HIENLTH

Chủ đề 1: Tổng quan về PTTK HĐT

Giới thiệu môn học

- Giảng viên:
  - **Ths. Lương Trần Hy Hiến (HIENLTH)**
  - Khoa CNTT, ĐH Công nghệ TpHCM (FIT – HUTECH)
  - Email: hienlth@hcmup.edu.vn
  - Điện thoại: **0125.4774.690**
  - Web môn học:

http://monhoc.weebly.com

Tài liệu tham khảo (1/2)

- ***Giáo trình OOAD, HUTECH.***
- ***Grady Booch*** (2007), Object-Oriented Analysis and Design with Applications, 3rd Edition, Addison Wesley.
- ***Dennis, Wixom, Tegarden*** (2009), System Analysis & Design with UML version 2.0, An Object-Oriented Approach 3rd Edition, Addison Wesley.
- ***Đặng văn Đức*** (2002), Phân tích thiết kế hướng đối tượng bằng UML, NXB Giáo dục.

Tài liệu tham khảo (2/2)

- http://www.agilemodeling.com/essays/umlDiagrams.htm
- http://www.omg.org/spec/UML/
- http://www.tutorialspoint.com/uml/

Thang điểm đánh giá

- Giữa kỳ: **30%**
  - Bài tập trên lớp + chuyên cần
  - Thi thực hành trên lớp
- Cuối kỳ: **70%**
  - Đồ án môn học

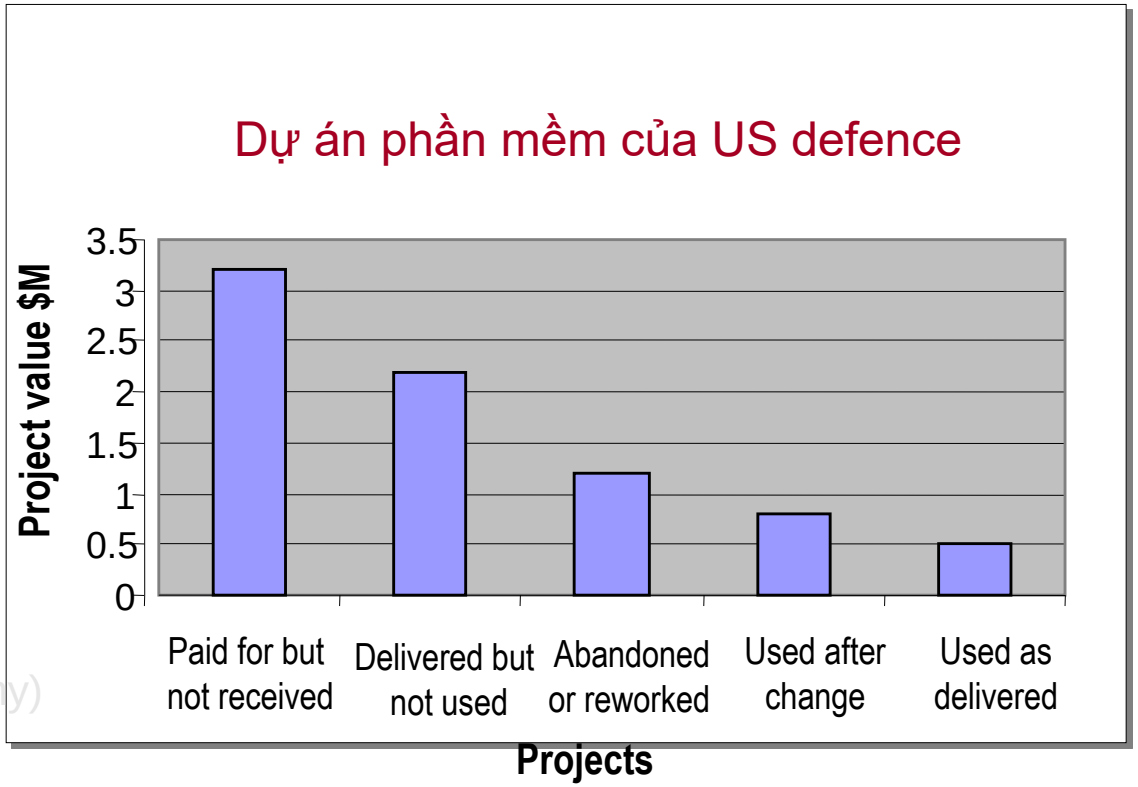
Nội dung

1. Khủng hoảng phần mềm
2. Công nghệ phần mềm
3. Quy trình công nghệ phần mềm
4. Phân tích thiết kế hướng chức năng
5. Phân tích thiết kế hướng đối tượng

1. Khủng hoảng phần mềm

NATO Software Engineering Conference, Germany, 1968

Thống kê của chính phủ Mỹ về các dự án SW của Bộ quốc phòng, 1970.



(E. Balagurusamy)

1. Khủng hoảng phần mềm

Genesis 11:1-9 Acts 2:1-4
The Tower Of Babel

1. Khủng hoảng phần mềm

How The Customer Explained It

1. Khủng hoảng phần mềm

How The Project Leader Understood It

1. Khủng hoảng phần mềm

How The Analyst Designed It

1. Khủng hoảng phần mềm

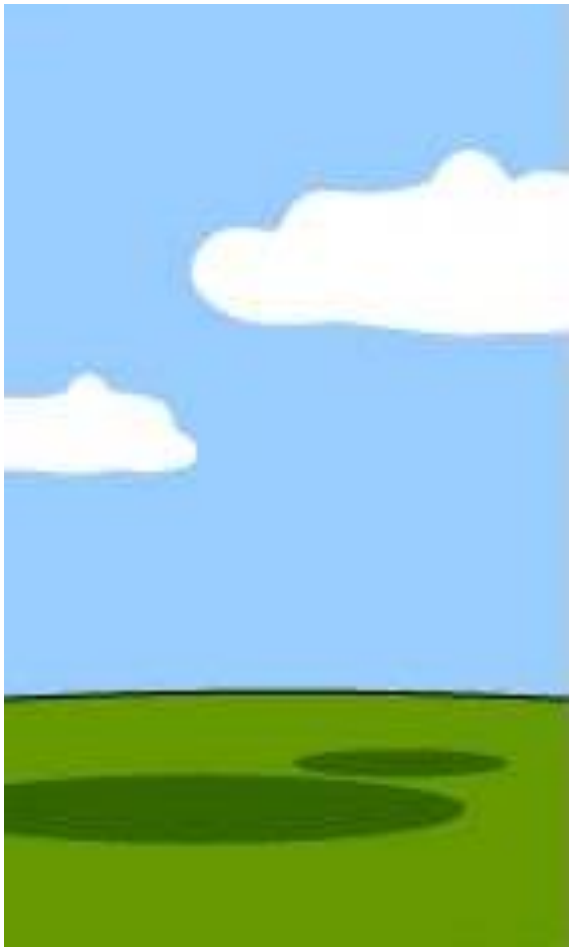
How The Programmer Wrote It

1. Khủng hoảng phần mềm

How The Business Consultant Described It

1. Khủng hoảng phần mềm

How The Project Was Documented



1. Khủng hoảng phần mềm

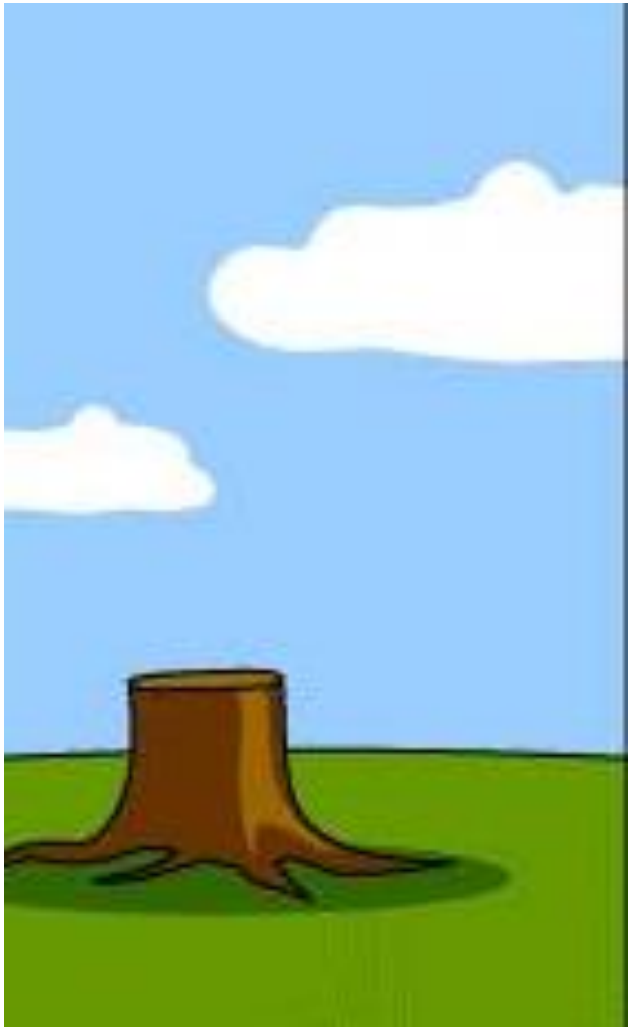
What Operations Installed

1. Khủng hoảng phần mềm

How The Customer Was Billed

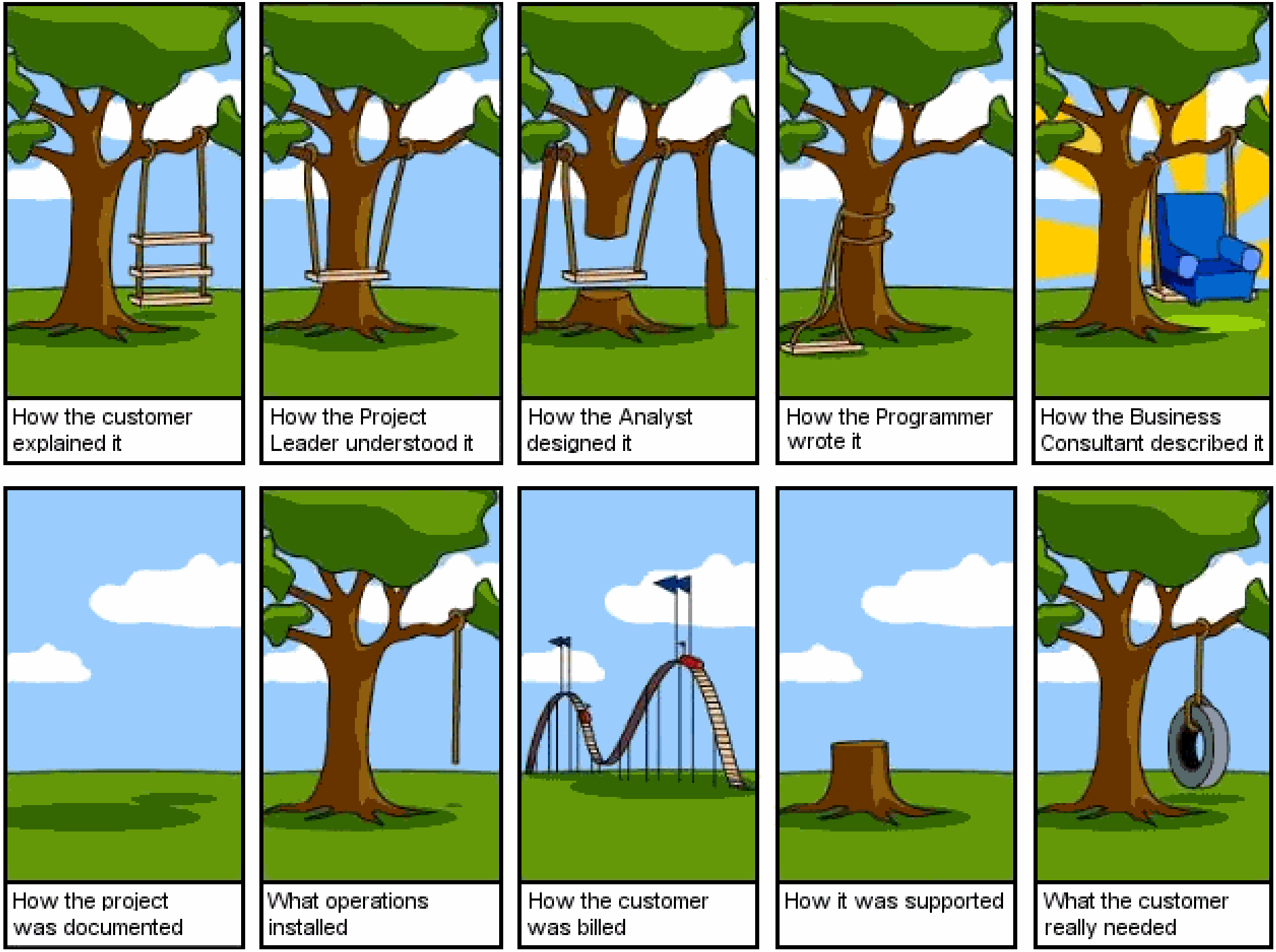
1. Khủng hoảng phần mềm

How It Was Supported



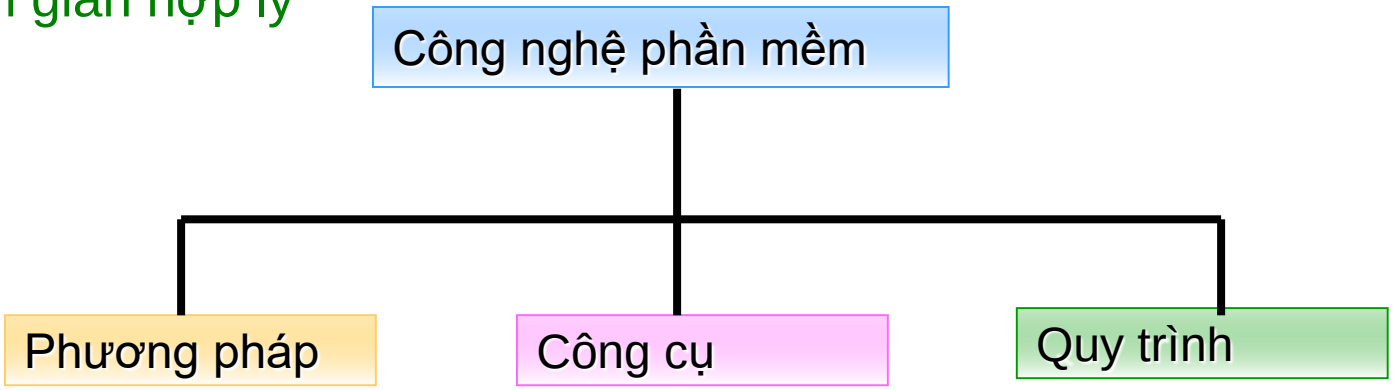
1. Khủng hoảng phần mềm

What The Customer Really Needed


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

2. Công nghệ phần mềm

- Khái niệm:
  - Công nghệ phần mềm là ngành khoa học nghiên cứu về việc xây dựng các phần mềm có chất lượng với chi phí hợp lý trong khoảng thời gian hợp lý



