Nhận xét của giáo viên

Contents

[Bảng thuật ngữ, 3](#_Toc470454043)

[I. Phần 1: Giới thiệu đề tài 4](#_Toc470454044)

[1.1 Tổng quan 4](#_Toc470454045)

[1.2 Phạm vi báo cáo 4](#_Toc470454046)

[1.3 Nội dung báo cáo 4](#_Toc470454047)

[II. Phần 2: Tổng quan về ngành Kỹ nghệ Phần mềm 5](#_Toc470454048)

[2.1 Khái niệm 5](#_Toc470454049)

[2.2 Lịch sử ra đời 5](#_Toc470454050)

[2.3 Sự phát triển của Kỹ nghệ Phần mềm 5](#_Toc470454051)

[2.4 Các thách thức hiện tại 7](#_Toc470454052)

[III. Phần 3: Kỹ thuật Yêu cầu 8](#_Toc470454053)

[3.1 Giới thiệu 8](#_Toc470454054)

[3.2 Các khái niệm 8](#_Toc470454055)

[3.3 Đặc điểm 8](#_Toc470454056)

[3.4 Mục tiêu 9](#_Toc470454057)

[3.5 Vai trò 9](#_Toc470454058)

[3.6 Các hoạt động chính 9](#_Toc470454059)

[IV. Phần 4: Đặc tả yêu cầu phần mềm 10](#_Toc470454060)

[4.1 Các Khái niệm 10](#_Toc470454061)

[4.2 Tìm kiếm Yêu cầu 13](#_Toc470454062)

[4.3 Mô hình hóa yêu cầu 14](#_Toc470454063)

[4.4 Lập tài liệu cho các yêu cầu 15](#_Toc470454064)

[V. Phần 5: Các phương pháp đặc tả yêu cầu 15](#_Toc470454065)

[VI. Tổng kết 24](#_Toc470454066)

[VII. Các tài liệu tham khảo 25](#_Toc470454067)

# Bảng thuật ngữ

Software Enigeering………………………………Kỹ nghệ phần mềm

Requirement Enigeering……………………….Kỹ thuật yêu cầu

# **Giới thiệu đề tài**

## **1.1 Tổng quan**

Trong quá trình thực hiện bất cứ dự án phần mềm nào, xử lý các yêu cầu mà khách hàng đưa ra bao giờ cũng là một vấn đề vô cùng quan trọng. Các yêu cầu nếu được xử lý đúng đắn sẽ giúp cho các nhà phát triển tham gia vào dự án có được cái nhìn tổng thể về dự án, các nhà quản lý có cơ sở để hoạch định các kế hoạch quản lý của mình. Như vậy việc đặc tả phần mềm đóng vai trò cốt lõi trong viêc tạo ra một sản phẩm phần mềm có giá trị. Trong nội dung bài báo này, chúng em xin trình bày một cách tổng quan và cô đọng nhất các vấn đề quan trọng trong quá trình đặc tả một phần mềm.

## **1.2 Phạm vi báo cáo**

Báo cáo này sẽ tập trung nói về đặc tả phần mềm trong cả một quy trình phát triển phần mềm, quy trình thực hiện, các tài liệu cuối cùng, và các kĩ thuật thường sử dụng trong quá trình đặc tả yêu cầu.

## **1.3 Nội dung báo cáo**

Trong phạm vi báo cáo này, chúng em xin trình bày theo cấu trúc như sau:

* Phần một: Giới thiệu chủ tổng quan về đề tài Đặc tả Phần mềm.
* Phần hai: Giới thiệu tổng quan về Kỹ nghệ Phần mềm.
* Phần ba: Giới thiệu chủ yếu về một ngành kỹ thuật quan trọng trong Kỹ nghệ Phần mềm là Kỹ thuật Yêu cầu, kỹ thuật được sử dụng chủ yếu trong quá trình xử lý yêu cầu phần mềm.
* Phần bốn: Trình bày về Quá trình Đặc tả yêu cầu Phần mềm, đây là một quá trình quan trọng trong Kỹ nghệ phần mềm.
* Phần năm: Trình bày về các kỹ thuật chủ yếu được sử dụng trong quá trình đặc tả yêu cầu phần mềm.
* Cuối cùng là phần tài liệu đặc tả cho dự án demo minh họa.

# **Sơ lược về Kỹ nghệ Phần mềm**

## **2.1 Khái niệm**

Muốn hiểu rõ khái niệm Kỹ nghệ Phần mềm, ta cần hiểu rõ hai khái niệm *Kỹ nghệ* (*Engineering)* và *Phần mềm (Software).*

*Kỹ nghệ* là một danh từ chỉ việc chúng ta áp dụng toán học và các kiến thức về kinh tế, xã hội, thực tiễn vào việc phát minh, sáng tạo, thiết kế, xây dựng, bảo trì, nghiên cứu và cải thiện các kiến trúc, các máy móc, các công cụ, các hệ thống, các thành phần, các nguyên liệu, các quy trình, các giải pháp và kể cả các tổ chức….

*Phần mềm* là một phần của hệ thống máy tính, tập hợp các câu lệnh và các chỉ thị được viết bằng một hay nhiều ngôn ngữ lập trình theo một trật tự nhất định nhằm tự động thực hiện một số nhiệm vụ hay chức năng hoặc giải quyết một vấn đề nào đó.

Vậy, Kỹ nghệ Phần mềm là một cách tiếp cận có hệ thống trong quá trình sản xuất phần mềm, ta sử dụng nó để phân tích, thiết kế, đánh giá, hiện thực, kiểm thử, bảo trì và tái cấu trúc phần mềm, hay nói cách khác là áp dụng kỹ nghệ sản xuất vào trong quá trình tạo ra phần mềm.

