Requirements Engineering
(Kỹ thuật yêu cầu)

Người dịch: Minh Nhựt

Software-Intensive Systems

(Hệ thống chuyên sâu về phần mềm)

- **Software (on its own) is useless**

(Phần mềm (chỉ có mình nó) là vô dụng

- Software is a set of computations

(Phần mềm là một bộ phận tính toán)

- Software only becomes useful when run on some hardware

(Phần mềm chỉ trở nên có ích khi nó được chạy trên phần cứng)

- Software + Hardware = "Computer System"

(Phần mềm + Phần cứng = "Hệ thốn máy tính")

- **A Computer System (on its own) is useless**

(Một Hệ thống Máy tính (chỉ mình nó) là vô dụng)

- Only useful in the context of some human activity that it can support

(Nó chỉ có ích trong ngữ cảnh của một số hoạt động mà nó có thể hỗ trợ cho con người)

- Software + Hardware + Human Activities = "Software-Intensive System"

(Phần mềm + Phần cứng + Hoạt động của con người = "Hệ thống chuyên sâu về phần mềm")

Quality = Fitness for purpose

(Chất lượng = Sự phù hợp của mục đích)

- **Software is designed for a purpose**

(Phần mềm được thiết kế cho một mục đích xác định)

- If it doesn't work well then either

(Nếu nó không làm việc tốt hơn sau đó)

- The designer didn't have an adequate understanding of the purpose

(Nhà thiết kế không hiểu một cách đầy đủ về mục đích)

- or we are using the software for a purpose different from the intended one

(hoặc là do chúng ta đang sử dụng phần mềm cho một mục đích khác với ý định ban đầu)

- Requirements analysis is about identifying this purpose

(Phân tích yêu cầu phần mềm là việc về xác định những mục đích đó)

- Inadequate or wrong understanding of the purpose leads to poor quality software

(Sự hiểu biết về mục đích không đầy đủ hoặc sai sẻ dẫn đến phần mềm kém chất lượng)

Quality = Fitness for purpose

(Chất lượng = Sự phù hợp của mục đích)

- **The purpose is found in human activities**

(Mục đích phải được tìm thấy trong những hoạt động của con người)

- E.g. Purpose of a student management system comes from the management activities of university and the needs of managers.

(Ví dụ: Mục đích của một hệ thống quản lý sinh vên đến từ những hoạt động quản lý của nhà trường và nhu cầu của những người quản lý.)

- **The purpose is often complex:**

(Mục đích là thường là phức tạp)

- Many different kinds of people and activities.

(Có nhiều loại hoạt động và con người khác nhau.)

- Conflicting interests among them.

(Sự xung đột lợi ích trong số những người đó.)

Where are the challenges?

(Những thách thức nằm ở đâu?)

Which systems are soft?

(Những hệ thống nào là phần mềm?)

- **Generic software components**

(Các thành phần chung của phần mềm)

- E.g. Core operating system functions, network services, …

(Ví dụ: Các chức năng lõi của hệ điều hành, dịch vụ mạng, …)

- Functionality relatively stable, determined by technical interfaces.

(Chức năng tương đối ổn định, được xác định bởi giao tiếp kỹ thuật.)

- But note that these systems still affect human activity.

(Nhưng cần chú ý rằng hệ thống này vẫn phải ảnh hưởng đến các hoạt động của con người.)

- E.g. concepts of a ‘URL’, etc.

(Ví dụ: Các khái niệm về ‘URL’, ….)

Which systems are soft?

(Những hệ thống nào là phần mềm?)

- **Control Systems**

(Hệ thống điều khiển)

- E.g. aircraft flight control, industrial process control, …

(Ví dụ: điều chuyến bay, kiểm soát quy trình công nghiệp, ….)

- Most requirements determined by the physical processes to be controlled.

(Ví dụ: Đa phần các yêu cầu điều được xác định bởi các tiến quy trình điều khiển vật lý bên ngoài.)