- Đối tượng nghiên cứu:
  - Quy trình công nghệ
  - Phương pháp xây dựng phần mềm
  - Công cụ hỗ trợ phát triển phần mềm

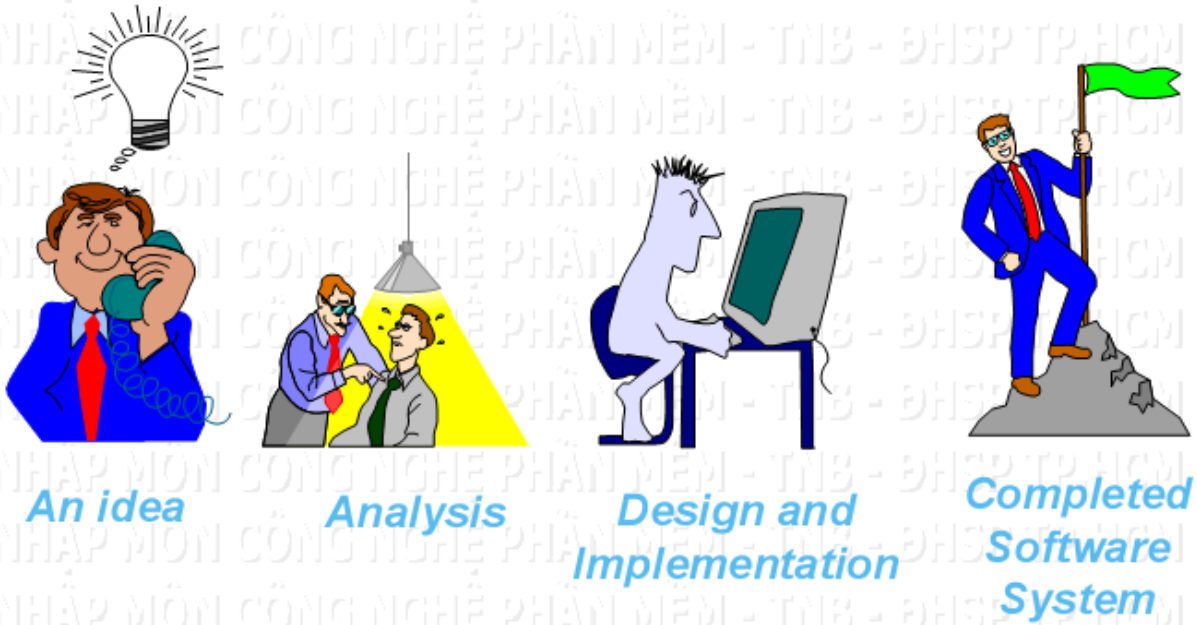
2. Công nghệ phần mềm

- Các đối tượng nghiên cứu của Công nghệ phần mềm :
  - Quy trình phần mềm:
    - Hệ thống các giai đoạn mà quá trình phát triển phần mềm phải trải qua,
    - với mỗi giai đoạn cần xác định rõ:
      - Mục tiêu, kết quả nhận từ giai đoạn trước đó,
      - Kết quả chuyển giao cho giai đoạn kế tiếp
  - Phương pháp phát triển phần mềm:
    - Hệ thống các hướng dẫn cho phép từng bước thực hiện một giai đoạn nào đó trong quy trình phần mềm
  - Công cụ và Môi trường phát triển phần mềm:
    - Hệ thống các phần mềm trợ giúp trong lĩnh vực xây dựng phần mềm
    - Hỗ trợ các chuyên viên tin học trong các bước xây dựng phần mềm theo một phương pháp nào đó với một quy trình được chọn trước

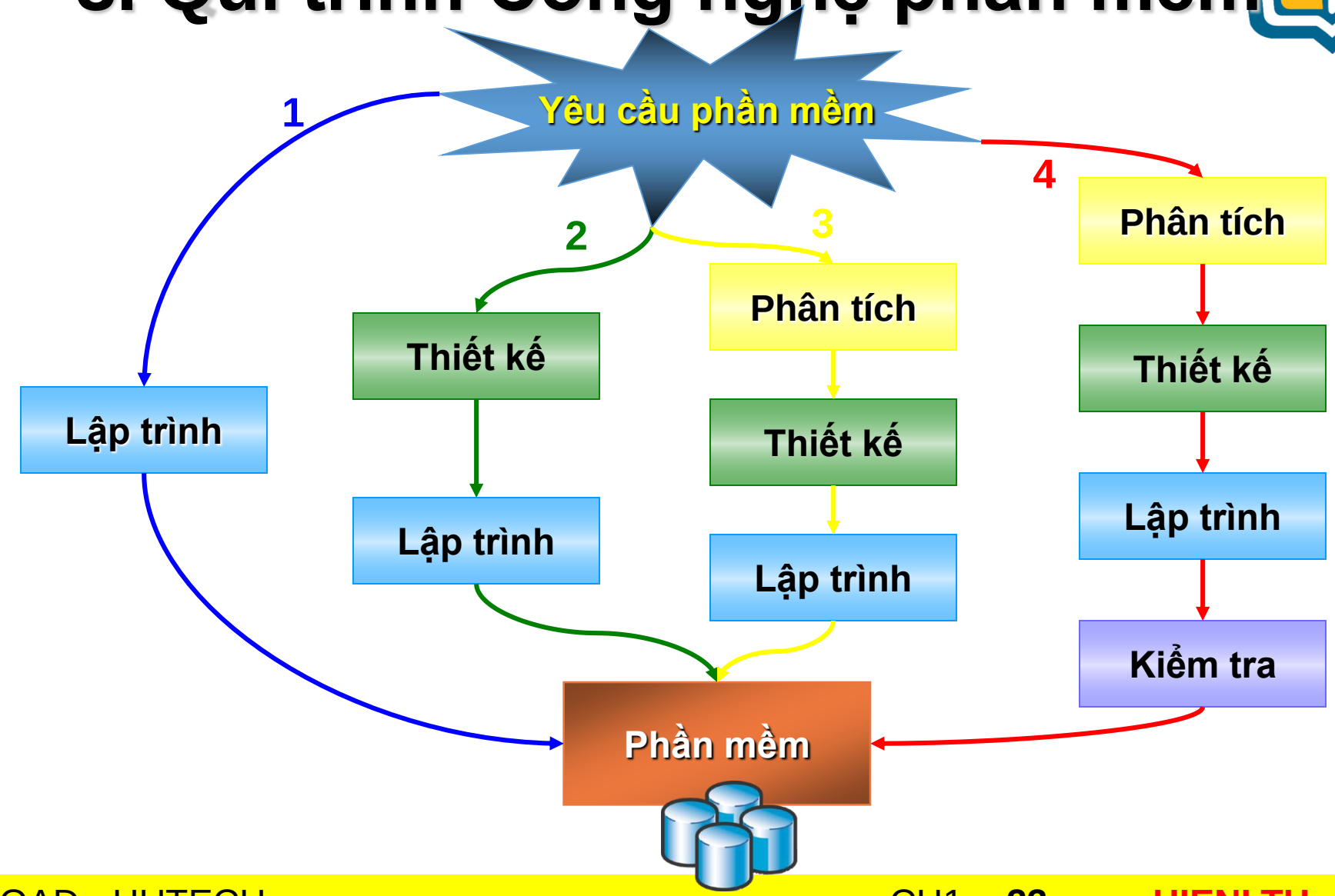
3. Quy trình Công nghệ Phần mềm

- Xây dựng phần mềm cần phải thực hiện theo trình tự nào?
- Cần bao nhiêu người tham gia? Vai trò của từng thành viên? Tổ chức quản lý các thành viên?
- Giao tiếp giữa các thành viên trong hệ thống?

Quy trình Công nghệ Phần mềm – Software Development Process



3. Qui trình Công nghệ phần mềm

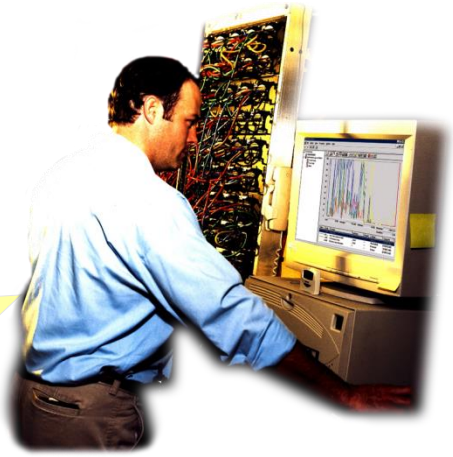


3. Quy trình Công nghệ Phần mềm

3. Quy trình Công nghệ Phần mềm

- Làm thế nào để tiếp nhận chính xác yêu cầu của khách hàng?
- Làm thế nào để đặc tả đúng yêu cầu của khách hàng?
- Làm thế nào để giao tiếp, tương tác với bộ phận phát triển hệ thống?
- Làm thế nào để kiểm tra hệ thống phát triển đúng theo yêu cầu trước khi thực hiện triển khai đến khách hàng?

Bộ phận tiếp nhận yêu cầu của khách hàng



Business Analyst

3. Quy trình Công nghệ Phần mềm

Bộ phận phát triển phần mềm

Developer

- Làm thế nào để thiết kế hệ thống đúng với yêu cầu của người dùng?
- Làm thế nào để giao tiếp, tương tác với các thành viên trong bộ phận phát triển phần mềm?
- Làm thế nào để quản lý, theo dõi tiến trình thực hiện phần mềm?

