MỤC LỤC

Chương 1. Giới thiệu

1.1. Khái niệm phần mềm

"Phần mềm là một tập hợp bao gồm:

- *Các lệnh (chương trình máy tính) khi thực hiện thì đưa ra hoạt động và kết quả mong muốn.*
- *Các cấu trúc dữ liệu làm cho chương trình thao tác thông tin thích hợp.*
- *Các tài liệu mô tả thao tác và cách dùng chương trình."*

Hoạt động của phần mềm là sự mô phỏng lại hoạt dộng của thế giới thực trong một góc độ

Lớp phần mềm là một hệ thống các phần mềm trên cùng một lĩnh vực hoạt động nào đó. Do cùng lĩnh vực nên các phần mềm này thường có cấu trúc và chức năng tương tự nhau.

- **Phân loại phần mềm:**
  - ✓ Phần mềm hệ thống là những phần mềm đảm nhận công việc tích hợp và điều khiển các thiết bị phần cứng đồng thời tạo ra một môi trường thuận lợi để cho các phần mềm khác và người sử dụng có thể thao tác trên đó như một khối thống nhất.
  - ✓ Phần mềm ứng dụng là những phần mềm được dùng để thực hiện một công việc xác định nào đó. Ví dụ: Phần mềm xem ảnh, Phần mềm xử lý văn bản.

1.2. Kiến trúc phần mềm

1.2.1. Thành phần giao diện

Cho phép nhận các yêu cầu về việc muốn thực hiện và cung cấp các dữ liệu nguồn liên quan đến công việc đó hoặc từ các thiết bị thu thập dữ liệu (cân, đo nhiệt độ, tế bào quang học…).

Cho phép trình bày các kết quả của vệc thực hiện các yêu cầu cho người dùng hoặc điều khiển các hoạt động của các thiết bị điều khiển (đóng mở cửa, bật mở máy…).

Một cách tổng quát, thành phần giao diện là hệ thống các hàm chuyên về việc nhập/xuất dữ liệu (hàm nhật/xuất) cùng với hình thức trình bày và tổ chức lưu trữ dữ liệu tương ứng. Mục tiêu chính của các hàm này là đưa dữ liệu từ thế giới bên ngoài vào trong máy tính và ngược lại.

1.2.2. Thành phần dữ liệu

Cho phép lưu trữ lại (hàm ghi) các kết quả đã xử lý trên bộ nhớ phụ với tổ chức lưu trữ được xác định trước (tập tin có cấu trúc, tập tin nhị phân, cơ sở dữ liệu).

Cho phép truy xuất lại (hàm đọc) các dữ liệu đã lưu trữ phục vụ các hàm xử lý tương ứng.

Một cách tổng quát, thành phần dữ liệu là hệ thống các hàm chuyên về đọc ghi dữ liệu (hàm đọc/ghi) cùng với mô hình tổ chức dữ liệu tương ứng. Mục tiêu chính của các hàm này là chuyển đổi dữ liệu giữa bộ nhớ phụ và bộ nhớ chính.

1.2.3. Thành phần xử lý

Kiểm tra tính hợp lệ của các dữ liệu nguồn được cung cấp từ người dùng theo các quy trình ràng buộc trong thế giới thực (ví dụ: chỉ cho phép điểm nhập từ 0 đến 10, một lớp học chỉ cho phép tối đa là 50 học sinh…).

Tiến hành xử lý cho ra kết quả mong đợi theo quy định tính toán có sẵn trong thế giới thực (ví dụ: quy tắc tính lương, tính phụ cấp, quy tác tính tiền điện, tiền nước…) hoặc theo thuật giải tự đề xuất (ví dụ: xếp thời khoá biểu tự động, trộn đề thi…).

Một cách tổng quát, thành phần xử lý là tập các hàm chuyên về xử lý, tính toán, biến đổi dữ liệu. Các hàm này sẽ dùng dữ liệu nguồn từ các hàm trong thành phần giao diện (hàm nhập) hay thành phần dữ liệu (hàm đọc dữ liệu) kiểm tra tính hợp lệ (hàm kiểm tra) và sau đó tiến hành xử lý (hàm xử lý) nếu cần thiết để cho ra kết quả mà sẽ được trình bày cho người dùng xem qua các hàm trong thành phần giao diện (hàm xuất) hoặc lưu giữ lại qua các hàm trong thành phần dữ liệu (hàm ghi).

1.3. Các đặc điểm của phần mềm

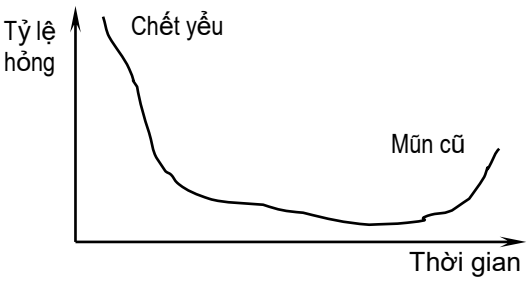
Phần mềm là phần tử của hệ thống logic chứ không phải hệ thống vật lý. Do vậy, phần mềm có một số đặc trưng khác biệt đáng kể đối với đặc trưng của phần cứng.

Đặc trưng 1: *Phần mềm được phát triển hay được kỹ nghệ hoá, nó không được chế tạo theo nghĩa cổ điển.*

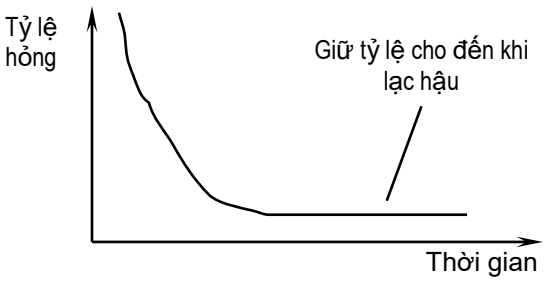
Mặc dầu có một số điểm tương đồng giữa phát triển phần mềm và chế tạo phần cứng, nhưng hai hoạt động này về cơ bản là khác nhau. Trong cả hai hoạt động này, chất lượng cao được đạt tới thông qua thiết kế tốt, nhưng giai đoạn chế tạo phần cứng có thể đưa vào vấn đề mà chất lượng không tồn tại (hay dễ được sửa đổi) cho phần mềm. Cả hai hoạt động này đều phụ thuộc vào con người, nhưng mối quan hệ giữa người được áp dụng và công việc được thực hiện hoàn toàn khác. Cả hai hoạt động này đòi hỏi việc xây dựng "sản phẩm", nhưng cách tiếp cận là hoàn toàn khác. Phần mềm được chế tạo ra là hoàn toàn mới, không có tiền lệ trước và nó cũng chỉ được tạo ra 1 lần duy nhất.

Đặc trưng 2: *Phần mềm không "hỏng đi".*

Phần mềm không cảm ứng với khiếm khuyết môi trường vốn gây cho phần cứng mòn cũ đi. Phần mềm nếu cứ với các bộ dữ liệu đầu vào hợp lý thì nó luôn cho kết quả có ý nghĩa giống nhau, không thay đổi theo thời gian, điều kiện khí hậu, …

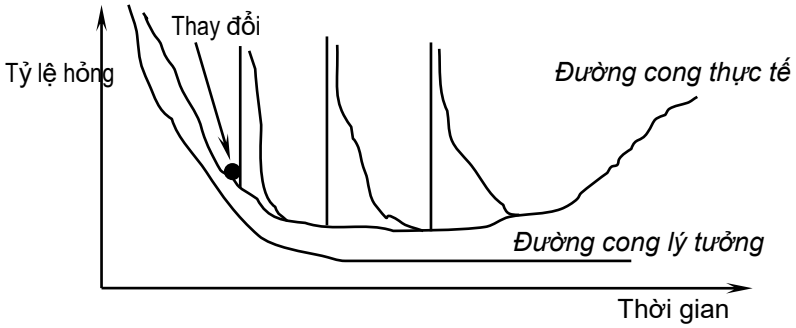


Đường cong hỏng hóc của phần cứng



Đường cong hỏng hóc của phần mềm (lý tưởng)

Thực tế, phần mềm sẽ trải qua sự thay đổi (bảo trì). Khi thay đổi được thực hiện, có thể một số khiếm khuyết sẽ được thêm vào, gây ra trong đường cong tỷ lệ hỏng có dấu hiệu như hình vẽ dưới đây. Trước khi đường cong đó có thể trở về tỷ lệ hỏng hóc ổn định ban đầu, thì một yêu cầu khác lại được đưa vào, lại gây ra đường cong phát sinh đỉnh nhọn một lần nữa. Dần dần, mức tỷ lệ hỏng tối thiểu tăng lên - phần mềm bị thoái hoá do sự thay đổi.



Hình 1: Đường cong hỏng hóc thực tế của phần mềm

Nhận xét: Phần cứng hỏng có "vật tư thay thế", nhưng không có phần mềm thay thế cho phần mềm. Mọi hỏng hóc của phần mềm đều chỉ ra lỗi trong thiết kế hay trong tiến trình chuyển thiết kế thành mã hoá lệnh máy thực hiện được. Do đó, việc bảo trì phần mềm bao gồm việc phụ thêm đáng kể so với bảo trì phần cứng.

Đặc trưng 3: *Phần lớn phần mềm được xây dựng theo đơn đặt hàng, chứ ít khi được lắp ráp từ các thành phần có sẵn.*

Cách thiết kế và xây dựng ***phần cứng*** điều khiển cho một sản phẩm dựa trên bộ vi xử lý: vẽ sơ đồ mạch số => thực hiện phân tích để đảm bảo chức năng đúng => phân loại các danh mục thành phần => gắn cho mỗi mạch tích hợp (thường gọi là *IC* hay *chip*) một số hiệu một chức năng đã định trước và hợp lệ; một giao diện đã xác định rõ; một tập các hướng dẫn tích hợp chuẩn hoá.

Đối với ***phần mềm***: Khi xây dựng ta không có danh mục các thành phần. Phần mềm được đặt hàng với đơn vị hoàn chỉnh, không phải là những thành phần có thể lắp ráp lại thành chương trình mới.

1.4. Các ứng dụng của phần mềm

Sản phẩm phần mềm là gì?

Sản phẩm phần mềm là một hoặc một nhóm các chương trình được xây dựng để giải quyết một vấn đề nào đó. Ví dụ: chương trình quản lý hoạt động của máy móc và các chương trình ứng dụng.

Nhóm các sản phẩm hiện có.

Hiện nay người ta phân chia thành 7 nhóm phần mềm chính.

Nhóm 1: *Phần mềm hệ thống.*

Là một tập hợp các chương trình được viết để phục vụ cho các chương trình khác. Chương trình này xử lý các thông tin phức tạp nhưng xác định cấp thấp, tạo môi trường hoạt động (trình biên dịch, trình soạn thảo, quản lý tệp tin, …).

Các chương trình này đặc trưng bởi tương tác chủ yếu với phần cứng máy tính, phục vụ nhiều người dùng, có cấu trúc dữ liệu phức tạp và nhiều giao diện ngoài.

Nhóm 2: *Phần mềm thời gian thực.*

Là phần mềm điều phối hoặc phân tích hay kiểm soát các sự kiện thế giới thực ngay khi chúng xuất hiện.

Phần mềm thời gian thực bao gồm các yếu tố:

- Một thành phần thu thập dữ liệu để thu và định dạng thông tin từ bên ngoài.
- Một thành phần phân tích để biến đổi thông tin theo yêu cầu của ứng dụng.
- Một thành phần kiểm soát hoặc đưa ra các đáp ứng cho môi trường ngoài.
- Một thành phần điều phối để điều hoà các thành phần khác sao cho có thể duy trì việc đáp ứng thời gian thực.

Hệ thống thời gian thực phải đáp ứng được những ràng buộc thời gian chặt chẽ.

Nhóm 3: *Phần mềm nghiệp vụ.*

Ngày nay, xử lý thông tin nghiệp vụ là lĩnh vự ứng dụng phần mềm lớn nhất. Phần mềm loại này phục vụ cho các hệ thống rời rạc: *hệ thông tin quản lý*. Các ứng dụng phần mềm nghiệp vụ

còn bao gồm cả tính toán tương tác (như xử lý các giao tác cho các điểm bán hàng) ngoài ứng dụng xử lý dữ liệu.

Nhóm 4: *Phần mềm khoa học công nghệ.*

Phần mềm này được đặc trưng bởi các thuật toán. Phần mềm tạo ra một ứng dụng mới, thiết kế có máy tính trợ giúp (computer aided of design - CAD), có chú ý đến các đặc trưng thời gian thực và phần mềm hệ thống.

Nhóm 5: *Phần mềm nhúng.*

Nằm trong bộ nhớ chỉ đọc và được dùng để điều khiển các sản phẩm và hệ thống cho người dùng và thị trường công nghiệp. Có thể thực hiện các chức năng đơn giản nhưng mang tính chuyên biệt (huyền bí), ví dụ: điều khiển chức năng cho lò vi sóng; hay có thể đưa ra các khả năng điều khiển và vận hành (chức năng số hoá ở ô-tô, kiểm soát xăng, biểu thị bảng đồng hồ, các hệ thống phanh…).