- But note that operator interaction is usually crucial

(Nhưng cần lưu ý rằng sự tương tác với nhà điều hành thường là rất quan trọng)

- E.g. accidents caused when the system doesn't behave as the operator expected.

(Ví dụ: Xảy ra tai nạn khi hệ thống không hoạt động như mong đợi của nhà điều hành.)

Which systems are soft?

(Những hệ thống nào là phần mềm?)

- **Information Systems**

(Hệ thống Thông tin)

- E.g. office automation, web services, business support,…

(Ví dụ: Tự động hóa công việc văn phòng, dịch vụ web, hỗ trợ doanh nghiệp, ….)

- These systems cannot be decoupled from the activities they support.

(Những hệ thống này không thể tách khỏi những hoạt động mà nó hỗ trợ.)

- Design of the software entails design of the human activity.

(Thiết kế của một phần mềm đòi hỏi thiết kế cho hoạt động của con người.)

- The software and the human activities co-evolve

(Phần mềm và các hoạt động của con người phải tiến triển đồng thời)

Definition of Requirement

(Định nghĩa yêu cầu)

- **Requirements are a specification of what should be implemented. They are descriptions of how the system should behave, or of a system property or attribute. They may be a constraint on the development process of the system.**

(Yêu cầu là một đặc tả của những gì nên được thực hiện. Chúng là những miêu tả cho việc hệ thống nên được hoạt vận hành như thế nào, hoặc một tính chất của hệ thống hoặc là thuộc tính. Chúng có thể là một phạm vi hạn chế về quá trình phát triển của hệ thống.)

Levels and types of requirements (Cấp độ và loại của các yêu cầu)

Business requirement (Yêu cầu nghiệp vụ)	A high-level business objective of the organization that builds a product or of a customer who procures it. (Là một mục tiêu kinh doanh cấp cao của tổ chức xây dựng một sản phẩm hoặc của một khách hàng mua sắm nó.)
Business rule (Quy tắc nghiệp vụ)	A policy, guideline, standard, or regulation that defines or constrains some aspect of the business. Not a software requirement in itself, but the origin of several types of software requirements. (Là một chính sách, hướng dẫn, tiêu chuẩn, hoặc quy định xác định hoặc một số khía cạnh hạn chế của nghiệp vụ. Không phải là một yêu cầu phần mềm trong nó tạo ra, nhưng là nguồn gốc của một vài loại yêu cầu phần mềm.)
Constraint (Khó khăn)	A restriction that is imposed on the choices available to the developer for the design and construction of a product. (Là một sự hạn chế được áp đặt vào các chọn lựa có sẵn buộc nhà phát triển thiết kiến và xây dựng cho một sản phẩm.)
External interface requirement (Yêu cầu giao tiếp bên ngoài)	A description of a connection between a software system and a user, another software system, or a hardware device. (Là một mô tả của một kết nối giữa một hệ thống phần mềm và một người dùng, một hệ thống phần mềm khác, hoặc một thiết bị phần cứng.)
Feature (Tính năng)	One or more logically related system capabilities that provide value to a user and are described by a set of functional requirements. (Một hoặc nhiều sự hợp lý có liên quan đến khả năng hệ thống đó cung cấp giá trị đến một người dùng và miêu tả bởi một tập hợp của yêu cầu chức năng.)

Levels and types of requirements (Cấp độ và loại của các yêu cầu)

Functional requirement (Yêu cầu chức năng)	A description of a behavior that a system will exhibit under specific conditions. (Một mô tả của một hành mà một hệ thống đó sẻ thể hiện dưới điều kiện xác định.)
Nonfunctional requirement (Yêu cầu phi chức năng)	A description of a property or characteristic that a system must exhibit or a constraint that it must respect. (Một sự mô tả của một tính chất hoặc đặc điểm mà một hệ thống phải thể hiện hoặc một hạn chế mà hệ thống phải quan tâm.)
Quality attribute (Thuộc tính chất lượng)	A kind of nonfunctional requirement that describes a service or performance characteristic of a product. (Là một loại của yêu cầu phi chức năng đó mô tả một dịch vụ hoặc đặc trưng hiệu suất của một sản phẩm.)
System requirement (Yêu cầu hệ thống)	A top-level requirement for a product that contains multiple subsystems, which could be all software or software and hardware. (Là một yêu cầu mức cao của một sản phẩm đó là chứa đựng nhiều hệ thống con, sản phẩm đó có thể là tất cả phần mềm hoặc phần mềm và phần cứng.)
User requirement (Yêu cầu người dùng)	A goal or task that specific classes of users must be able to perform with a system, or a desired product attribute. (L à một mục tiêu hoặc nhiệm vụ mà các lớp cụ thể của người dùng phải khả năng để thực thi với một hệ thống, hoặc một biểu hiện mong muốn ở sản phẩm)