3. Quy trình Công nghệ Phần mềm

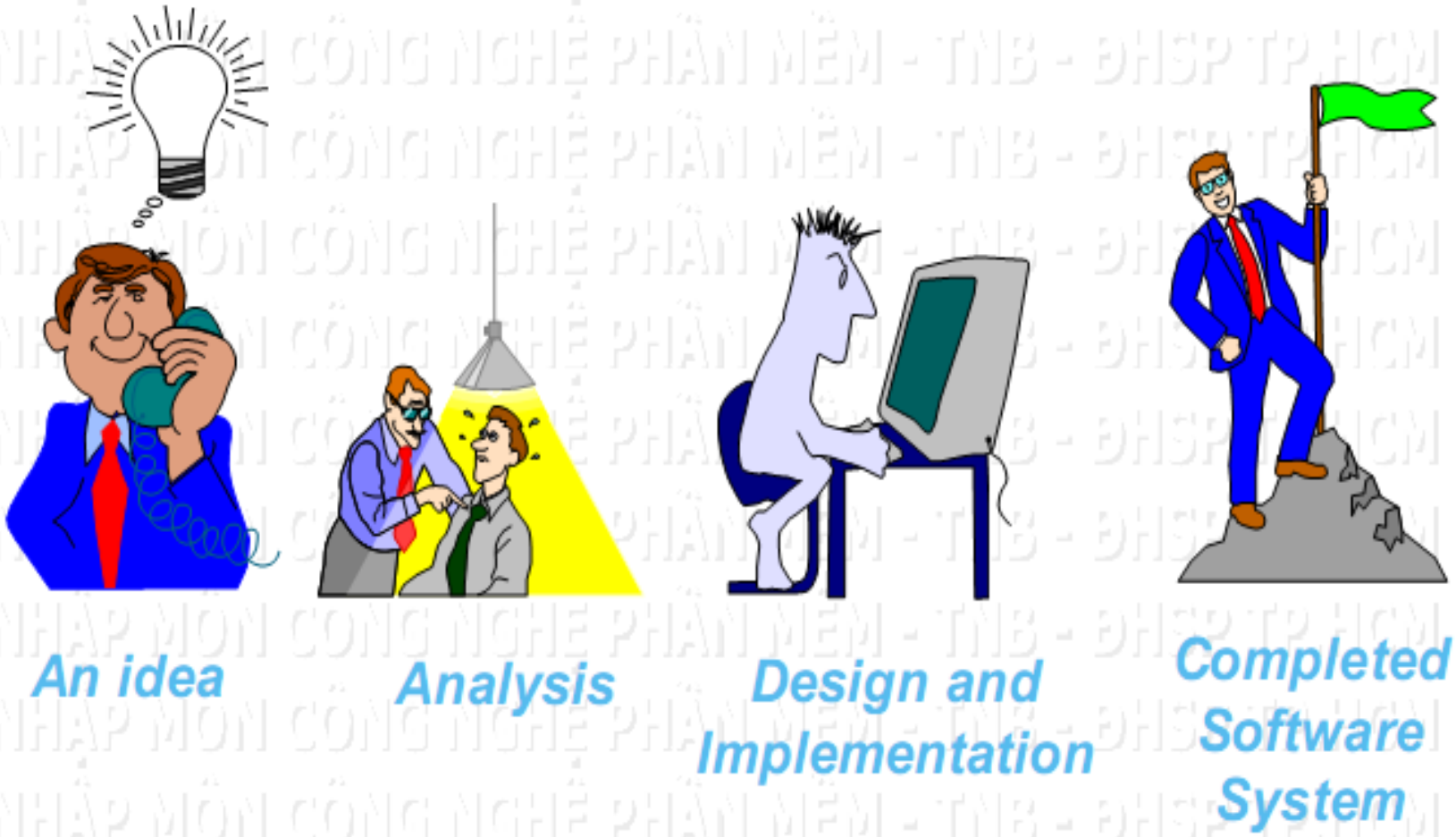
Bộ phận phát triển phần mềm

Bộ phận tiếp nhận yêu cầu của khách hàng

Development

Business Analyst

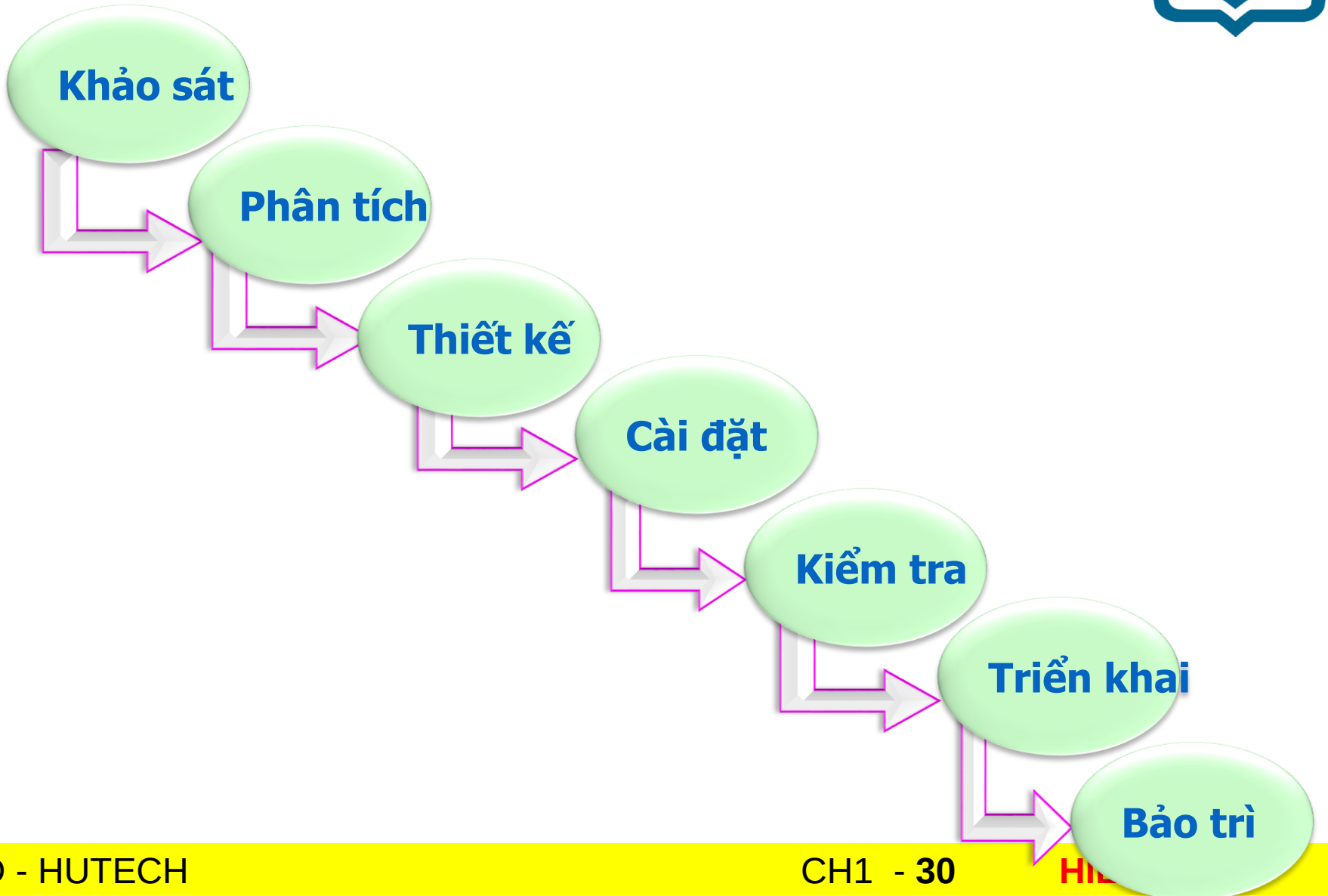
3. Quy trình Công nghệ Phần mềm



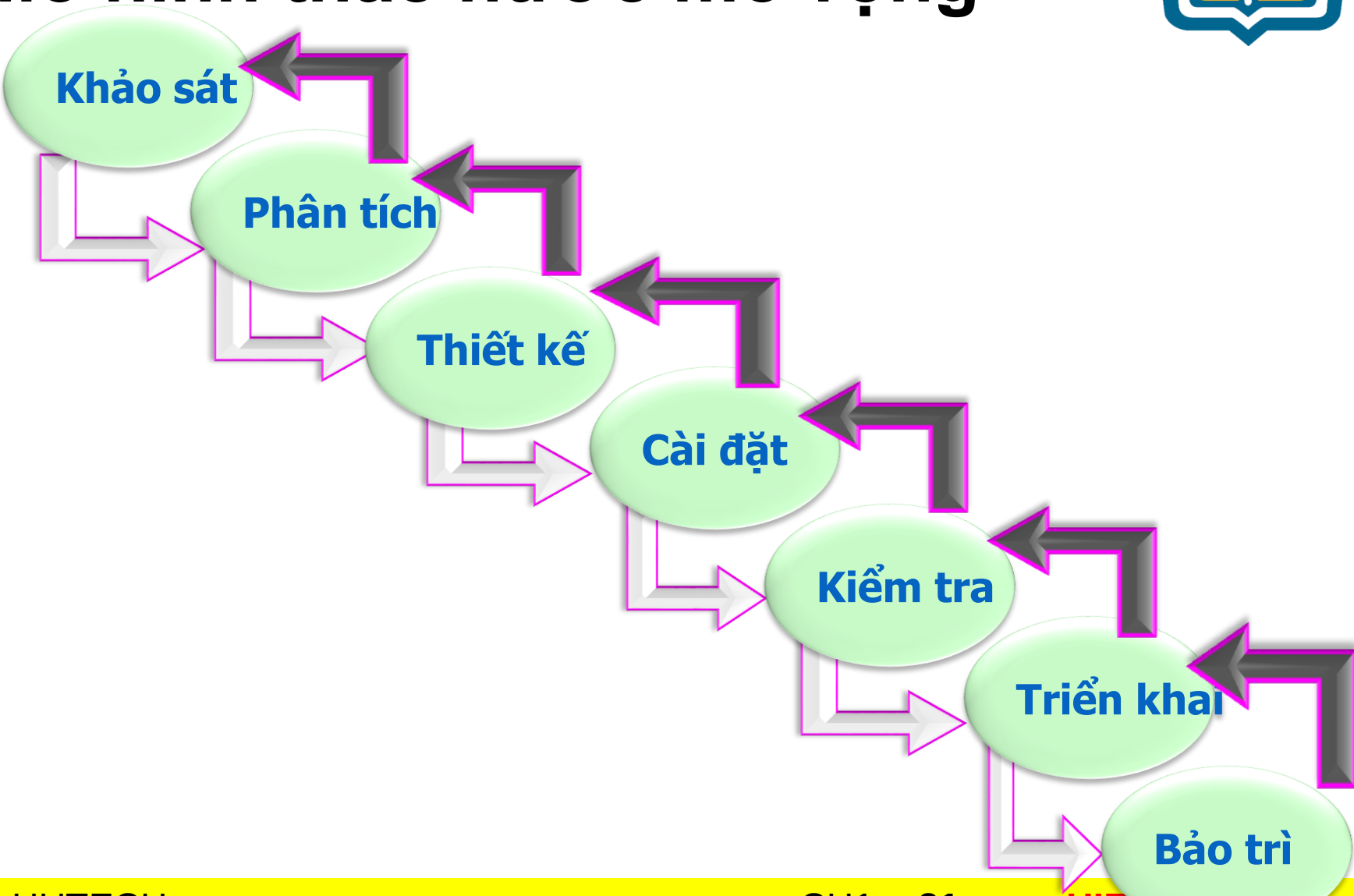
3. Qui trình Công nghệ phần mềm

- Phân tích: Mô tả mức phát thảo các thành phần của phần mềm (đã có yêu cầu)
- Thiết kế: Mô tả mức chi tiết các thành phần của phần mềm (đã phân tích)
- Lập trình: Thực hiện các thành phần của phần mềm (đã thiết kế)
- Kiểm tra: kiểm chứng các thành phần của phần mềm (đã thực hiện)

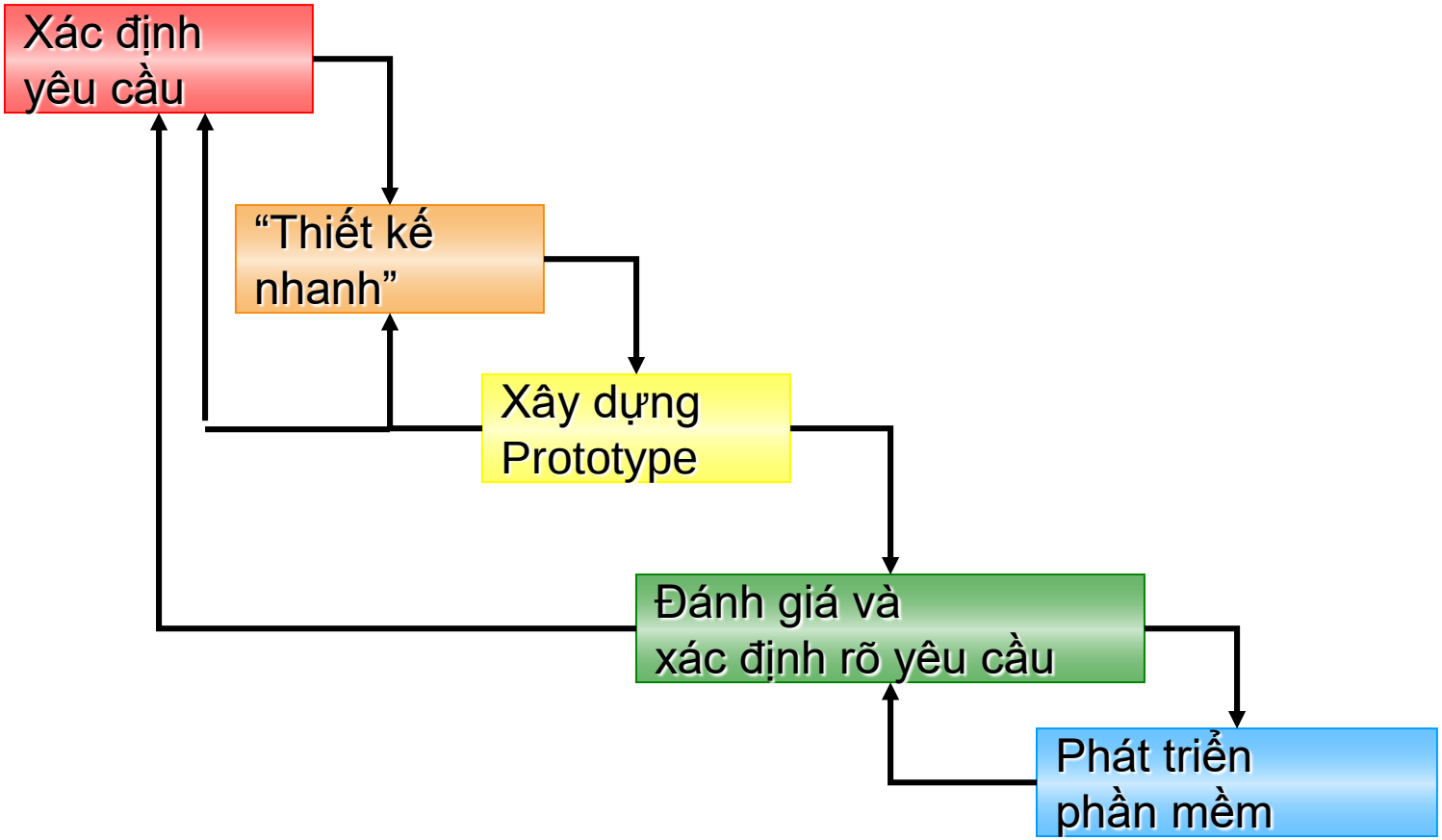
Mô hình thác nước



Mô hình thác nước mở rộng

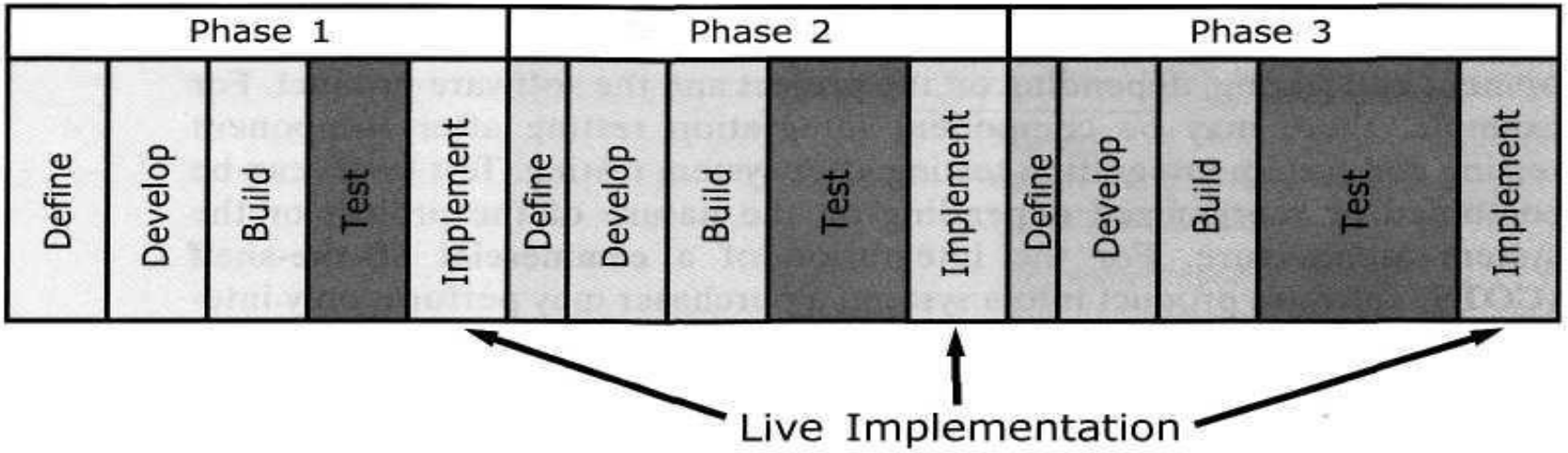


Quy trình Prototype



Quy trình phát triển lặp

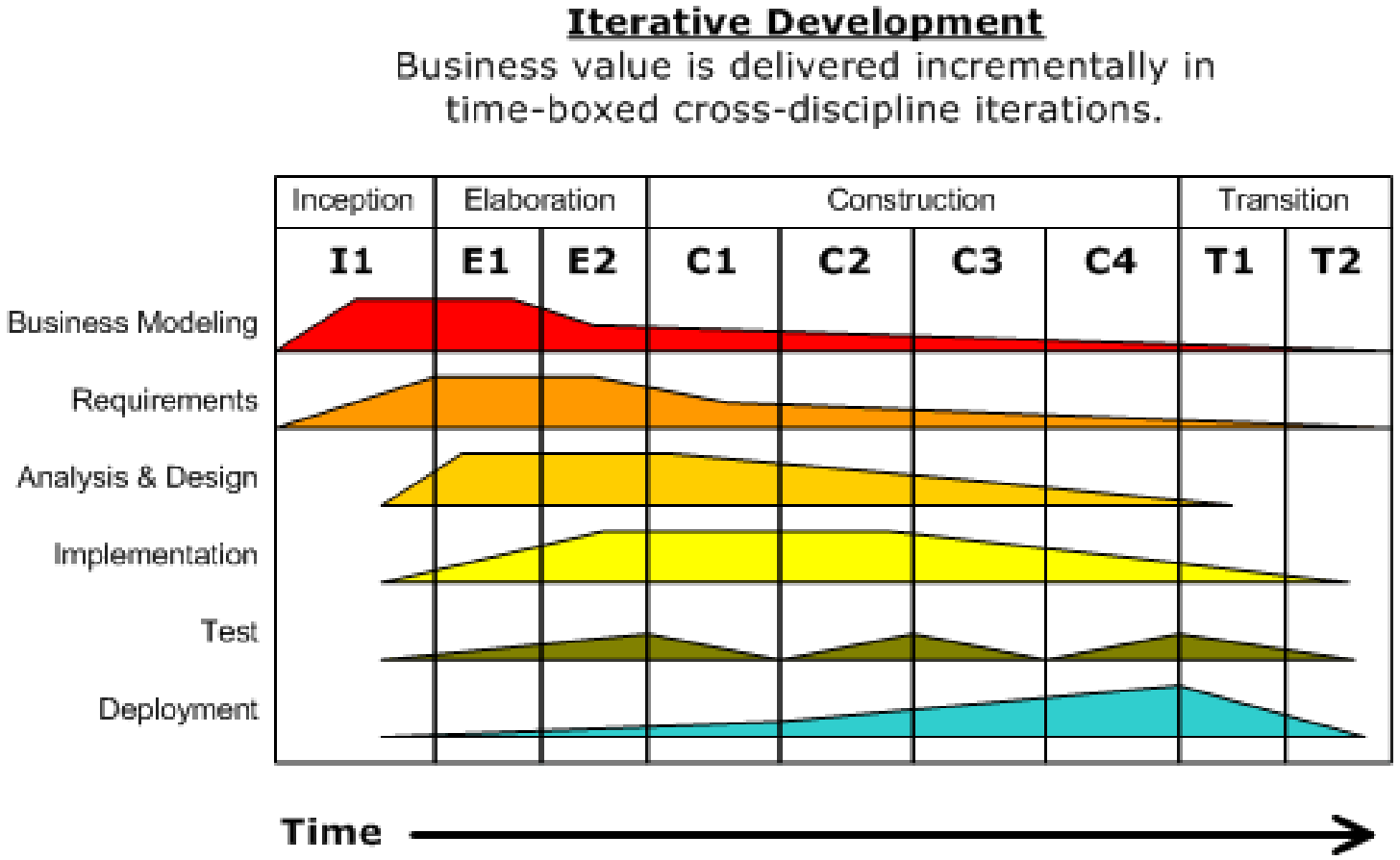
- Chúng ta có thể chia nhỏ phần mềm ra làm nhiều giai đoạn thay vì làm một lần từ đầu đến cuối.



