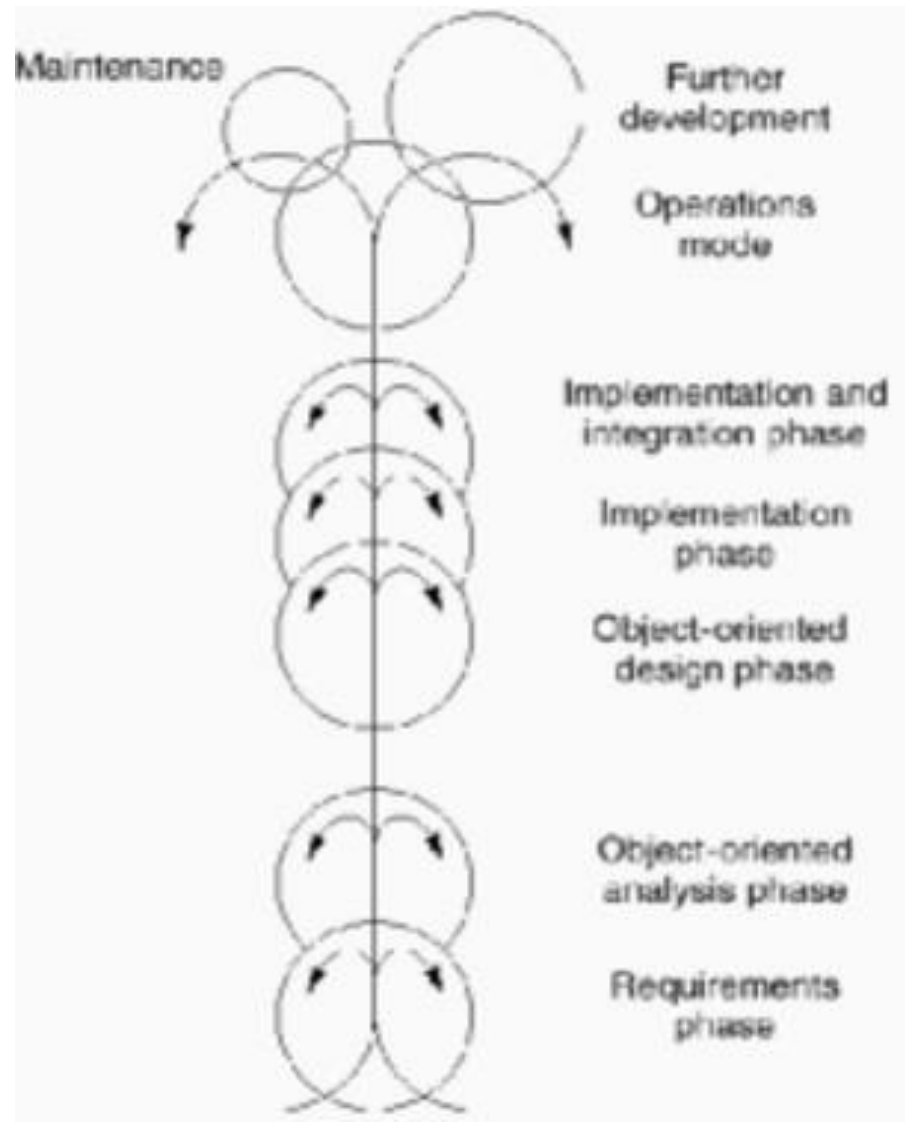
- ✓ Water Fall + phân tích rủi ro trước mỗi giai đoạn
- ✓ Trước khi vào giai đoạn cố gắng kiểm soát rủi ro



7-Object-oriented life-cycle models

□ Fountain Model

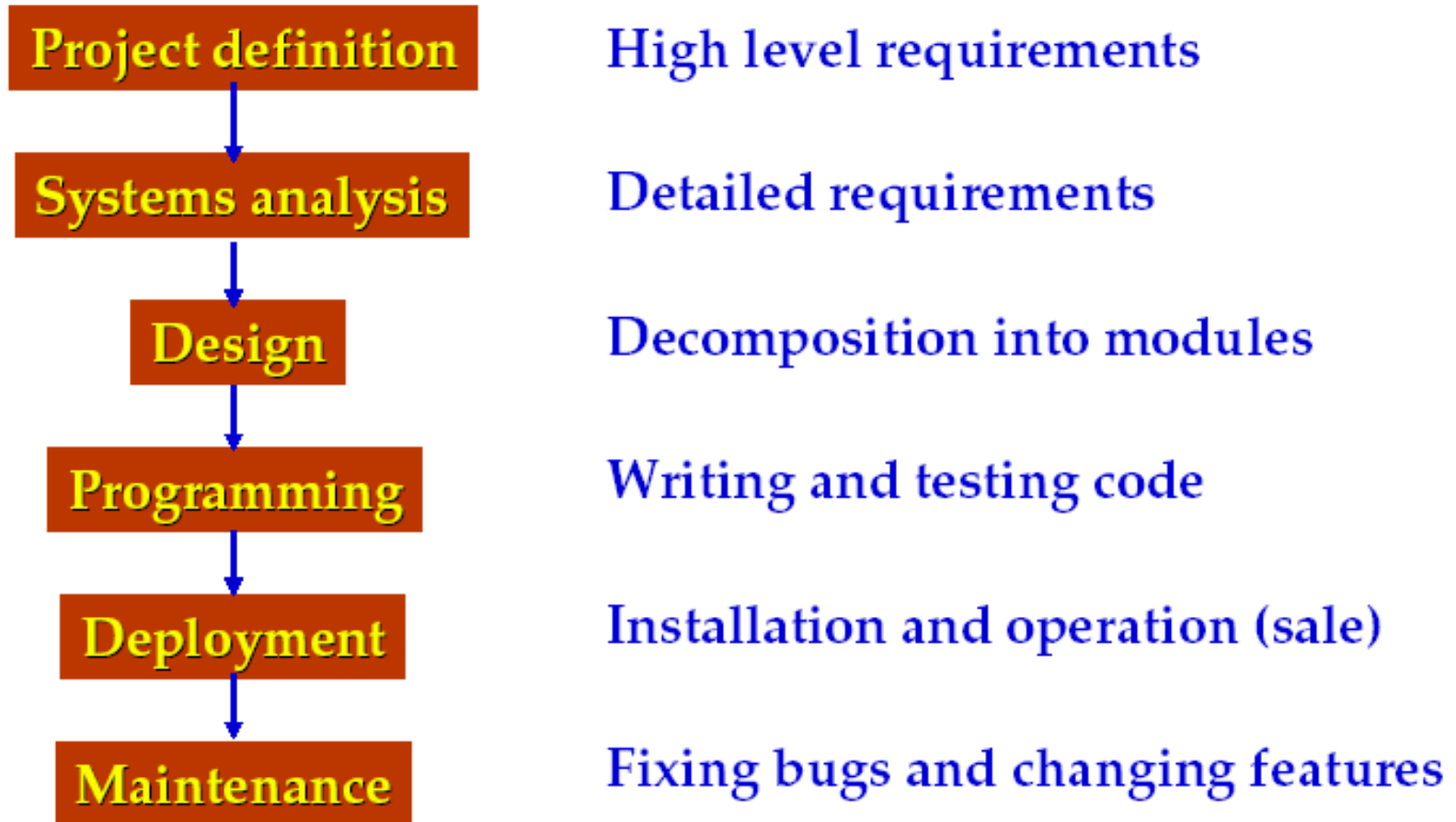
- by Henderson-Sellers and Edward, Communications of the ACM, Sept. 1990
- Vòng tròn thể hiện giai đoạn khác trùng lặp
- Mũi tên thể hiện bước lặp của mỗi giai đoạn
- Vòng tròn bảo trì nhỏ hơn, để biểu trưng giảm nỗ lực bảo trì khi thành phần hướng đối tượng được sử dụng