Nhóm 6: *Phần mềm máy tính cá nhân.*

Loại phần mềm này bùng nổ trong hơn thập kỷ vừa qua (như xử lý văn bản, trang tính, đồ hoạ, quản trị cơ sở dữ liệu). Hiện nay được tiếp tục phát triển biểu thị giao diện người máy, tạo ra sự thân thiện, dễ sử dụng cho người dùng.

Nhóm 7: *Phần mềm trí tuệ nhân tạo.*

Dùng các thuật toán phi số để giải quyết các vấn đề phức tạp mà tính toán hay phân tích trực tiếp đều không thể quản lý nổi. Phần mềm này hoạt động mạnh ở hệ chuyên gia (hệ cơ sở tri thức); trong lĩnh vực nhận dạng và xử lý hình ảnh và âm thanh; chứng minh các định lý và chơi trò chơi. Hiện nay phát triển mạnh mạng nơ-ron nhân tạo: mô phỏng cấu trúc việc xử lý trong bộ não của con người.

1.5. Chất lượng phần mềm

1.5.1. Tính đúng đắn

Tính đúng đẫn của phần mềm được thể hiện ở chỗ sản phẩm đó thực hiện đầy đủ và chính xác các yêu cầu của người dùng. Tính đúng đắn phải được hiểu theo nghĩa rộng là chương trình cần phải thực hiện được cả trong trường hợp mà dữ liệu đầu vào không phù hợp.

Tính đúng đắn của một sản phẩm phần mềm được xác minh qua các căn cứ sau:

- *Tính đúng đắn của thuật toán.*
- *Tính tương đương của chương trình với thuật toán. Thuật toán có thể đúng nhưng chương trình lập ra không tương đương với thuật toán nên khi thực hiện sẽ cho kết quả sai.*

- *Tính đúng đắn của chương trình có thể được chứng minh trực tiếp trong văn bản của chương trình.*
- *Tính đúng đắn cũng có thể được khẳng định dần qua kiểm thử, việc áp dụng chương trình trong một khoảng thời gian dài trên một diện rộng với tần suất sử dụng cao.*

1.5.2. Tính tiến hoá

Cho phép người dùng có thể khai báo các thay đổi về quy định với phần mềm tuỳ theo thay đổi của thế giới thực liên quan (ví dụ: thay đổi về quy định tính thuế thu nhập cá nhân, định mức thu bảo hiểm….).

Sản phẩm có thể mở rộng, tăng cường về mặt chức năng một cách dễ dàng.

1.5.3. Tính hiệu quả

Tính hiệu quả của một phần mềm được xác định qua các tiêu chuẩn sau:

- *Hiệu quả kinh tế hoặc ý nghĩa, giá trị thu được do áp dụng sản phẩm đó.*
- *Tốc độ xử lý phần mềm (v) được tính bằng tỉ lệ giữa khối lượng đối tượng cần xử lý (m) và tổng thời gian (t) cần thiết để xử lý các đối tượng đó.*
- *Sử dụng tối ưu tài nguyên của máy tính (CPU, bô nhớ…).*

1.5.4. Tính tiện dụng

Sản phẩm phải tính đến những yếu tố sau đây của người dùng:

- *Dễ học, có giao diện trực quan tự nhiên*
- *Dễ thao tác, dễ sử dụng…*

1.5.5. Tính tương thích

Sản phẩn có thể trao đổi dữ liệu (import/export) với các phần mềm khác có liên quan (Microsoft Excel, Microsoft Word…).

- *Giao tiếp nội bộ*
- *Giao tiếp bên ngoài*

1.5.6. Tính tái sử dụng

Sản phẩn phần mềm có thể áp dụng cho nhiều lĩnh vực theo nhiều chế độ làm việc khác nhau.

- *Các phần mềm cùng lớp*
- *Các phần mềm khác lớp*

1.6. Giới thiệu về Công nghệ phần mềm (Software engineering)

1.6.1. Định nghĩa

Công nghệ phần mềm là một lĩnh vực nghiên cứu của tin học nhằm đưa ra các *nguyên lý, phương pháp, công cụ, cách tiếp cận* phục vụ cho việc thiết kế và cài đặt các sản phẩm phần mềm đạt được các yêu cầu về chất lượng phần mềm.

Có thể định nghĩa tóm tắt về công nghệ phần mềm như sau:

"Công nghệ phần mềm là một ngành khoa học nghiên cứu về việc xây dựng các phần mềm có chất lượng trong một khoảng thời gian và chi phí hợp lý".

1.6.2. Đối tượng nghiên cứu

- Quy trình công nghệ phần mềm: hệ thống các giai đoạn mà quá trình phát triển phần mềm cần trải qua.

- 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 công nghệ phần mềm.

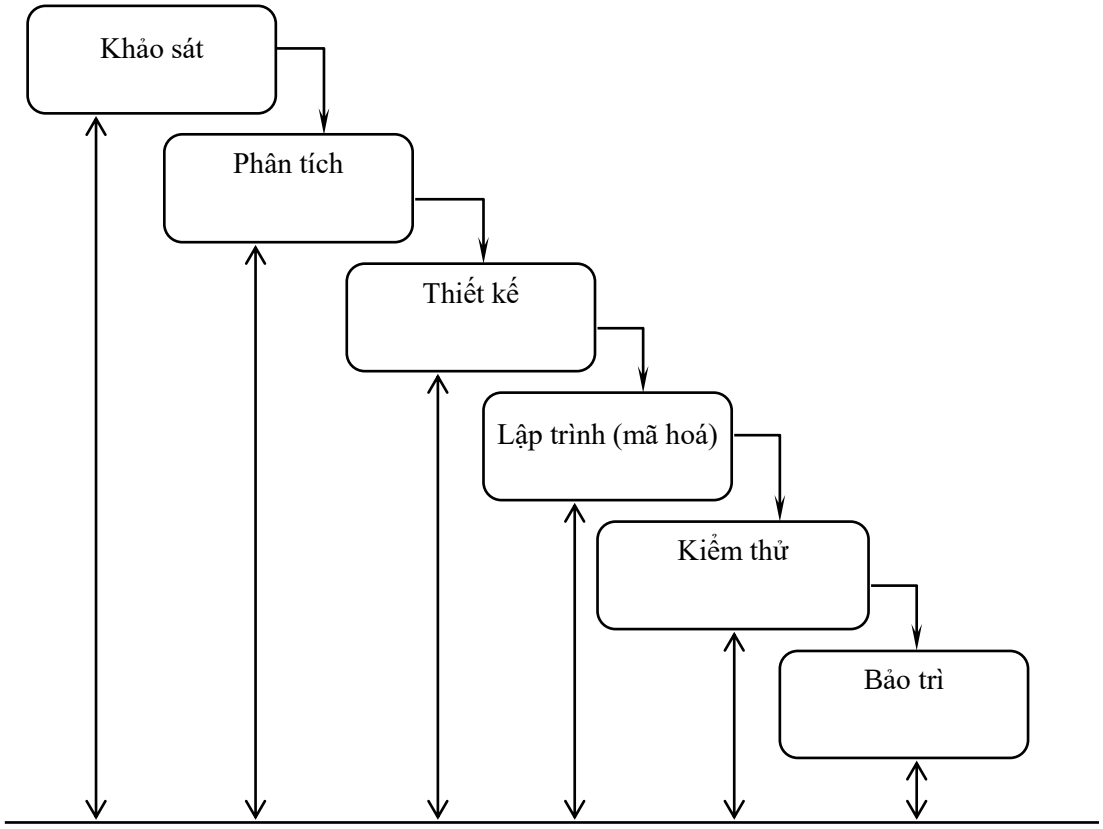
Bài tập:

1. Trình bày vai trò của phần mềm
2. Trình bày các đặc điểm của phần mềm
3. Các ứng dụng của phần mềm
4. Trình bày yêu cầu chất lượng phần mềm

Chương 2. Các mô hình phát triển phần mềm

2.1. Mô hình thác nước (Waterfall model)

Đôi khi còn được gọi là *mô hình tuần tự tuyến tính* hay *mô hình thác nước*, mô hình này gợi ý một cách tiếp cận tuần tự, có hệ thống tới việc phát triển phần mềm vốn bắt đầu từ mức hệ thống và tiến dần qua phân tích, thiết kế, mã hoá, kiểm thử và hỗ trợ. Dưới đây minh hoạ mô hình thác nước cho kĩ nghệ phần mềm. Được mô hình hoá theo chu kì kĩ nghệ qui ước, mô hình thác nước



Hình 2: Mô hình thác nước

bao gồm các hoạt động sau:

Khảo sát. Bởi vì phần mềm bao giờ cũng là một phần của một hệ thống (hay nghiệp vụ) lớn hơn nên công việc bắt đầu từ việc thiết lập yêu cầu cho mọi phần tử hệ thống và rồi cấp phát một tập con các yêu cầu đó cho phần mềm. Quan điểm hệ thống này là điều bản chất khi phần mềm phải tương tác với các thành phần khác như phần cứng, con người và CSDL. Kĩ nghệ và phân tích hệ thống bao gồm việc thu thập yêu cầu ở mức hệ thống với một lượng nhỏ thiết kế và phân tích mức đỉnh. Kĩ nghệ thông tin bao gồm việc thu thập yêu cầu tại mức nghiệp vụ chiến lược và tại mức lĩnh vực nghiệp vụ.

Phân tích. Tiến trình thu thập yêu cầu được tăng cường và hội tụ đặc biệt vào phần mềm. Để hiểu được bản chất của các chương trình phải xây dựng, kĩ sư phần mềm ("nhà phân tích") phải hiểu về lĩnh vực thông tin (được mô tả trong phần sau) đối với phần mềm cũng như chức năng cần

có, hành vi, hiệu năng và giao diện. Các yêu cầu cho cả hệ thống và phần mềm cần phải được lập tư liệu và xét duyệt cùng với khách hàng.

Thiết kế. Thiết kế phần mềm thực tế là một tiến trình nhiều bước tập trung vào bốn thuộc tính phân biệt của chương trình: cấu trúc dữ liệu, kiến trúc phần mềm, biểu diễn giao diện và chi tiết thủ tục (thuật toán). Tiến trình thiết kế dịch các yêu cầu thành một biểu diễn của phần mềm có thể được định giá về chất lượng trước khi giai đoạn mã hoá bắt đầu. Giống như các yêu cầu, việc thiết kế phải được lập tư liệu và trở thành một phần của cấu hình phần mềm.

Lập trình (mã hoá). Thiết kế phải được dịch thành dạng máy đọc được. Bước mã hoá thực hiện nhiệm vụ này. Nếu thiết kế được thực hiện theo một cách chi tiết thì việc sinh mã có thể được thực hiện một cách máy móc.

Kiểm thử. Một khi mã đã được sinh ra thì việc kiểm thử chương trình bắt đầu. Tiến trình kiểm thử hội tụ vào nội bộ logic của phần mềm, đảm bảo rằng tất cả các câu lệnh đều được kiểm thử, và vào bên ngoài chức năng; tức là tiến hành các kiểm thử để làm lộ ra các lỗi và đảm bảo những cái vào đã định sẽ tạo ra kết quả thống nhất với kết quả muốn có.

Bảo trì. Phần mềm chắc chắn sẽ phải trải qua những thay đổi sau khi nó được bàn giao cho khách hàng (một ngoại lệ có thể là những phần mềm nhúng). Thay đổi sẽ xuất hiện bởi vì gặp phải lỗi, bởi vì phần mềm phải thích ứng với những thay đổi trong môi trường bên ngoài (chẳng hạn như sự thay đổi do hệ điều hành mới hay thiết bị ngoại vi mới), hay bởi vì khách hàng yêu cầu nâng cao chức năng hay hiệu năng. Việc bảo trì phần mềm phải áp dụng lại các bước vòng đời nói trên cho chương trình hiện tại chứ không phải chương trình mới.

Mô hình tuần tự tuyến tính là mô hình cũ nhất và được sử dụng rộng rãi nhất cho kĩ nghệ phần mềm. Tuy nhiên, những chỉ trích về mô hình này đã làm cho những người ủng hộ nó tích cực phải đặt vấn đề về tính hiệu quả của nó. Một số các vấn đề thỉnh thoảng gặp phải khi dùng mô hình tuần tự tuyến tính này là:

Các dự án thực hiếm khi tuân theo dòng chảy tuần tự mà mô hình đề nghị. Mặc dầu mô hình tuyến tính có thể cho phép lặp, nhưng điều đó chỉ làm gián tiếp. Kết quả là những thay đổi có thể gây ra lẫn lộn khi tổ dự án tiến hành.

Khách hàng thường khó phát biểu mọi yêu cầu một cách tường minh. Mô hình tuần tự tuyến tính đòi hỏi điều này và thường khó thích hợp với sự bất trắc tự nhiên tồn tại vào lúc đầu của nhiều dự án.