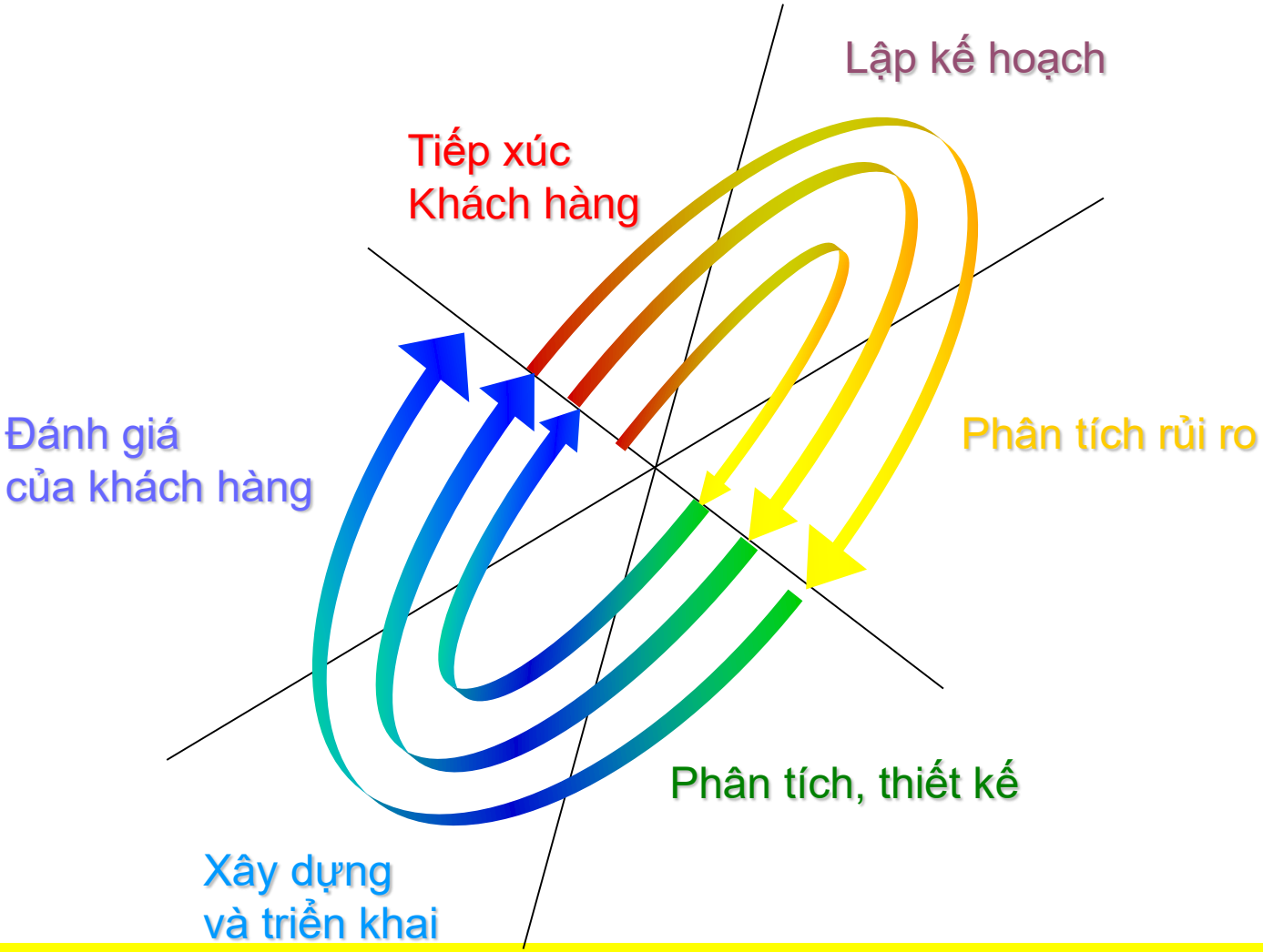
Iterative development model

Quy trình phát triển lặp

- IBM Rational Unified Process (RUP)



Quy trình xoắn ốc

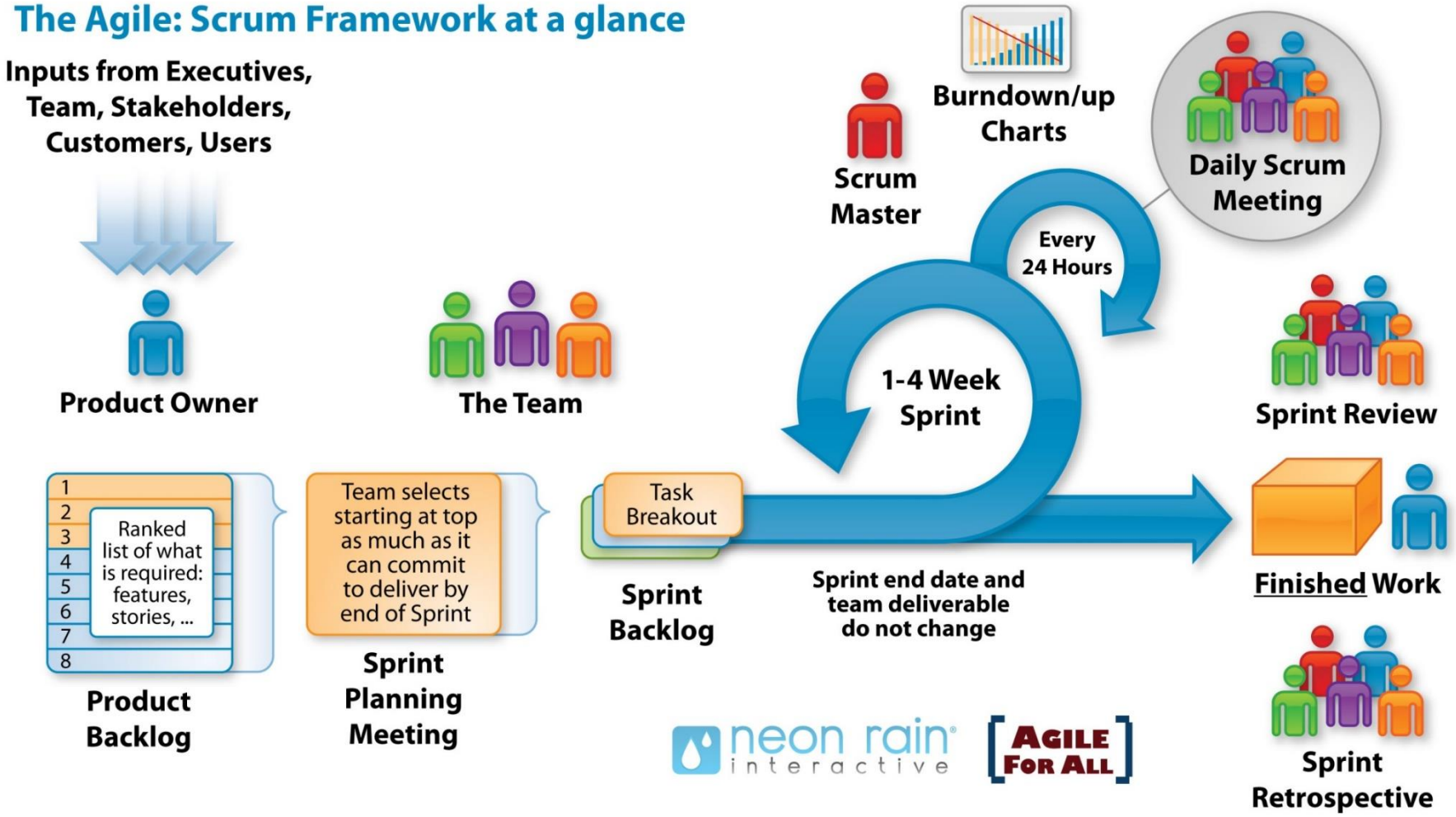


Quy trình xoắn ốc

- Qui trình được biểu diễn ở dạng xoắn ốc thay vì một dãy các hoạt động với quay lui.
- Mỗi lần lặp trong xoắn ốc biểu diễn một pha trong qui trình.
- Không có các pha cố định như đặc tả hay thiết kế - số lần lặp trong xoắn ốc được chọn phụ thuộc vào nhu cầu.
- Các rủi ro được đánh giá và giải tỏa một cách rõ ràng xuyên suốt qui trình.

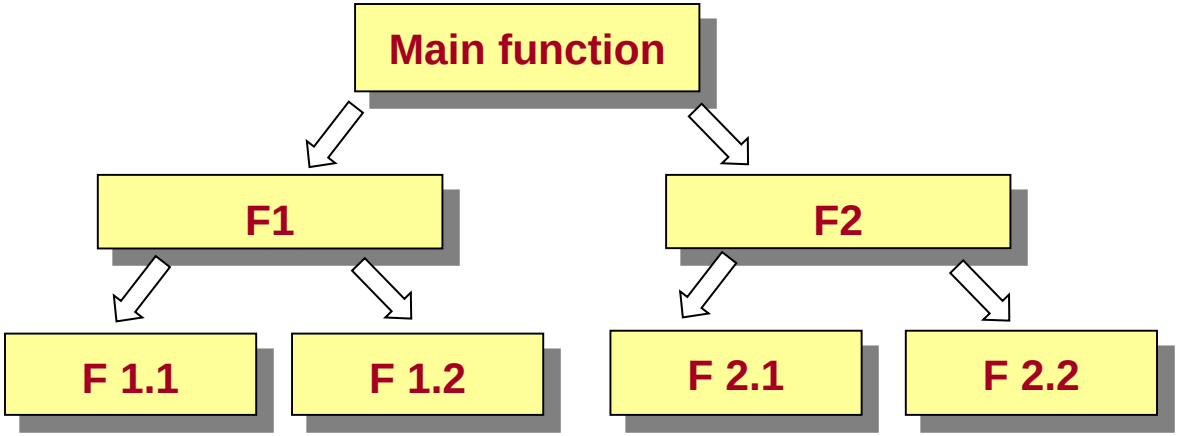
Agile methods

Agile methods



4. Phân tích thiết kế chức năng

- Cho đến giữa 1990: Phần lớn các kỹ sư phần mềm sử dụng phương pháp thiết kế chức năng top-down (thiết kế kiến trúc)



4. Phân tích thiết kế chức năng

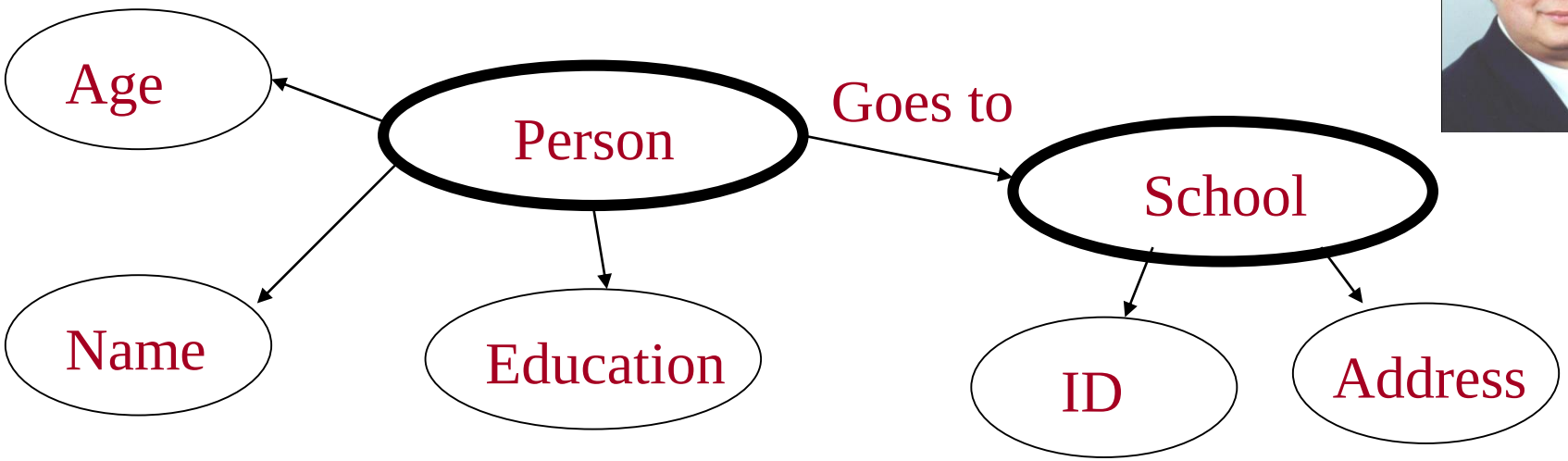
- Tiến trình phát triển tập trung vào thông tin mà hệ thống quản lý
- Chỉ tập trung vào thông tin, ít quan tâm đến cái gì thực hiện với thông tin hay hành vi hệ thống
- Tiếp cận này gọi là tiếp cận hướng dữ liệu