Levels and types of requirements

(Cấp độ và loại của các yêu cầu)

- **Software requirements include three distinct levels: business requirements, user requirements, and functional requirements.**

(Yêu cầu phần mềm bao gồm ba mức độ rõ rệt: yêu cầu nghiệp vụ, yêu cầu người dùng, và yêu cầu chức năng.)

Definition of RE

(Định nghĩa yêu cầu)

Cost of getting it wrong

(Chi phí bỏ ra nếu làm sai)

- **Cost of fixing errors**

(Chi phí của việc sửa các lỗi)

- Typical development process

(Qúa trình phát triển điển hình)

requirements analysis ⇒ software design ⇒ programming ⇒ development testing ⇒ acceptance testing ⇒ operation

(Phân tính yêu cầu => thiết kế phần mềm => Lập trình => Kiểm thử phát triển => Kiểm thử chấp nhận => Vận hành)

- **Errors cost more to fix the longer they are undetected**

(Chi phí sửa lỗi sẻ nhiều hơn nếu được phát hiện trễ)

- E.g. A requirements error found in testing costs 100 times more than a programming error found in testing

(Ví dụ: Một lỗi yêu cầu tìm thấy trong lúc kiểm thử có chi phí gấp 100 lần hơn một lỗi lập trình được tìm thấy kiểm thử)

Cost of getting it wrong

(Chi phí bỏ ra nếu làm sai)

Causes of project failure

(Nguyên nhân của dự án thất bại)

Causes of fails: (Nguyên nhân của thất bại)

1. Large problems (Vấn đề lớn)

2. Lack of training in software engineering (Thiếu tập huấn về công nghệ phần mềm)

3. Objectives are unclear (Mục tiêu không rõ ràng)

4. Requirement specifications are uncompleted, wrong (Đặc tả yêu cầu là chưa được hoàn thành, sai)

5. Changing requirements (Sự thay đổi yêu cầu)

6. Errors in designing and implementing phases (Lỗi trong thiết kế và sự thực hiện các giai đoạn)

7. Lack of plans (Thiếu kế hoạch)

Cost of getting it wrong

(Chi phí bỏ ra nếu làm sai)

- It is very cost if errors can't early detected in the development process

(Nó tốn rất nhiều chi phí nếu các lỗi không thể phát hiện trong quá trình phát triển)

- Boehm and Papaccio(1988) estimated:

(Ước tính)

requirements analysis ($1) ⇒ software design ($5) ⇒ programming ($10) ⇒
development testing ($20) ⇒ operation ($100)

- It is necessary to gather and analysis requirements carefully

(Cần thiết để thu thập và phân tích yêu cầu một cách cẩn thận)

What do Requirements Analysts do?

(Phân tích yêu cầu là làm gì ?)

- **Starting point**

(Điểm bắt đầu)

- Some notion that there is a “problem” that needs solving

(Một số quan niệm cho rằng có một “vấn đề” là cần giải quyết vấn đề đó)

- e.g. dissatisfaction with the current state of affairs

(Ví dụ: Sự không hài lòng với tình hình hiện tại của công việc)

- e.g. a new business opportunity

(Ví dụ: Một cơ hội kinh doanh mới)

- e.g. a potential saving of cost, time, resource usage, etc.

(Tiết kiệm chi phí, thời gian, sử dụng tài nguyên, ….)