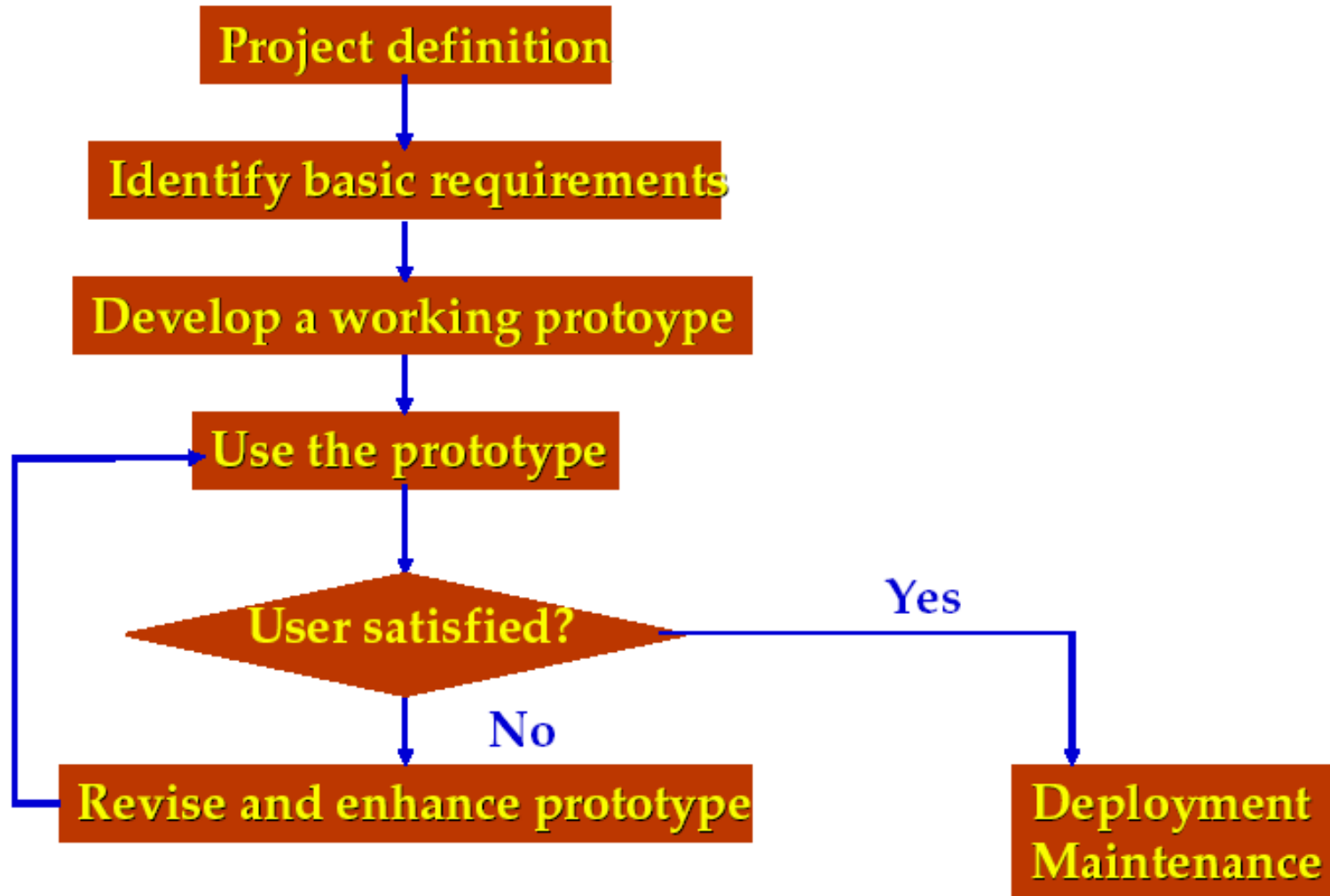
Các tiếp cận để phát triển phần mềm

- ☐ Traditional systems development life cycle
 - ☐ Prototyping
 - ☐ Packaged software
 - ☐ End-user development
 - ☐ Outsourcing
 - ☐ Open source
- **Thảo luận Thuận lợi và Bất lợi các tiếp cận trên**

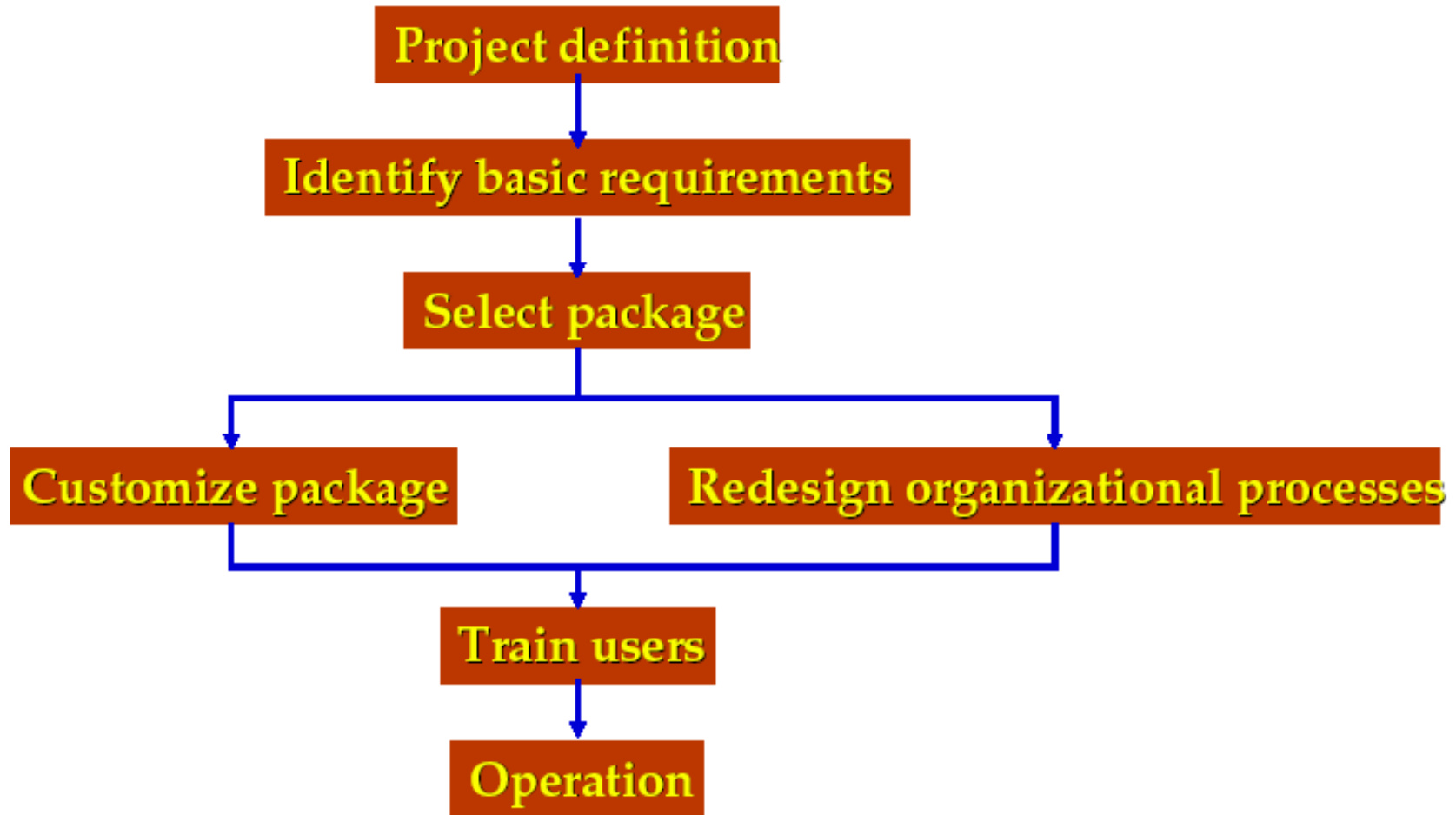
Traditional systems development life cycle