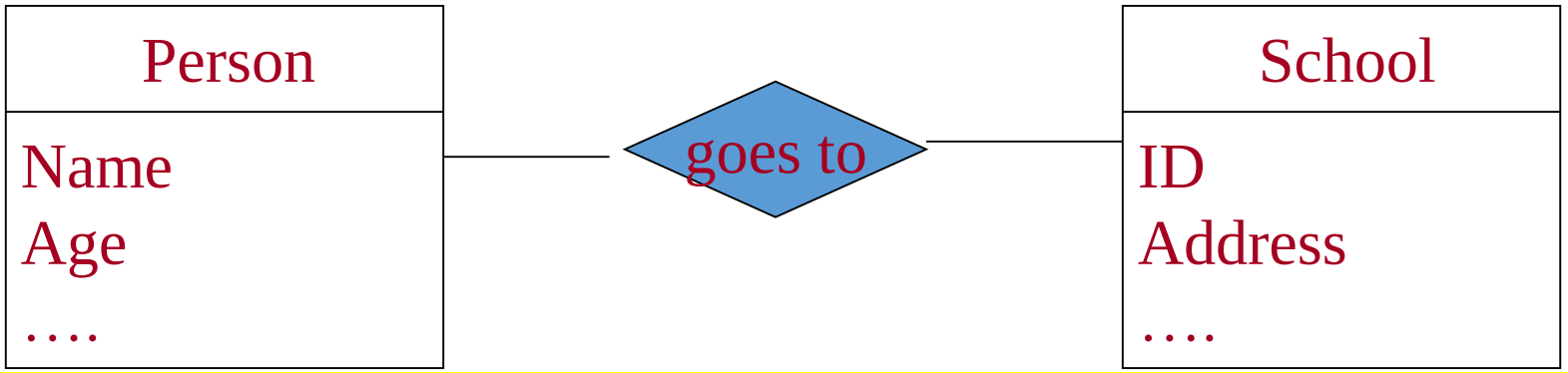
4. Phân tích thiết kế chức năng

- Công nghệ hướng chức năng có các hạn chế sau
  - Sản phẩm hình thành từ giải pháp này khó bảo trì
  - Tiến trình phát triển không ổn định
  - Tiệm cận này không hỗ trợ lập trình bằng ngôn ngữ hướng đối tượng như C++, Java, Smalltalk, Eiffel.

4. Phân tích thiết kế chức năng

Entity-Relationship Model





5. Phân tích thiết kế hướng đối tượng

5.1. Sự phức tạp của việc phân tích hệ thống

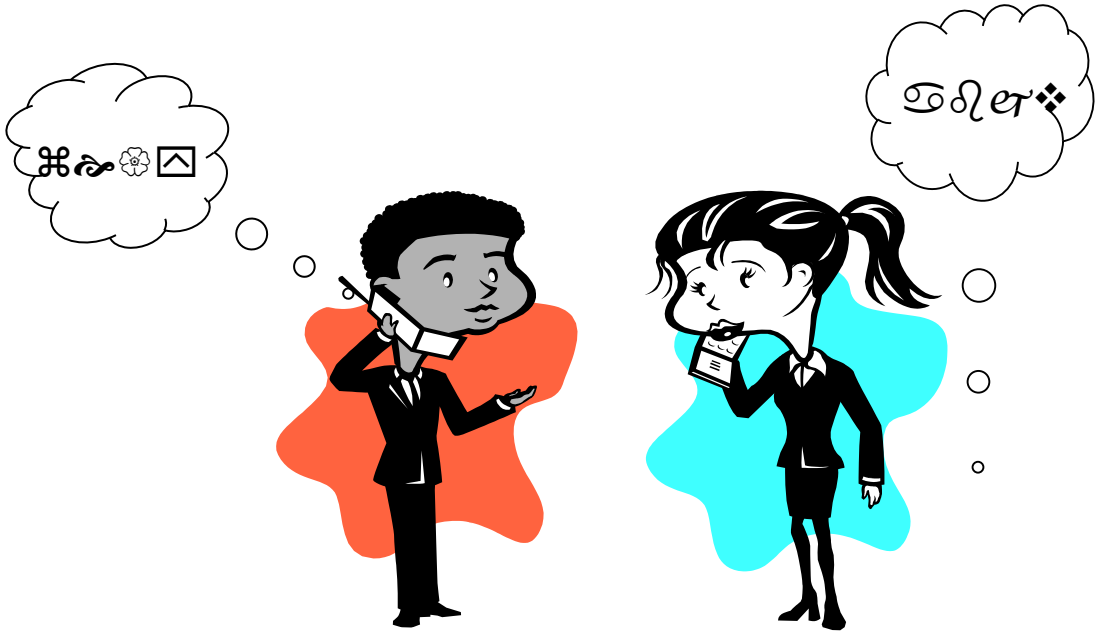
5.2. Thế nào là hướng đối tượng

5.3. Thế nào là phân tích hướng đối tượng

5.4. Chu trình phát triển hệ thống hướng đối tượng

5. Phân tích Thiết kế Hướng đối tượng

5.1. Sự phức tạp của việc phân tích hệ thống



What kind of language can alleviate difficulties with communication & complexity hopefully well?

5. Phân tích Thiết kế Hướng đối tượng

5.1. Sự phức tạp của việc phân tích hệ thống

5. Phân tích Thiết kế Hướng đối tượng

5.1. Sự phức tạp của việc phân tích hệ thống

- Tính phức tạp của lĩnh vực vấn đề
- Khó khăn trong quản lý tiến trình phát triển
- Vấn đề xác định đặc điểm hành vi hệ thống

5. Phân tích Thiết kế Hướng đối tượng

5.1. Sự phức tạp của việc phân tích hệ thống

Làm chủ hệ thống phức tạp

- Nhiệm vụ cơ bản của kỹ nghệ phần mềm là làm chủ độ phức tạp trong tiến trình phát triển phần mềm
- Thí dụ hệ thống phức tạp
  - Máy vi tính

5. Phân tích Thiết kế Hướng đối tượng

5.1. Sự phức tạp của việc phân tích hệ thống

Năm tính chất của hệ thống phức tạp

- Tính phức tạp có hình thức phân cấp
- Việc chọn thành phần nào làm cơ sở trong hệ thống là tương đối tùy ý
- Kết nối bên trong thành phần mạnh hơn kết nối giữa các thành phần
- Thông thường các hệ thống phân cấp hình thành từ vài loại phân hệ khác nhau, theo các tổ hợp và sắp xếp khác nhau
- Mọi hệ thống phức tạp được tiến hóa từ hệ thống đơn giản

5. Phân tích Thiết kế Hướng đối tượng

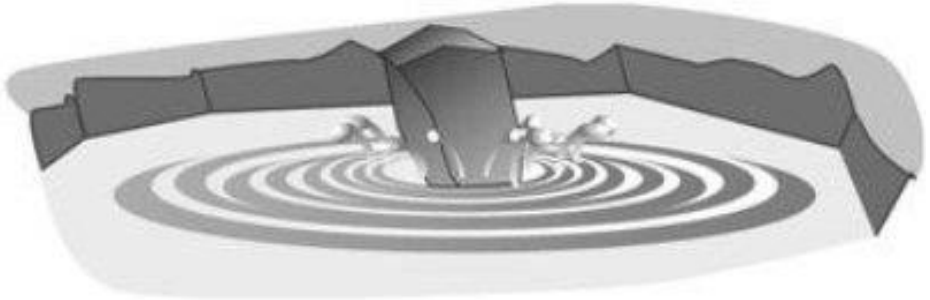
5.2. Thế nào là hướng đối tượng

- Chiến lược phát triển phần mềm hướng đối tượng là quan sát thế giới như tập các đối tượng
- Các tính chất của đối tượng
  - Đối tượng có thể là
    - thực thể nhìn thấy được trong thế giới thực (trong pha phân tích yêu cầu)
    - biểu diễn thực thể hệ thống (trong pha thiết kế)
  - Các đối tượng được phân thành class
    - Các đối tượng thuộc cùng lớp đều có đặc tính (thuộc tính và thao tác) chung

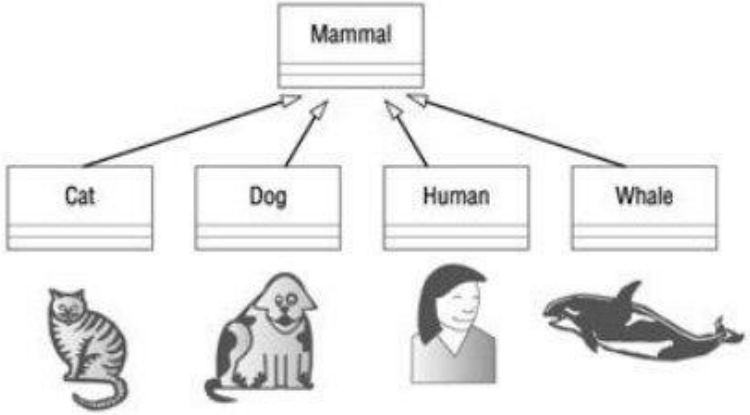
5. Phân tích Thiết kế Hướng đối tượng

5.2. Thế nào là hướng đối tượng

- Tiếp cận hướng đối tượng tập trung vào cả thông tin và hành vi
- Cho khả năng xây dựng hệ thống mềm dẻo, "co dãn"
- Phương pháp này dựa trên các nguyên tắc sau
  - Tính gói
  - Kế thừa
  - Đa trị



Lake Model



Natural Model

Thành phần cơ bản của đối tượng

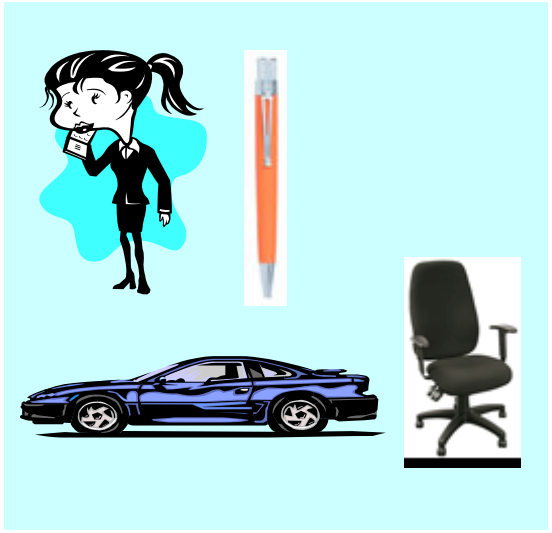
- Đối tượng gồm 2 thành phần cơ bản: trạng thái (state) và hành vi (behavior).
- **Trạng thái** (state):
  - Giúp phân biệt giữa đối tượng này với đối tượng khác
  - Mô tả cấu trúc cơ bản của đối tượng.
  - Bao gồm những thuộc tính (attribute) và những giá trị của những thuộc tính đó.
- **Hành vi** (behavior):
  - Cho biết đối tượng có thể làm được những việc gì.
  - Bao gồm những phương thức (method) để chúng ta có thể điều khiển đối tượng đó.

5. Phân tích Thiết kế Hướng đối tượng

5.2. Thế nào là hướng đối tượng

What is Object-Orientation?

- What is Object?



- A structure that has identity and properties and behavior
- It is an instance of a collective concept, i.e., a class.

What is Object-Orientation
- Abstraction and Encapsulation

Abstraction

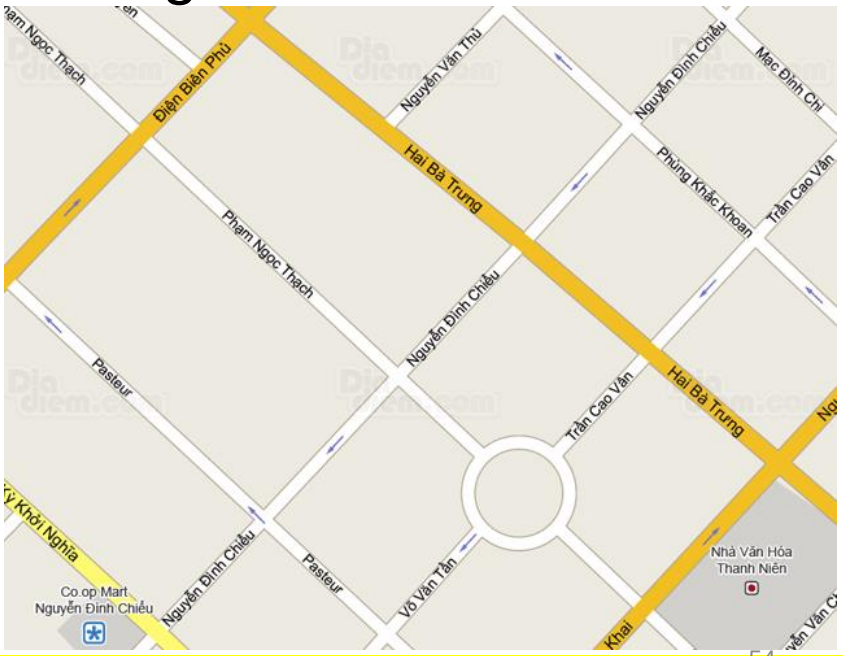
Focus on the essential

Encapsulation

a.k.a. information hiding

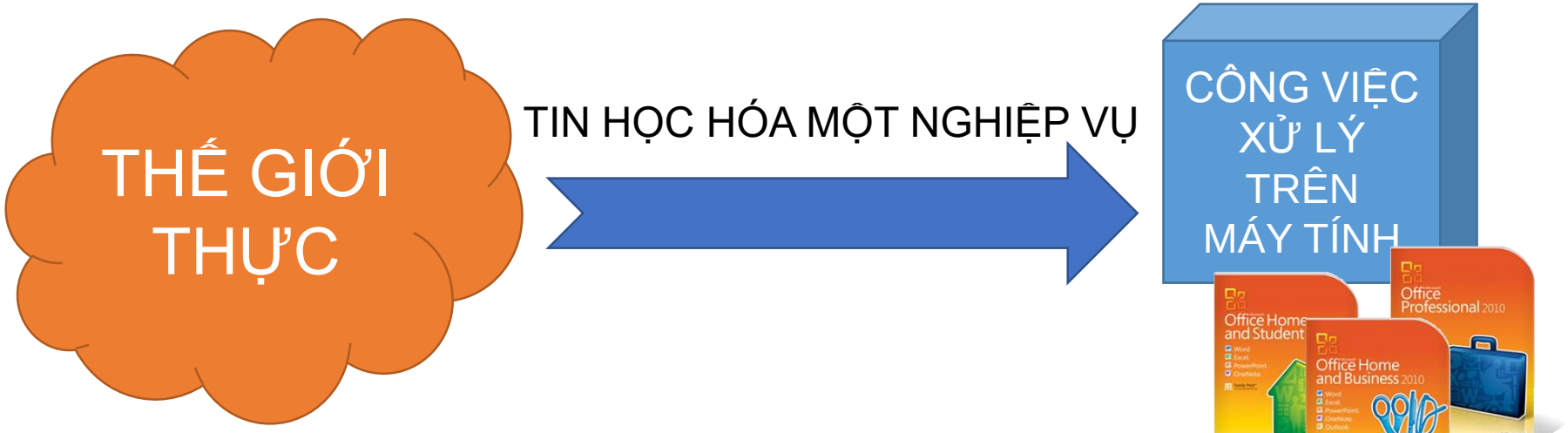
Sự trừu tượng hóa

- Là quá trình bao gồm việc nhận ra và tập trung những tính chất quan trọng của một tình huống hay đối tượng và bỏ qua những chi tiết không quan trọng.
- An abstraction is something more general



Sự trừu tượng hóa trong quá trình phát triển PM

- Sự trừu tượng hóa là khâu quan trọng và cơ bản trong quá trình phát triển phần mềm.
- Sự trừu tượng hóa giúp lượt bỏ những chi tiết không cần thiết và tập trung những chi tiết quan trọng trong thế giới thực để từ đó xây dựng phần mềm.