Trong cách tiếp cận kỹ nghệ phần mềm, ta sẽ định nghĩa một số mô hình cho vòng đời sống của phần mềm, mỗi mô hình bao gồm nhiều pha khác nhau. Để đánh giá các pha nói chung và các mô hình nói riêng, ta sẽ cần sử dụng nhiều phương pháp luận khác nhau.

Kỹ nghệ phần mềm sử dụng kiến thức của nhiều lĩnh vực khác nhau như kỹ thuật máy tính, khoa học máy tính, quản lý, toán học, quản lý dự án, quản lý chất lượng, công thái học phần mềm (software ergonomics), và kỹ nghệ hệ thống (systems engineering).

## **2.2 Lịch sử ra đời**

Lịch sử của ngành kỹ nghệ phần mềm gắn liền cùng vơi sự phát triển của máy tính cá nhân (personal computer). Từ những năm 1950, phát triển phần mềm và kỹ nghệ phần mềm đã có những bước đi đầu tiên, tuy nhiên nhiều người tin rằng phải đến năm 1968, kỹ nghệ phần mềm mới chính thức trở thành khái niệm được công nhận bởi một hội nghị do NATO tài trợ. Hội nghị này chính là động lực thúc đẩy các nghiên cứu sau này về kỹ nghệ phần mềm.

Nhiều người trong ngành tự gọi mình là “kỹ sư phần mềm” từ năm 1989. Đến cuối những năm 1980, đầu những năm 1990, các trường đại học bắt đầu giảng dạy về kỹ nghệ phần mềm.

## **2.3 Sự phát triển** **của Kỹ nghệ Phần mềm**

Sự phát triển của kỹ nghệ phần mềm đi liên với sự phát triển của máy tính cá nhân (personal computer), từ các chương trình trên bìa đục lỗ hết sức đơn giản dành cho các máy tính mainframe khổng lồ có kích cỡ ngang một tòa nhà cho tới các ứng dụng chạy trên các thiết bị di động có kích thước chỉ bằng bàn tay. Ta có thể tóm tắt sự phát triển của Kỹ nghệ phần mềm qua các giai đoạn chính sau:

* *Những năm đầu từ 1950 đến 1960*

Giai đoạn này phần cứng thay đổi liên tục, số lượng máy tính rất ít và phần lớn mỗi máy đều được đặt hàng chuyên dụng cho một ứng dụng đặc biệt. Phương thức chính là xử lý theo lô (batch), tức là “gói” các chương trình có sử dụng kết quả của nhau lại thành một khối dể tăng tốc độ thực hiện.

Thời kỳ này việc lập trình các chương trình máy tính được coi là “theo bản năng”, không hề có một quy trình hay một phương pháp nào có hệ thống. Việc phát triển phần mềm chưa được quản lý.

Môi trường lập trình có tính chất cá nhân; thiết kế, tiến trình phần mềm không tường minh, thường không có tài liệu. Sản xuất có tính đơn chiếc, theo đơn đặt hàng. Người lập trình thường là người sử dụng và kiêm cả việc bảo trì và sửa lỗi.

* *Thời kỳ mở rộng từ những năm 1960 đến giữa những năm 1970*

Các hệ thống đa nhiệm, đa người sử dụng (ví dụ: Multics, Unix,...) xuất hiện dẫn đến khái niệm mới về tương tác người máy. Kỹ thuật này mở ra thế giới mới cho các ứng dụng và đòi hỏi mức độ tinh vi hơn cho cả phần mềm và phần cứng.

Nhiều hệ thống thời gian thực với các đặc trưng thu thập, phân tích và biến đổi dữ liệu từ nhiều nguồn khác nhau và phản ứng (xử lý, tạo output) trong một khoảng thời gian nhất định xuất hiện. Các tiến bộ trong việc lưu trữ trực tuyến làm xuất hiện các hệ quản trị CSDL thế hệ đầu tiên.

Số lượng các hệ thống dựa trên máy tính phát triển, nhu cầu phân phối mở rộng, thư viện phần mềm phát triển, quy mô phần mềm ngày càng lớn làm nẩy sinh nhu cầu sửa chữa khi gặp lỗi, cần sửa đổi khi người dùng có yêu cầu hay phải thích nghi với những thay đổi của môi trường phần mềm (phần cứng, hệ điều hành, chương trình dịch mới). Công việc bảo trì phần mềm dần dần tiêu tốn nhiều công sức và tài nguyên đến mức báo động.

* *Thời kỳ từ giữa những năm 1970 đến đầu những năm 1990*

Hệ thống phân tán (bao gồm nhiều máy tính, mỗi máy thực hiện một chức năng và liên lạc với các máy khác) xuất hiện làm tăng quy mô và độ phức tạp của phần mềm ứng dụng trên chúng.

Mạng toàn cục và cục bộ, liên lạc số giải thông cao phát triển mạnh làm tăng nhu cầu thâm nhập dữ liệu trực tuyến, nảy sinh yêu cầu lớn phát triển phần mềm quản lý dữ liệu.

Công nghệ chế tạo các bộ vi xử lý tiến bộ nhanh khiến cho máy tính cá nhân, máy trạm để bàn, và các thiết bị nhúng (dùng cho điều khiển trong robot, ô tô, thiết bị y tế, đồ điện gia dụng,...) phát triển mạnh khiến cho nhu cầu về phần mềm tăng nhanh.

Thị trường phần cứng đi vào ổn định, chi phí cho phần mềm tăng nhanh và có khuynh hướng vượt chi phí mua phần cứng.