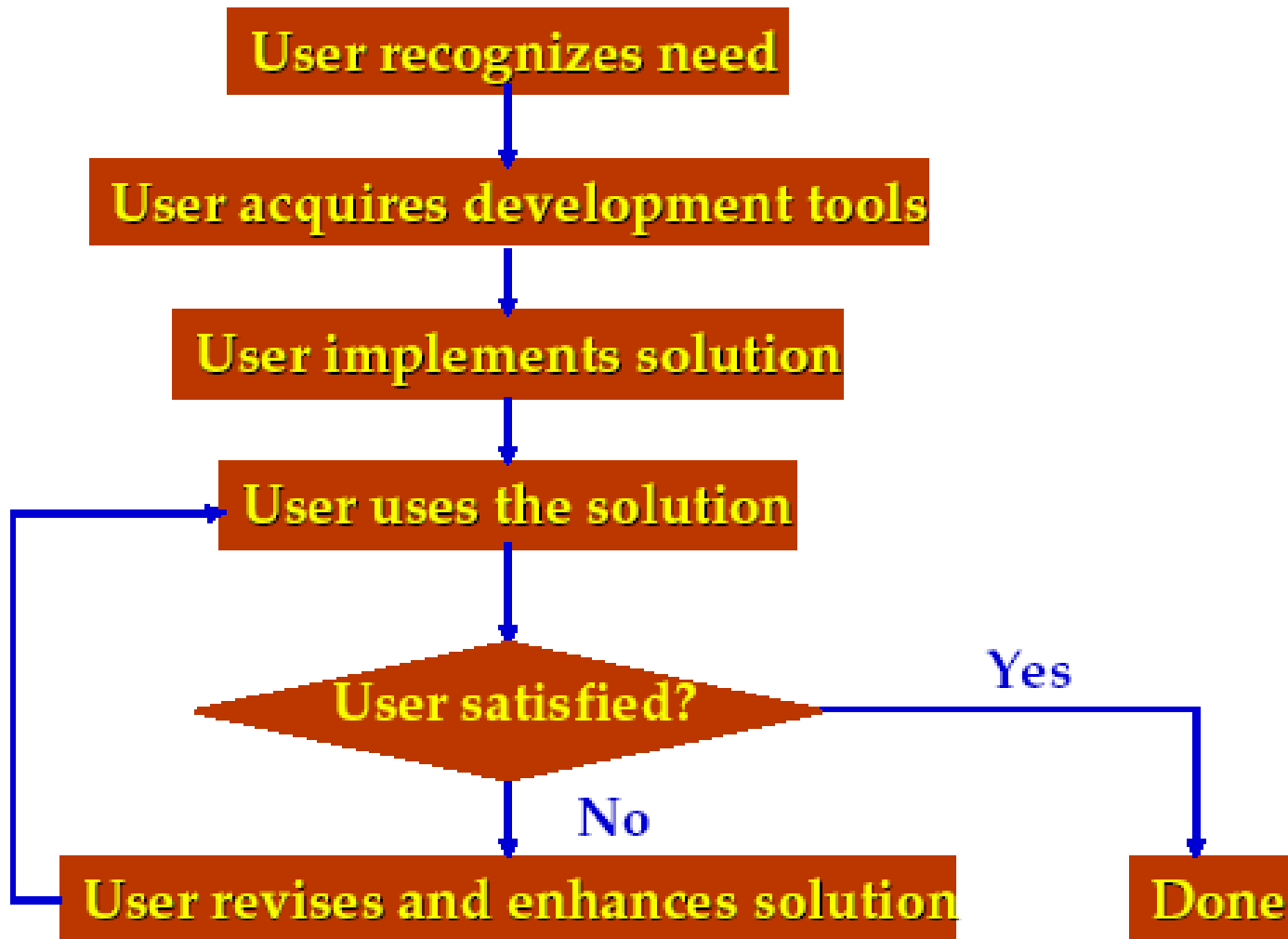
Prototyping



Packaged software



End-User Development



Open-source

Someone develops working prototype

Use the prototype

**Some
developer wants
to do more?**

No

Yes

Revise and enhance prototype

Project ends

Điểm mạnh & yếu Life-Cycle Model

Life-Cycle Model	Strengths	Weaknesses
Evolution-tree model (Section 2.2)	Closely models real-world software production Equivalent to the iterative-and-incremental model	
Iterative-and-incremental model (Section 2.5)	Closely models real-world software production Underlies the Unified Process	
Code-and-fix model (Section 2.9.1)	Fine for short programs that require no maintenance	Totally unsatisfactory for nontrivial programs
Waterfall model (Section 2.9.2)	Disciplined approach Document driven	Delivered product may not meet client's needs
Rapid-prototyping model (Section 2.9.3)	Ensures that the delivered product meets the client's needs	Not yet proven beyond all doubt
Extreme programming (Section 2.9.4)	Works well when the client's requirements are vague	Appears to work on only small-scale projects
Synchronize-and-stabilize model (Section 2.9.5)	Future users' needs are met Ensures that components can be successfully integrated	Has not been widely used other than at Microsoft
Spiral model (Section 2.9.6)	Risk driven	Can be used for only large-scale, in-house products Developers have to be competent in risk analysis and risk resolution

Tổng kết life cycle's model

- ❑ Mỗi life-cycle model khác nhau có thể mạnh và điểm yếu
- ❑ Điều kiện để quyết định chọn model bao gồm:
 - Tổ chức
 - Cách quản lý của tổ chức
 - Kỹ năng của nhân viên
 - Bản chất của sản phẩm
- ❑ Giải pháp tốt nhất:
 - “Mix-and-match” life cycle model
 - Nhưng, phải lên kế hoạch

Maintenance Process (extended to real life)

- ❑ Processing requests before starting to work on them, like:
 - ◆ Capturing maintenance requests
 - ◆ Investigating those requests – like testing to verify a bug and decide how hard to fix it
 - ◆ Deciding the time / cost to do, getting customer ok
 - ◆ Prioritizing requests – versus other requests!
 - ◆ Assigning to a sub-team to do
- ❑ Coding and documenting (as per standards)
- ❑ Testing with various configurations, other legacy code issues
- ❑ Deciding to send it out (special, or in which sub-release)

Ngữ cảnh người bảo trì phần mềm

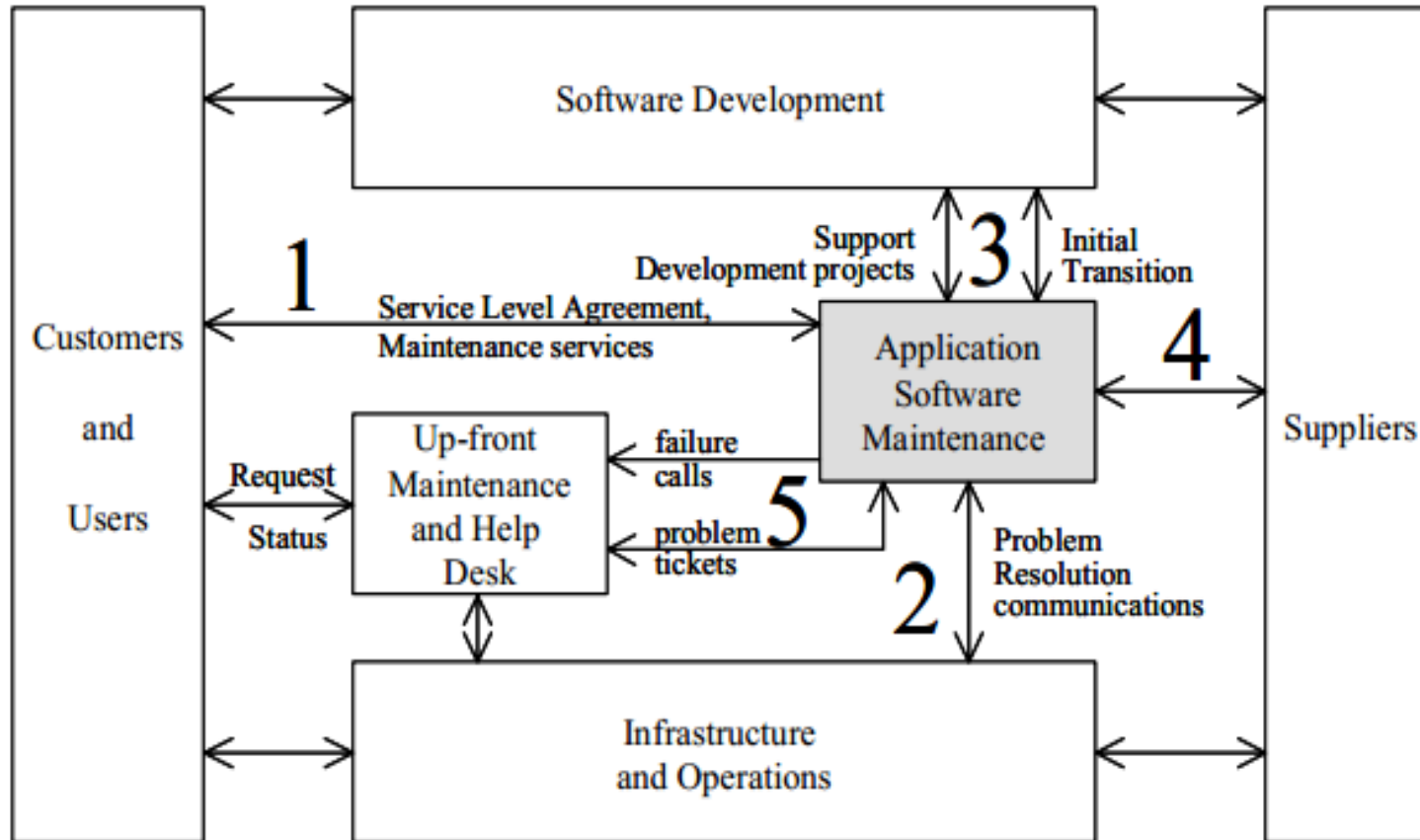
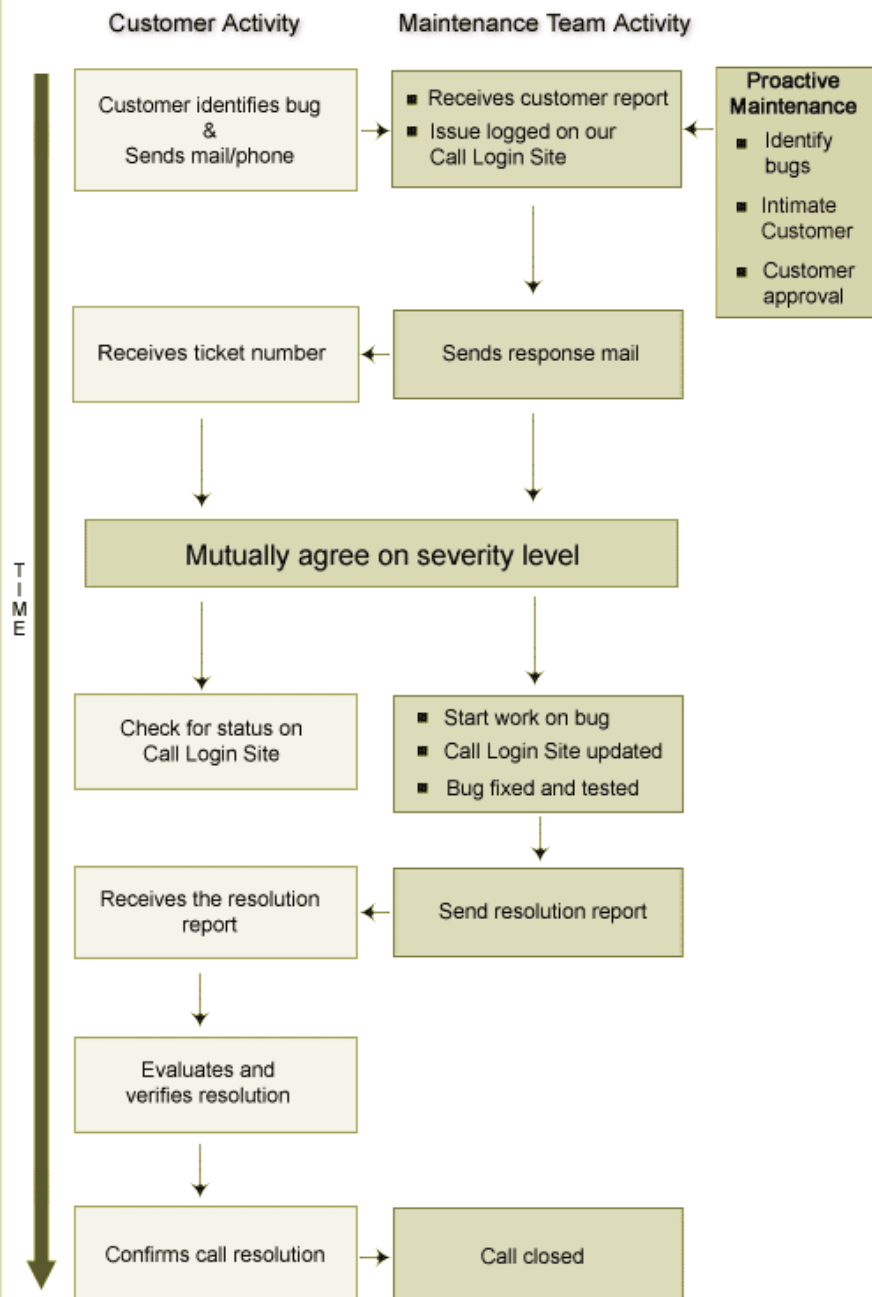