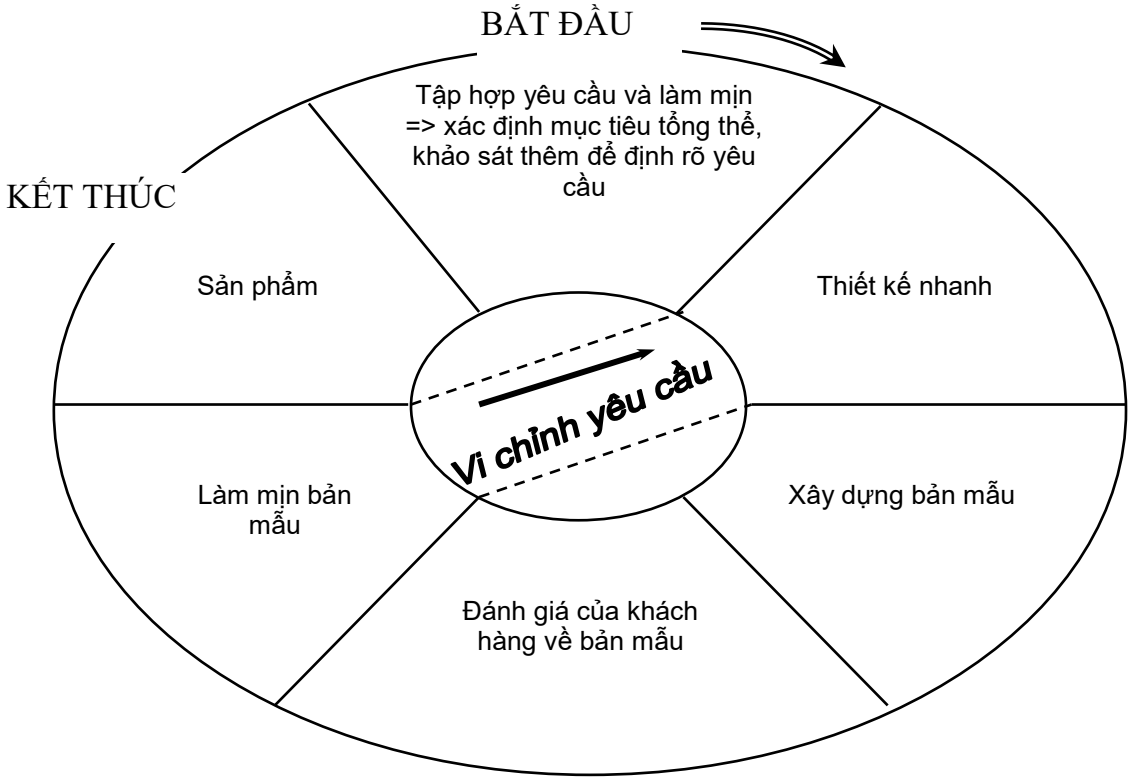
Khách hàng phải kiên nhẫn. Bản làm việc được của chương trình chỉ có được vào lúc cuối của thời gian dự án. Một sai lầm ngớ ngẩn, nếu đến khi có chương trình làm việc mới phát hiện ra, có thể sẽ là một thảm hoạ.

Trong một phân tích thú vị về các dự án hiện tại, Brada thấy rằng bản chất tuyến tính của vòng đời cổ điển dẫn tới "các trạng thái nghẽn" mà trong đó một số thành viên tổ dự án phải đợi cho các thành viên khác của tổ hoàn thành các nhiệm vụ phụ thuộc. Trong thực tế, thời gian mất cho việc chờ đợi có thể vượt quá thời gian dành cho công việc sản xuất. Trạng thái nghẽn có khuynh hướng phổ biến vào lúc đầu và cuối của tiến trình tuần tự tuyến tính.

Từng vấn đề trên đều là thực. Tuy nhiên, mô hình vòng đời cổ điển có một vị trí quan trọng và xác định trong công việc về kĩ nghệ phần mềm. Nó đưa ra một tiêu bản trong đó có thể bố trí các phương pháp cho phân tích, thiết kế, mã hoá, kiểm thử và bảo trì. Bên cạnh đó, vòng đời cổ điển vẫn còn là một mô hình thủ tục được dùng rộng rãi cho kĩ nghệ phần mềm. Trong khi nó quả thực còn điểm yếu, nó vẫn tốt hơn đáng kể nếu so với cách tiếp cận ngẫu nhiên tới việc phát triển phần mềm.

2.2. Mô hình làm bản mẫu (Prototyping model)

Thông thường khách hàng đã xác định một tập các mục tiêu tổng quát cho phần mềm, nhưng còn chưa định danh các yêu cầu cái vào chi tiết, hay xử lí cái ra. Trong các trường hợp khác, người phát triển có thể không chắc về tính hiệu quả của thuật toán, việc thích nghi hệ điều hành hay dạng giao diện người máy cần có. Trong những trường hợp này và nhiều trường hợp khác *mô hình làm bản mẫu* có thể đưa ra cách tiếp cận tốt nhất.



Mô hình làm bản mẫu (hình dưới) bắt đầu với việc *thu thập yêu cầu*. Người phát triển và khách hàng gặp nhau và *xác định các mục tiêu tổng thể* cho phần mềm, xác định các yêu cầu nào đã biết, và miền nào bắt buộc phải xác định thêm. Rồi đến việc *"thiết kế nhanh"*. Thiết kế nhanh tập trung vào việc biểu diễn các khía cạnh của phần mềm thấy được đối với người dùng (như cách đưa vào và định dạng đưa ra). Thiết kế nhanh dẫn tới việc xây dựng một bản mẫu. Bản mẫu được khách hàng / người dùng *đánh giá* và được dùng để *làm mịn các yêu cầu* đối với phần mềm cần phát triển. Tiến trình lặp đi lặp lại xảy ra để cho bản mẫu được "*vi chỉnh*" thoả mãn nhu cầu của khách hàng trong khi đồng thời lại làm cho người phát triển hiểu được kĩ hơn cần phải thực hiện nhu cầu nào.

Một cách lí tưởng, bản mẫu phục vụ như một cơ chế để xác định các yêu cầu phần mềm. Nếu một bản mẫu làm việc được xây dựng thì người phát triển có thể dùng được các đoạn chương trình đã có hay áp dụng các công cụ (như bộ sinh báo cáo, bộ quản lí cửa sổ, v.v..) để nhanh chóng sinh ra chương trình làm việc.

Nhưng chúng ta nghĩ về bản mẫu thế nào khi nó được dùng cho mục đích được nêu trên? Brook đã nêu ra câu trả lời:

Trong hầu hết các dự án, hệ thống đầu tiên hiếm khi sử dụng được. Nó có thể là quá chậm, quá lớn, cồng kềnh trong sử dụng hay tất cả những nhược điểm này. Không có cách nào khác là bắt đầu lại, đau đớn nhưng tinh khôn hơn, và xây dựng một phiên bản được thiết kế lại trong đó những vấn đề này đã được giải quyết... Khi một khái niệm hệ thống mới hay một kĩ nghệ mới được dùng, người ta phải xây dựng một hệ thống để rồi vứt đi, cho dù việc lập kế hoạch được thực hiện chu đáo nhất thì nó cũng không thể bao quát hết để chạy đúng được ngay lần đầu. Do đó câu hỏi quản lí không phải là liệu chúng ta có nên xây dựng một hệ thống thử nghiệm và rồi vứt nó đi hay không. Bạn sẽ làm như vậy. Câu hỏi duy nhất là liệu nên lập kế hoạch trước để xây dựng một cái vứt đi hay để hứa hẹn bàn giao cái vứt đi đó cho khách hàng...

Bản mẫu có thể phục vụ như "hệ đầu tiên" - cái mà Brook lưu ý chúng ta nên vứt đi. Nhưng điều này có thể là một cách nhìn lí tưởng hoá. Giống như mô hình tuyến tính tuần tự (thác nước), việc làm bản mẫu tựa như một mô hình cho kĩ nghệ phần mềm có thể trở thành có vấn đề bởi những lí do sau:

1. Khách hàng thấy được cái dường như là phiên bản làm việc của phần mềm mà không biết rằng bản mẫu được gắn lại "bằng kẹo cao su và dây gói hàng", không biết rằng trong khi xô đẩy để cho nó làm việc thì chẳng ai xem xét tới chất lượng phần mềm tổng thể hay tính bảo trì thời gian dài. Khi được thông báo rằng sản phẩm phải được xây dựng lại để cho có thể đạt tới mức độ chất lượng cao, thì khách hàng kêu trời và đòi hỏi rằng "phải ít sửa chữa" để làm bản mẫu thành sản phẩm làm việc. Rất thường là việc quản lí phát triển phần mềm bị buông lỏng.

2. Người phát triển thường hay thoả hiệp cài đặt để có được bản mẫu làm việc nhanh chóng. Hệ điều hành hay ngôn ngữ lập trình không thích hợp có thể được dùng đơn giản bởi vì nó có sẵn và đã biết; một thuật toán không hiệu quả có thể được cài đặt đơn giản để chứng tỏ khả năng. Sau một thời gian, người phát triển mới có thể trở nên quen thuộc với những chọn lựa này và quên mất mọi lí do tại sao chúng lại không thích hợp. Việc chọn lựa không được theo lí tưởng bây giờ lại trở thành một phần tích hợp của hệ thống.

Mặc dầu vấn đề có thể xuất hiện, việc làm bản mẫu có thể là một mô hình hiệu quả cho kĩ nghệ phần mềm. Chìa khoá là định nghĩa ra các qui tắc của trò chơi từ ngay lúc bắt đầu; tức là khách hàng và người phát triển phải cùng đồng ý rằng bản mẫu được xây dựng để phục vụ làm cơ chế xác định yêu cầu. Thế rồi nó phải bị bỏ đi (ít nhất cũng một phần) và phần mềm thực tại được đưa vào kĩ nghệ với con mắt hướng về chất lượng và tính bảo trì được.

2.3. Mô hình phát triển nhanh (RAD model)

Mô hình phát triển nhanh (RAD – Rapid Application Development) chính là mô hình tăng dần với chu kỳ phát triển cực ngắn. Để đạt được mục tiêu này, RAD dựa trên phương pháp phát triển trên cơ sở thành phần hoá hệ thống cùng với việc tái sử dụng các thành phần thích hợp. RAD thích hợp cho những hệ thống quản lý thông tin.

RAD - dựa vào phương pháp luận, điều chỉnh các giai đoạn SDLC để tạo ra một sốphần của hệthống phát triển nhanh và vào các thao tác thủ công của người sử dụng.

Phần lớn RAD - dựa vào phương pháp luận mà người phân tích sử dụng các kỹ thuật đặc biệt và công cụ máy tính để tăng tốc các giai đoạn phân tích, thiết kế, và thực hiện, như công cụ CASE (computer-aided software engineering).

2.4. Mô hình tăng trưởng (Incremental model)

Thay vì chuyển giao một lần, quá trình phát triển và chuyển giao được chia làm nhiều lần, mỗi chuyển giao đáp ứng một phần chức năng.

- Yêu cầu người dùng được phân loại ưu tiên, mức cao sẽ thuộc phần chuyển giao sớm.
- Khi phát triển một bản tăng, yêu cầu tương ứng là cố định, tuy nhiên, yêu cầu cho bản tăng sau vẫn phát triển.

2.5. Mô hình xoắn ốc (Spiral model)

Mô hình xoắn ốc, ban đầu do Boehm đề xuất, là mô hình tiến trình phần mềm tiến hoá vốn cặp đôi bản chất lặp của làm bản mẫu với các khía cạnh hệ thống và có kiểm soát của mô hình trình tự tuyến tính. Nó cung cấp tiềm năng cho việc phát triển nhanh các phiên bản tăng dần của phần mềm. Dùng mô hình xoắn ốc này, phần mềm được phát triển thành từng chuỗi các lần đưa ra

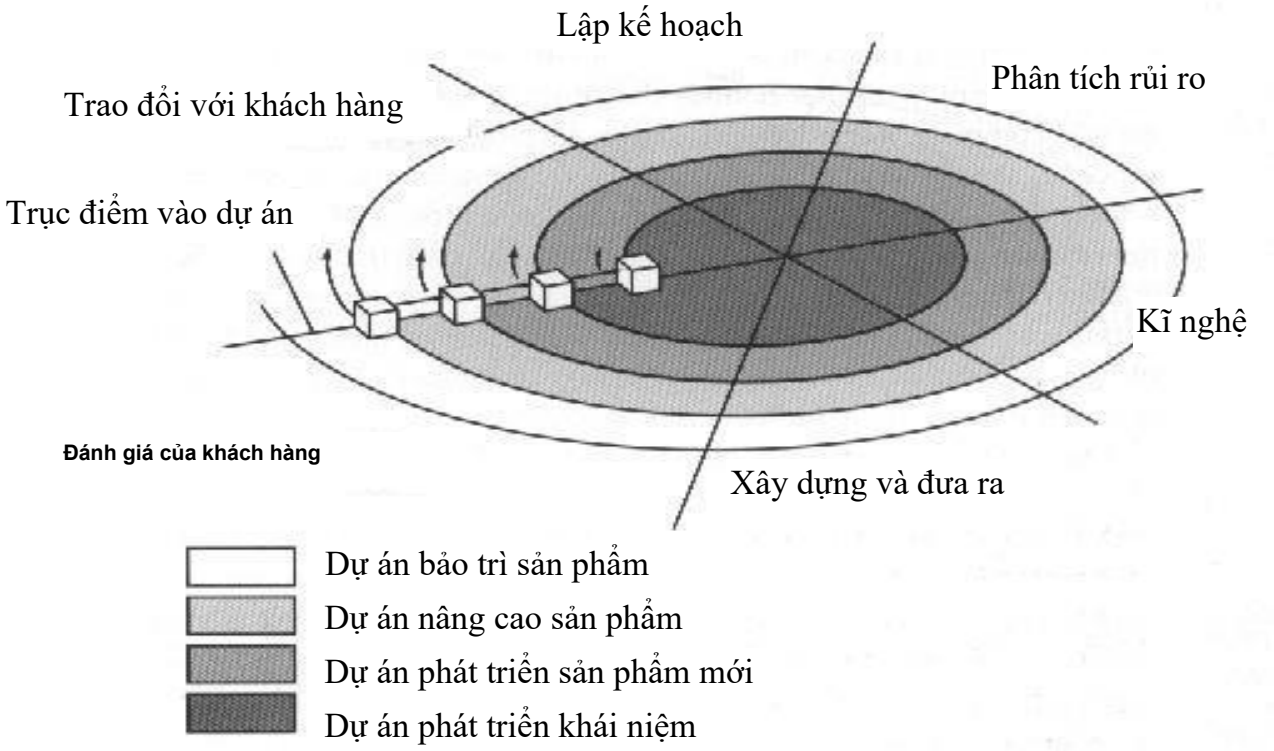
tăng dần. Trong những lần lặp đầu, việc đưa ra tăng dần có thể là mô hình trên giấy hay bản mẫu. Trong các lần lặp sau, các phiên bản đầy đủ tăng dần của hệ thống được kĩ nghệ hoá sẽ được tạo ra.

