**CHƯƠNG 1: Tổng Quan về Yêu Cầu Phần Mềm**

**1.1. Mục Tiêu**

Chương này nhằm giúp người học:

* Hiểu các thuật ngữ chính trong lĩnh vực yêu cầu phần mềm (software requirements domain).
* Phân biệt yêu cầu sản phẩm (product requirements) và yêu cầu dự án (project requirements).
* Phân biệt phát triển yêu cầu (requirements development) và quản lý yêu cầu (requirements management).
* Nhận biết và cảnh giác với một số vấn đề thường gặp liên quan đến yêu cầu.

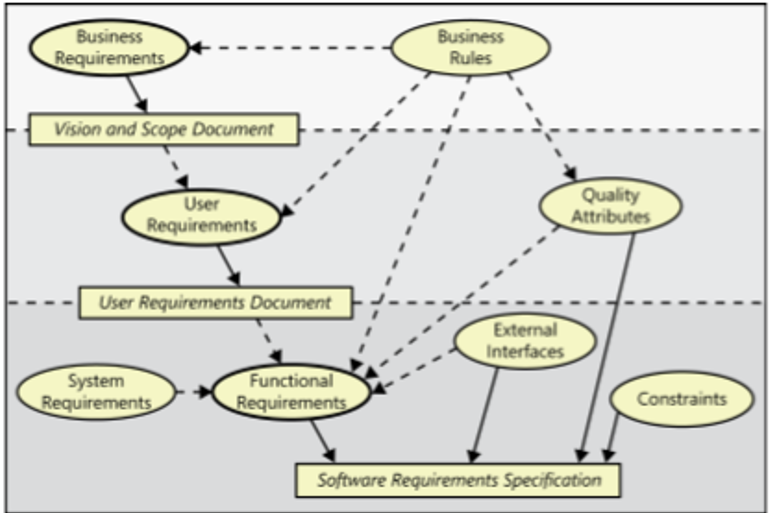
**1.2. Kỹ Thuật Yêu Cầu (Requirements Engineering) và Tầm Quan Trọng**

**1.2.1. Yêu cầu phần mềm là gì?**

Theo Brian Lawrence, yêu cầu là "bất cứ thứ gì định hướng cho các lựa chọn thiết kế" (anything that drives design choices). Ian Sommerville và Pete Sawyer định nghĩa yêu cầu là "một đặc tả về những gì nên được triển khai" (a specification of what should be implemented). Chúng mô tả cách hệ thống nên hoạt động hoặc một thuộc tính/đặc tính của hệ thống. Ngoài ra, chúng cũng có thể là một ràng buộc (constraint) đối với quá trình phát triển hệ thống.

**1.2.2. Phân loại Yêu cầu (Types of Requirements)**

Yêu cầu được phân cấp và phân loại dựa trên mục đích và phạm vi của chúng:

* **Business Requirement**: Mục tiêu kinh doanh cấp cao của tổ chức.
* **Business Rule**: Một chính sách, hướng dẫn hoặc quy định ràng buộc một khía cạnh của doanh nghiệp.
* **User Requirement**: Một mục tiêu hoặc tác vụ mà người dùng cần thực hiện với hệ thống.
* **System Requirement**: Yêu cầu cấp cao nhất cho một sản phẩm chứa nhiều hệ thống con.
* **Functional Requirement**: Mô tả một hành vi mà hệ thống sẽ thể hiện trong các điều kiện cụ thể.
* **Nonfunctional Requirement**: Mô tả một thuộc tính hoặc đặc tính mà hệ thống phải có.
* **Constraint**: Một hạn chế áp đặt lên các lựa chọn thiết kế và xây dựng sản phẩm.
* **External Interface Requirement**: Mô tả kết nối giữa hệ thống phần mềm với người dùng, hệ thống khác hoặc thiết bị phần cứng.
* 

**1.2.3. Tầm Quan Trọng của Yêu cầu**

Việc xác định và quản lý yêu cầu kém có thể dẫn đến hậu quả nghiêm trọng. Số liệu từ Martin cho thấy:

* **Phân bố Lỗi (Distribution of Defects)**: 56% lỗi phát sinh trong dự án đến từ giai đoạn yêu cầu. Con số này cao hơn nhiều so với thiết kế (27%), code (7%) và các yếu tố khác (10%).
* **Công sức để Sửa Lỗi (Effort to Fix Defects)**: Sửa một lỗi phát sinh từ yêu cầu chiếm tới 82% tổng công sức sửa lỗi. Trong khi đó, sửa lỗi từ thiết kế chỉ chiếm 13% và từ code chỉ là 1%.