Các phương pháp quản lý mới được nghiên cứu và lần đầu đưa vào sử dụng trong quá trình phát triển phần mềm.

* *Thời kỳ sau 1990*

Kỹ nghệ hướng đối tượng là cách tiếp cận mới đang nhanh chóng thay thế nhiều cách tiếp cận phát triển phần mềm truyền thống trong các lĩnh vực ứng dụng.

Sự phát triển của Internet làm cho người dùng máy tính tăng lên nhanh chóng, nhu cầu phần mềm ngày càng lớn, quy mô và độ phức tạp của những hệ thống phần mềm mới cũng tăng đáng kể.

Phần mềm trí tuệ nhân tạo ứng dụng các thuật toán phi số như hệ chuyên gia, mạng nơ ron nhân tạo được chuyển từ phòng thí nghiệm ra ứng dụng thực tế mở ra khả năng xử lý thông tin và nhận dạng kiểu con người.

Càng ngày càng có nhiều các phương pháp luận, các công cụ, các nền tảng, các framework, các ngôn ngữ, các kiến trúc mới được phát triển, kỹ nghệ phần mềm đạt được một sự đa dạng chưa từng có kể từ những năm đầu sơ khai.

## **2.4 Các thách thức hiện tại**

Phát triển nhanh chóng nhưng kỹ nghệ phần mềm vẫn chưa khắc phục hết các thách thức mà bản thân nó còn tồn đọng.

1. **Kỹ thuật Yêu Cầu**

Trước khi đi vào phần đặc tả yêu cầu phần mềm, chúng em xin giới thiệu sơ qua về Kỹ thuật yêu cầu: các khái niệm, đặc điểm, mục tiêu, vai trò và các hoạt động chính trong Kỹ thuật Yêu cầu. Sau đó giới thiệu các hoạt động chính.

Một vấn đề quan trọng trong Kỹ thuật Yêu cầu là Mô hình hóa Yêu cầu. Nó sẽ liên quan đến nhiều kỹ thuật khác nhau để mô tả các yêu cầu theo đúng như mong đợi của khách hàng. Các kỹ thuật này cáo thể ở mức trừu tượng cao như các đặc tả có dạng mã giả, các logic hình thức và các biễu diễn đồ họa. Bất cứ kỹ thuật nào được dùng, các kỹ sư cũng cần phải cố gắng để thu thập các đặc tả yêu cầu hệ thống một cách đầy đủ, chính xác và chi tiết.

## **3.1 Tổng quan**

**Khái niệm:**

Ta có thể coi *Kỹ thuật yêu cầu* là một phạm vi con (subdiscipline) của Kỹ nghệ Phần mềm mà ở đó các công việc chủ yếu liên quan đến việc xác định các mục tiêu, các chức năng, các ràng buộc của hệ thống phần mềm. Kỹ thuật Yêu cầu cũng bao gồm các yếu tố kỹ thuật cũng như các đặc tả chính xác về hành vi của phần mềm và sự tiến hóa của nó qua thời gian.

Kỹ thuật yêu cầu gồm các giai đoạn tìm kiếm, ghi chép, phân tích, xác thực và quản lý các yêu cầu. Có nhiều cách tiếp cận kỹ thuật yêu cầu, mỗi cách tiếp cận có mức độ hoàn thiện khác nhau, tuy nhiên chúng đều có phương pháp luận đầy đủ và đều đảm bảo mỗi giai đoạn đều có tài liệu riêng.

**Khi nào bắt đầu:**

Theo lý thuyết, kỹ thuật yêu cầu sẽ bắt đầu với một hoạt động nghiên cứu tính khả thi (feasibility study) để sinh ra tài liệu về tính khả thi của dự án (feasibility report). Tùy vào kết quả của tài liệu này mà ta sẽ quyết định có tiếp tục thực hiện dự án hay không. Nếu nghiên cứu tính khả thi đề nghị tiếp tục phát triển sản phẩm thì giai đoạn phân tích yêu cầu sẽ được tiến hành.

**Mục tiêu:**

Các đặc tả phần mềm chính xác cung cấp các thông tin cơ bản cho các pha phân tích thiết kết sau này, xác thực những mong muốn của stakeholder, định nghĩa những gì mà các kỹ sư cần xây dựng, và bằng chứng để chứng mình những gì họ làm là chính xác.

**Vai trò:**

Đặc tả yêu cầu phần mềm cho phép chúng ta biết được động lực để phát triển hệ thống phần mềm. Các yêu cầu phần mềm được đặc tả một cách kỹ càng giúp các nhà quản lý hoạch định kế hoạch tương lai và nắm bắt được sự phát triển của phần mềm qua thời gian. Cách tiếp cận này phản ánh thực tế của một thế giới luôn thay đổi và nhu cầu sử dụng lại một phần đặc tả đã được dùng trong một hệ thống phần mềm khác.

**Các hoạt động chính:**

Có 5 hoạt động chính trong quá trình đặc tả yêu cầu phần mềm:

* Tìm kiếm yêu cầu – **Requirements Elicitation.**
* Mô hình hóa yêu cầu – **Requirements Modeling.**
* Phân tích yêu cầu – **Requirements Analysis.**
* Quản lý các yêu cầu – **Requirements Documentation.**
* Xác thực các yêu cầu – **Requirements Varifiaction.**

Tìm kiếm các yêu cầu có thể coi là hoạt động khó khăn nhất trong trong kỹ thuật yêu cầu. Bởi vì các yêu cầu không phải lúc nào cũng rõ ràng, chúng thường được stakeholders diễn tả không chính xác, vì đôi khi stakeholders cũng không hiểu rõ các yêu cầu của chính mình, nhiều yêu cầu có thể rất phức tạp hoặc chúng có thể mâu thuẫn với nhau. Các kỹ sư cần phải tự mình tìm tòi và suy ra các yêu cầu chính xác từ những gì mà khách hàng cung cấp cho họ.