Mô hình xoắn ốc được chia thành một số khuôn khổ hoạt động, cũng còn được gọi là *vùng nhiệm vụ*. Về cơ bản, có từ ba tới sáu vùng. Hình sau mô tả cho mô hình xoắn ốc có chứa sáu vùng:

1. ***Trao đổi với khách hàng*** - nhiệm vụ đòi hỏi thiết lập việc trao đổi có hiệu quả giữa người phát triển và khách hàng.
2. ***Lập kế hoạch*** - nhiệm vụ đòi hỏi định nghĩa các tài nguyên, hạn thời gian và các thông tin liên quan tới dự án.
3. ***Phân tích rủi ro*** - nhiệm vụ đòi hỏi định giá cả những rủi ro kĩ thuật và quản lí
4. ***Kĩ nghệ*** - nhiệm vụ đòi hỏi xây dựng một hay nhiều biểu diễn cho ứng dụng
5. ***Xây dựng và đưa ra*** - nhiệm vụ đòi hỏi xây dựng, kiểm thử, thiết đặt và cung cấp sự hỗ trợ cho người dùng (như tài liệu và huấn luyện)
6. ***Đánh giá của khánh hàng*** - nhiệm vụ đòi hỏi thu được phản hồi của khách hàng dựa trên đánh giá về biểu diễn phần mềm được tạo ra trong giai đoạn kĩ nghệ và được cài đặt trong giai đoạn cài đặt.

Mỗi một trong các vùng đều được đặt vào một tập các nhiệm vụ, được gọi là *tập nhiệm vụ*, vốn được thích ứng với các đặc trưng của dự án được tiến hành. Với các sự án nhỏ, số các nhiệm vụ công việc và tính hình thức của chúng là thấp. Với các dự án lớn, nhiều căng thẳng hơn, thì mỗi vùng nhiệm vụ lại chứa nhiều nhiệm vụ công việc vốn được xác định để đạt tới mức độ hình thức cao hơn. Trong mọi trường hợp, hoạt động hỗ trợ (như quản lí cấu hình phần mềm và đảm bảo chất lượng phần mềm) - được nêu trong phần sau - sẽ được áp dụng.

Khi tiến trình tiến hoá này bắt đầu, tổ kĩ nghệ phần mềm đi vòng xoắn ốc theo chiều ngược kim đồng hồ, bắt đầu từ trung tâm. Mạch đầu tiên quanh xoắn ốc có thể làm phát sinh việc phát triển đặc tả sản phẩm; các bước tiếp theo quanh xoắn ốc có thể được dùng để phát triển bản mẫu và thế rồi các phiên bản phức tạp dần thêm. Mỗi bước qua vùng lập kế hoạch lại làm nảy sinh việc điều chỉnh kế hoạch dự án. Chi phí và lịch biểu được điều chỉnh dựa trên phản hồi được suy từ đánh giá của khách hàng. Bên cạnh đó, người quản lí dự án điều chỉnh số việc lặp đã lập kế hoạch cần để hoàn chỉnh phần mềm.



Không giống như mô hình tiến trình cổ điển vốn kết thúc khi phần mềm được chuyển giao, mô hình xoắn ốc có thể được thích ứng để áp dụng trong toàn bộ cuộc đời của phần mềm máy tính. Một cái nhìn khác có thể được xem xét bằng việc *kiểm tra trục điểm vào dự án*, như được vẽ trong hình trên. Mỗi hình hộp được đặt theo trục có thể được dùng để biểu diễn cho điểm bắt đầu cho các kiểu dự án khác nhau. "Dự án phát triển khái niệm" bắt đầu tại cốt lõi của xoắn ốc và sẽ tiếp tục (nhiều lần lặp xuất hiện theo con đường xoắn ốc mà vốn gắn với vùng tô đậm trung tâm) cho tới khi việc phát triển khái niệm là đầy đủ. Nếu khái niệm này được phát triển thành một sản phẩm thực tại, thì tiến trình tiến qua hình hộp tiếp (điểm vào dự án phát triển sản phẩm mới) và một "dự án phát triển mới" được khởi đầu. Sản phẩm mới sẽ tiến hoá qua một số lần lặp quanh xoắn ốc, đi theo con đường vốn gắn vùng có tô mầu sáng hơn vùng lõi. Về bản chất, xoắn ốc, khi được đặc trưng theo cách này, vẫn còn làm việc cho tới khi phần mềm được cho nghỉ. Có những

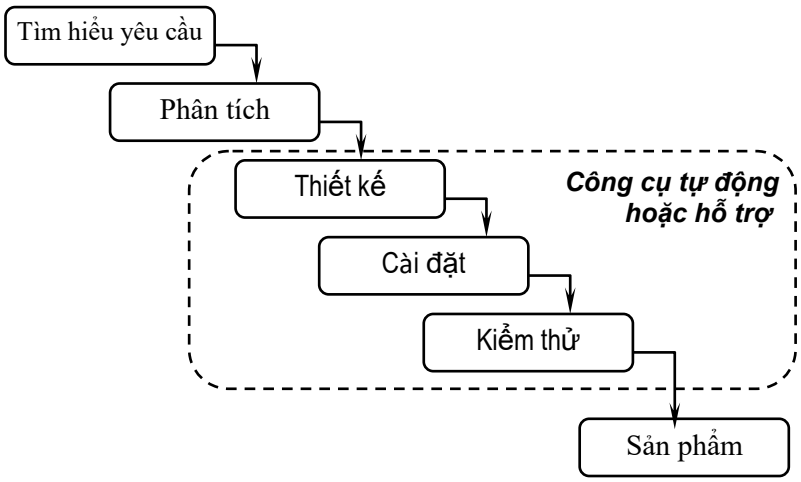
lúc tiến trình này *"ngủ"*, nhưng bất kì khi nào một thay đổi được khởi đầu, thì tiến trình này lại bắt đầu tại điểm vào thích hợp (tức là nâng cao sản phẩm).

Mô hình xoắn ốc là cách tiếp cận thực tế cho việc phát triển cho các hệ thống và phần mềm qui mô lớn. Bởi vì phần mềm tiến hoá khi tiến trình tiến hoá, nên người phát triển và khách hàng hiểu rõ hơn và phản ứng với rủi ro tại từng mức tiến hoá. Mô hình xoắn ốc dùng cách làm bản mẫu như một cơ chế làm giảm bớt rủi ro, nhưng điều quan trọng hơn, làm cho người phát triển có khả năng áp dụng được cách tiếp cận làm bản mẫu tại mọi giai đoạn trong tiến hoá của sản phẩm. Nó duy trì cách tiếp cận từng bước một cách có hệ thống do cách tiếp cận vòng đời cổ điển gợi ý, nhưng tổ hợp cách tiếp cận này vào một khuôn khổ lặp lại, vốn phản ánh được sát thực hơn thế giới thực. Mô hình xoắn ốc đòi hỏi việc xem xét trực tiếp các rủi ro kĩ thuật tại mọi giai đoạn của dự án, và nếu được áp dụng đúng thì nó có thể làm giảm rủi ro trước khi chúng trở thành vấn đề thực sự.

Nhưng giống như các mô hình khác, mô hình xoắn ốc không phải là một liều thuốc bách bệnh. Có thể khó thuyết phục những khách hàng (đặc biệt trong tình huống có hợp đồng) rằng cách tiếp cận tiến hoá là kiểm soát được. Nó đòi hỏi tri thức chuyên gia đánh giá rủi ro chính xác và dựa trên tri thức chuyên gia này mà đạt được thành công. Nếu một rủi ro chính không được phát hiện và quản lí thì không nghi ngờ gì nữa vấn đề sẽ xuất hiện. Cuối cùng, chính bản thân mô hình này cũng còn chưa được sử dụng rộng rãi như mô hình trình tự tuyến tính hoặc làm bản mẫu. Cần phải có thêm một số năm nữa trước khi tính hiệu quả của mô hình quan trọng này có thể được xác định với sự chắc chắn hoàn toàn.

2.6. Các mô hình hiện đại (Fourth generation techniques)

Thuật ngữ *kĩ thuật thế hệ thứ tư* (4GT) bao gồm một phạm vi rộng các công cụ phần mềm có một điểm chung: mỗi công cụ đều cho phép người kĩ sư phần mềm xác định đặc trưng nào đó của phần mềm ở mức cao. Rồi công cụ đó tự động sinh ra mã chương trình gốc dựa trên đặc tả của người phát triển. Người ta gần như không còn bàn cãi về việc phần mềm có thể được xác định đối với một máy càng ở mức cao thì chương trình có thể được xây dựng càng nhanh hơn. Mô hình 4GT cho kĩ nghệ phần mềm tập trung vào khả năng xác định phần mềm bằng việc dùng các khuôn mẫu ngôn ngữ đặc biệt hay kí pháp đồ hoạ vốn mô tả cho vấn đề cần được giải quyết dưới dạng khách hàng có thể hiểu được.



Hình 3: Mô hình kỹ nghệ thứ 4 - 4GT

Hiện tại, một môi trường phát triển phần mềm hỗ trợ cho mô hình 4GT bao gồm một số hay tất cả các công cụ sau:

- Ngôn ngữ phi thủ tục để hỏi đáp cơ sở dữ liệu.
- Bộ sinh báo cáo.
- Bộ thao tác dữ liệu.
- Bộ tương tác và xác định màn hình.
- Bộ sinh chương trình.
- Khả năng đồ hoạ mức cao.
- Khả năng làm trang tính và việc sinh tự động HTML.
- Các ngôn ngữ tương tự được dùng cho việc tạo ra trang Web thông qua việc dùng các công cụ phần mềm tiên tiến.

Ban đầu nhiều trong những công cụ đã được nhắc tới đó đã có sẵn chỉ cho những lĩnh vực ứng dụng rất đặc thù, nhưng ngày nay môi trường 4GT đã được mở rộng để đề cập tới hầy hết các loại ứng dụng phần mềm.

Giống như các mô hình khác, 4GT bắt đầu từ bước thu thập yêu cầu. Một cách lý tưởng, khách hàng sẽ mô tả các yêu cầu và các yêu cầu đó sẽ được dịch trực tiếp thành một bản mẫu vận hành được. Nhưng điều này không thực hiện được. Khách hàng có thể không chắc chắn mình cần gì, có thể có sự mơ hồ trong việc xác định các sự kiện đã biết, có thể không có khả năng hay không sẵn lòng xác định thông tin theo cách thức mà công cụ 4GT có thể giải quyết được. Bởi lí do này, đối thoại khách hàng/ người phát triển được mô tả cho các mô hình tiến trình khác vẫn còn là phần bản chất của cách tiếp cận 4GT.

Với những ứng dụng nhỏ, có thể chuyển trực tiếp từ bước thu thập yêu cầu sang cài đặt bằng cách dùng *ngôn ngữ sinh thế hệ thứ tư phi thủ tục* (4GL) hay một mô hình bao gồm một mạng các biểu tượng đồ hoạ. Tuy nhiên với nỗ lực lớn hơn, cần phải phát triển một chiến lược thiết kế cho hệ thống, ngay cả nếu có dùng 4GL. Việc dùng 4GT thiếu thiết kế (với các dự án lớn) sẽ gây ra cùng những khó khăn (chất lượng kém, khó bảo trì, người dùng khó chấp nhận) mà chúng ta đã gặp phải khi phát triển phần mềm bằng cách dùng các cách tiếp cận qui ước.