- A Requirements Analyst is an agent of change

(Nhà phân tích yêu cầu là một tác nhân thay đổi)

What do Requirements Analysts do?

(Phân tích yêu cầu là làm gì ?)

- **The requirements analyst must:**

(Nhà phân tích yêu cầu phải)

- Identify the “problem”/”opportunity”

(Xác định “vấn đề”/”cơ hội”)

- Which problem needs to be solved? (identify problem Boundaries)

(Vấn đề nào cần được giải quyết? (xác định ranh giới vấn đề))

- Where is the problem? (understand the Context/Problem Domain)

(Vấn đề ở đâu? (Không hiểu được ngữ cảnh/Miền vấn đề))

- Whose problem is it? (identify Stakeholders)

(Vấn đề là ai? (xác định các bên liên quan))

- Why does it need solving? (identify the stakeholders’ Goals)

(Tại sao nó cần giải quyết? (xác định mục tiêu các bên liên quan))

- How might a software system help? (collect some Scenarios)

(Một hệ thống phần mềm có thể giúp đỡ như thế nào? (thu thập một số kịch bản))

- When does it need solving? (identify Development Constraints)

(Khi nào nó cần được giải quyết? (xác định thời hạn phát triển))

- What might prevent us solving it? (identify Feasibility and Risk)

(Cái gì có thể ngăn chặn chúng ta giải quyết nó? (Xác định tính khả thi và rủi ro)

Some observations about RE

(Một số quan sát về yêu cầu)

- **RE is not necessarily a sequential process**

(Yêu cầu không nhất thiết là một quá trình tuần tự)

- Don't have to write the problem statement before the solution statement.

(Không phải viết báo cáo vấn đề trước khi báo cáo giải pháp)

- (Re-)writing a problem statement can be useful at any stage of development

(viết báo cáo một vấn đề có giúp ích ở bất kỳ giai đoạn nào của việc phát triển)

- **The problem statement will be imperfect**

(Báo cáo vấn đề sẽ là không hoàn hảo)

- RE models are approximations of the world

(Các mẫu yêu cầu là xấp xỉ của thế giới)

- will contain inaccuracies and inconsistencies

(Sẽ chứa đựng sự không chính xác và không nhất quán)

- will omit some information

(Sẽ bỏ qua một số thông tin)

- analysis should reduce the risk that will cause serious problems…

(Phân tích nên giảm rủi ro đó sẽ là nguy cơ sẽ gây ra những vấn đề nghiêm trọng….)

Some observations about RE

(Một số quan sát về yêu cầu)

- **Perfecting a specification may not be cost-effective**

(Hoàn thiện một đặc điểm kỹ thuật có thể không hiệu quả về chi phí)

- Requirements analysis has a cost

(phân tích yêu là có chi phí)

- For different projects, the cost-benefit balance will be different

(Của những dự án khác nhau, sự cân bằng giữa chi phí và lợi ích sẽ khác nhau)

- **Problem statement should never be treated as fixed**

(Báo cáo vấn đề không bao giờ được coi là đã sửa nó)

- Change is inevitable, and therefore must be planned for

(Thay đổi là chắc chắn xảy ra, và do đó phải lên kế hoạch trước)

Describing a problem

(Mô tả một vấn đề)

- **E.g. “prevent unauthorized access to machines”**

(Ngăn chặn truy cập trái phép đến các máy)

What are requirements?

(Yêu cầu là gì?)

- **Domain Properties**

(Miền các thuộc tính)

- Things in the application domain that are true whether or not we ever build the proposed system.

(Những thứ trong miền ứng dụng là đúng dù chúng ta có xây dựng được hệ thống đề xuất hay không.)

- **Requirements**

(Các yêu cầu)

- Things in the application domain that we wish to be made true by delivering the proposed system.

(Những thứ trong miền ứng dụng đó chúng ta muốn làm đúng được thực hiện bằng cách cung cấp hệ thống được đề xuất.)