Mô hình hóa yêu cầu liên quan đến việc biểu diễn các yêu cầu dưới các dạng văn bản, đồ họa, hình ảnh, các công thức toán học,.. Nhờ vậy ta có thể hiểu được các yêu cầu một cách trực quan và rõ ràng.

Phân tích yêu cầu liên quan đến việc xác định xem yêu cầu đó có chính xác hay không. Ngoài ra còn một số yếu tố mà ta cần quan tâm đến là sự chặt chẽ, tính hoàn thiện, độ chi tiết, v.v… Mô hình của các yêu cầu và các thuộc tính của chúng cần phải được liên hệ với các bên liên quan.

Xác thực yêu cầu là hoạt động mà các kỹ sư cần làm việc với các bên liên quan để kiểm chứng độ đúng đắn của các mô hình yêu cầu mà họ đã lập nên. Bất kì thay đổi nào trong quá trình cũng này đều dẫn tới việc quay trở lại thực hiện các hoạt động bên trên.

Quản lý các yêu cầu có trách nhiệm đảm bảo các yêu cầu sau khi được tạo ra sẽ được lưu trữ cẩn thận, bất kì thay đổi này theo thời gian cũng cần được cập nhật kịp thời và đồng bộ.

## **3.2 Các hoạt động chính**

### **3.2.1 Tìm kiếm yêu cầu**

Là các công việc liên quan đến việc làm việc với khách hàng để xác định lĩnh vực ứng dụng, các dịch vụ mà hệ thống cần cung cấp cũng như các ràng buộc của hệ thống.

Vấn đề đặt ra là không phải lúc nào khách hàng cũng hiểu rõ cái mà họ cần, hoặc các yêu cầu thường không rõ ràng, hoặc các bên liên quan (stackholder) diễn tả không chính xác. Do đó, các kỹ sư cần phải cảm nhận được yêu cầu của khách hàng và nhắc nhở họ bằng kinh nghiệm của bản thân, sau đó thông qua giao tiếp với khách hàng bằng các cuộ gặp gỡ, trao điển để tìm ra các yêu cầu của họ là gì.

Những nhân tố tham gia vào quá trình này bao gồm: Các bên liên quan (stakeholder): người dùng cuối, quản lý, các kỹ sư, các chuyên gia có lên quan đến lĩnh vực của ứng dụng.

Có 3 cách cách tiếp cận chủ yếu để giúp cho các phân tích viên có thể tìm kiếm và nắm bắt được các yêu cầu:

* Thiết kế ứng dụng chung (Joint Application Design – JAD)
* Triển khai chức năng chất lượng (Quality Function Deployment – QFD)
* Thiết kế như người học việc (Desginer as apprentice)

**Thiết kế ứng dụng chung (Joint Application Design)**

Được đánh giá cao trong việc tổ chức và thực hiện các cuộc họp nhằm lấy yêu cầu hệ thống**.** Mỗi cuộc họp có sự tham gia của người dùng hệ thống, người sử hữu hệ thống và các phân tích viên**.** Quy trình này giúp các kỹ sư phần mềm trong việc tìm ra các yêu cầu cho việc đặc tả yêu cầu phần mềm, thiết kế và mô tả thiết kế hệ thống, code, kiểm thử, viết tài liệu hướng dẫn người dùng.

Quy trình thực hiện JAD:

* Chọn người tham gia
* Chuẩn bị tài liệu (agenda)
* Chọn địa điểm tiến hành.
* Thực hiện hội thảo

**Triển khai chức năng chất lượng (Quality Function Deployment)**

Được giới thiệu vào năm 1966 bởi Yoji Akao sử dụng trong sản xuất, công nghiệp nặng và hệ thống kỹ thuật. Là kỹ thuật để xác định yêu cầu của khách hàng và xác định được các điểm mà chất lượng sản phẩm được đảm bảo trong toàn bộ quá trình phát triển. Cung cấp cấu trúc cho việc lắng nghe cái khách hàng muốn và cần. Sau đó chuyển chúng (tiếng nói của khách hàng) thành tài liệu kỹ thuật. Giúp cải thiện việc tham gia của người dùng và người quản lý. Rút ngắn vòng đời phát triển sản phẩm. Cung cấp công cụ dự phòng trong việc tránh mất mát thông tin trong mỗi giai đoạn phát triển sản phẩm

**Thiết kế như người học việc (Designer as apprentice)**

Người thiết kế phải học tất cả các công việc của khách hàng đã, đang và sẽ làm. Phương pháp này mang lại nhiều khó khăn cho người thiết kế, bởi vì đôi khi không phải khách hàng nào cũng có khả năng truyền đạt toàn bộ những gì họ hiểu hoặc có thể không hiểu hoàn toàn những gì họ đang làm. Do đó cần có sự hợp tác chặc chẽ của khách hàng trong suốt quá trình “học tập” của người thiết kế.

Thuận lợi mang lại cũng rất đáng để lưu ý: người phân tích sẽ hiểu sâu hơn về cái mình sẽ làm, đồng lời tích lũy thêm kinh nghiệm, học hỏi cái mới. Khách hàng nhận được sản phẩm sát với yêu cầu nhất và thời gian phát triển sản phẩm sẽ được rút ngắn.