Việc cài đặt dùng 4GL làm cho người phát triển phần mềm biểu diễn được các kết quả mong muốn theo cách là phát sinh tự động chương trình tính ra chúng. Hiển nhiên, một cấu trúc dữ liệu với những thông tin có liên quan cần phải có sẵn và sẵn sàng cho 4GL truy nhập vào.

Để biến đổi một cài đặt 4GT thành một sản phẩm, người phát triển phải tiến hành việc kiểm thử toàn diện, xây dựng các tài liệu có ý nghĩa và thực hiện mọi hoạt động tích hợp giải pháp khác vốn cần tới trong các mô hình kĩ nghệ phần mềm khác. Bên cạnh đó, phần mềm được phát triển theo 4GT phải được xây dựng theo cách làm cho việc bảo trì có thể được tiến hành một cách chóng vánh.

Giống như mọi mô hình kĩ nghệ phần mềm, mô hình 4GT có ưu điểm và nhược điểm. Những người ủng hộ cho là làm giảm đáng kể thời gian phát triển phần mềm và làm tăng rất nhiều hiệu suất của người xây dựng phần mềm. Những người phản đối cho là các công cụ 4GT hiện tại không phải tất cả đều dễ dùng hơn các ngôn ngữ lập trình, rằng chương trình gốc do các công cụ này tạo ra là "không hiệu quả," và rằng tính bảo trì cho các hệ thống phần mềm lớn được phát triển bằng cách dùng 4GT vẫn còn là vấn đề mở.

Có đôi điều lợi ích trong các luận điểm của cả hai phía và có thể tóm tắt trạng thái hiện tại của cách tiếp cận 4GT như sau:

1. Việc dùng 4GT là cách tiếp cận có thể tồn tại được cho nhiều lĩnh vực ứng dụng khác nhau. Gắn với các công cụ kĩ nghệ phần mềm có máy tính hỗ trợ và bộ sinh mã, 4GT cung cấp một giải pháp tin cậy được cho nhiều vấn đề phần mềm.
2. Dữ liệu được thu thập từ các công ty có dùng 4GT chỉ ra rằng thời gian cần cho việc tạo ra phần mềm được giảm đáng kể đối với các ứng dụng vừa và nhỏ và rằng khối lượng thiết kế và phân tích cho các ứng dụng nhỏ cũng được rút bớt.
3. Tuy nhiên, việc dùng 4GT cho các nỗ lực phát triển phần mềm lớn đòi hỏi nhiều phân tích, thiết kế và kiểm thử (các hoạt động kĩ nghệ phần mềm) để đạt tới việc tiết kiệm thời gian vốn nảy sinh từ việc xoá bỏ mã hoá.

Tóm lại, các kĩ thuật thế hệ thứ tư đã trở thành một phần quan trọng của kĩ nghệ phần mềm. Khi đi đôi với cách tiếp cận dựa trên cấu phần (sẽ được trình bày ở mục tiếp theo), mô hình 4GT có thể trở thành cách tiếp cận thống trị cho việc phát triển phần mềm.

Bài tập:

1. Trình bày mô hình thác nước
2. Phân biệt mô hình bản mẫu với mô hình thác nước
3. Phân biệt mô hình tăng trưởng với mô hình thác nước
4. Đánh giá ưu nhược điểm của từng mô hình phát triển phần mềm

Chương 3. Quy trình phát triển phần mềm eXtreme Programming (XP)

3.1. Giới thiệu về XP

Lập trình cực độ (eXtreme Programming, viết tắt là XP) là một phương pháp linh hoạt dành cho các nhóm phát triển phần mềm nhỏ và trung bình xây dựng các phần mềm có yêu cầu thay đổi một cách nhanh chóng. Đối với lập trình viên, XP đảm bảo rằng họ sẽ làm những công việc hữu ích theo khả năng của họ. Đối với khách hàng và người quản lý, XP đảm bảo rằng mang lại những lợi ích tốt nhất có thể sau mỗi thời gian làm việc. Họ sẽ nhìn thấy mục đích của quá trình thực hiện và có thể thay đổi nó mà không phát sinh nhiều chi phí.

XP sử dụng các nhóm làm việc kết hợp gồm những người lập trình, khách hàng và các nhà quản trị để phát triển phần mềm có chất lượng cao trong thời gian nhanh chóng. Một chương trình chạy được là thước đo đầu tiên của tiến trình theo XP. XP có thể phát triển và tồn tại được là do sự hiểu biết ngày một tiến bộ về các vấn đề đang giải quyết và cũng là vì các công cụ sẵn có cho phép ta thay đổi được cái giá của sự thay đổi (cost-of-change). XP giữ cho cái giá phải trả này ở mức thấp do vậy sẽ thúc đẩy môi trường sản xuất phần mềm.

3.2. Vai trò, quyền hạn và trách nhiệm của các tác nhân trong XP

Có nhiều vai trò khác nhau trong XP đối với những tác vụ khác nhau và mục đích trong suốt quá trình xử lý và vận hành.

Programmer

Programmer sẽ viết những testing và giữ cho code của chương trình đơn giản và rõ ràng như có thể. Vấn đề đầu tiên tạo nên sự thành công của XP đó là việc giao tiếp và kết hợp với những Programmer và Team khác.

Customer

Customer viết những story và kiểm tra chức năng, quyết định khi yêu cầu được thỏa mãn. Khách hàng sẽ thiết lập độ ưu tiên cài đặt cho từng yêu cầu.

Tester

Tester giúp đỡ Customer thực hiện việc kiểm tra các chức năng. Họ sẽ thực hiện việc kiểm trac các chứ năng thường xuyên, công bố kết quả và duy trì những công cụ kiểm tra.

Tracker

Tracker gửi feedback vào XP. Anh ta sẽ dò và ước lượng thành viên mỗi Team và gửi feedback để xác định làm thế nào để có thể cải thiện chức năng đánh giá 1 cách đúng đắn.

Coach

Coach là người chịu trách nhiệm về quá trình xử lý sao cho đầy đủ. Hiểu biết đúng đắn có cơ sở của XP rất quan trọng trong vai trò kích hoạt coach để hướng dẫn thành viên Team khác tiếp tục xử lý.

Consultant

Consultant là thành viên bên ngoài xử lý những kiến thức về công nghệ đặc trưng cần thiết. Consultant hướng dẫn Team xử lí những vấn đề đặc trưng.

Manager (Big Boss)

Manager nắm vai trò đưa ra quyết định. Để làm được việc này, anh ta phải kết nối với Team thực hiện dự án để xác định trạng thái hiện tại và phân biệt những khó khăn hoặc xử lý những thiếu hụt trong quá trình xử lý.

3.3. Các giá trị cốt lõi của XP

XP là một phương pháp có khả năng thích nghi, thích ứng. Điều đó có nghĩa là sẽ không có hai dự án XP nào giống nhau cả. XP cung cấp một tập hợp các thực hành và được sử dụng như là điểm khởi đầu, và sau đó được làm cho thích ứng để phù hợp với các ràng buộc của từng dự án riêng.

3.3.1. Sự giao tiếp (Communication)

Mục đích của XP là giữ một luồng giao tiếp đúng đắn bằng các hoạt động cần sự giao tiếp như: kiểm thử đơn vị, lập trình theo đôi, …

Một XP team lớn mạnh dựa trên các kiến thức, sự hiểu biết bài toán và hiểu biết phần mềm được chia sẻ. Các phương pháp giải quyết vấn đề được trao đổi trực tiếp. Những thứ cản trở đến công việc đều được loại bỏ.

XP chú trọng việc trao đổi thông tin một cách 'trong suốt' giữa các thành viên trong nhóm phát triển. Đề cao việc trao đổi trực tiếp, giảm việc trao đổi gián tiếp hay hình thức thông qua các văn bản.

Với XP, khách hàng tham gia trực tiếp vào việc thực hiện dự án với tư cách là một thành viên chính thức của nhóm phát triển. Khách hàng sẽ giúp nhóm phát triển hiểu và nắm bắt được

và kịp thời các yêu cầu của người sử dụng (cũng như sự thay đổi về yêu cầu) trong suốt quá trình thực hiện dự án.

Tất cả các thành viên đều tham gia vào mọi hoạt động trong quá trình phát triển phần mềm.

3.3.2. Sự đơn giản (Simplicity)

Quan điểm trong XP là thực hiện một công việc đơn giản hôm nay, bổ sung thêm vào ngày mai và thay đổi nếu cần thiết chứ không phải là làm một công việc phức tạp mà có thể không cần thiết. Sự giao tiếp và sự đơn giản có mối liên hệ chặt chẽ với nhau.

XP đảm bảo chỉ phát triển những chức năng mà khách hàng yêu cầu. Phần thiết kế và mã nguồn được thiết lập một cách đơn giản nhất, cho phép có được đặc tính 'mở' cao nhằm đáp ứng với các thay đổi liên tục và luôn duy trì được một tốc độ phát triển nhanh trong suốt quá trình phát triển phần mềm.

3.3.3. Sự phản hồi (Feedback)

Sự phản hồi được thực hiện ở nhiều mức độ khác nhau: giữa các lập trình viên hằng ngày, giữa khách hàng và người kiểm thử hàng tuần.

Thường các đội làm dự án và khách hàng của họ không nhận ra những vấn đề rắc rối cho tới khi sắp bàn giao sản phẩm. Nhưng các đội XP thường xuyên lấy phản hồi – trong quá trình làm việc, kiểm thử, bàn giao sản phẩm … Khi đó sẽ điều khiển được các vấn đề phát sinh.

Phản hồi sớm và liên tục từ khách hàng cũng như từ nhóm phát triển giúp cho dự án luôn đi theo đúng hướng. XP đều đặn giao sản phẩm cho khách hàng để kiểm tra, theo đó khách hàng có thể 'làm mịn' và hoàn thiện yêu cầu sản phẩm dựa trên các kết quả cụ thể.

3.3.4. Sự dũng cảm (Courage)

Khi nhóm phát triển thấy rằng không thể tiếp tục quá trình hiện tại, họ sẽ thay đổi nó. Điều này có thể phải bỏ đi một nữa các trường hợp kiểm thử họ đã làm trước đó, và sẽ tốn thêm một vài ngày cố gắng sau đó. Tuy vậy, họ có thể hướng đến mục đích hoàn thành.

Các đội làm phần mềm thành công cần phải kiểm soát được ngay cả khi xuất hiện các lỗi. XP đưa ra 12 phương án thực hành, và điểm mạnh của XP chính là đã kết hợp được các phương án này lại. Mỗi một phương án tuy đơn giản nhưng rất cần thiết phải nắm vững, sẽ góp phần làm giảm bớt đáng kể cái giá của sự thay đổi.

XP cho rằng phải có lòng dũng cảm thì mỗi thành viên mới thực hiện được các nguyên tắc kể trên. Tuy XP không chỉ ra một cách rõ ràng, nhưng cũng cần phải nhấn mạnh rằng tính kỷ luật là yêu cầu quan trọng để thực hiện có hiệu quả phương pháp phát triển phần mềm XP.

3.4. Vòng đời phát triển của một dự án XP

3.4.1. Khởi tạo (Exploration)

Khách hàng sẽ viết toàn bộ một Story Card mà họ hi vọng sẽ được thêm vào trong phiên bản đầu tiên. Mỗi Story Card mô tả chức năng được thêm vào trong chương trình. Tại cùng một thời điểm đó Project Team sẽ làm quen với Story Card bằng các Tools, Công nghệ và thực hành, họ sẽ sử dụng trong Project. Công nghệ sử dụng sẽ được kiểm tra kỹ lưỡng, còn kiến trúc hệ thống nếu như có triển vọng sẽ được khảo sát một cách tỉ mỉ bằng việc xây dựng các mẫu ban đầu. Thời gian để hoành thành pha này mất khoảng vài tuần cho tới vài tháng, điều đó còn phụ thuộc vào độ phức tạp của công nghệ đối với các Programmers.

3.4.2. Lập kế hoạch (Planning)

Pha này sẽ thiết lập các vị trí ưu tiên cho các Story. Ngoài ra còn xác nhận đồng ý cho nội dung của Version đầu tiên. Programmer trước tiên sẽ ước lượng độ yêu cầu của mỗi Story và lên kế hoạch làm việc sau thời điểm xác nhận đồng ý nội dung. Quãng thời gian của kế hoạch làm việc cho version đầu tiên không vượt quá 2 tuần.

3.4.3. Chuyển giao từng phần (Iterations to Release)

Pha này bao gồm một vài bước lặp của hệ thống trước khi cho ra đời Version đầu tiên. Kế hoạch làm việc được thiết lập trong bản kế hoạch bị hỏng tới một số lượng bước lặp nào đó sẽ mất từ 1 tới 4 tuần để cài đặt. Bước lặp đầu tiên sẽ tạo hệ thống với kiến trúc của một hệ thống đầy đủ. Đó là bản lưu trữ bởi việc lựa chọn các story sẽ thúc đây việc xây dựng cấu trúc cho hệ thống đầy đủ. Khách hàng chấp nhận story được lựa chọn cho mỗi bước lặp. Quá trình test các chức năng sẽ được tạo bởi khách hàng được thực thi ở cuối mỗi bước lặp. Tại cuối bước lặp cuối cùng hệ thống đủ sẵn sang để có thể sản xuất.