Figure 1. Software Maintainers Context Diagram.

1: Customers & Users 2:Operation Department 3: Developers 4: Suppliers
5: Help Desk



Một ví dụ ...

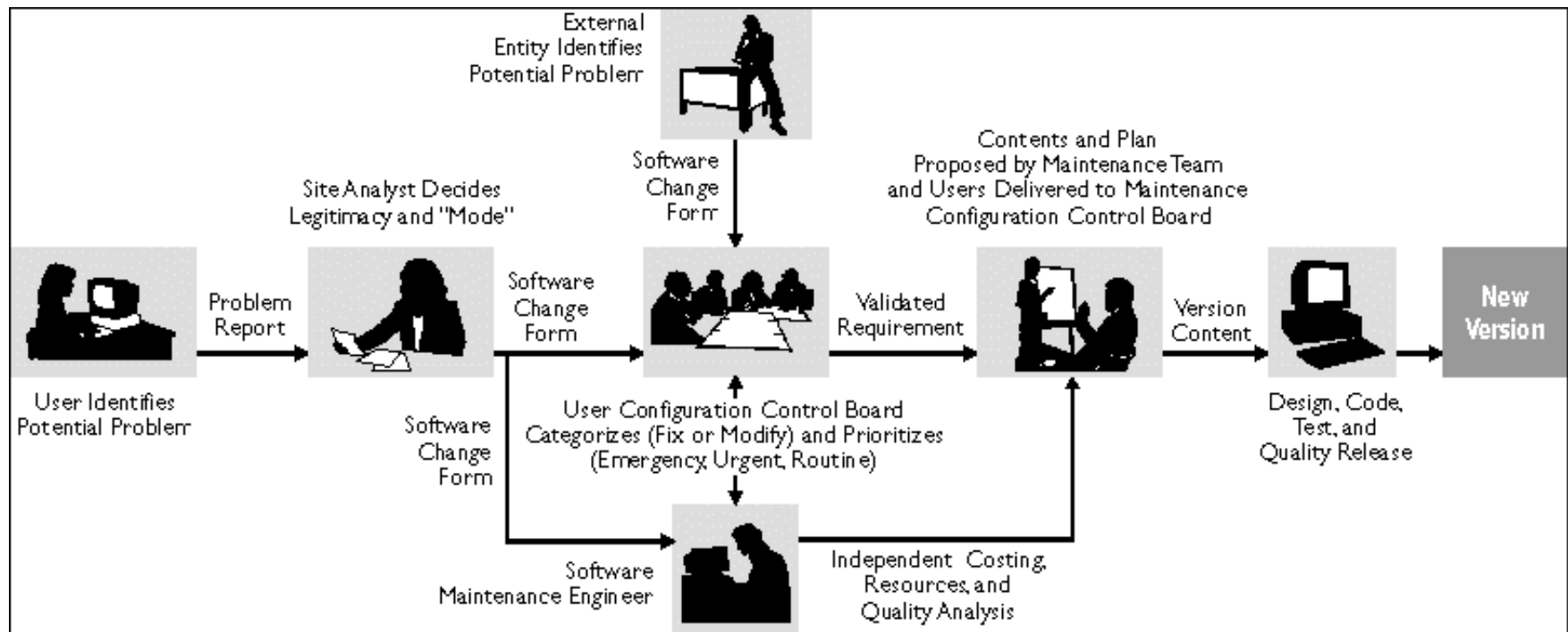
- Chú ý đến số lượng “pre-fixing” và những hoạt động truyền thông khác!

From

http://www.indiawebdevelopers.com/CustomerSupport/maintenance_process.asp.