### **3.2.2 Mô hình hóa yêu cầu**

Có rất nhiều phương pháp khác nhau để mô hình hóa các yêu cầu phần mềm. Bởi vì tính chất của ngôn ngữ tự nhiên, nó ít khi được dùng trong các tài liệu đặc tả yêu cầu phần mềm, hoặc nếu được dùng thì cần phải đi kèm các cấu trúc, ràng buộc ngữ nghĩa để đảm bảo tính đúng đắn và mạch lạc của mô hình.

Các giải pháp thay thế cho ngôn ngữ tự nhiên:

* Ngôn ngữ đặc tả có cấu trúc (structured language specification)
* Ngôn ngữ thiết kế phần mềm (Program Design Language – PDL)
* Kí hiệu đồ họa
* Use case/User Story
* Formal method (ngôn ngữ Z, VDM)
* Informal/Semifomal method (DFD, UML)

### **3.3.3 Phân tích yêu cầu**

Trong công đoạn này, các nhà phát triển sẽ bắt tay vào xem liệu các yêu cầu được mô hình hóa có rõ ràng, minh bạch, hoàn chỉnh, v.v… hay không. Sau đó giải quyết các vấn đề còn tồn đọng trong mô hình bằng cách làm việc với khách hàng. Đồng thời họ sẽ xem xét các nhu cầu/điều kiện đã có để quyết định xem nên làm một sản phẩm mới hay chỉnh sửa một sản phẩm có sẵn.

Các yêu cầu sau khi đã qua bước phân tích cần đáp ứng các tiêu chí: có thể tiến hành, có thể đo lường được, có thể kiểm tra được, liên quan đến các nhu cầu kinh doanh, và được định nghĩa ở mức độ chi tiết đủ để thiết kế hệ thống

### **3.2.3 Lập tài liệu cho các yêu cầu**

Các yêu cầu sau khi đã mô hình hóa và được phân tích kỹ càng sẽ được tập hợp lại trong một tài liệu đặc tả yêu cầu phần mềm. Tài liệu này bao gồm các định nghĩa và các đặc tả của các yêu cầu, tuy nhiên nó không phải là tài liệu thiết kế. Nó cho chúng ta biết hệ thống của chúng ta sẽ làm được cái gì chứ không phải làm thế nào mà chúng được thực hiện.

Những người sẽ sử dụng SRS:

* Khách hàng: cho biết làm thế nào để thỏa mãn yêu cầu của họ; tiếp tục quá trình này trong suốt quá trình sản xuất.
* Người quản lý: lập kế hoạch cho dự án và quản lý quá trình sản xuất phần mềm
* Nhà phát triển : tạo ra các thiết kế phù hợp với các yêu cầu.
* Kỹ sư kiểm thử: là căn cứ để xác định hệ thống có phù hợp hay không.
* Kỹ sư bảo trì: hiểu được hệ thống sẽ làm gì và nó trở nên như thế nào qua thời gian.

**IV. Đặc tả Yêu cầu Phần mềm**

## **4.1 Tổng quát**

Đặc tả yêu cầu phần mềm là một tập hợp các hoạt động được thiết kế nhằm nắm bắt các khía hạn chức năng và phi chức năng của hệ thống, sau đó đặc tả chúng trong các tài liệu SRS (tài liệu đặc tả yêu cầu phần mềm). Mục tiêu chính của quá trình này là nhằm cung cấp một mô tả đầy đủ về hành vi của hệ thống (cái gì mà hệ thống cần làm) mà không cần phải mô tả chi tiết cấu trúc bên trong (các yêu cầu đó được hệ thống thực hiện như thế nào, ra sao). Đặc tả yêu cầu phần mềm thường được phân loại dựa theo mức độ trừu tượng của chúng:

* Đặc tả yêu cầu người dùng
* Đặc tả yêu cầu hệ thống
* Đặc tả thiết kế phần mềm

*Đặc tả yêu cầu người dùng*

Thường được dùng để User requirements specifications are usually used to attract bidders, often in the form of a request for proposal (RFP)

User requirements may contain abstract statements in natural language, for example, English, with accompanying informal diagrams, even back-ofthe-napkin drawings\*. User requirements specify functional and nonfunctional requirements as they pertain to externally visible behavior in a form understandable by clients and system users. These kinds of representation techniques, however, are fraught with danger, as will be discussed shortly.

*Yêu cầu hệ thống*

System level requirements, or SRSs mentioned previously, are detailed descriptions of the services and constraints. Systems requirements are derived from analysis of the user requirements. Systems requirements should be structured and precise because they act as a contract between client and contractor (and can literally be enforced in court). Appendix A contains an SRS for a wet well control system. This specification is nearly complete (it is lacking some elements, mostly formal specifications and some narrative that are omitted for brevity). I will be referring to this document occasionally, and now would be a good time for you to browse the first few pages of Appendix A to become familiar with this simple engineering application.

*Đặc tả thiết kế phần mềm*

Software design specifications are usually the most detailed level requirements specifications that are written and used by software engineers as a basis for the system’s architecture and design. Appendix B contains the corresponding software design specification for the wet well control system, which was derived from Appendix A. The software design specification is described in Chapter 4. Within these three specification types are there different requirements types? Yes, there are. Within the family of user, system, and functional requirements specifications, all kinds of things can be described. These include external constraints to the software and user needs. The user needs are usually called “functional requirements” and the external constraints are called “nonfunctional requirements.”

**Yêu cầu chức năng**

Functional requirements describe the services the system should provide.

Sometimes the functional requirements state what the system should not do.

Functional requirements can be high-level and general or detailed, expressing inputs, outputs, exceptions, and so on. See Appendix A, Section A.3 for

specific functional requirements for the wet well control system.

**Yêu cầu phi chức năng**

Nonfunctional requirements are imposed by the environment in which the

system is to exist. These requirements could include timing constraints,

quality properties, standard adherence, programming languages to be used,

compliance with laws, and so on.