Một số ví dụ về đối tượng

Những quyển sách

Những cây bút

Những quả bong bóng

Relationships of classes

- Có 3 mối quan hệ cơ bản:
  - Association
  - Dependency
  - Generalization

Behavioral relationships vs Structural relationships

- Behavioral relationship between object X and object Y:
  - object X is temporarily handed a reference to object Y
  - when X is finished communicating with Y, object X often discards the reference to Y
- Structural relationship:
  - A permanent relationship
  - In order to keep track of such relationships, an object actually maintains lasting references to its related objects in the form of fields
- Behavioral relationships = Dependency relationships
- Structural relationships = Association relationships

Ví dụ minh họa

```
class Circle
{
   ……….……
}
class Rectangle
{
   protected double w;
   protected double h;
   protected double top;
   protected double left;
   public bool CheckOverlap(Circle c)
   {
        ……..……
   }
}
```

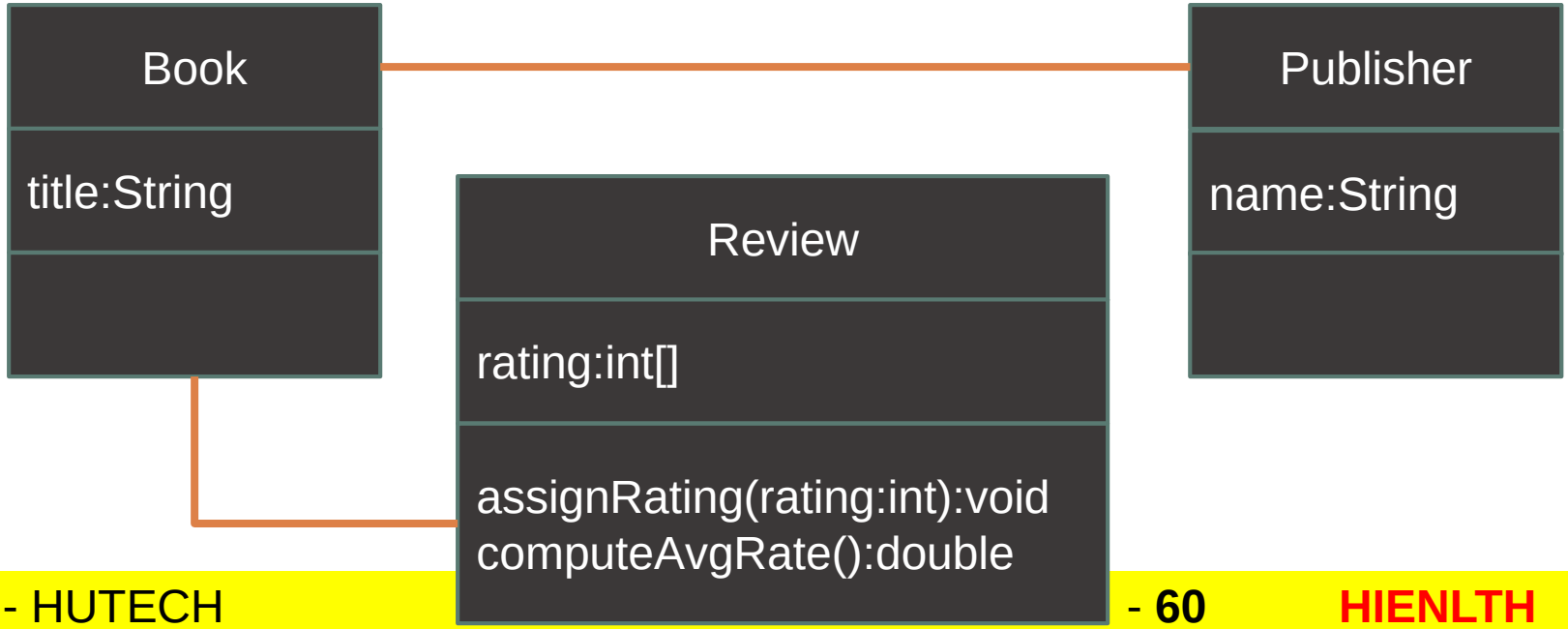
Behavioral relationship between Circle & Rectangle

```
class SinhVien
{
   protected string name;
   protected string id;
}
class LopHoc
{
   protected SinhVien[] dsSinhVien;
}
```

Structural relationship between SinhVien & LopHoc

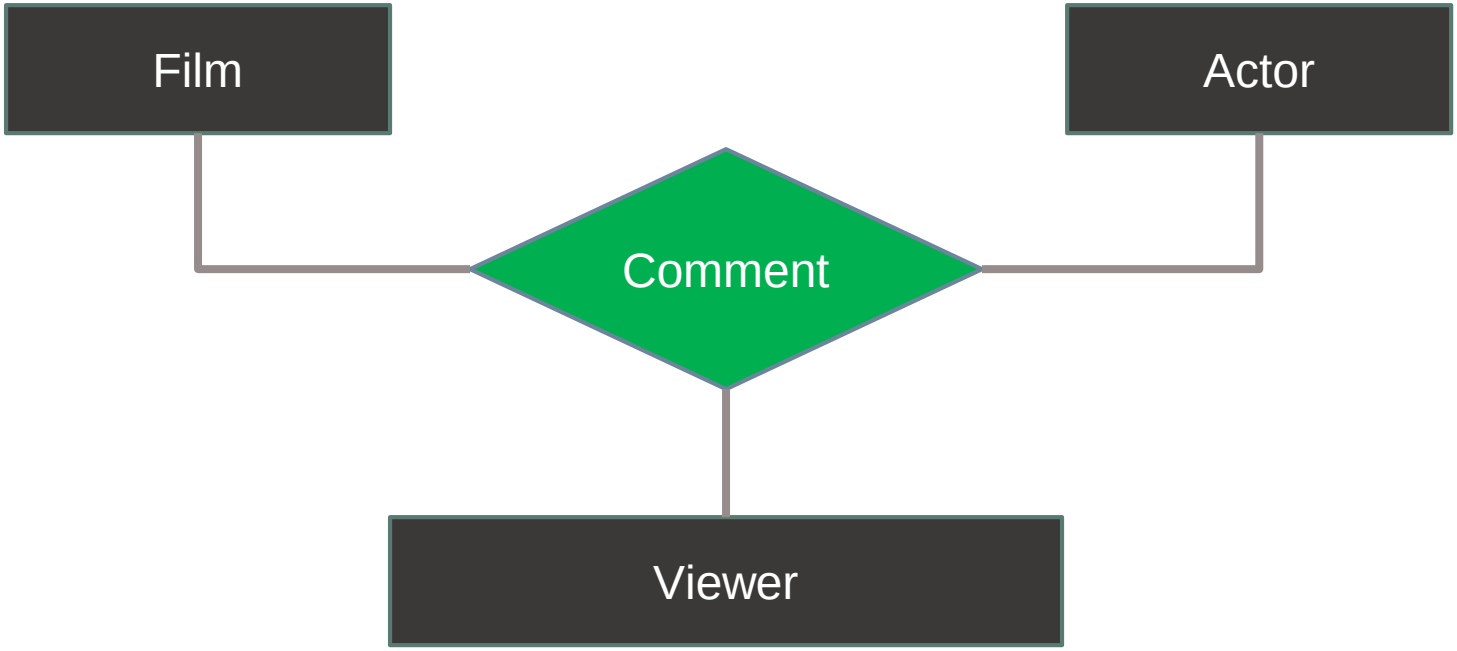
Association relationship (mối quan hệ kết hợp)

- Mối quan hệ kết hợp là mối quan hệ trong đó đối tượng lớp này là thành phần dữ liệu của đối tượng lớp kia.
- Dựa vào số lượng lớp tham gia mối quan hệ, có 2 loại: bin-ary, n-ary
  - bin-ary (mối quan hệ 2 ngôi): là mối quan hệ chỉ có 2 lớp tham gia



Association relationship (cont.)

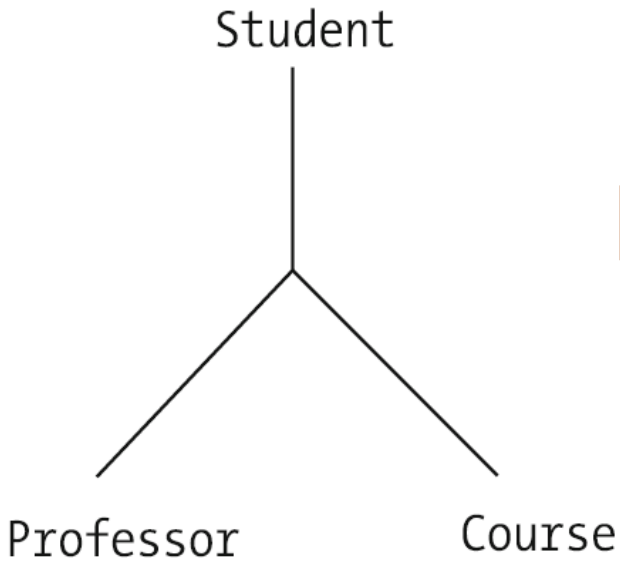
- n-ary (mối quan hệ đa ngôi): là mối quan hệ có nhiều hơn 2 lớp đối tượng tham gia

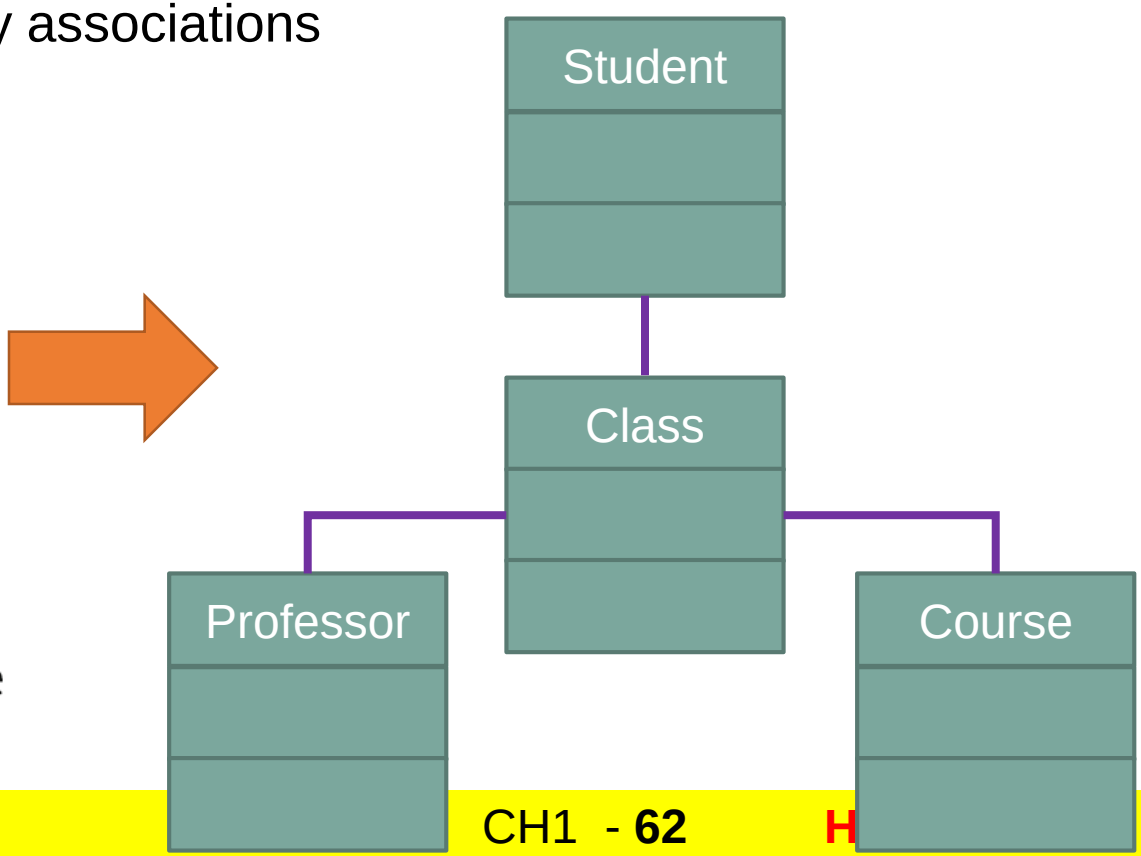


Quan hệ 3 ngôi

Association relationship (cont.)