3.4.4. Triển khai hoàn thiện sản phẩm (Productionizing)

Pha này yêu cầu mở rộng hơn về việc Testing cũng như Checking về khả năng thực thi của hệ thống trước khi hệ thống chính thức được giao tới tay của khách hàng. Tại pha này, sự thay đổi mới vẫn có thể được tìm ra và quyết đinh thực hiện chúng nếu như chúng được chứa trong phiên bản hiện tại. Trong suốt quá trình của pha này, các bước lặp có thể sẽ trở nên cần thiết để xử lí nhanh chóng từ 1 tới 3 tuần.

Sau khi phiên bản được phát hành đầu tiên được sản xuất hóa cho khách hàng sử dụng, dự án XP phải giữ được hệ thống trong quá trình thực thi sản phẩm khi hầu hết việc thực hiện các bước lặp mới. Để làm được việc đó, thì pha Bảo trì cần sự nỗ lực từ việc hỗ trợ của khách hàng. Theo cách đó tốc độ phát triển có thể được hãm lại sau khi hệ thống đã trở thành sản phẩm.

3.4.5. Duy trì sản phẩm (Maintenance)

Cũng gần giống như việc khi khách hàng không còn bất cứ 1 story nào để phát triển. Quy định này có nghĩa như việc khách hàng cần thỏa mãn hệ thống ở nhiều khía cạnh. Đây là thời điểm trong quá trình XP xử lý khi những tài liệu cần thiết của hệ thống được hoàn thành hay như việc không có bất kì thay đổi nào trong kiến trúc, thiết kế hoặc code. Death hầu hết xảy ra nếu như hệ thống không được giao đúng như yêu cầu hoặc nếu như nó trở nên quá đắt đỏ so với chi phí để phát triển.

3.5. Các công việc cốt lõi trong XP

3.5.1. Lập kế hoạch (The Planning Game)

Với XP, khách hàng tham gia trực tiếp vào quá trình lập kế hoạch phát triển phần mềm. Vai trò của khách hàng và nhóm phát triển được định ra một cách rõ ràng.

Trách nhiệm của khách hàng:

- Mô tả tính năng phần mềm cần phát triển thông qua các ' câu chuyện' (user story). User story có ý nghĩa tương tự như use case trong UML nhưng mức độ mô tả thì không chi tiết bằng. Phân loại các user story theo mức độ quan trọng từ quan điểm người sử dụng (dựa trên giá trị kinh doanh-business value). Từ đó sẽ định ra tính năng nào cần phải phát triển và phát triển theo thứ tự như thế nào.

- Định ra thời điểm và chu kỳ bàn giao sản phẩm

Trách nhiệm của nhóm phát triển:

- Ước lượng yêu cầu kỹ thuật (để phát triển) cho từng user story (ước lượng độ phức tạp).
- Ước lượng thời gian, nhân công cũng như giá thành để phát triển từng user story.

3.5.2. Chuyển giao từng phần (Small releases)

Do nhóm XP làm việc trong các bước nhỏ cho nên việc phát hành cũng chia ra thành các phát hành nhỏ (khoảng vài tuần một lần). Và các thành viên sẽ phải tích hợp liên tục. Có những đề án XP thực hiện việc phát hành hàng ngày.

Theo quy cách này, nhóm phát triển sẽ phát triển dần dần phần mềm, từ đơn giản đến phức tạp. Từng phần sẽ được chuyển giao cho khách hàng để có được ngay sự phản hồi của khách hàng. Từ đó sẽ có thể điều chỉnh ngay được sản phẩm cho phù hợp với yêu cầu của khách hàng. Khách hàng cũng có điều kiện để bổ sung hay thay đổi yêu cầu phần mềm.

3.5.3. Bảng định danh (Metaphor)

Nhóm phát triển XP dùng chung một hệ thống các thuật ngữ để biểu diễn hệ thống cần phát triển. Các thuật ngữ này sẽ được dùng trong khi trao đổi giữa các thành viên trong nhóm cũng như khi trao đổi với khách hàng.

3.5.4. Thiết kế đơn giản (Simple design)

Để giảm giá phải trả cho sự thay đổi đồng nghĩa với việc làm cho hệ thống càng đơn giản càng tốt. Điều đó có nghĩa là không nên bỏ quá nhiều thời gian và công sức vào những việc mà sau này có thể cần hoặc cũng có thể không. Các đề án XP thường được đơn giản một cách tối đa, đảm bảo rằng sau này nếu có cần thay đổi thì chi phí cũng rất nhỏ.

XP khuyến khích tìm kiếm giải pháp đơn giản khi thiết kế phần mềm. Chỉ thiết kế phần mềm thoả mãn yêu cầu hiện tại của khách hàng, không nên tìm kiếm một giải pháp cho một hệ thống tương lai. Theo đó, chỉ cần một thiết kế làm sao cho chương trình chạy được và thỏa mãn yêu cầu của khách hàng.

3.5.5. Kiểm thử liên tục (Testing)

Các lập trình viên sẽ viết các đơn vị thử nghiệm trước khi viết mã (thiết kế thử nghiệm đầu tiên). Tất cả các đơn vị thử nghiệm (trường hợp thử nghiệm) đều được thực hiện thường xuyên trong quá trình phát triển sản phẩm trong từng module mã của từng lập trình viên. Các lập trình viên có thể thay đổi một cách linh hoạt vì quy trình thử nghiệm có thể mắc lỗi hay sai so với thiết kế ban đầu.

XP yêu cầu rất cao trong khâu kiểm thử và kiểm định chương trình. Với mỗi phần của chương trình, lập trình viên phải viết chương trình kiểm thử cho phần đó trước khi thực sự bắt đầu khi viết chương trình (cho phần đó). Khách hàng sẽ chịu trách nhiệm thực hiện kiểm định sản phẩm.

3.5.6. Hoàn thiện liên tục (Refactoring)

Tái chế là kỹ thuật làm tăng hiệu quả của việc thết kế các mã có sẵn mà không làm thay đổi chức năng. Tái chế là rất khả thi vì một nhóm XP có các quy trình thử nghiệm tự động bắt lỗi,

cho phép ta thay đổi mã (phản ánh khả năng hiểu bài toán ngày càng cao của các thành viên). Qua đó cũng mở rộng các thiết kế lên.

Quan điểm của XP là chất lượng phần mềm được thể hiện bằng chất lượng của mã nguồn (code). Một chương trình được viết rõ ràng, đơn giản thì sẽ dễ bảo dưỡng và thay đổi. XP khuyến khích tổ chức lại chương trình một cách đều đặn để nâng cao tính sáng sủa của chương trình, dễ bổ sung các chức năng mới, nâng cao hiệu suất của chương trình.

3.5.7. Lập trình theo đôi (Pair programming)

Bất kỳ người nào trong đội dự án đều có quyền thay đổi mã trong quá trình làm việc với khách hàng chỉ cần tuân theo Tiêu chuẩn mã hoá và phải đảm bảo thực hiện thử nghiệm lại toàn bộ sau khi hoàn tất công việc sửa đổi. Điều này sẽ loại bỏ các vấn đề như là sai lệch về cấu trúc chương trình, … có thể xảy ra khi một cá nhân mã hoá độc lập.

Tất cả các phần chương trình do một hay nhiều nhóm hai người viết. Hai người này sẽ sử dụng chung một máy tính, cùng đồng thời viết chương trình. Quy cách này sẽ giúp cho có được giải pháp lập trình tốt hơn, chương trình sẽ có chất lượng và hiệu quả hơn.

3.5.8. Chia sẻ công việc (Collective ownership)

Mọi thành viên trong nhóm đều phải học và sử dụng thành thạo các Nguyên lý và dạng thức thiết kế. Thứ nhất, nó giúp cho cả nhóm làm việc với nhau một cách ăn ý (liên lạc tốt). Sau đó là giúp cho việc viết mã của từng thành viên được tốt và nhanh do tái sử dụng được kinh nghiệm từ người đi trước, điều này rất quan trọng, vì trong XP không có thiết kế chi tiết, từng đoạn mã/từng module phải do từng thành viên của nhóm thể hiện, vì vậy nếu áp dụng được thì sẽ giảm thiểu được quá trình điều chỉnh/tái chế.

Tất cả mã nguồn đều thuộc quyền sở hữu của mọi thành viên trong nhóm phát triển. Theo đó, mã nguồn có thể được sửa đổi ngay khi cần. Với cách quản lý thông thường, mỗi phần mã nguồn thường do một người quản lý, nên khi cần sửa đổi thì phải cần sự thông qua chủ sở hữu, đôi khi điều này gây mất nhiều thời gian.

3.5.9. Tích hợp liên tục (Continuous integration)

Các nhóm XP chia công việc ra thành các bước nhỏ và tích hợp mã của họ một vài lần trong một ngày. Do vậy, các vấn đề sẽ được xem xét ngay sau khi thực hiện và có thể dễ dàng sửa chữa khi gặp sự cố. Quá trình này đảm bảo cho mọi người luôn làm việc với phiên bản mới nhất của hệ thống.

Việc tích hợp sẽ được tiến hành một cách liên tục. Khi một đoạn chương trình mới được phát triển, đã vượt qua phần kiểm thử, thì sẽ được tích hợp ngay vào hệ thống. Điều này sẽ giúp cho việc phát hiện và sửa lỗi thích hợp nhanh hơn và rẻ hơn. Trong một ngày có thể thực hiện nhiều lần tích hợp hệ thống.

3.5.10. Làm việc cùng khách hàng (On-site customer)

Các lập trình viên phải luôn tiếp xúc với khách hàng để xác định rõ nhu cầu bất kể nỗ lực tốn bao nhiêu. Các nhà lập trình XP không nên suy đoán các vấn đề cụ thể của một chức năng mà phải hỏi trực tiếp khách hàng.

Với XP, khách hàng sẽ tham gia cách trực tiếp trong suốt quá trình phát triển phần mềm. Sự tham gia này sẽ giúp cho nhóm phát triển có điều kiện tham khảo trực tiếp ý kiến của khách hàng, trao đổi về hệ thống cần được phát triển, tránh được nhầm lẫn trong cách hiểu về hệ thống cần phát triển. Mục tiêu cuối cùng là sản phẩm làm ra phù hợp với yêu cầu của khách hàng.

3.5.11. Sử dụng các chuẩn viết mã (Coding standards)

Đây là một loạt các quy ước về mã hoá để các thành viên của dự án theo đó làm. Khi đó mọi người có thể xem xét lẫn nhau và có thể bàn giao được cho nhau.

Để chương trình (mã nguồn) có thể hiểu được một cách dễ dàng, nhất là đối với các quy cách lập trình đôi và sở hữu tập thể, nhóm phát triển phải thống nhất cách viết chương trình. Cần phải có một quy định cụ thể, rõ ràng về cách viết (ví dụ, cách đặt tên biến, cách bổ sung chú thích...v.v.) để làm sao tất cả đều hiểu được.

3.5.12. Giới hạn 40 giờ/tuần (40-hour week)

Việc phát triển phần mềm là một công việc sáng tạo, và họ sẽ không thể sáng tạo được nếu họ kiệt sức. Việc giới hạn số giờ làm việc trong tuần sẽ đảm bảo được sức khoẻ của các thành viên và tăng cường chất lượng sản phẩm.

Hiện tượng làm việc quá giờ rất phổ biến trong giới phát triển phần mềm. Thực tế cho thấy khi người lao động làm việc quá giờ thường hay mệt mỏi, dẫn đến làm việc không hiệu quả, chất lượng sản phẩm giảm. XP khuyến cáo không nên làm việc quá giờ, chỉ làm đúng giờ quy định để đảm bảo chất lượng sản phẩm.

Bài tập

1) Các giá trị cốt lõi trong XP
2) Vòng đời của một dự án XP

3) Các công việc cốt lõi trong XP

Chương 4. Quy trình phát triển phần mềm thống nhất Rational Unified Process (RUP)

4.1. Giới thiệu

Trong phát triển phần mềm, có những sai sót làm ảnh hưởng không nhỏ đến chất lượng sản phẩm. Các sai sót này có thể phát sinh từ nhiều nguồn khác nhau trong quá trình xây dựng hệ thống, chẳng hạn như không quản lý được các yêu cầu, không phát hiện lỗi kịp thời, không quản lý được các thay đổi của dự án.

- RUP là một quy trình vòng lặp phát triển phần mềm được tạo ra bởi công ty Rational Software, một bộ phận của IBM từ năm 2002 (IBM Rational).

- RUP không phải là một quy trình bó hẹp cụ thể đơn nhất nhưng là một nền tảng quy trình thích ứng với sự phát triển các tổ chức và các nhóm dự án phần mềm, tất cả sẽ chọn các yếu tố cần thiết của quy trình để phù hợp với nhu cầu, quy mô của công ty, dự án và sản phẩm.