**Yêu cầu domain**

Domain requirements are a type of nonfunctional requirement from which the

application domain dictates or derives. Domain requirements might impose

new functional requirements or constraints on existing functional requirements.

For example, in the baggage inspection system, industry standards and restrictions on baggage size and shape will place certain constraints on the system.

**Đặc tả giao diện**

Interface specifications are functional software requirements specified in

terms of interfaces to operating units. Most systems must operate with other

systems and the operating interfaces must be specified as part of the requirements. There are three types of interface that may have to be defined:

1. procedural interfaces

2. data structures that are exchanged

3. data representations

Formal notations are an effective technique for interface specification

**Yêu cầu hiệu suất**

Performance requirements are functional requirements that include the static

and dynamic numerical requirements placed on the software or on human

interaction with the software as a whole. For an imaging system, static requirements might include the number of simultaneous users to be supported. The

dynamic requirements might include the numbers of transactions and tasks

the amount of data to be processed within certain time periods for both normal

and peak workload conditions. For the wet well control system in Section A.3.2,

many of the requirements must be achieved within five seconds. These time

constraints represent performance requirements.

**What are logical database requirements?**

Logical database requirements are functional requirements that include the

types of information used by various functions such as frequency of use,

accessing capabilities, data entities and their relationships, integrity constraints, and data retention requirements.

**What are design constraint requirements?**

Design constraint requirements are nonfunctional requirements that are

related to standards compliance and hardware limitations.

**What are system attribute requirements?**

System attribute requirements are functional requirements that include reliability,availability, security, maintainability, and portability. Many of these

requirements can be found in Appendix A.

What is a feasibility study and what is its role?

A feasibility study is a short focused study that checks if the system contributes to organizational objectives the system can be engineered using current

technology and within budgetthe system can be integrated with other systems that are used.

A feasibility study is used to decide if the proposed system is worthwhile.

Feasibility studies can also help answer some of the following questions.

• What if the system wasn’t implemented?

• What are current process problems?

• How will the proposed system help?

• What will be the integration problems?

• Is new technology needed? What skills?

• What facilities must be supported by the proposed system?

Therefore, it is important to conduct a feasibility study before a large investment is made in a system that is not needed or which does not solve some

inherent business process problem.

Are their social and organizational factors in requirements engineering?

You bet there are. Software engineering involves many human activities, and

none is more obviously so than requirements engineering. Software systems

are used in a social and organizational context; thus, political factors almost

always influence the software requirements.

Moreover, stakeholders often don’t know what they really want and they

express requirements in their own terms. Different stakeholders may have conflicting requirements, new stakeholders may emerge, and the business environment may change. Finally, the requirements change during the analysis process

for a host of reasons. Therefore, good requirements engineers must be sensitive

to these factors.

**Các phương pháp đặc tả** **yêu cầu**

## **5.1 Phân loại**

Theo trạng thái ngôn ngữ:

* Ngôn ngữ tự nhiên có cấu trúc
* Ngôn ngữ mô tả thiết kế
* Ký hiệu đồ họa
* Đặc tả toán học

Theo mức độ hình thức:

* Hình thức – formal
* Không hình thức – informal
* Bán hình thức – semiformal

## **5.2 Ngôn ngữ tự nhiên có cấu trúc**

* Là một hình thức giới hạn của ngôn ngữ tự nhiên được dùng để đặc tả các yêu cầu.
* Chúng sử dụng một tập từ vựng và ngữ pháp nhất định để thể hiện các yêu cầu dưới một sự kiểm soát chặt chẽ để có thể được phân tích (parse)
* Ưu điểm:
  + Giúp ta loại bỏ các vấn đề phát sinh từ sự mơ hồ và tang mức độ đồng nhất trên tài liệu đặc tả.
* Nhược điểm:
  + Đòi hỏi các bên có liên quan (stakeholder) phải được đào tạo ở một trình độ nhất đỉnh để có thể sử dụng.
* Cách tiếp cận:
  + Sử dụng các template dạng biểu mẫu (form-based template)

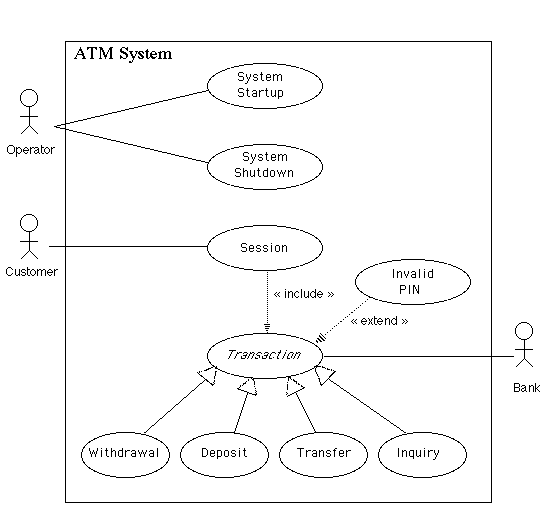
|  |  |
| --- | --- |
|  |  |
| **Function** |  |
| **Description** |  |
| **Inputs** |  |
| **Outputs** |  |
| **Destination** |  |
| **Requires** |  |
| **Precondition** |  |
| **Postcondition** |  |
| **Side-effects** |  |