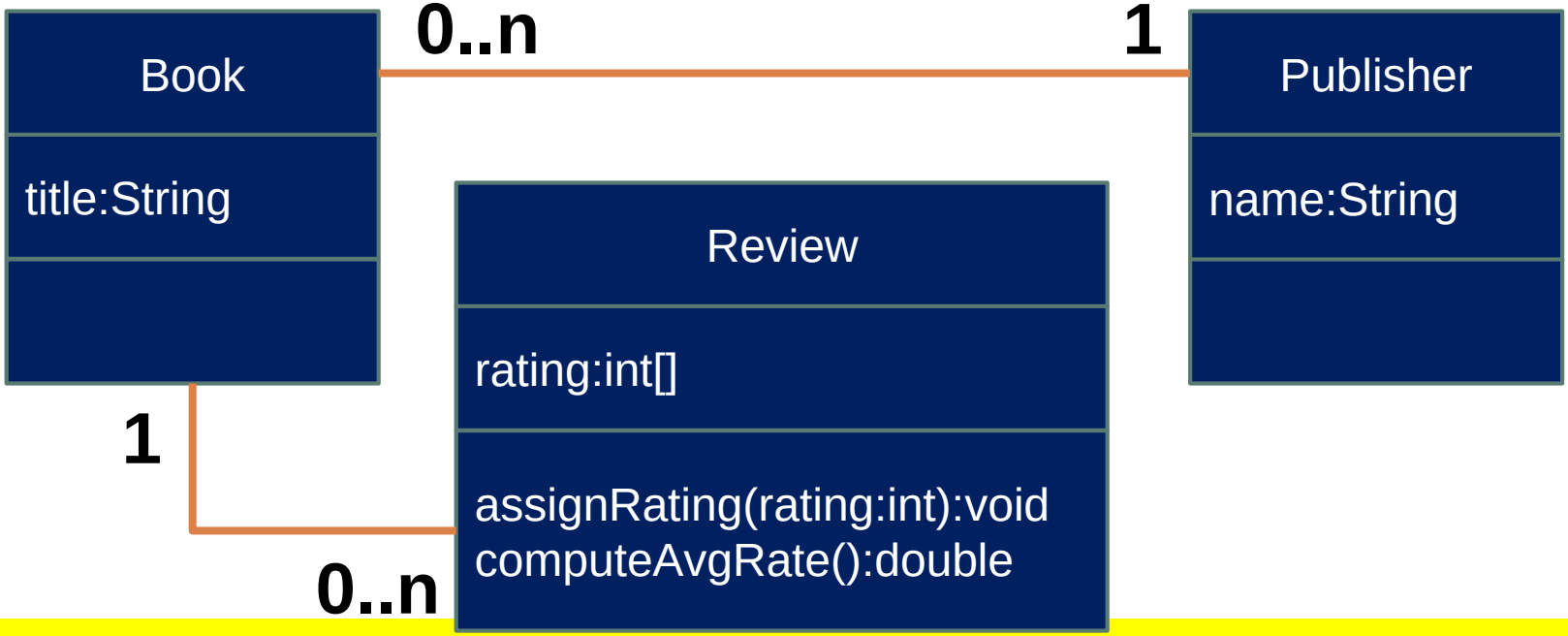
- Higher-order associations are possible, but rare.
- A *ternary association* involves three classes.
- For example, a Student takes a Course from a particular Professor
- However, we usually decompose higher-order associations into an appropriate number of binary associations





Multiplicity (bản số)

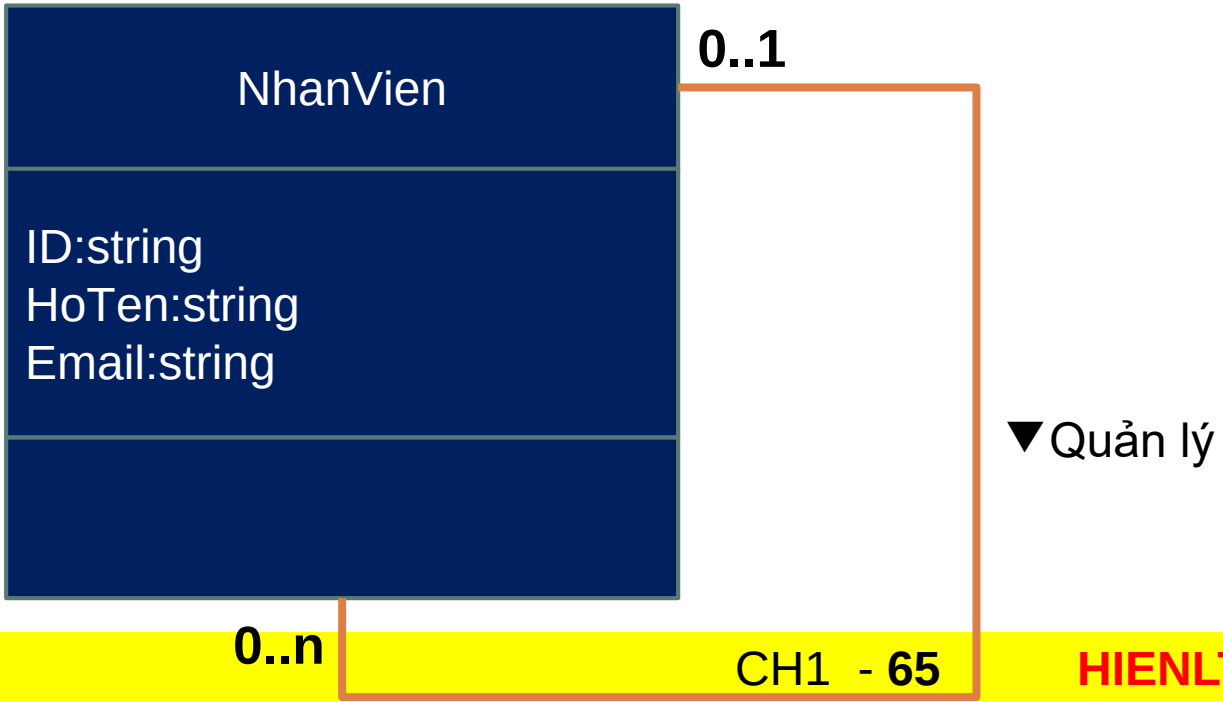
- For a given association type X between classes A and B, the term ***multiplicity*** refers to **the number of objects** of type A that may be associated with a given instance of type B
- There are three basic categories: one-to-one, one-to-many, and many-to-many



Some special forms of association

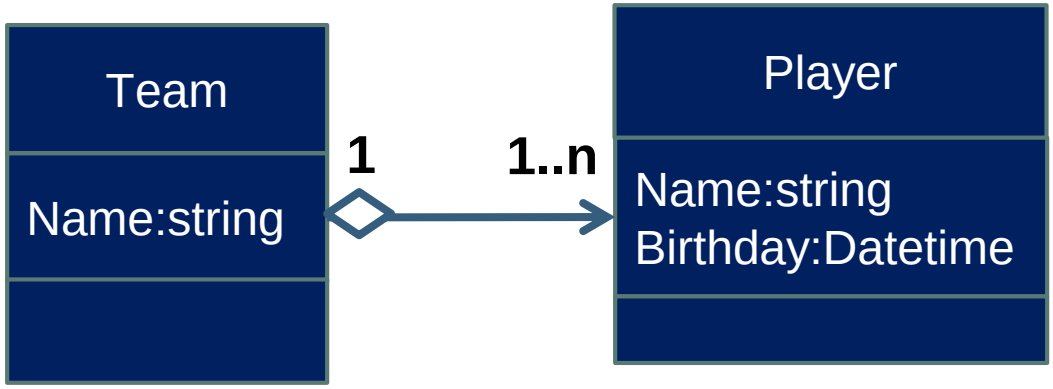
Reflexive Association (quan hệ phản thân)

- Đây là trường hợp đặc biệt của quan hệ kết hợp
- Là mối quan hệ mà 1 lớp đối tượng có quan hệ với chính nó
- Do vậy, người ta còn gọi là recursive association

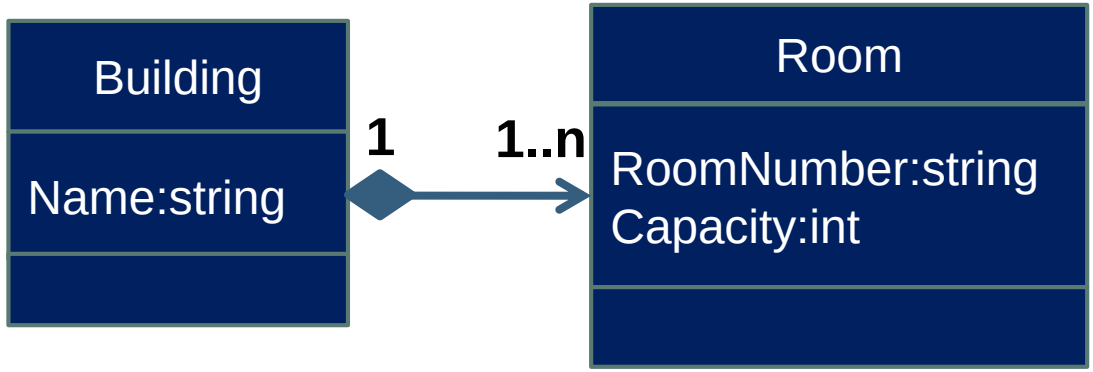


Aggregation relationship

- Aggregation:
  - Is normally understood as a "has-a" relationship
  - Both the entities continue to have their own independent existence



- Composition:
  - Is normally understood as a “part of" relationship
  - The 'part' entity doesn't have its own independent existence



Generalization relationship

Inheritance is the principle that you can apply your knowledge of a general category to more specific objects

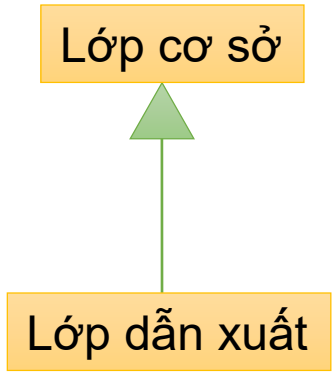
When you create a class by making it **inherit** from another class, you are provided with data fields and methods **automatically**; you can **reuse** fields and methods that are already written and tested.

Khái niệm về kế thừa

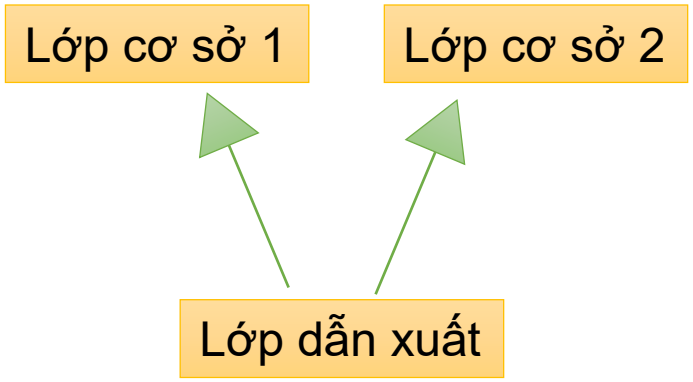
- Biểu diễn mối quan hệ “cha – con” (hay “1 dạng của”) giữa các lớp đối tượng
- Lớp mang ý nghĩa khái quát được gọi là lớp cha, lớp cơ sở (base class)
- Lớp mang ý nghĩa chi tiết, cụ thể gọi là lớp con, lớp dẫn xuất (derived class)
- Kế thừa tạo khả năng xây dựng lớp mới từ lớp đã có
- Kế thừa giúp tận dụng mã nguồn đã có
- Kế thừa giúp dễ dàng sửa chữa, nâng cấp, mở rộng hệ thống

Phân loại kế thừa

- Có 2 loại kế thừa: đơn thừa kế và đa thừa kế
  - Đơn thừa kế: một lớp chỉ kế thừa từ 1 lớp cơ sở
  - Đa thừa kế: một lớp được kế thừa từ nhiều lớp cơ sở
- Trong C#, không hỗ trợ đa thừa kế (có thể dùng khái niệm interface để khắc phục vấn đề này)



Đơn thừa kế



Đa thừa kế

Một số nguyên tắc trong kế thừa

- Các thành phần của lớp dẫn xuất (lớp con) sẽ bao gồm:
  - Các thành phần được khai báo ở lớp dẫn xuất
  - Các thành phần được khai báo ở lớp cơ sở (lớp cha)
- Lớp dẫn xuất không được quyền xóa đi những thành phần đã được khai báo ở lớp cơ sở

Mô hình biểu diễn

Mối quan hệ kế thừa

- Có thể kế thừa lớp đối tượng:
  - Khi cần bổ sung thêm thành phần cho lớp cơ sở
  - Khi cần chuyên biệt hóa các phương thức xử lý của lớp cơ sở

Một số lời khuyên khi dùng quan hệ kế thừa

- Khi kế thừa lớp đối tượng, ĐỪNG:
  - Thay đổi ngữ nghĩa của các thành phần của lớp cha
  - Loại bỏ 1 số thành phần của lớp cha

5. Phân tích Thiết kế Hướng đối tượng

5.3. Thế nào là phân tích hướng đối tượng

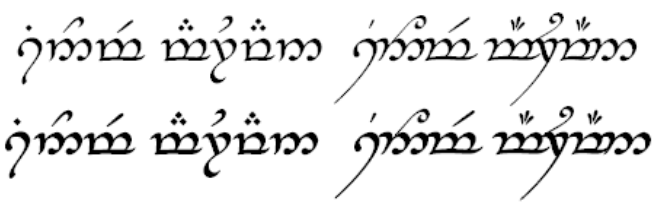
What is OOAD?

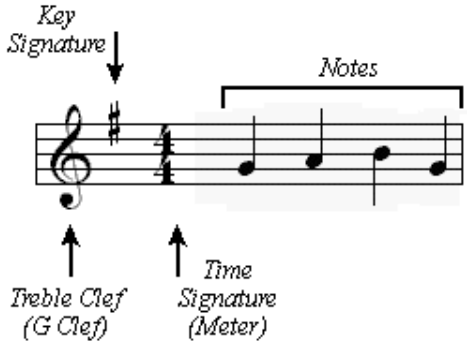
- Analysis.
- Design.
- OOAD

Involves both a notation and a process

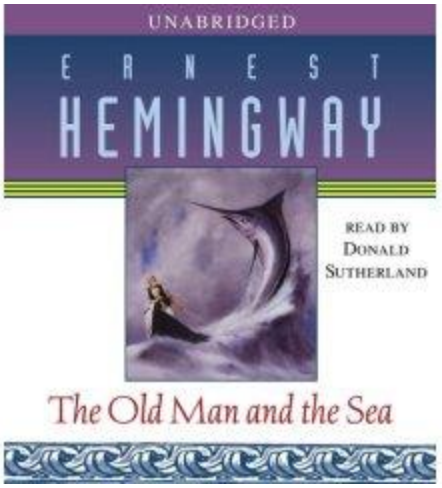
How to do OOAD
- notation vs. process

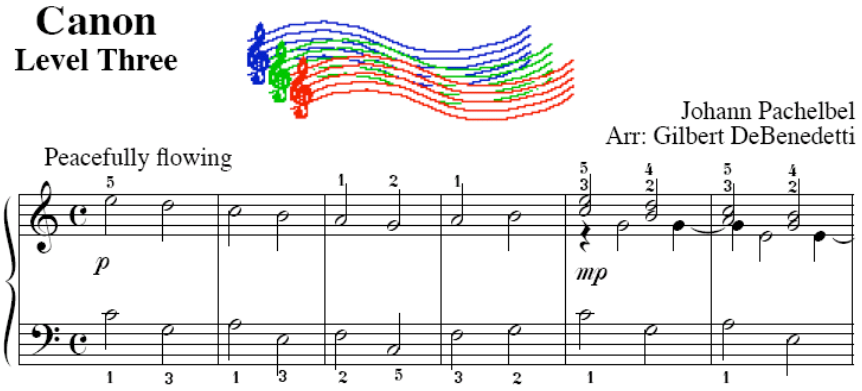
- UML is a notation.
- So are English, Elvish, Ku, …