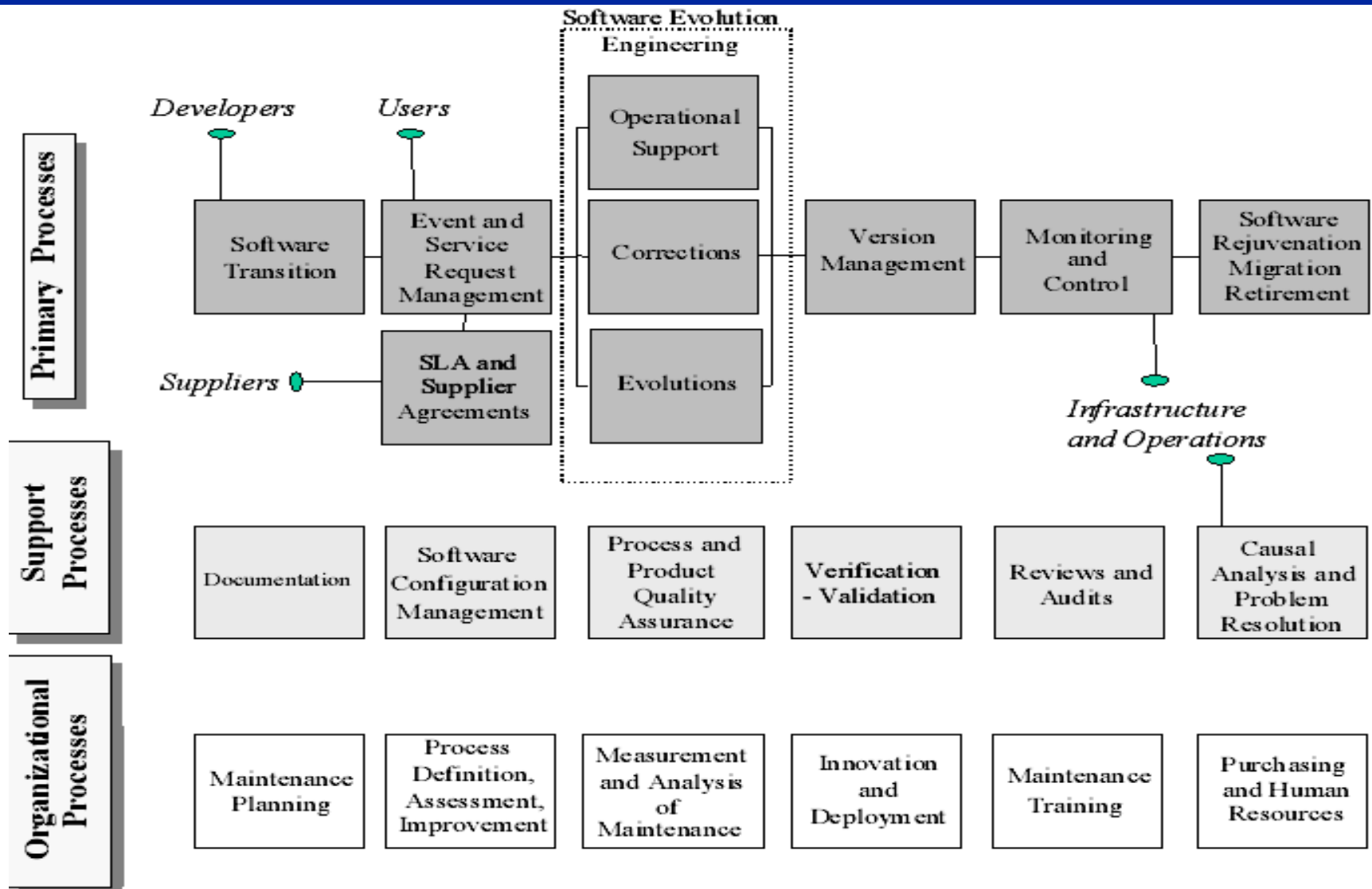
Ví dụ khác ...

□ Như trên...



From <http://www.stsc.hill.af.mil/crosstalk/1997/07/stark1.gif>.

Phân lớp qui trình then chốt (KEY) bảo trì phần mềm



Basic Strategies for Software Enhancement (one more review topic)

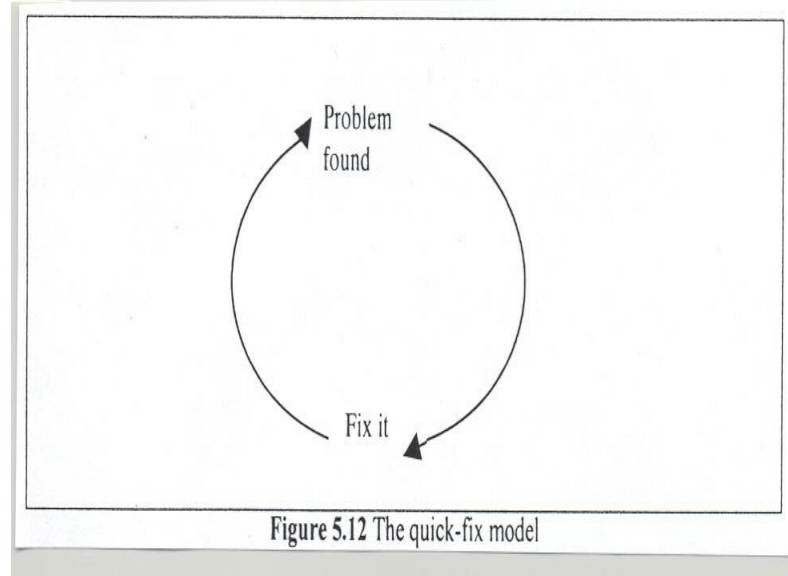
- ☐ New versions coming out at regular intervals
- ☐ Ongoing (technical) support – between or instead of releases

3.2 CÁC MÔ HÌNH BẢO TRÌ PHẦN MỀM

- ❑ Mô hình Quick-Fix
- ❑ Mô hình Boehm
- ❑ Mô hình Osborne
- ❑ Iterative Enhancement Model
- ❑ Mô hình hướng sử dụng lại (Reuse-Oriented)

Quick-Fix Model

- ❑ Thuận lợi
 - Nhanh
 - Có thể hữu ích cho dự án nhỏ
- ❑ Không thuận lợi
 - Nhỏ hay không sưu liệu
 - Bất kỳ thiết kế trở nên ít hiệu quả vượt quá thời gian



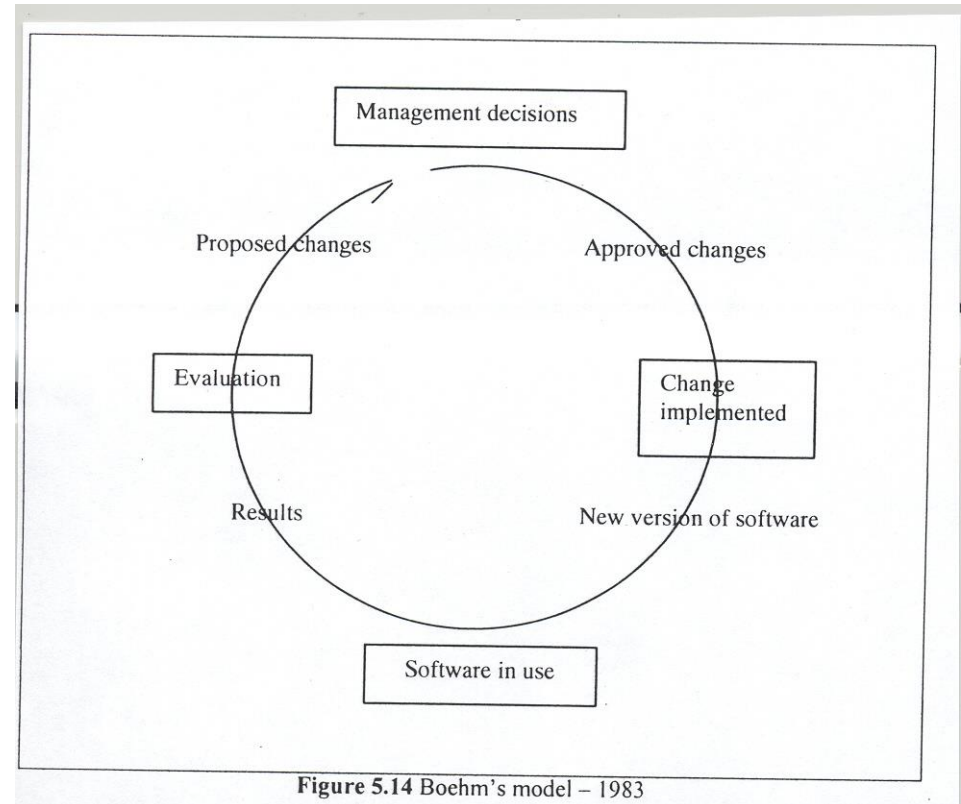
- ❑ It is a 'firefighting' approach, chờ cho vấn đề xảy ra và sau đó cố gắng fix nó nhanh như có thể
- ❑ Việc sửa lỗi có thể được hoàn tất mà không có phân tích chi tiết những tác động về lâu dài
- ❑ Tại sao nó vẫn được sử dụng?

