## Chapter 1: Giới thiệu về PTTKHT

### Giới thiệu:

* Vòng phát triển hệ thống là quá trình hiểu hệ thống thông tin có thể hỗ trợ nghiệp vụ như thế nào bằng cách thiết kế hệ thống, xây dựng nó và đưa nó tới người sử dụng
* Key person trong 1 vòng đời sp là người pttk, người có khả năng phân tích nghiệp vụ, xác định thuận lợi để củng cố, thiết kế hệ thống để thực hiện nó
* Nhiệm vụ chính của pttk không phải là tạo ra một hệ thống hoàn hảo, thay vào đó, nếu nó tạo ra lợi nhuận cho cty. Nhiều dự án thất bại do người phân tích cố gắng tạo ra 1 hệ thống hoàn hảo nhưng không rõ ràng để lập trình.

### Vòng đời phát triển hệ thống

* Lập kế hoạch (planning)
* Phân tích (Analysis)
* Thiết kế (Design)
* Thực hiện (Implementation)

### Phương pháp phát triển phần mềm

* Structured Design
  + Waterfall
  + Parallel
  + Phased
* Rapid Application Development (RAD)
  + Protyping
  + Throwaway Protyping
* Agile Development
  + XP

### Phân tích thiết kế hệ thống hướng đối tượng

* Ca sử dụng dẫn dắt (Usecase Driven)
* Kiến trúc trung tâm (Architecture Centric)
* Lặp và tăng (Iterative and Incremental)

### The Unified Process

* Pha
* Dòng việc

### Vai trò và kỹ năng của nhóm dự án

* Phân tích nghiệp vụ
* Phân tích hệ thống
* Phân tích cơ sở hạ tầng
* Change Management Analyst
* Quản lý dự án (PM)

### Tổng kết

* The Systems Development Life Cycle: Tất cả dự án phát triển hệ thống đều tuân theo những tiến trình cơ bản gọi là SDLC. SDLC bắt đầu với pha lập kế hoạch là quá trình định nghĩa lợi nhuận, kiểm tra tính khả thi và lập kế hoạch. Pha thứ 2 là pha phân tích là quá trình đội phát triển phân tích chiến lược, thu thập thông tin và xây dựng mô hình phân tích. Trong pha tiếp thoe là pha thiết kế, đội phát triển thiết kế vật lý, thiết kế kiến trúc, thiết kế giao diện, csdl, tệp và chương trình. Pha cuối – thực thi, hệ thống được xây dựng, cài đặt và bảo trì.
* Sự phát triển của các phương pháp phát triển hệ thống: các phương pháp phát triển hệ thống (System Development Methodologies) là cách tiếp cận chính thức để thực thi 1 SDLC. SDM phát triển qua hàng thập kỷ. Thiết kế cấu trúc, như mô hình thác nước hoặc song song,

The Evolution of Systems Development Methodologies

System development methodologies are formalized approaches to implementing an SDLC.

System development methodologies have evolved over the decades. Structured design

methodologies, such as waterfall and parallel development, emphasize decomposition of a

problem by either focusing on process decomposition (process-centric methodologies) or

data decomposition (data decomposition). They produce a solid, well-thought-out system

but can overlook requirements because users must specify them early in the design process

before seeing the actual system. RAD-based methodologies attempt to speed up development and make it easier for users to specify requirements by having parts of the system

developed sooner either by producing different versions (phased development) or by using

prototypes (prototyping, throwaway prototyping) through the use of CASE tools and

fourth-generation/visual programming languages. However, RAD-based methodologies

still tend to be either process-centric or data-centric. Agile development methodologies,

such as XP, focus on streamlining the SDLC by eliminating many of the tasks and time

associated with requirements definition and documentation. Several factors influence the

choice of a methodology: clarity of the user requirements; familiarity with the base technology; system complexity; need for system reliability; time pressures; and the need to see

progress on the time schedule.

# Phần 1: Khởi tạo dự án, quản lý dự án và lấy yêu cầu

## Chapter 2: Khởi tạo dự án

## Chapter 3: Quản lý dự án

## Chapter 4: Lấy yêu cầu

# Phần 2: Mô hình phân tích

## Chapter 5: Mô hình hàm

## Chapter 6: Mô hình cấu trúc

## Chapter 7: Mô hình hành vi

# Phần 3: Mô hình thiết kế

## Chapter 8: Đi vào thiết kế

Design contains many steps that guide the project team through planning out exactly how

a system needs to be constructed. The requirements that were identified and the models

that were created during analysis serve as the primary inputs for the design activities. In

object-oriented design, the primary activity is to evolve the analysis models into design

models by optimizing the problem-domain information already contained in the analysis

models and adding system environment details to them.