- Many of which will involve phenomena the machine has no access to

(Nhiều trong số đó sẽ liên quan đến máy không có quyền truy cập vào)

- **A Specification**

(Một đặc tả)

- is a description of the behaviours that the program must have in order to meet the requirements.

(Là một mô tả của các hành vi mà chương trình phải có để đáp ứng các yêu cầu)

Fitness for purpose?

(Sự phù hợp của mục đích?)

- **Two correctness (verification) criteria**

(Hai tiêu chuẩn chính xác)

- The Program running on a particular Computer satisfies the Specification.

(Chương trình khi đang chạy trên một máy tính cụ thể thì phải đáp ứng các đặc tả kỹ thuật)

- The Specification, in the context of the given domain properties, satisfies the requirements.

(Đặc tả, trong ngữ cảnh của các miền thuộc tính, đáp ứng các yêu cầu)

- **Two completeness (validation) criteria**

(Hai tiêu chuẩn hoàn chỉnh)

- We discovered all the important requirements.

(Chúng ta đã phát hiện tất cả các yêu cầu quan trọng)

- We discovered all the relevant domain properties.

(Chúng ta đã phát hiện tất cả các miền thuộc tính có liên quan)

- **Example**

But we can also move the boundaries…

(Nhưng chúng ta cũng có thể di chuyển ranh giới …)

- **E.g. Elevator control system:**

(Ví dụ: Hệ thống điều khiển thang máy)

- **We can shift things around:**

(Chúng ta có thể thay đổi những thứ xung quanh)

- E.g. Add some sensors to detect when people are waiting

(Ví dụ: Thêm một số cảm biến để phát hiện khi nào mọi người đang chờ đợi)

- This changes the nature of the problem to be solved

(Điều này làm thay đổi bản chất của vấn đề được giải quyết)

What is engineering?

(Kỹ thuật là gì?)

"Engineering is the development of cost-effectivesolutions to practical problems, through the application of scientific knowledge"

("Kỹ thuật là sự phát triển của chi phí – giải pháp hiệu quả đến vấn đề thực tế, thông qua việc áp dụng kiến thức khoa học")

- **"…cost-effective…"**

(Hiệu quả về chi phí)

- Consideration of design trade-offs, resource usage.

(Sự cân nhắc của sự cân bằng thiết kế, sử dụng tài nguyên)

- Minimize negative impacts (e.g. environmental and social cost)

(Giảm thiểu tác động tiêu cực (ví dụ: chi phí môi trường và xã hội))

- **"…solutions…"**

(Giải pháp)

- Emphasis on building devices

(Nhấn mạnh vào việc xây dựng các thiết bị)

- **"…practical problems…"**

(Vấn đề thực tế)

- solving problems that matter to people.

(Giải quyết những vấn đề quan trọng đối với mọi người)

- improving human life in general through technological advance.

(Nâng cao đời sống con người nói chung thông qua tiến bộ công nghệ)

Project Management

(Quản lý dự án)

- **A manager can control 4 things:**

(Một người quản lý có thể điều khiển 4 thứ:)

- Resources (can get more dollars, facilities, personnel)

(Tài nguyên (có thể được nhiều tiền, cơ sở vật chất, nhân sự))

- Time (can increase schedule, delay milestones, etc.)

(Thời gian (có thể tăng tiến độ kế hoạch, trì hoãn cột mốc, ….))

- Product (can reduce functionality)

(Sản phẩm (có thể giảm chức năng))

- Risk (can decide which risks are acceptable)

(Rủi ro (có thể quyết định rủi ro nào được chấp nhận))

Project Management

(Quản lý dự án)

- **To do this, a manager needs to keep track of:**

(Để làm điều này, một người quản lý cần chú ý đến:)

- Effort- How much effort will be needed? How much has been expended?

(Cố gắng- sẽ cần bao nhiêu nỗ lực? bao nhiêu chi tiêu đã dùng?)

- Time - What is the expected schedule? How far are we deviating from it?

(Thời gian – Lịch dự kiến là gì? Chúng ta chệch hướng bao xa từ nó?)