Quick-fix model

- ❑ Trong môi trường phù hợp nó có thể làm việc hiệu quả
- ❑ Ví dụ:
 - Một hệ thống được phát triển và bảo trì bởi 1 người. Người này có thể hiểu hệ thống đủ để quản lý mà không cần sự liệu
 - Áp lực về hạn cuối và nguồn lực
- ❑ chiến lược để thích ứng là gắn kết kỹ thuật quick-fix với kỹ thuật khác

Boehm's model

- ❑ Thuận lợi
 - Qui trình được kiểm soát
 - Nhấn mạnh vào phản hồi
- ❑ Không thuận lợi
 - Thấp hơn quick-fix
- ❑ Dựa trên mô hình kinh tế và nguyên tắc.
- ❑ Qui trình bảo trì dẫn xuất từ quyết định của người quản lý dựa trên cân bằng mục tiêu và ràng buộc



Osborne's

❑ Thuận lợi

- Liên quan đến tất cả giai đoạn chu trình sống
- Sửa lỗi được cập nhật

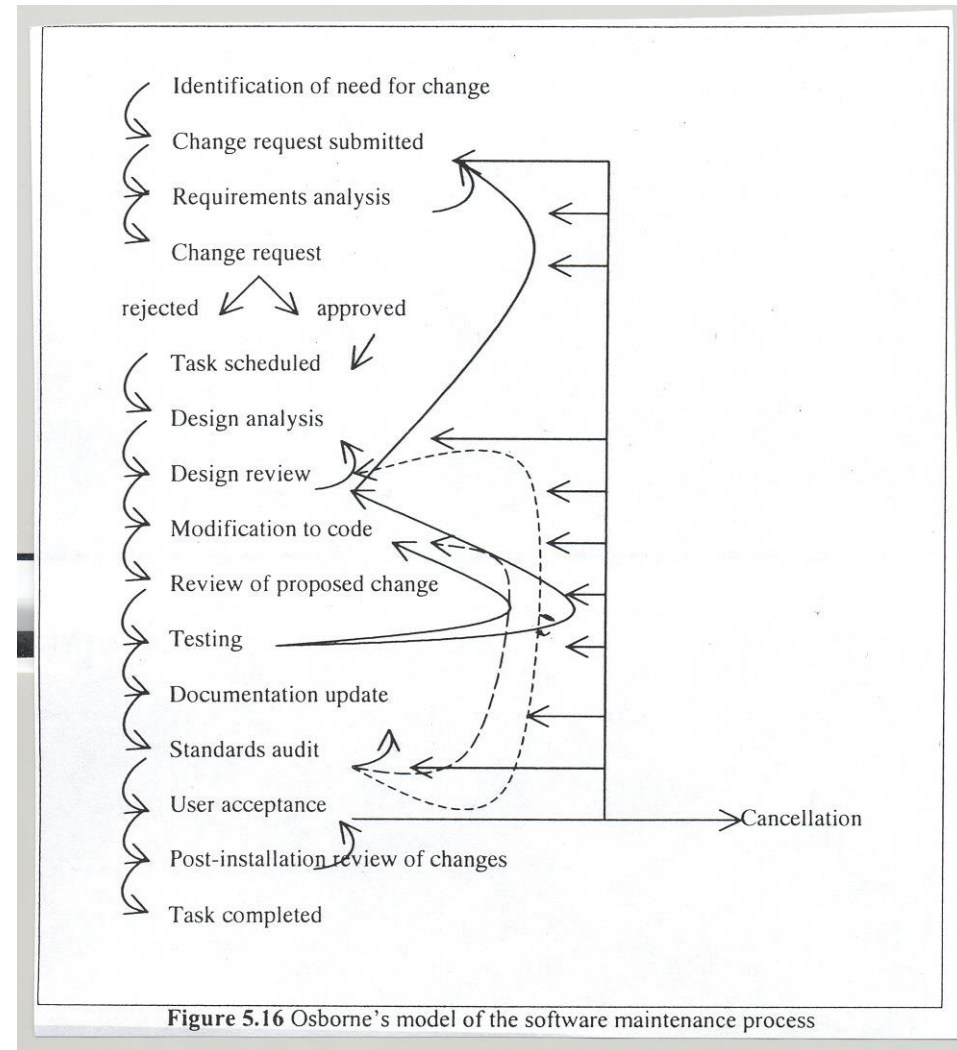
❑ Không thuận lợi

- Phức tạp
- Nhiều công sức chi phí

❑ đối phó trực tiếp với thực tế của môi trường bảo trì.

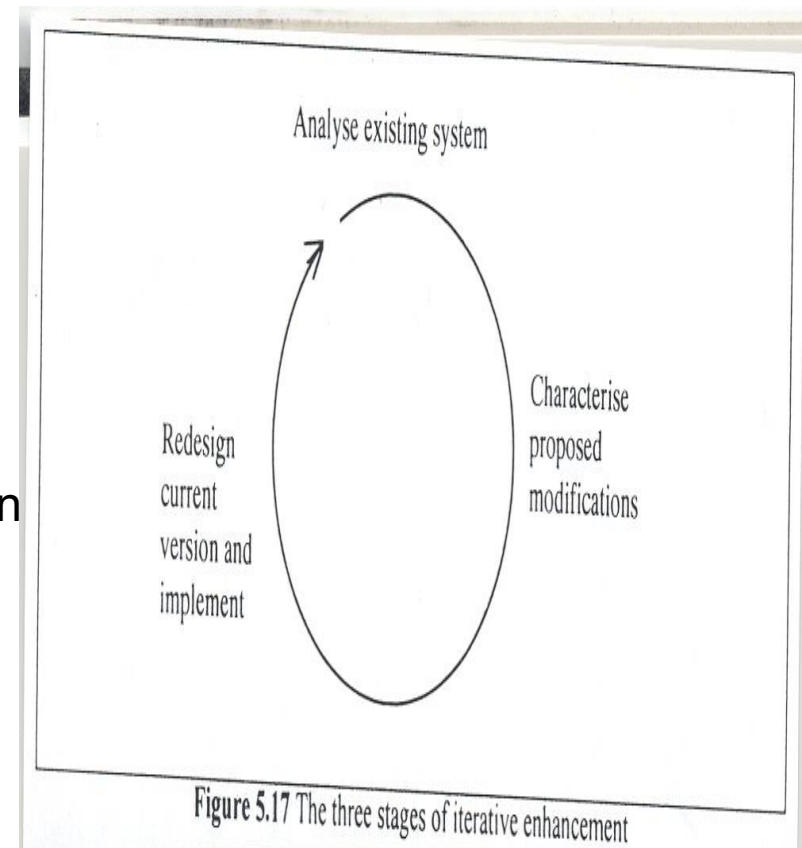
❑ Mô hình khác hướng đến giả định khía cạnh của tình huống lý tưởng

❑ Mô hình bảo trì được xử lý như lặp tiếp diễn của vòng đời phần mềm



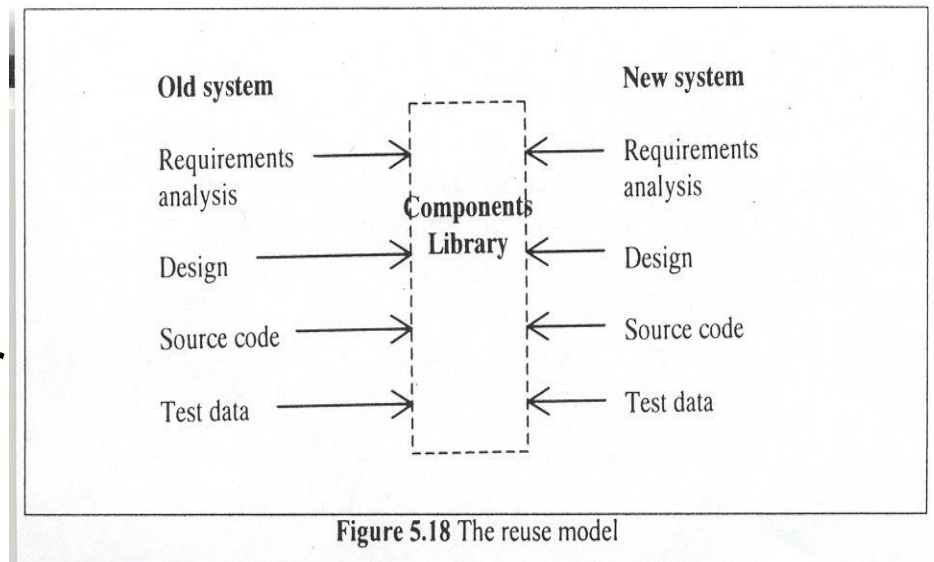
Iterative Enhancement Model

- ❑ Thuận lợi
 - Khá đơn giản
 - Cho phép phân tích
- ❑ Không thuận lợi
 - Những quyết định bao gồm không rõ ràng
 - Appears informally to be on a tilt!
- ❑ Thực hiện thay đổi đối với hệ thống phần mềm qua thời gian sống của nó là qui trình lặp và liên quan đến cải thiện hệ thống theo bước lặp.