Thiết kế bao gồm nhiều bước mà dẫn dắt đội dự án xuyên suốt quá trình lập kế hoạch, làm thế nào để xây dựng hệ thống. Các yêu cầu đã được xác định và mô hình đã được xây dựng trong quá trình phân tích giống như các nguyên liệu chính cho quá trình thiết kế. Trong thiết kế hướng đối tượng, mục đích chính là đưa các mô hình phân tích thành mô hình thiết kế qua việc tối giản các miền vấn đề chứa trong mô hình phân tích và thêm vào môi trường chi tiết.

Verifying and Validating the Analysis Models

Before actually adding system environment details to the analysis models, the various representations need to be verified and validated. One very useful approach to test the fidelity

of the representations is to perform a walkthrough in which developers walk through the

representations by presenting the different models to members of the analysis team, members of the design team, and representatives of the client. The walkthrough must validate

each model to be sure that the different representations within the model all agree with each

other; for example, for the functional model, activity diagrams, use-case descriptions, and

use-case diagrams must be consistent with each other. Furthermore, the different models

(functional, structural, and behavioral) must be consistent. Finally, care must be taken during the walkthroughs to ensure that the presenter is not simply degraded and destroyed.

Evolving the Analysis Models into Design Models

When evolving the analysis models into design models, the analysis models: activity diagrams, use-case descriptions, use-case diagrams, CRC cards, class and object diagrams,

sequence diagrams, communication diagrams, and behavioral state machines should first be

carefully reviewed. During this review, factoring, refinement, and abstraction processes can

be used to polish the current models. During this polishing, it is possible that the analysis

models may become overly complex. If this occurs, then the models should be partitioned

based on the interactivity (message sending) and relationships (generalization, aggregation,

and association) shared among the classes. The more a class has in common with another

class (i.e., the more relationships shared), the more likely they belong in the same partition.

The second thing to do to evolve the analysis mode is to add the system environment

(physical architecture, user interface, and data access and management) information to the

problem domain information already contained in the model. To accomplish this and to

control the complexity of the models, layers are used. A layer represents an element of the

software architecture of the system. We recommend five different layers to be used: foundation, physical architecture, human–computer interaction, data access and management,

and problem domain. Each layer supports only certain types of classes (e.g., database

manipulation classes would be allowed only on the data access and management layer).

Packages and Package Diagrams

A package is a general UML construct used to represent collaborations, partitions, and layers.

Its primary purpose is to support the logical grouping of other UML constructs together (e.g.,

use cases and classes by the developer and user to simplify and increase the understandability

of a UML diagram). There are instances in which a diagram that contains only packages is

useful. A package diagram contains packages and dependency relationships. A dependency

relationship represents the possibility of a modification dependency existing between two

packages (i.e., changes in one package could cause changes in the dependent package).

Identifying packages and creating a package diagram is accomplished using a five-step

process. The five steps can be summed up as setting the context, clustering similar classes,

placing the clustered classes into a package, identifying dependency relationships among

the packages, and placing the dependency relationship on the package diagram.

Design Strategies

During the design phase, the project team also needs to consider three approaches to creating the new system, including developing a custom application in-house, buying a packaged system and customizing it, and relying on an external vendor, developer, or system

provider to build and/or support the system.

Summary 313

Custom development allows developers to be flexible and creative in the way they solve

business problems, and it builds technical and functional knowledge within the organization. But, many companies have a development staff that is already overcommitted to filling huge backlogs of systems requests, and they just don’t have time to devote to a project

where a system is built from scratch. It can be much more efficient to buy programs that

have been created, tested, and proven, and a packaged system can be bought and installed

in a relatively short period of time as compared with a custom solution. Workarounds can

be used to meet the needs that are not addressed by the packaged application.

The third design strategy is to outsource the project and pay an external vendor, developer, or service provider to create the system. It can be a good alternative to approaching

the new system; however, it does not come without costs. However, if a company decides to

leave the creation of a new system in the hands of someone else, the organization could

compromise confidential information or lose control over future development.

Each of the design strategies discussed here has its strengths and weaknesses, and no

one strategy is inherently better than the others. Thus, it is important to consider such

issues as the uniqueness of business need for the system, the amount of in-house experience that is available to build the system, and the importance of the project skills to the

company. Also, the existence of good project management and the amount of time available to develop the application play a role in the selection process.

Developing the Actual Design

Ultimately, the decision must be made regarding the specific type of system that needs to

be designed. An alternative matrix can help make this decision by presenting feasibility

information for several candidate solutions in a way in which they can be compared easily.

Both the Request for Proposal and Request for Information are two ways to gather accurate information regarding the alternatives.

314 Chapter 8Moving on to Design

## Chapter 9: Thiết kế lớp và phương thức

## Chapter 10: Thiết kế layer quản lý dữ liệu

## Chapter 11: Tương tác người – máy

## Chapter 12: Thiết kế kiến trúc vật lý