- Unified Process được thiết kế từ đặc điểm chung, quy trình phạm vi rộng lớn và RUP là một mô tả chi tiết cụ thể.

- RUP hỗ trợ các hoạt động giữa các nhóm, phân chia công việc cho từng thành viên trong nhóm, trong từng giai đoạn khác nhau của quá trình phát triển phần mềm.

- RUP sử dụng hệ thống ký hiệu trực quan của UML và RUP được phát triển song song với UML.

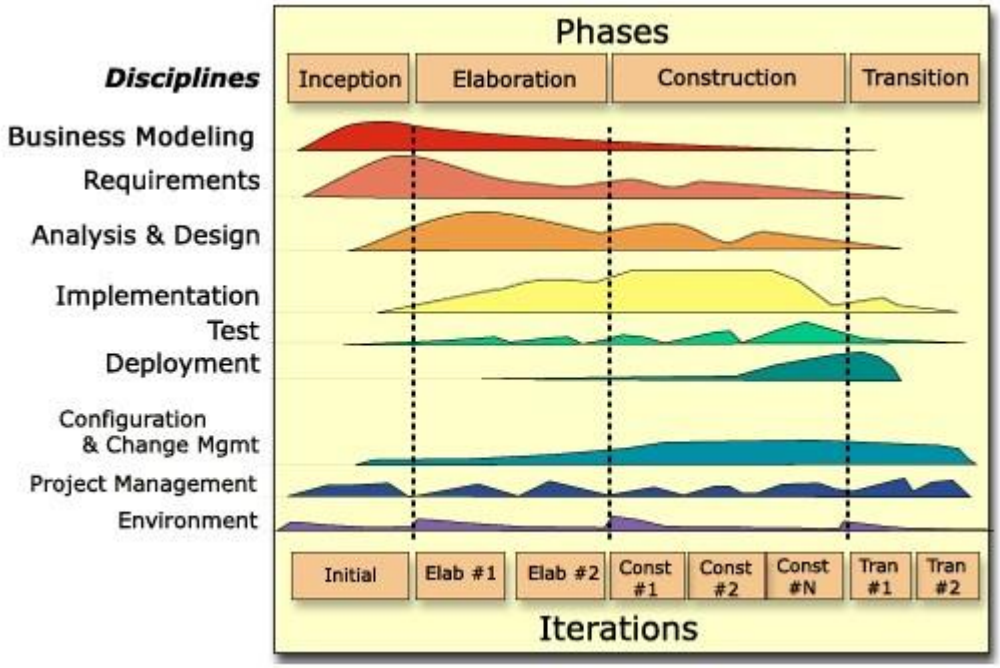
- RUP là một sản phẩm tiến trình có thể tùy biến.

4.1.1. Kiến trúc của RUP

Cấu trúc của quy trình RUP, được thể hiện theo hai chiều:

- Trục hoành: là chiều biểu diễn thời gian và vòng đời của quy trình: thể hiện mặt động của chu kì (cycles), được biểu diễn dưới dạng các giai đoạn (phase), các vòng lặp (interations) và các cột mốc thời gian (milestones).

- Trục tung: là chiều biểu diễn các tiến trình của quy trình, là các công việc được nhóm lại một cách logic theo bản chất của chúng, thể hiện mặt tĩnh dưới dạng các thành phần của chu trình như các tiến trình, các kết quả sinh ra (artifacts_WHAT), cá nhân hay một nhóm thực hiện (worker_WHO), giai đoạn công việc hoạt động liên quan với nhau (workflows_WHEN) và các đơn vị công việc (activities_HOW).



- *Luồng công việc chính:*
  - ✓ Business modeling
  - ✓ Requirement
  - ✓ Analysis & Design
  - ✓ Implementation
  - ✓ Test
  - ✓ Deployment
- *Luồng công việc hỗ trợ:*
  - ✓ Project Management
  - ✓ Configuration and Change Management
  - ✓ Enviroment

4.1.2. So sánh RUP với một số quy trình phát triển phần mềm khác

So sánh RUP với XP:

RUP không đề cập tỉ mỉ đến những định lý, tuy nhiên những nguyên lý cơ bản của phương pháp luận được đề ra rõ ràng và ta có thể thấy rõ, ví dụ như phản hồi từ khách hàng, sự thay đổi tăng dần, đầu tư ban đầu ít, thử nghiệm cụ thể và tuỳ biến theo từng nơi.

Có một số điểm tương đồng trong chiến lược lập kế hoạch, cả hai phương pháp đều phát biểu là không lập kế hoạch quá cụ thể ngay từ ban đầu bởi vì bạn không thể biết công việc gì là

quan trong ngay từ lúc đầu. Cả hai phương pháp sử dụng quan niệm vòng quay của dự án, và nhấn mạnh sự ưu tiên theo mức độ quan trọng của các chức năng. User story và use case đều được dùng để định hướng kiến trúc phần mềm và đóng vai trò quyết định cho việc lập kế hoạch cũng như quá trình phát triển.

Phương pháp luận hướng đối tượng đều là công cụ chính của cả hai phương pháp.

User story trong XP giống hệt với Use case trong RUP.

Trong các yếu tố quan trọng cấu thành dự án là phạm vi dự án, tính đúng hạn, chất lượng và tài nguyên dự án, XP khuyến cáo chúng ta cần giữ cố định mục tiêu tính đúng hạn, chất lượng và tài nguyên. Phạm vi dự án có thể được phép điều chỉnh. RUP không đặt quan điểm một cách chặt chẽ như vậy, tuy nhiên RUP cho phép chúng ta dùng phương pháp xây dựng ma trận quyết định, trong đó các yếu tố đó có thể được điều chỉnh để phù hợp theo đặc tính của từng dự án. Dẫu RUP không phát biểu cụ thể như vây về sự cho phép thay đổi phạm vi dự án, tuy nhiên quan điểm tương đồng cũng thể hiện ở phương pháp phát triển phần mềm "to dần" của RUP.

Việc kiểm tra chương trình một cách tự động đều được XP và RUP khuyến cáo. XP dựa trên việc này để đảm bảo chi phí thấp cho mỗi sự thay đổi, tuy nhiên RUP thì không đòi hỏi.

Khác nhau:

Sự khác biệt đáng kể giữa RUP và XP là RUP hướng đến những dự án lớn hơn so với XP và vì thế, nó phức tạp hơn.

Chi phí cho thay đổi và rủi ro:

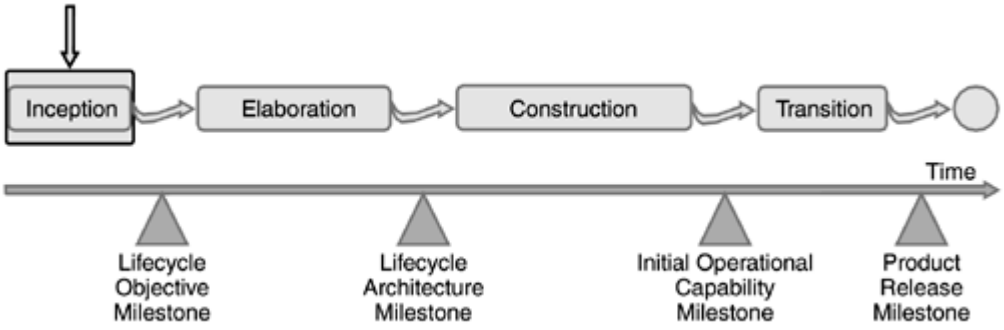
RUP cho rằng chi phí thay đổi tăng theo hàm mũ và tập trung vào cho những bước đầu tiên để giảm thiểu những chi phí về sau trong quá trình phát triển phần mềm. RUP quan tâm nhiều đến việc giảm thiểu rủi ro, theo dõi để tìm kiếm rủi ro và tiếp cận mục tiêu này từng bước từ quá trình xây dựng kiến trúc đến các quá trình phát triển phần mềm sau này.

XP cho rằng chi phí thay đổi không lớn lắm. XP cũng được xây dựng với mục tiêu giảm thiểu rủi ro (rủi ro ở đây định nghĩa là "những vấn đề cơ bản") và hướng tới một thiết kế và kiến trúc tốt cũng như với mục tiêu trước tiên là cho ra lò những chức năng quan trọng nhất. Tuy nhiên, với sự giả định là giá cho sự thay đổi thấp, kiến trúc của phần mềm được phép phát triển một cách hữu cơ, như một bộ xương chỉ cần phát triển vừa đủ để nâng đỡ cơ thể tại thời điểm nhất định. Điều then chốt trong XP là sự đơn giản.

4.2. Vòng đời của một dự án RUP

Từ phương diện quản lý, vòng đời của một phần mềm theo RUP được chia theo thời gian qua bốn giai đoạn nối tiếp nhau, mỗi giai đoạn có một mốc quan trọng, mỗi giai đoạn thực chất là khoảng giữa của 2 điểm mốc. Cuối mỗi giai đoạn, bộ phận kiểm định sẽ thực hiện thẩm định các đối tượng của giai đoạn này, nếu việc kiểm tra thích hợp thì dự án sẽ được chuyển sang giai đoạn tiếp theo.

4.2.1. Khởi tạo (Inception)



Trong giai đoạn khởi động cần đưa ra tình huống về mặt nghiệp vụ có thể có đối với hệ thống và xác định phạm vi của dự án. Các tình huống nghiệp vụ gồm: đánh giá sự thành công, đánh giá rủi ro, xác định các nguồn lực cần thiết cho dự án và một bản kế hoạch tóm tắt chỉ ra lịch trình của các điểm mốc chủ yếu của dự án, rủi ro của các yêu cầu, các chức năng nghiệp vụ cũng phải được chỉ ra trước khi dự án bắt đầu.

Trong giai đoạn này cũng đề cập đến việc cải tiến từ các hệ thống đã có, tuy nhiên vẫn chỉ tóm tắt, giai đoạn này chú trọng đến cả 2 điều quan trọng của dự án: đáng để làm hay không và khả năng thực hiện.

Cuối giai đoạn này cần kiểm tra các mục tiêu của quá trình phát triển của dự án và quyết định có tiếp tục quá trình phát triển hay không. Kết quả của giai đoạn này là đạt được sự nhất trí giữa tất cả những người đóng vai trò chủ chốt về các mục tiêu của dự án.

- *Mục tiêu chính của giai đoạn Inception:*

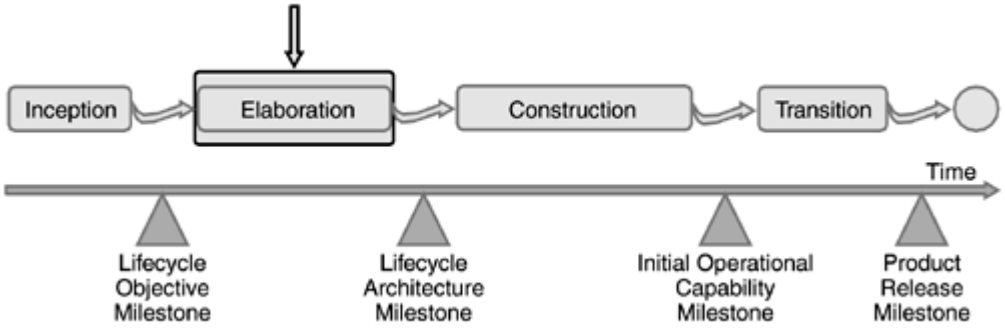
Thiết lập phạm vi phần mềm và các điều kiện biên của dự án, bao gồm: nhìn nhận khả năng thực hiện, các điều kiện thỏa thuận và những gì sản phẩm mong đợi và không mong đợi. Nhận định đúng đắn về các chức năng của hệ thống, kịch bản của các hành vi trong hệ thống sẽ đóng vai trò định hướng quan trọng cho kết quả của phần thiết kế.

Trình bày, demo một số kiến trúc ứng cử viên cho một vài kịch bản chính. Dự trù tất cả chi phí, lập kế hoạch cho toàn bộ dự án.

Dự trù các rủi ro tiềm ẩn.

Chuẩn bị môi trường hỗ trợ cho dự án.

4.2.2. Phác thảo (Elaboration)

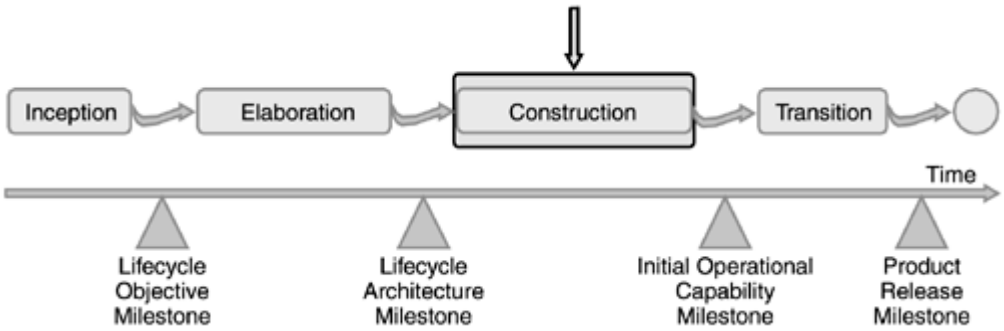


Kết quả của giai đoạn này là tạo ra một baseline cho kiến trúc của hệ thống, tạo cơ sở cho quá trình thiết kế và thực thi trong giai đoạn construction. Kiến trúc này mở rộng việc phân tích các yêu cầu quan trọng của hệ thống (các yêu cầu có sự ảnh hưởng lớn đến hệ thống) và đánh giá các rủi ro. Sự ổn định của kiến trúc được đánh gia qua nhiều nguyên bản của kiến trúc.