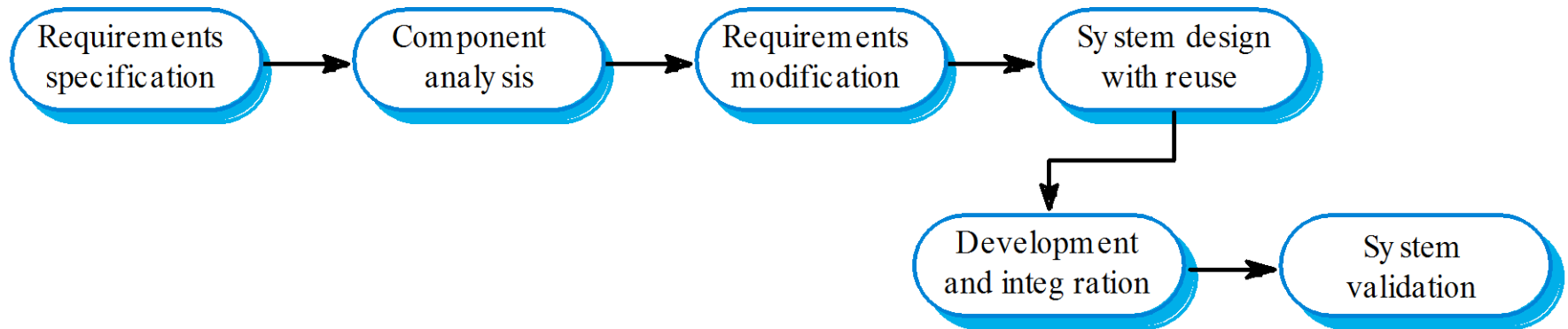
Reuse Model

- ❑ Thuận lợi
 - Có thể dùng các thành phần từ các dự án khác
 - Code is modular
- ❑ Không thuận lợi
 - Overhead in designing for reuse



- ❑ Nhận diện phần của hệ thống cũ là những ứng viên có thể tái sử dụng,
- ❑ Hiểu phần hệ thống này,
- ❑ Thay đổi phần hệ thống cũ phù hợp với yêu cầu,
- ❑ Tích hợp những phần được thay đổi vào trong hệ thống mới.

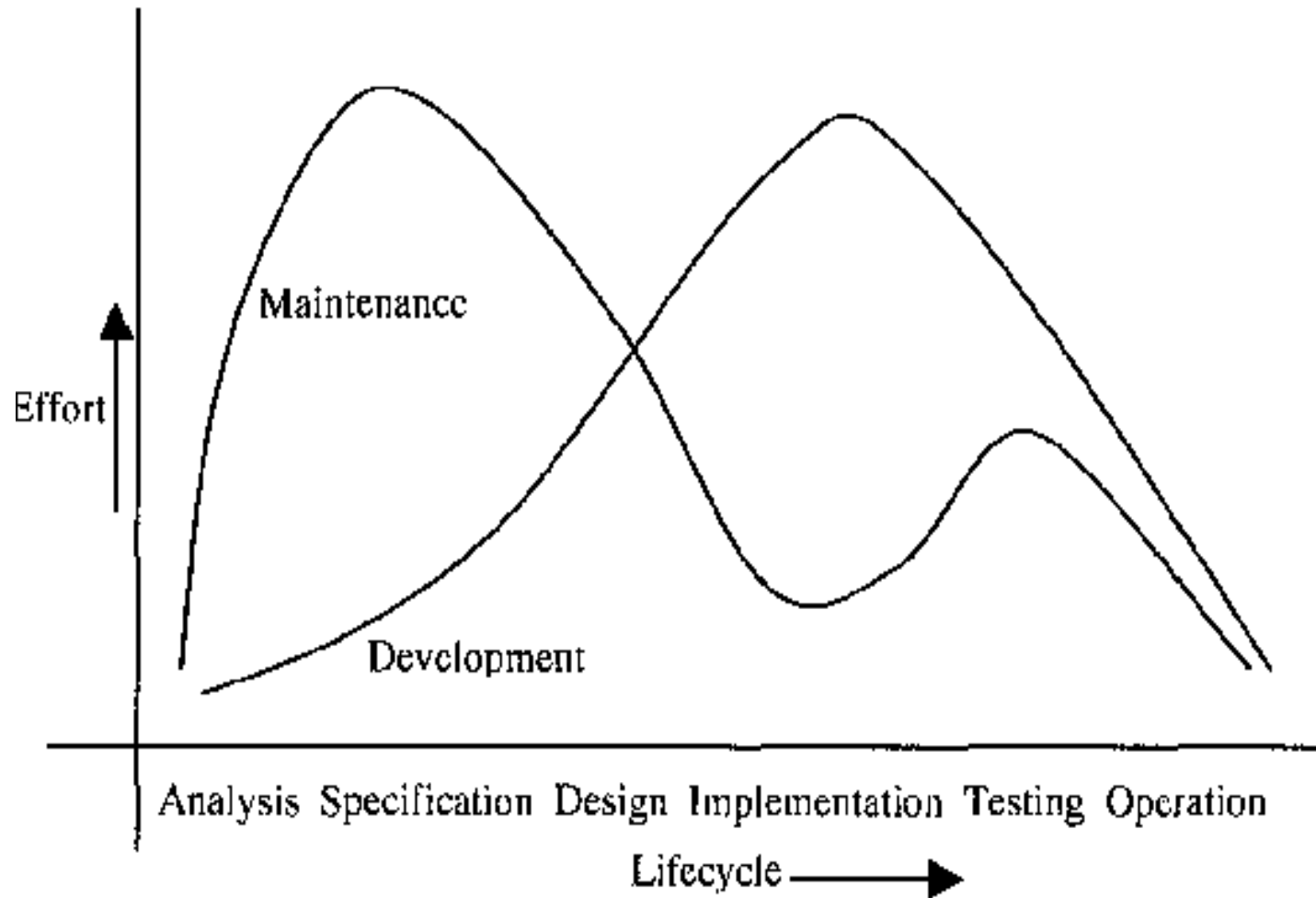
Reuse-oriented development



3.3 KHI THỰC HIỆN THAY ĐỔI

- ☐ Tăng trưởng qui trình
- ☐ Mô hình tăng trưởng CMM (Capability Maturity Model) cho phần mềm
- ☐ Cơ sở kinh nghiệm phần mềm

So sánh nỗ lực bảo trì & phát triển



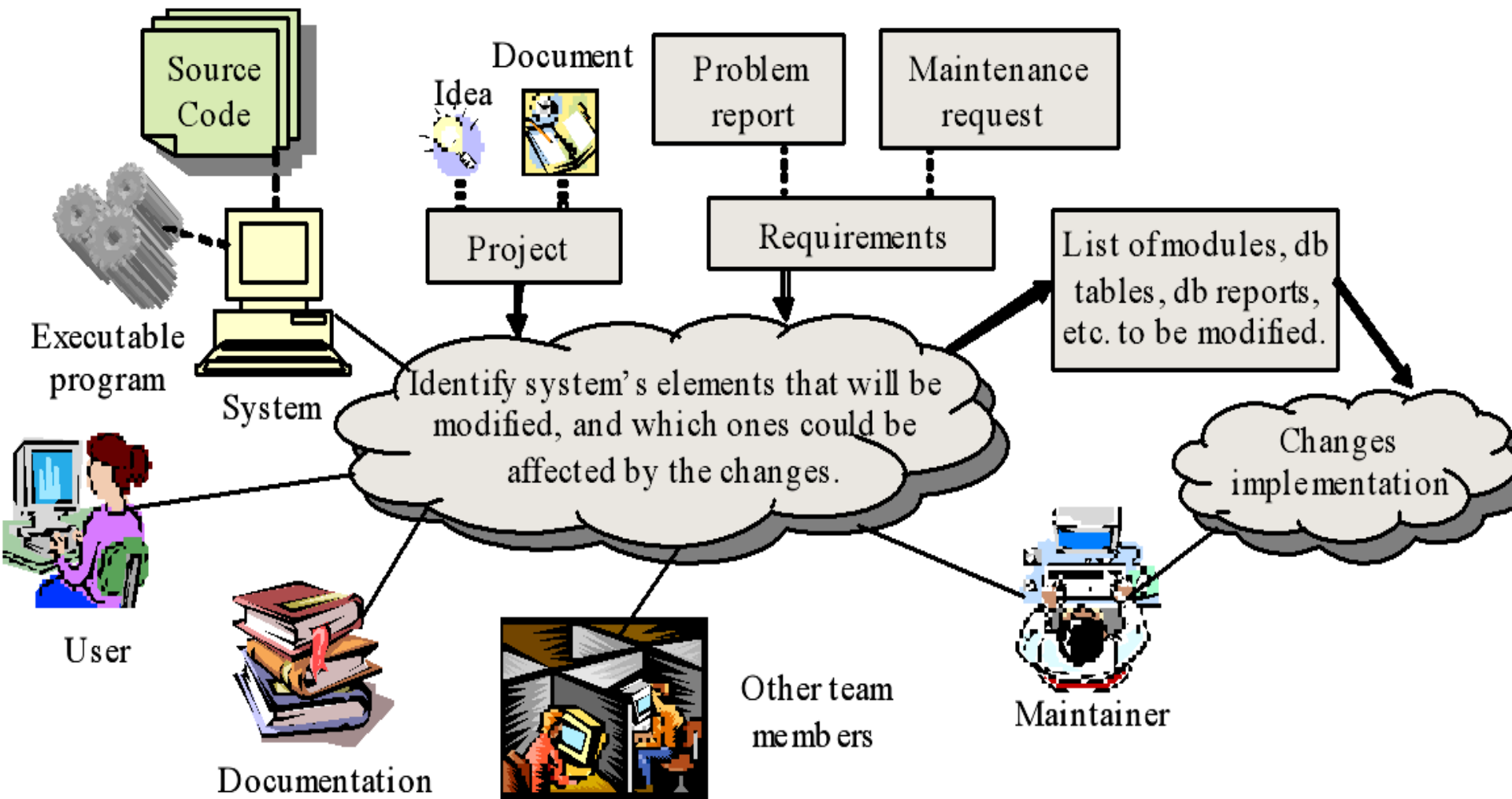
QUI TRÌNH – Cải tiến nâng cao chất lượng