- Size - How big is the planned system? How much have we built?

(Kích thước – Hệ thống lên kế hoạch lớn bao nhiêu? Chúng ta mất bao lâu để xây dựng?)

- Defects - How many errors are we making? How many are we detecting?

(Khiếm khuyết – Chúng ta làm ra bao nhiêu lỗi? Chúng ta phát hiện ra bao nhiêu?)

- And how do these errors impact quality?

(Và những lỗi này ảnh hưởng đến chất lượng như thế nào?)

Project Management

(Quản lý dự án)

- **Requirements are the foundation for both the software development and the project management activities.**

(Yêu cầu là nền tảng cho cả việc phát triển phần mềm và các hoạt động quản lý dự án.)

- **Initially, a manager needs good estimates**

(Ban đầu, một người quản lý cần ước tính tốt)

- …and these can only come from a thorough analysis of the problem.

(... và những điều này chỉ có thể đến từ một phân tích toàn diện của vấn đề.)

You cannot control that which you cannot measure!

(Bạn không thể điều khiển những thứ bạn không thể đo lường!)

Project Types

(Các Loại dự án)

- **Reasons for initiating a software development project**

(Lý do để bắt đầu một dự án phát triển phần mềm)

- Problem-driven: competition, crisis,…

(Hướng vấn đề: cạnh tranh, khủng hoảng, ….)

- Change-driven: new needs, growth, change in business or environment,…

(Hướng Thay đổi: nhu cầu mới, tăng trưởng, thay đổi trong kinh doanh hoặc môi trường, ...)

- Opportunity-driven: exploit a new technology,…

(Hướng cơ hội: khai thác công nghệ mới, ….)

- Legacy-driven: part of a previous plan, unfinished work, …

(Hướng kế thừa: một phần trước đó của kế hoạch, công việc chưa hoàn thành, ….)

Project Types

(Các Loại dự án)

- **Relationship with Customer(s):**

(Mối quan hệ với khách hàng)

- Customer-specific - one customer with specific problem

(Khách hàng cụ thể – một khác hàng với những vấn đề cụ thể)

- May be another company, with contractual arrangement

(có thể là một công ty khác, với một thỏa thuận hợp đồng)

- May be a division within the same company

(có thể là một phần trong cùng công ty)

Project Types

(Các Loại dự án)

- **Market-based - system to be sold to a general market**

(Dựa vào thị phần – hệ thống được bán cho một thị trường chung)

- In some cases the product must generate customers

(Trong số các trường hợp sản phẩm phải tạo khách hàng)

- Marketing team may act as substitute customer

(Đội ngũ tiếp thị có thể thay thế như là một khách hàng)

- **Community-based - intended as a general benefit to some community**

(Dựa trên cộng đồng – như là một lợi ích chung cho cộng đồng)

- E.g. open source tools, tools for scientific research

(Ví dụ: các công cụ mã nguồn mở, các công cụ cho việc nghiên cứu)

- **Hybrid(a mix of the above)**

(Hỗn hợp (kết hợp của các mối quan hệ trên))

Lifecycle of an Engineering Project

(Chu kỳ sống của một dự án công nghệ)

- **Examples:**

- Sequential models: Waterfall, V model

(Mô hình tuần tự: MH thác nước, mô hình chữ V)

- Rapid Prototyping

(Tạo mẫu nhanh)

- Phased Models: Incremental, Evolutionary

(Mô hình phân chi giai đoạn: Tăng trưởng, thích nghi)

- Iterative Models: Spiral

(Mô hình lặp: xoắn ốc)

- Agile Models: eXtreme Programming

(Mô hình phát triển nhanh: Lập trình eXtreme)

Agile Models

- **Basic Philosophy**

(Nguyên lý cơ bản)

- Reduce communication barriers

(giảm thiểu rào cảng giao tiếp)

- Programmer interacts with customer

(Có sự tương tác giữa người lập trình và khách hàng)

- Reduce document-heavy approach

(Không nặng về tài liệu)

- Documentation is expensive and of limited use

(Tài liệu được dùng hệ chế và chiếm chi phí cao)

Agile Models

- Have faith in the people

(Có sự tin cậy của nhau)

- Don’t need process models to tell them what to do!