- But as yet I can't





5. Phân tích Thiết kế Hướng đối tượng

5.3. Thế nào là phân tích hướng đối tượng

A Unified Language + A Good Process + *A Good Goal, perhaps*

UNIFIED MODELING LANGUAGE

5. Phân tích Thiết kế Hướng đối tượng

5.3. Thế nào là phân tích hướng đối tượng

Introduction to OOAD - Summary

Why

- Once Software Crisis due to Communication and Complexity
- Languages, Concepts, Models
- OO for Conceptual Modeling

What

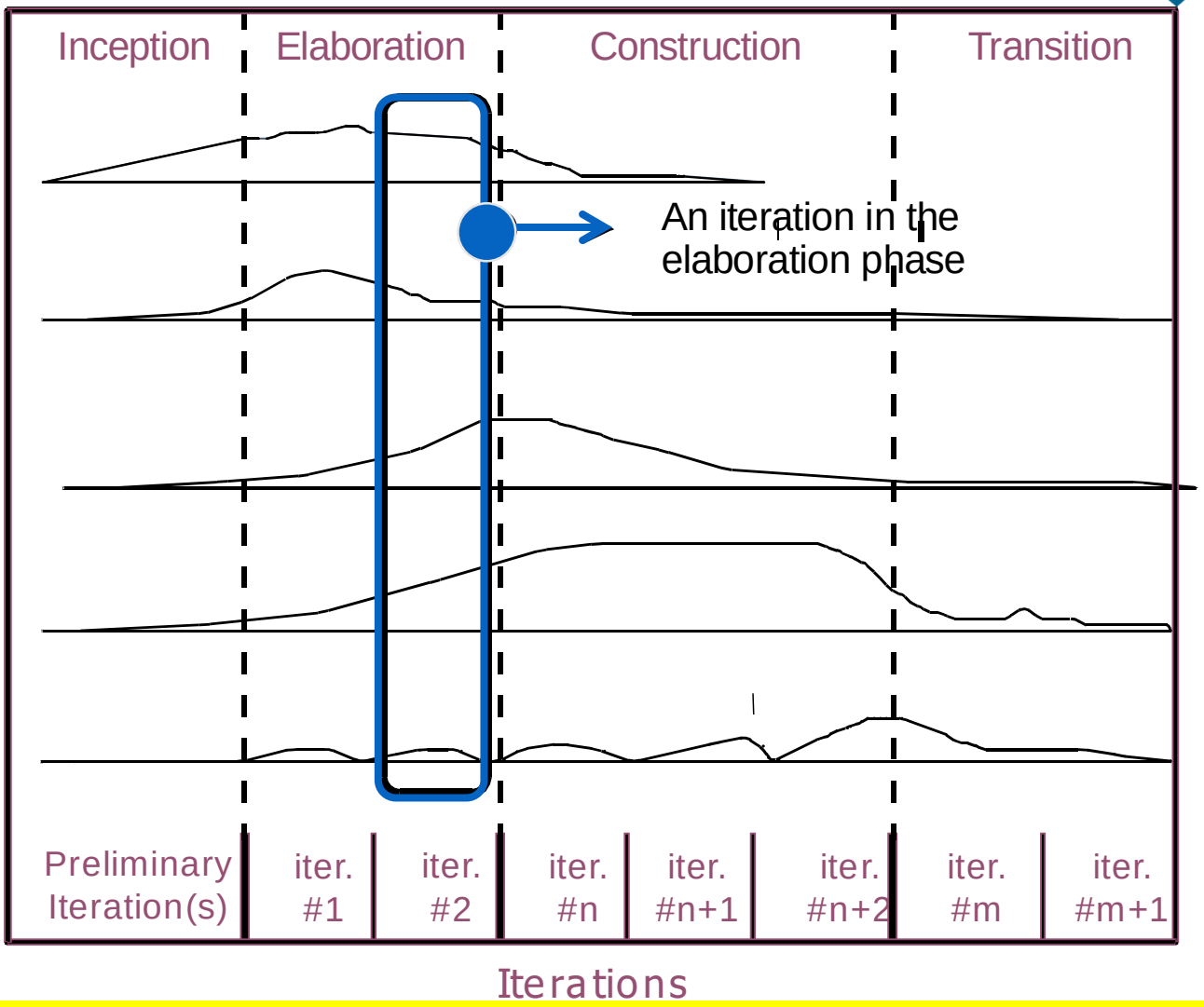
- Fundamental OO Concepts
- A little taste of UML

How

- OO development processes & (Design) Patterns

5. Phân tích Thiết kế Hướng đối tượng

5.4. Chu trình Phát triển Hệ thống Hướng đối tượng



5. Phân tích Thiết kế Hướng đối tượng

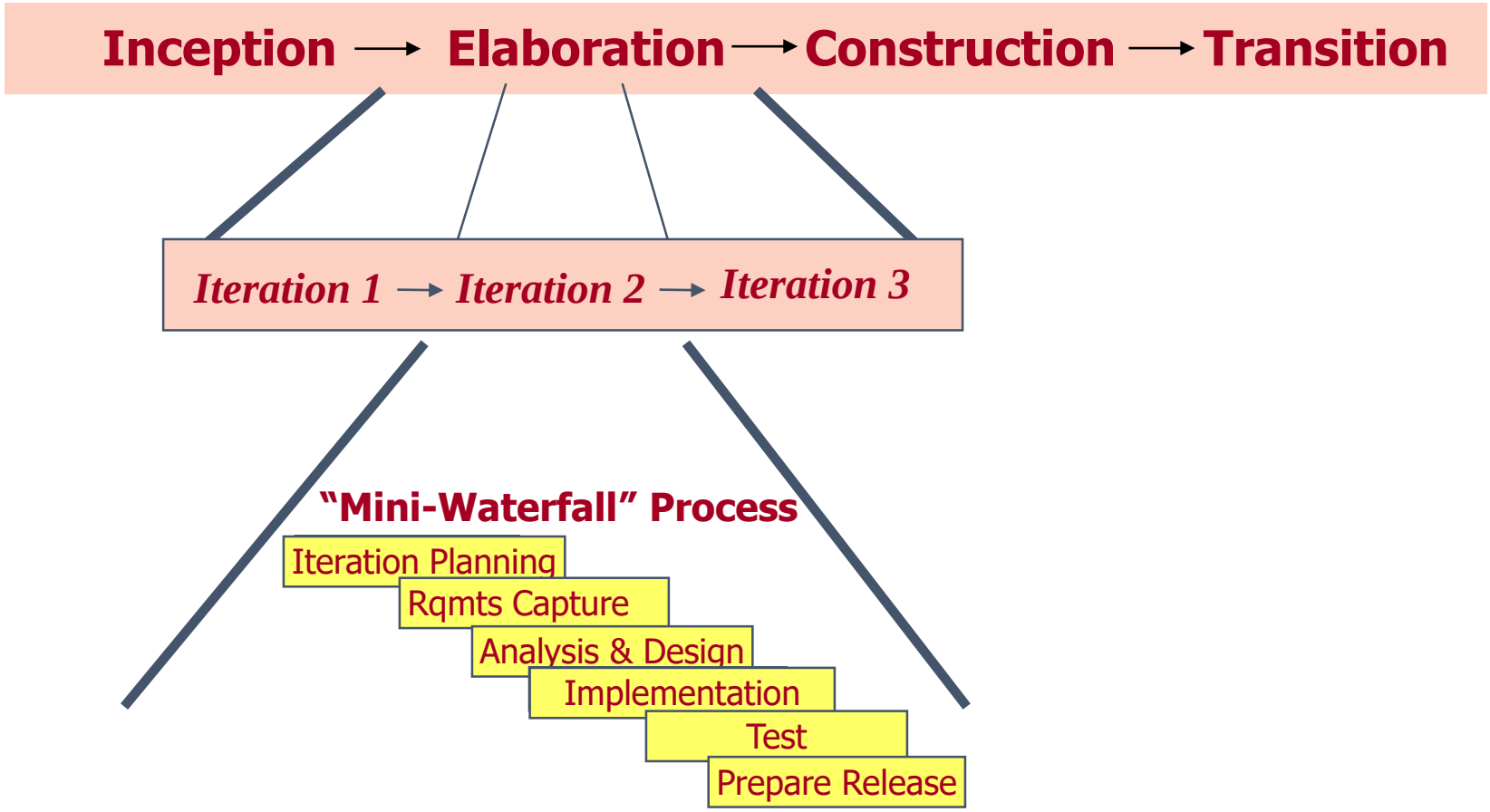
5.4. Chu trình Phát triển Hệ thống Hướng đối tượng

Các pha của chu trình

- Inception — Define the scope of the project and develop business case
- Elaboration — Plan project, specify features, and baseline the architecture
- Construction — Build the product
- Transition — Transition the product to its users

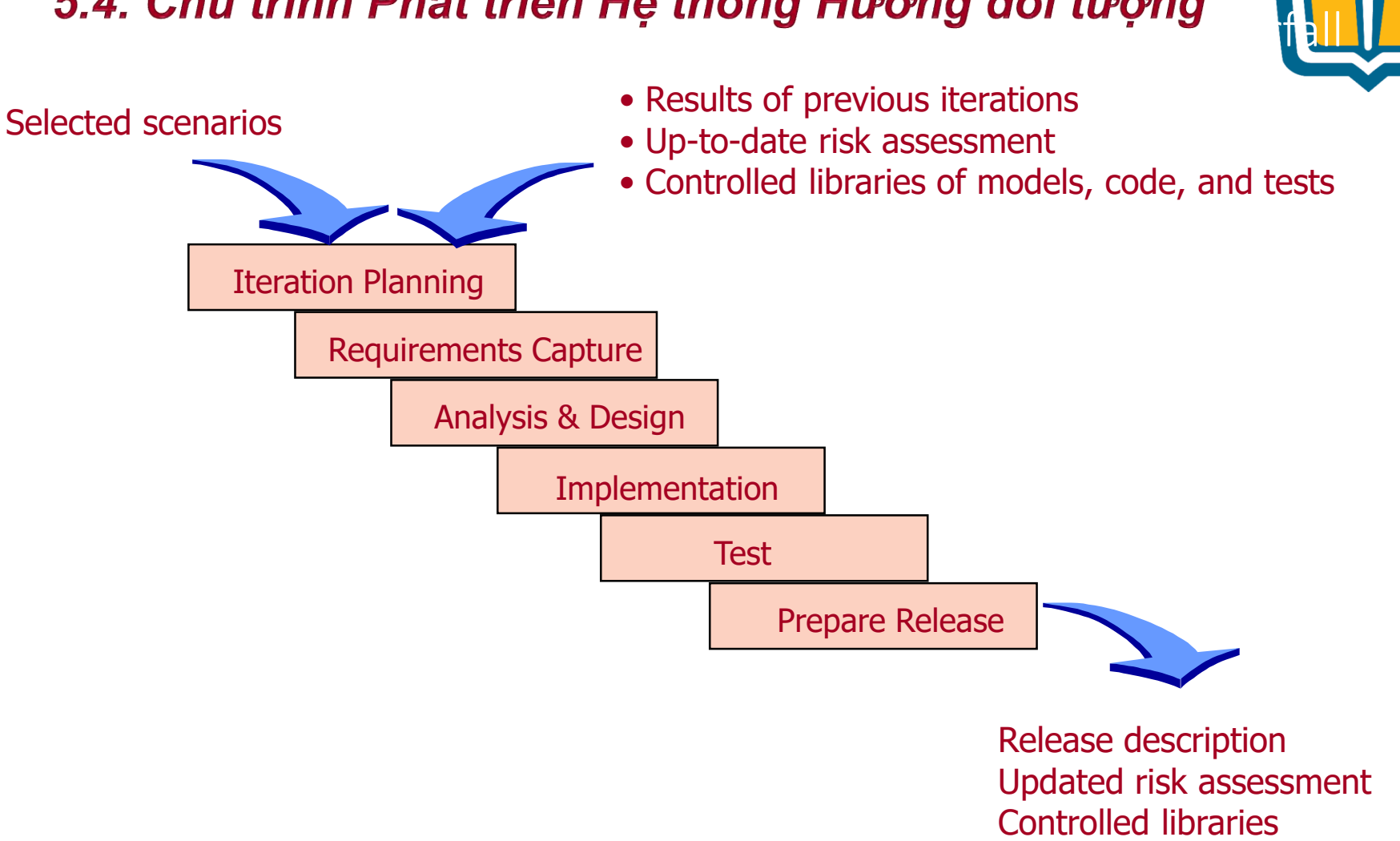
5. Phân tích Thiết kế Hướng đối tượng

5.4. Chu trình Phát triển Hệ thống Hướng đối tượng



5. Phân tích Thiết kế Hướng đối tượng

5.4. Chu trình Phát triển Hệ thống Hướng đối tượng



5. Phân tích Thiết kế Hướng đối tượng

5.4. Chu trình Phát triển Hệ thống Hướng đối tượng

Các hoạt động của lặp

- Kế hoạch lặp
  - Trước khi lặp bắt đầu thực hiện, mục tiêu chính của lặp cần được hình thành trên cơ sở
    - Các kết quả của các lặp trước (nếu có)
    - Cập nhật đánh giá rủi ro của dự án
  - Xác định tiêu chí đánh giá cho lặp này
  - Chuẩn bị kế hoạch chi tiết cho lặp
    - Bao gồm intermediate milestones để điều khiển tiến trình
    - Bao gồm walkthroughs và reviews

5. Phân tích Thiết kế Hướng đối tượng

5.4. Chu trình Phát triển Hệ thống Hướng đối tượng

Các hoạt động của vòng đời lặp

- Requirements Capture
- Analysis & Design
- Implementation
- Test
- Prepare the release description

5. Phân tích Thiết kế Hướng đối tượng

5.4. Chu trình Phát triển Hệ thống Hướng đối tượng

Các hoạt động của vòng đời lặp

- Requirements Capture
- Analysis & Design
- Implementation
- Test
- Prepare the release description

5. Phân tích Thiết kế Hướng đối tượng

5.4. Chu trình Phát triển Hệ thống Hướng đối tượng

Ích lợi của tiếp cận lặp

- Compared to the traditional waterfall process, the iterative process has the following advantages:
  - Risks are mitigated earlier
  - Change is more manageable
  - Higher level of reuse
  - The project team can learn along the way
  - Better overall quality

5. Phân tích Thiết kế Hướng đối tượng

5.4. Chu trình Phát triển Hệ thống Hướng đối tượng

– A New Paradigm with Evolving Object Orientation

- OOP: Object-Oriented Programming
  - Simula (1967), Smalltalk (70's), C++ (mid 80's), Eiffel, Ada95, Turing, …
- OOD: Object-Oriented Design
  - Taxis (1976), Adaplex, …, Grady Booch (1980)
- OOA: Object-Oriented Requirements
  - RML (1981), James Rumbaugh (late 80's)
- OO-Databases (OODBs): 1980-90's
- OLE/DCOM, VisualBasic, CORBA, Java: mid 90's
- .Net, C#, (eb/voice…/-)XML, J2EE: into 2000+
- UML: mid 90's and still evolving

Câu hỏi và thảo luận

Thank you!!!