## **5.3 Ngôn ngữ thiết kế chương trình – PDL**

* Sử dụng các ngôn ngữ tự nhiên giống như là ngôn ngữ lập trình nhưng sự biểu đạt linh hoạt hơn, không bị gò bó như ngôn ngữ lập trình.
* Gần giống với mã giả (Pseudocode) nhưng không phải là mã giả.
* Được sử dụng trong 2 trường hợp:
  + Khi một hoạt động được mô tả như là một choỗi các hành động và thứ tự giữa chúng rất quan trọng
  + Khi giao tiếp (interface) phần cứng và phần mềm cần phải được quy định.
* Ưu:
  + Có cấu trúc giống như ngôn ngữ lập trình, có vốn từ vựng đa dạng của ngôn ngữ tự nhiên
* Nhược:
  + PDL có thể chưa đủ khả năng để biểu diễn các khái niệm trong miền ứng dụng (domain)
  + Đặc tả được tạo ra bởi PDL có thể được dùng cho việc thiết kế hơn là một đặc tả tuần túy.
  + Chỉ có thể hiểu được bởi những người có kiến thức về ngôn ngữ lập trình

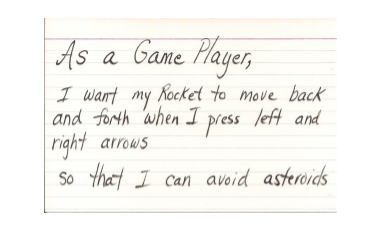
## Kết quả hình ảnh cho program design language**5.4 Use case**

* Use case mô tả sự tương tác đặc trưng giữa người dùng bên ngoài (actor) và hệ thống. Nó thể hiện ứng xử của hệ thống đối với bên ngoài, trong một hoàn cảnh nhất định, xét từ quan điểm của người sử dụng.
* Nó mô tả các yêu cầu đối với hệ thống, có nghĩa là những gì hệ thống phải làm chứ không phải mô tả hệ thống làm như thế nào.
* Mỗi use case mô tả cách thức actor tương tác với hệ thống để đạt được mục tiêu nào đó.
* Một hoặc nhiều kịch bản (scenario) có thể được tạo ra từ mỗi use case, tương ứng với chi tiết về mỗi cách thức đạt được mục tiêu nào đó.
* Khi mô tả Use case, người ta thường tránh dùng thuật ngữ kỹ thuật, thay vào đó họ sử dụng ngôn ngữ của người dùng cuối hoặc chuyên gia về lĩnh vực đó.



## **5.5 User story**

* Là một tài liệu sơ giản về yêu cầu sản phẩm với góc nhìn người dùng.
* Thông thường, User Story do khách hàng, hoặc đại điện của khách hàng viết, tuy nhiên nếu có sự cộng tác của các nhà phát triẻn thì nhóm và khách hàng sẽ có sự chia sẻ hiểu biết về sản phẩm tốt hơn.
* Thường được sử dụng trong các phương pháp luận Agile.
* Mô tả hệ thống dưới cái nhìn của người dùng, nói cách khác là hệ thống cần đáp ứng cho họ những gì.
* Thường có dạng:
  + “Là *<người dùng cụ thể\vai trò>* , tôi muốn *<làm gì đó>* để *<phục vụ mục đích nào đó>”*



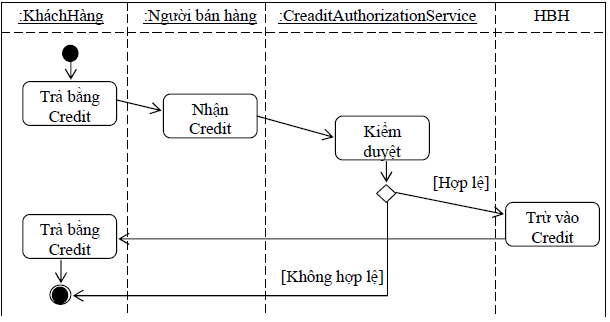
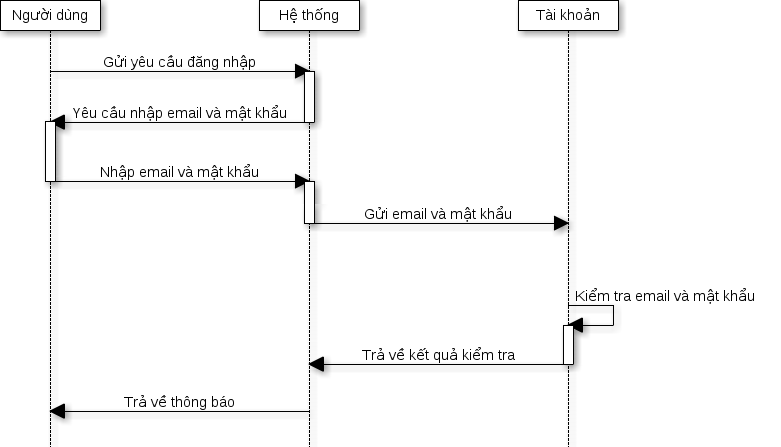
## **5.6 Formal method**

* Phương pháp hình thức được sử dụng để cải thiện việc xây dựng các yêu cầu bằng cách áp dụng toán học và logic.
* Phương pháp hình thức sử dụng một số sự kết hợp của:
  + Tính toán vị ngữ - predicate calculus (first order logic).
  + Lý thuyết hàm đệ quy - recursive function theory,
  + Tính toán Lambda - lambda calculus,
  + Ngữ nghĩa ngôn ngữ lập trình - programming language semantics,
  + Toán rời rạc - discrete mathematics,
  + Lý thuyết số - number theory,
  + Đại số trừu tượng - abstract algebra.
* Cách tiếp cận này hấp dẫn bởi vì cung cấp một phương pháp khoa học cho việc đặc tả yêu cầu
* Các ngôn ngữ thường dùng:
  + Ngôn ngữ Z
  + Ngôn ngữ VDM
  + CSP (cating sequential processes)
* Được sử dụng trong:
  + Kiểm tra tính nhất quán – các yêu cầu về hành vi hệ thống được mô tả dựa trên các ký hiệu toán học.
  + Kiểm tra mô hình – các trạng thái máy (state machine) được sử dụng để xác minh nếu một thuộc tính được cho là thỏa mãn mọi điều kiện
  + Chứng minh định lý – tiên đề của hành vi hệ thống được sử dụng để chứng minh rằng hệ thống sẽ hoạt động theo cách định sẵn.