Những con số này nhấn mạnh rằng việc đầu tư vào một quy trình yêu cầu chất lượng cao là vô cùng cần thiết để giảm thiểu rủi ro và chi phí dự án.

**1.3. Phân Biệt Yêu Cầu Sản Phẩm và Yêu Cầu Dự Án**

* **Yêu cầu sản phẩm (Product Requirements)**: Là các đặc tính (properties) của một hệ thống phần mềm cần được xây dựng.
* **Yêu cầu dự án (Project Requirements)**: Là các kỳ vọng và sản phẩm bàn giao (deliverables) không phải là một phần của phần mềm nhưng cần thiết để hoàn thành dự án một cách thành công. Ví dụ: kế hoạch dự án, báo cáo tiến độ, tài liệu đào tạo.

**1.4. Phát Triển và Quản Lý Yêu Cầu**

**Kỹ thuật Yêu cầu (Requirements Engineering)** là một quy trình lớn, bao gồm hai nhánh chính:

* **Phát triển Yêu cầu (Requirements Development)**: Quá trình xác định và định nghĩa các yêu cầu của hệ thống. Nhánh này bao gồm các hoạt động:
  + **Khơi gợi (Elicitation)**: Thu thập thông tin từ các bên liên quan (stakeholders).
  + **Phân tích (Analysis)**: Phân tích, làm rõ và tổ chức các yêu cầu.
  + **Đặc tả (Specification)**: Viết các yêu cầu thành tài liệu chính thức.
  + **Xác thực (Validation)**: Kiểm tra và xác nhận các yêu cầu.
* **Quản lý Yêu cầu (Requirements Management)**: Quá trình kiểm soát, theo dõi và duy trì các yêu cầu đã được xác định.

**1.5. Những Vấn Đề Thường Gặp**

Một số vấn đề phổ biến có thể phát sinh khi làm việc với yêu cầu:

* **Insufficient user involvement**: Không đủ sự tham gia của người dùng.
* **Inaccurate planning**: Lập kế hoạch không chính xác.
* **Creeping user requirements**: Yêu cầu của người dùng thay đổi liên tục.
* **Ambiguous requirements**: Yêu cầu mơ hồ, không rõ ràng.
* **Gold plating**: Thêm các tính năng không cần thiết.
* **Overlooked stakeholders**: Bỏ sót các bên liên quan quan trọng.

**1.6. Lợi Ích của Quy Trình Yêu Cầu Chất Lượng Cao**

Một quy trình yêu cầu được thực hiện tốt sẽ mang lại nhiều lợi ích cho dự án và sản phẩm:

* Ít lỗi hơn trong cả yêu cầu và sản phẩm bàn giao.
* Giảm công việc làm lại (rework) trong quá trình phát triển.
* Tăng tốc độ phát triển và bàn giao sản phẩm.
* Giảm chi phí nâng cấp và bảo trì.
* Giảm sự mơ hồ, nhầm lẫn trong giao tiếp.
* Hạn chế tình trạng "phạm vi dự án bị nới rộng không kiểm soát" (scope creep).
* Tăng sự hài lòng của khách hàng và đội ngũ.
* Sản phẩm đáp ứng đúng mục đích ban đầu.

Phần tích sơ đồ ở trên

**1. Cấp độ Tổng quát**

Đây là cấp độ đầu tiên, nơi các yêu cầu được định hình ở mức cao nhất.