- ❑ Quy trình khung là cơ sở để cải tiến tiến trình nâng cao chất lượng, năng suất
- ❑ Quy trình khung phổ biến (Các chuẩn)
 - ISO
 - CMM (Capability Maturity Model)
 - CMMI (Capability Maturity Model Integration): 5 level
 - ✓ Initial (chaotic, ad hoc, heroic) the starting point for use of a new process.
 - ✓ Repeatable (project management, process discipline) the process is used repeatedly.
 - ✓ Defined (institutionalized) the process is defined/confirmed as a standard business process.
 - ✓ Managed (quantified) process management and measurement takes place.
 - ✓ Optimising (process improvement) process management includes deliberate process optimization/improvement.

Cơ sở tri thức kinh nghiệm

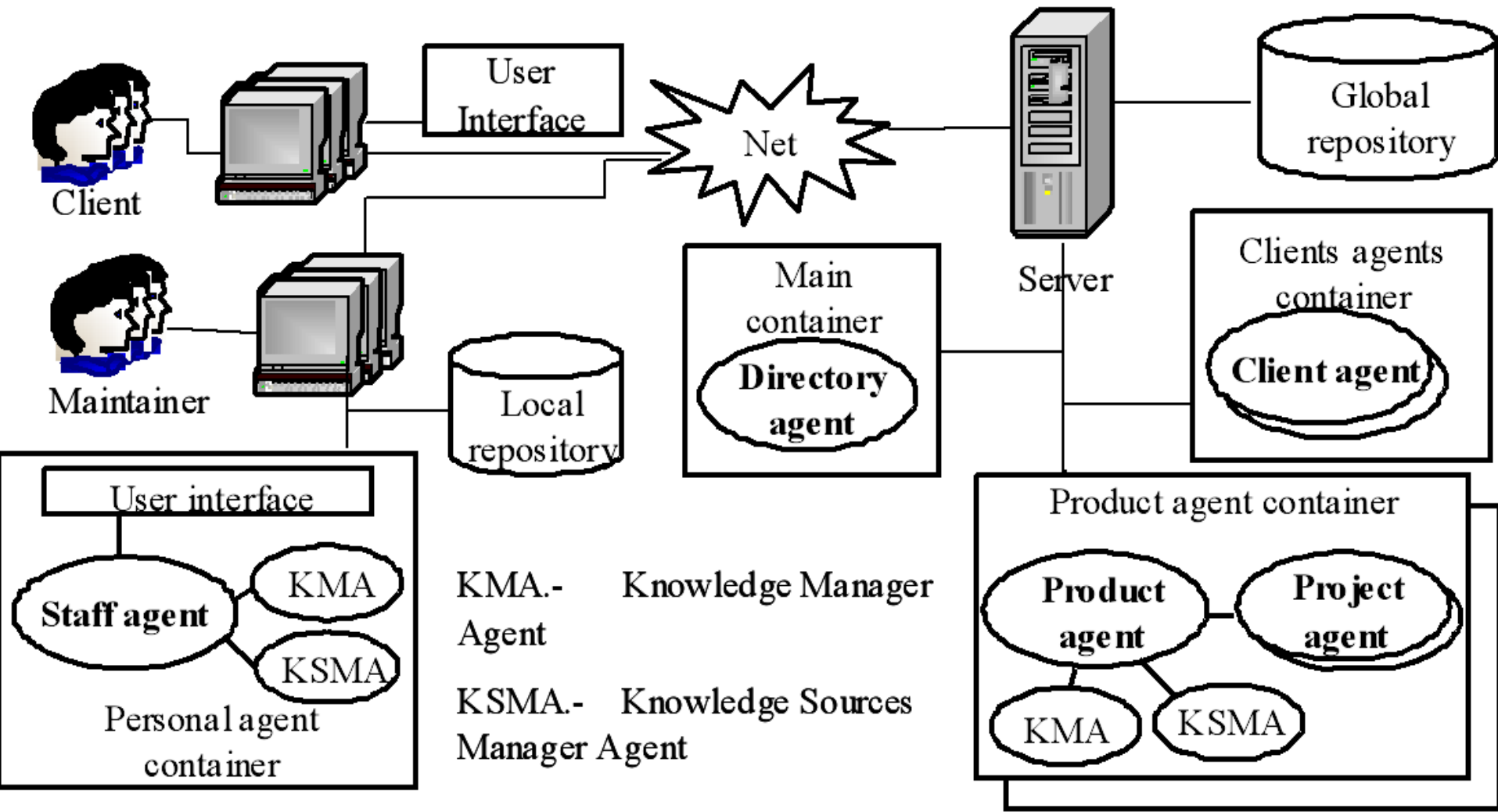
- ❑ Tri thức được hình thành từ hướng dẫn (guideline), mô hình hay thể hiện rõ ràng kỹ năng nhân sự.
- ❑ Tổ chức tạo ra hệ thống để hỗ trợ và chia sẻ kinh nghiệm. 4 yếu tố đòi hỏi thực thi thành công cơ sở tri thức kinh nghiệm:
 - Thay đổi văn hoá
 - Tính ổn định
 - Giá trị nghiệp vụ (business value)
 - Thực thi tăng cường

Các khía cạnh cơ bản của bảo trì – kiến thức ...



Các khía cạnh cơ bản của kỹ sư bảo trì khi thực hiện thay đổi, và tri thức hỗ trợ trong bảo trì

Vận dụng Công cụ KM Agent hỗ trợ Bảo trì



Vấn đề tham dự trong quá trình bảo trì ?

☐ Hiểu hệ thống hiện hành

- Hệ thống thực sự làm được gì?
- Ở đâu cần thay đổi?
- Các phần của phần mềm liên quan với nhau như thế nào?

☐ Thực thi sự thay đổi

☐ Kiểm thử

☐ Nhận diện nhu cầu thay đổi

☐ Thảo luận

- Hiểu chương trình ?
- Tác động thay đổi?
- Kiểm thử ?
- Vấn đề Quản lý?

Bài tập & thảo luận

- ❑ **Exercise 5.6:** Bạn thực hiện gì ở dự án phần mềm vừa qua? Đó là dự án thương mại & dự án cấp đại học hay dự án nhân sự. Hãy viết đánh giá mô hình chu trình sống mà bạn đã làm. Nó có đảm bảo có cấu trúc hay không dự định. Bạn có thực hiện mô hình khác nếu như bạn bắt đầu dự án này một lần nữa
- ❑ **Exercise 5.7:** Bạn là IT manager phải có trách nhiệm với hệ thống thư viện lớn bị lỗi không mong đợi vào sáng thứ hai. Bạn đã trải qua các bước để thực hiện nó như thế nào:
 - 1. Nó cấp bách phải mất 2 giờ để thực hiện
 - 2. Nếu thư viện có chức năng tương đương cho vài ngày mà không có hệ thống phần mềm

Yêu cầu thực hiện tuần tiếp theo

- ☐ **Viết lại các báo cáo cho các thảo luận trên lớp**
- ☐ **Bài tập 1:** Tìm hiểu qui trình nâng cao đảm bảo chất lượng (slide 58)
- ☐ **Bài tập 2:** Tìm hiểu Knowledge Management Agent hỗ trợ qui trình bảo trì (software maintenance process)
- ☐ **Deadline tối hôm trước của buổi học kế tiếp**
- ☐ Chuẩn bị báo cáo khả thi cho đề án của nhóm, kết hợp thảo luận với nhóm khách hàng. Sẽ dành thời gian cho các nhóm luân phiên lên trình bày với các khách hàng thứ ... (Trên lớp)