forse

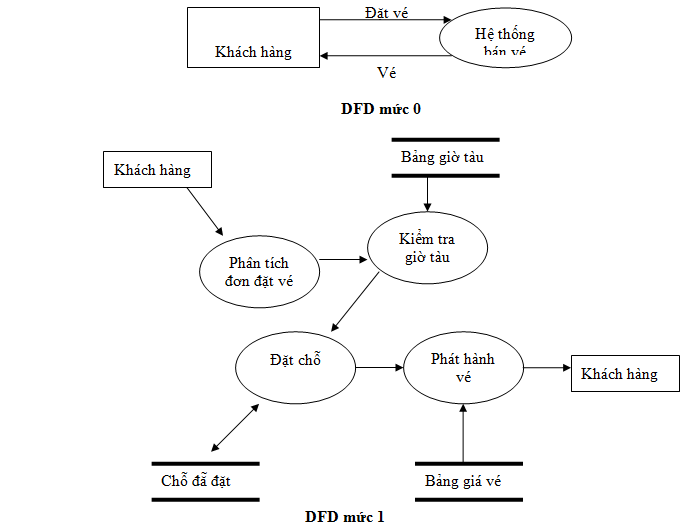
## **5.7 Semi/Informal method**

* Các cách tiếp cận khác không phải phải là phương pháp hình thức thì hoặc là không hình thức (như các biểu đồ dòng chảy – flow-charting) hoặc là bán hình thức (như UML)
* Cách tiếp cận bán hình thức có nghĩa là chúng không xuất hiện dưới dạng các kí hiệu toán học, mà các công cụ mô hình hóa có thể chuyển đổi một phần hoặc toàn bộ chúng thành các biễu diễn toán học khác nhau.
  + 1. *Ngôn ngữ mô hình hóa thống nhất - UML*
* Là một cách tiếp cận bán hình thức.
* Được sử dụng trong hầu hết các giai đoạn của quá trình phát triển phần mềm.
* Trong quá trình đặc tả yêu cầu phần mềm người ta thường sử dụng các sơ đồ tuần tự và sơ đồ hoạt động
  + Sơ đồ tuần tự: là bản vẽ mô tả sự tương tác của các đối tượng để tạo nên các chức năng của hệ thống. Bản vẽ này mô tả sự tương tác theo thời gian nên rất phù hợp với việc sử dụng để thiết kế và cài đặt chức năng cho hệ thống phần mềm.
  + Sơ đồ hoạt động: là bản vẽ tập trung vào mô tả các hoạt động, luồng xử lý bên trong hệ thống. Nó có thể được sử dụng để mô tả các qui trình nghiệp vụ trong hệ thống, các luồng của một chức năng hoặc các hoạt động của một đối tượng.



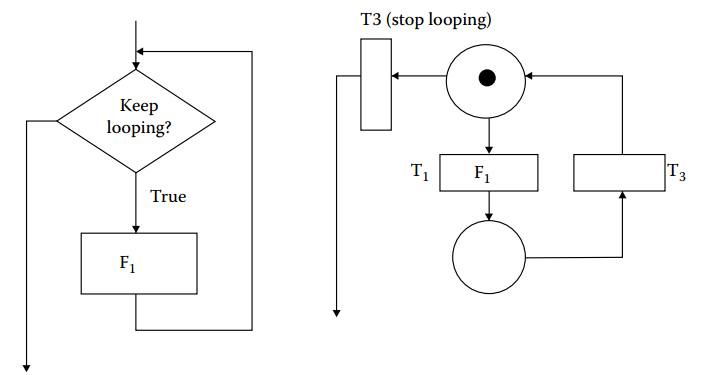
### 5.7.2 *Sơ đồ dòng chảy dữ liệu - DFD*

* Là một cách tiếp cận không hình thức.
* Được sử dụng rộng rãi, tập trung vào các chức năng được thực hiện trong hệ thống
* Xem hệ thống như một mạng lưới các dữ liệu chuyển đổi thông qua các luồng dữ liệu
* Thường được sử dụng bằng cách chia nhỏ các chức năng cần mô hình hóa và áp dụng DFD cho các phần được chia.
* Phương pháp Phân tích và Thiết kế hệ thống (SSAD) sử dụng DFD để tổ chức thông tin và hướng dẫn phân tích



## **5.8 Petri net**

* Là một phương pháp hình thức (formal) được dùng để xác định các hoạt động sẽ được thực hiện trong môi trường đa xử lý hoặc môi trường đa nhiệm.
  + Các bong bóng tròn được gọi là “place” dùng để đại diện cho kho dữ liệu hoặc các xử lý.
  + Các hộp chữ nhật đại diện cho các quá trình chuyển đổi hoặc các hoạt động.
  + Các quy trình và chuyển tiếp (transition) được đánh số và được nối bằng các đường vòng cung
* Được xử dụng cho việc thể hiện các hệ thống đa xử lý và đa tác vụ, đặc biệt là khi các hàm thực hiện có mức độ phức tạp thấp.
* Bởi vì chúng thuần túy là toán học, các kỹ thuật cho việc tối ưu và hình thức hóa chương trình có thể được sử dụng.



**Tổng kết**

**Các tài liệu tham khảo**