(Không cần các mẫu tiến trình chỉ để nói công việc là phải làm gì!)

- Respond to the customer

(Có trách nhiệm với khách hàng)

- Rather than focusing on the contract

(Hơn là chú trọng trên bản ký kết)

Agile Models

- **Weaknesses**

(Những điểm yếu)

- Relies on programmer's memory

(Lập trình viên tin vào trí nhớ của họ)

- Code can be hard to maintain

(Mã lệnh có khó bảo trì)

- Relies on oral communication

(Phụ thuộc giao tiếp trao đổi bằng lời)

- Mis-interpretation possible

(Giải thích sai)

Agile Models

- Assumes single customer representative

(Chỉ có duy nhất một đại diện khách hàng)

- Multiple viewpoints not possible

(Có nhiều có nhiều quan điểm không thể làm)

- Only short term planning

(Chỉ lên kế hoạch với kỳ hạn ngắn)

- No longer term vision

(Không có tầm nhìn xa)

Software requirement process (cont.)

(Tiến trình yêu cầu phần mềm)

- **Feasibility study**

(Nghiên cứu tính khả thi)

- For all new systems.

(Cho tất cả các hệ thống mới)

- It is a set of preliminary business requirements.

(Là một tập hợp của các yêu cầu nghiệp vụ sơ bộ)

- An outline description of the system and how the system is intended to support business processes.

(Một miêu tả phác thảo của hệ thống và hệ thống được dự định để hỗ trợ cho quy trình nghiệp vụ như thế nào)

=> The results of the feasibility study should be a report that recommends whether or not it is worth to carry out.

(Kết quả của việc nghiên cứu khả thi nên là một báo cáo tham khảo để quyết định có mang nó vào dự án hay không)

Software requirement process (cont.)

(Tiến trình yêu cầu phần mềm)

- **Requirements elicitation and analysis**

(Yêu cầu và phân tích yêu cầu)

- Software engineers work with customers and system end-users to find out about the application domain. What services the system should provide, the required performance of the system, hardware constraints, and so on.

(Các kỹ sư về phần mềm làm việc với các khách hàng và hệ thống người dùng đầu cuối để tìm ra miền ứng dụng. Những dich vụ gì nên cung cấp, hiệu suất của hệ thống, ràng buộc về phần cứng, và vân2)

Software requirement process (cont.)

(Tiến trình yêu cầu phần mềm)

- **Requirements elicitation and analysis**
  - The process activities are:

(Các quy trình hoạt động là:)

- Requirements discovery

(Khá phá yêu cầu)

- Requirements classification and organization

(Tổ chức và phân loại yêu cầu)

- Requirements prioritisation and negotiation

(Thỏa thuận và ưu tiên yêu cầu)

- Requirements documentation

(Tài liệu yêu cầu)

Software requirement process (cont.)

(Tiến trình yêu cầu phần mềm)

- **Requirements validation**

(Xác nhận yêu cầu)

- To show that the requirements actually define the system that the customer wants.

(Thể hiện yêu cầu đó thật sự xác định của khách hàng mong muốn)

Software requirement process (cont.)

(Tiến trình yêu cầu phần mềm)

- **Requirements validation**
- It is important because errors in a requirements document can lead to extensive rework costs. These documents should be checked:

(Điều quan trọng là bởi vì lỗi trong một tài liệu yêu cầu có thể dẫn đến nhiều chi phí làm lại. Các tài liệu này cần được kiểm tra)

- Validity checks

(Kiểm tra tính hợp lệ)

- Consistency checks: Requirements in the document should not conflict

(Kiểm tra tính nhất quán: Yêu cầu trong tài liệu không được xung đột)

- Completeness checks

(Kiểm tra tính toàn vẹn)

- Realism checks

(Kiểm tra tính hiện thực)

- Verifiability

(Kiểm chứng)