* **Business Requirements (Yêu cầu kinh doanh):** Đây là những mục tiêu và nhu cầu cấp cao của tổ chức hoặc doanh nghiệp. Chúng thường liên quan đến lý do tại sao dự án được thực hiện, các lợi ích kinh doanh mong đợi, và các mục tiêu chiến lược mà dự án phải đáp ứng. Ví dụ: "Tăng doanh thu online lên 20% trong năm đầu tiên."
* **Business Rules (Quy tắc nghiệp vụ):** Đây là các quy tắc, chính sách, hoặc ràng buộc của tổ chức chi phối cách thức hoạt động của doanh nghiệp. Chúng ảnh hưởng đến cách phần mềm được thiết kế và hoạt động. Ví dụ: "Khách hàng phải có tài khoản để được giảm giá 10%." Nó khác constraints là : rules nó ràng buộc các quy định để liên quan đến kinh tế còn constraints thì liên quan đến các ràng buộc để làm ra sản phẩm
* **Vision and Scope Document (Tài liệu tầm nhìn và phạm vi):** Đây là tài liệu đầu ra từ Business Requirements. Nó xác định mục tiêu của dự án, phạm vi của hệ thống (những gì sẽ và sẽ không được xây dựng), và những người liên quan (stakeholders) chính. Tài liệu này đóng vai trò là "la bàn" định hướng cho toàn bộ dự án.

**2. Cấp độ Người dùng**

Sau khi xác định được các yêu cầu kinh doanh, cấp độ này sẽ tập trung vào nhu cầu của người dùng cuối.

* **User Requirements (Yêu cầu người dùng):** Đây là những gì người dùng cần làm với hệ thống. Chúng thường được mô tả dưới dạng các câu chuyện người dùng (user stories) hoặc các trường hợp sử dụng (use cases), tập trung vào chức năng từ góc độ của người dùng. Ví dụ: "Là một khách hàng, tôi muốn xem lịch sử mua hàng của mình."
* **User Requirements Document (Tài liệu yêu cầu người dùng):** Đây là tài liệu thu thập và tổ chức các User Requirements. Nó mô tả các tác vụ mà người dùng cần thực hiện và cách hệ thống sẽ hỗ trợ họ, thường ít đi sâu vào chi tiết kỹ thuật.
* **Quality Attributes (Thuộc tính chất lượng):** Còn được gọi là các yêu cầu phi chức năng (Non-Functional Requirements). Đây là những tiêu chí đánh giá chất lượng của hệ thống, không phải là chức năng cụ thể. Ví dụ: "Hệ thống phải hoạt động 24/7", "Thời gian phản hồi của trang chủ không quá 2 giây". Chúng ảnh hưởng đến thiết kế kiến trúc của phần mềm.

**3. Cấp độ Hệ thống**

Đây là cấp độ chi tiết nhất, chuyển đổi các yêu cầu ở cấp cao hơn thành các yêu cầu kỹ thuật mà nhóm phát triển có thể xây dựng.

* **System Requirements (Yêu cầu hệ thống):** Đây là những yêu cầu cụ thể, chi tiết về các chức năng và thuộc tính của hệ thống. Đây là sự cụ thể hóa của User Requirements và các yêu cầu phi chức năng. Chúng thường bao gồm các yêu cầu chức năng (functional) và phi chức năng (non-functional) chi tiết.
* **Functional Requirements (Yêu cầu chức năng):** Đây là mô tả chi tiết các tính năng cụ thể mà hệ thống phải thực hiện. Chúng mô tả "hệ thống làm gì". Ví dụ: "Hệ thống phải cho phép người dùng đăng nhập bằng email và mật khẩu", "Hệ thống phải xử lý thanh toán bằng thẻ tín dụng". Chúng được hình thành từ User Requirements và Business Rules.
* **External Interfaces (Giao diện bên ngoài):** Mô tả cách hệ thống của bạn tương tác với các hệ thống, thiết bị hoặc người dùng khác. Ví dụ: giao diện người dùng (UI), giao diện lập trình ứng dụng (API) với hệ thống thanh toán bên ngoài, hoặc các cổng giao tiếp với phần cứng.
* **Constraints (Ràng buộc):** Đây là các giới hạn hoặc điều kiện bắt buộc mà dự án và hệ thống phải tuân thủ. Ví dụ: "Phần mềm phải được xây dựng bằng ngôn ngữ Java", "Hệ thống phải tuân thủ các quy định bảo mật GDPR".
* **Software Requirements Specification (SRS) (Đặc tả yêu cầu phần mềm):** Đây là tài liệu cuối cùng, chi tiết nhất. Nó tổng hợp tất cả các loại yêu cầu đã được xác định ở các cấp độ trên, bao gồm Business, User, Functional, và Non-Functional Requirements. SRS là tài liệu cốt lõi để các nhà phát triển, kiểm thử viên và các bên liên quan khác sử dụng trong suốt quá trình phát triển phần mềm.