Mục tiêu của giai đoạn này là phân tích các vấn đề nghiệp vụ, xác định kiến trúc hợp lý, xây dựng kế hoạch cho dự án, giới hạn các yếu tố rủi ro cao nhất. Những quyết định về mặt kiến trúc cần được đưa ra cho toàn bộ hệ thống, đồng thời cần mô tả hầu hết các yêu cầu của hệ thống.

Cuối giai đoạn này cần kiểm tra các mục tiêu và phạm vi chi tiết của hệ thống, sự lựa chọn về kiến trúc và cách xử lý các rủi ro có thể đồng thời quyết định có tiếp tục chuyển sang giai đoạn xây dựng hay không.

4.2.3. Xây dựng (Construction)

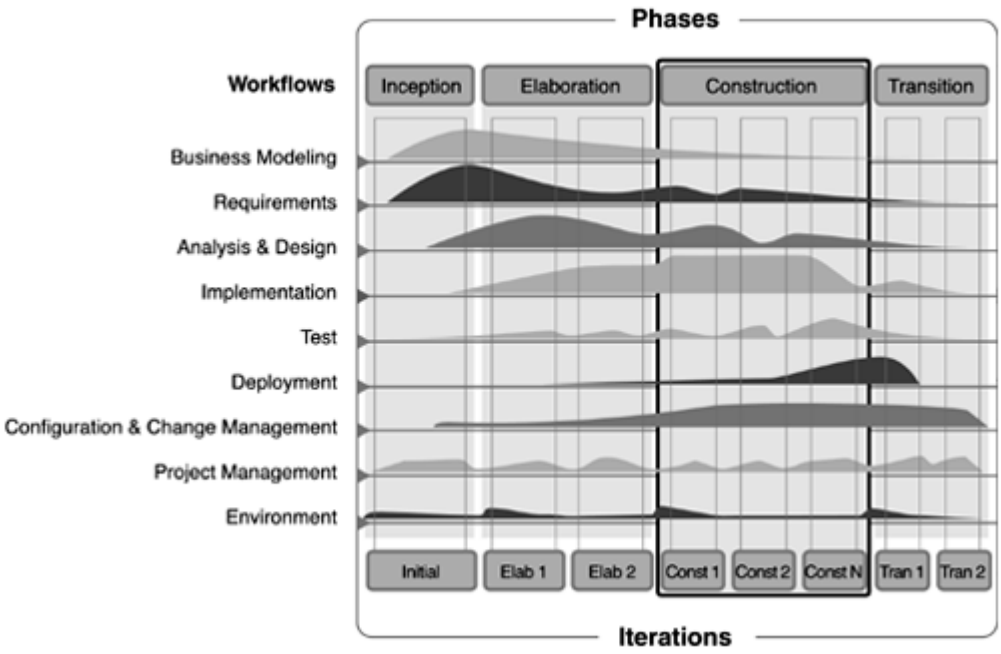


Trong giai đoạn này bạn phát triển một cách tái lập và tăng dần toàn bộ sản phẩm đầy đủ, xây dựng sản phẩm và phát triển các phiên bản, kiến trúc, các kế hoạch cho đến khi đạt được phiên bản hoàn thiện nhất sẵn sàng chuyển giao tới người sử dụng. Giai đoạn này bao gồm việc

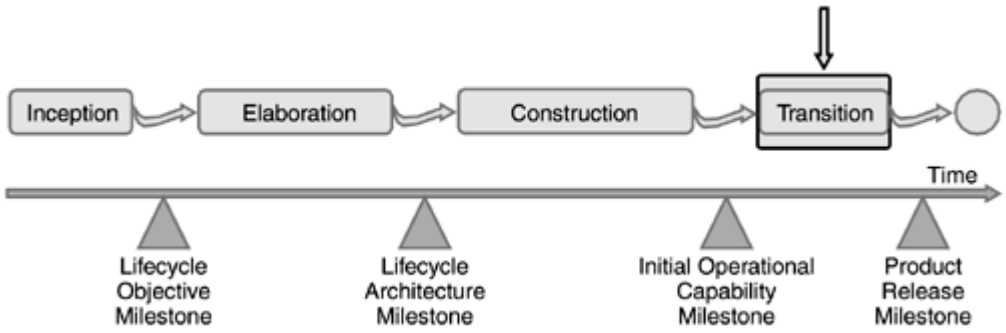
mô tả các yêu cầu còn lại chưa được xác định, xác định các tiêu chuẩn, làm mịn thiết kế và hoàn thành việc lập trình ứng dụng.

Cuối giai đoạn này cần xác định liệu hệ thống phần mềm, các điểm triển khai và người dùng đã sẵn sàng đi vào hoạt động chưa để có thể chuyển giao cho người dùng.

Giai đoạn này sẽ được kết luận dựa vào các mốc là khả năng thực hiện các chức năng yêu cầu ban đầu đã xác định.



4.2.4. Chuyển giao (Transition)



Trong giai đoạn này, cần đưa hệ thống phần mềm tới người sử dụng. Khi hệ thống đã tới tay người sử dụng thì các vấn đề thường phát sinh đòi hỏi những bước tiếp theo là căn chỉnh hệ thống, xác định các vấn đề chưa được phát hiện trước đó hay hoàn thiện các chức năng trước đó bị trì hoãn. Giai đoạn này thường bắt đầu với việc tung ra phiên bản Beta và sau đó là thay thế bởi bản chương trình đầy đủ.

Chuyển giao sản phẩm cho những người sử dụng bao gồm: hoàn chỉnh sản phẩm, phân phối, huấn luyện, hỗ trợ và bảo trì cho đến khi người sử dụng hài lòng.

Giai đoạn này được kết luận thông qua mốc các phiên bản của sản phẩm, kết thúc từng chu trình lặp của giai đoạn này.

4.3. Các luồng công việc chính trong RUP

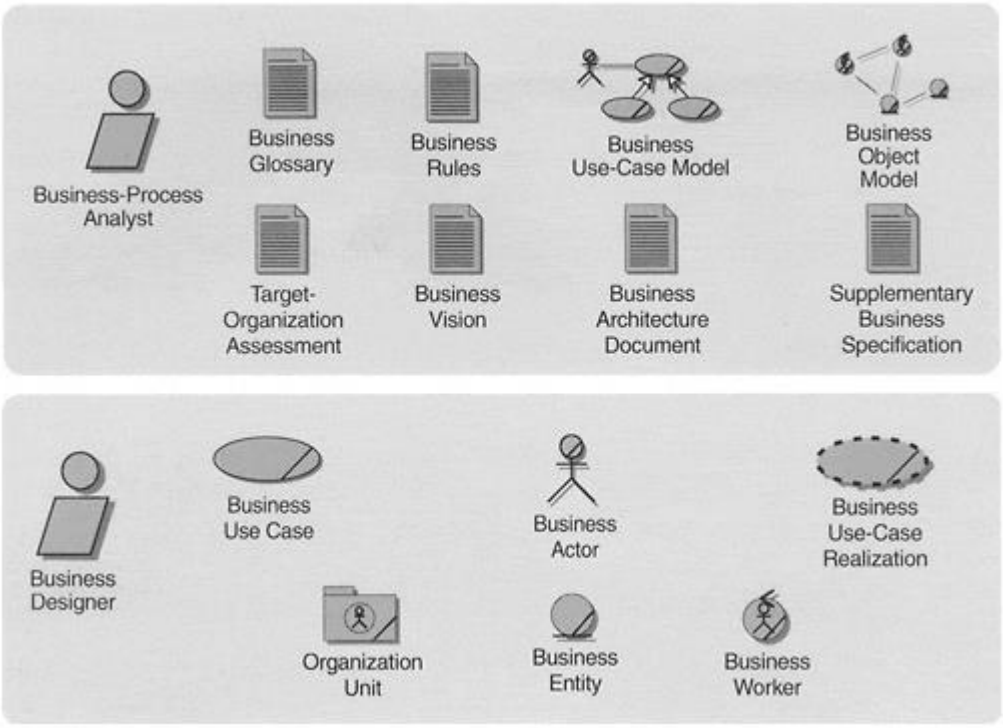
4.3.1. Mô hình hoá nghiệp vụ (Business modeling)

Mô tả cấu trúc và quy trình nghiệp vụ. Mục tiêu của mô hình hoá nghiệp vụ là:

- Hiểu được vấn đề đang tồn tại trong tổ chức và đề xuất cải tiến.

- Để đảm bảo khách hàng, người dùng cuối, người phát triển có sự hiểu biết thống nhất về hệ thống.

- Để tìm ra những yêu cầu hệ thống cần thiết.



Các ký hệu quy ước cho mô hình hoá nghiệp vụ

4.3.2. Quản lý yêu cầu (Requirements management)

Mô tả nghiệp vụ bằng phương pháp "tình huống sử dụng" (use case base method). Mục tiêu của luồng công việc này là:

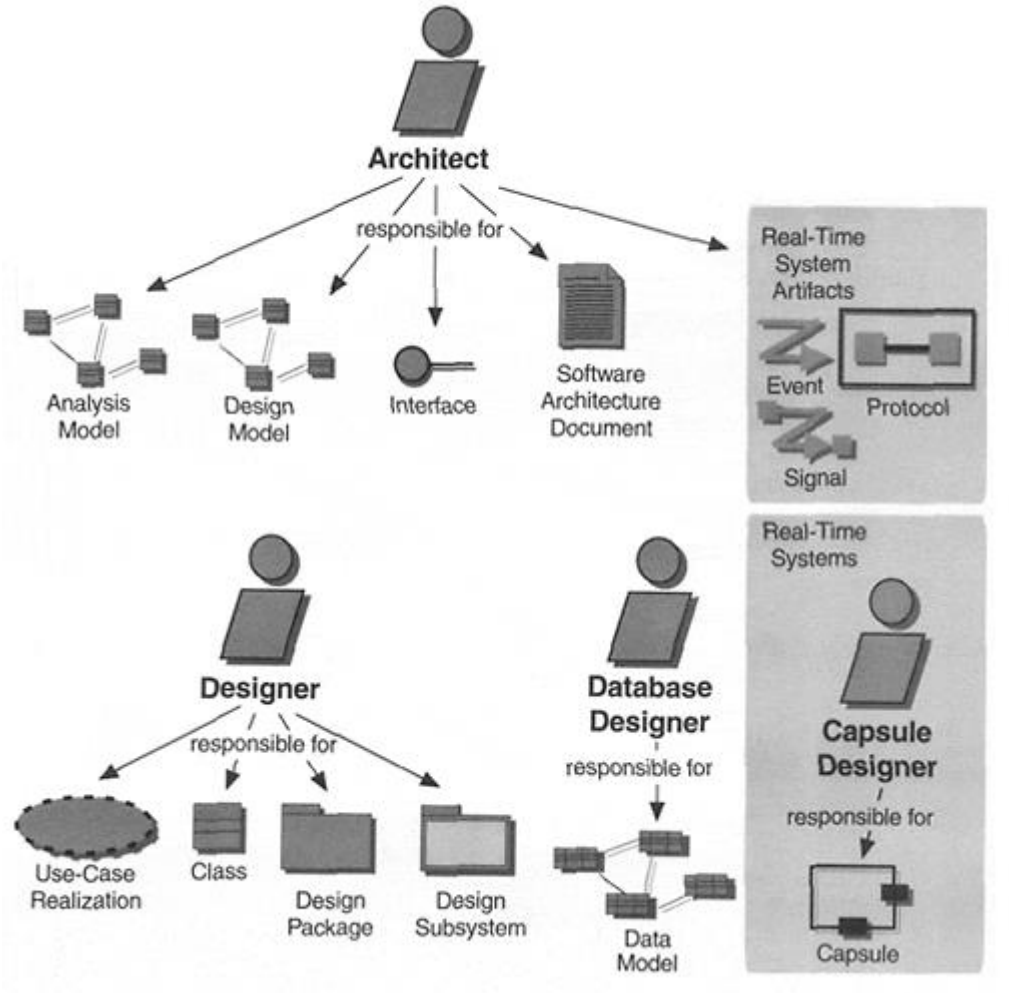
- Thiết lập và duy trì sự đúng đắn về yêu cầu của khách hàng hoặc các nhân tố khác về những gì mà hệ thống sẽ thực hiện.

- Giúp cho người phát triển hiểu rõ hơn về những yêu cầu của hệ thống
- Xác định giới hạn của hệ thống.
- Giúp cho việc ước lượng thời gian và chi phí phát triển hệ thống

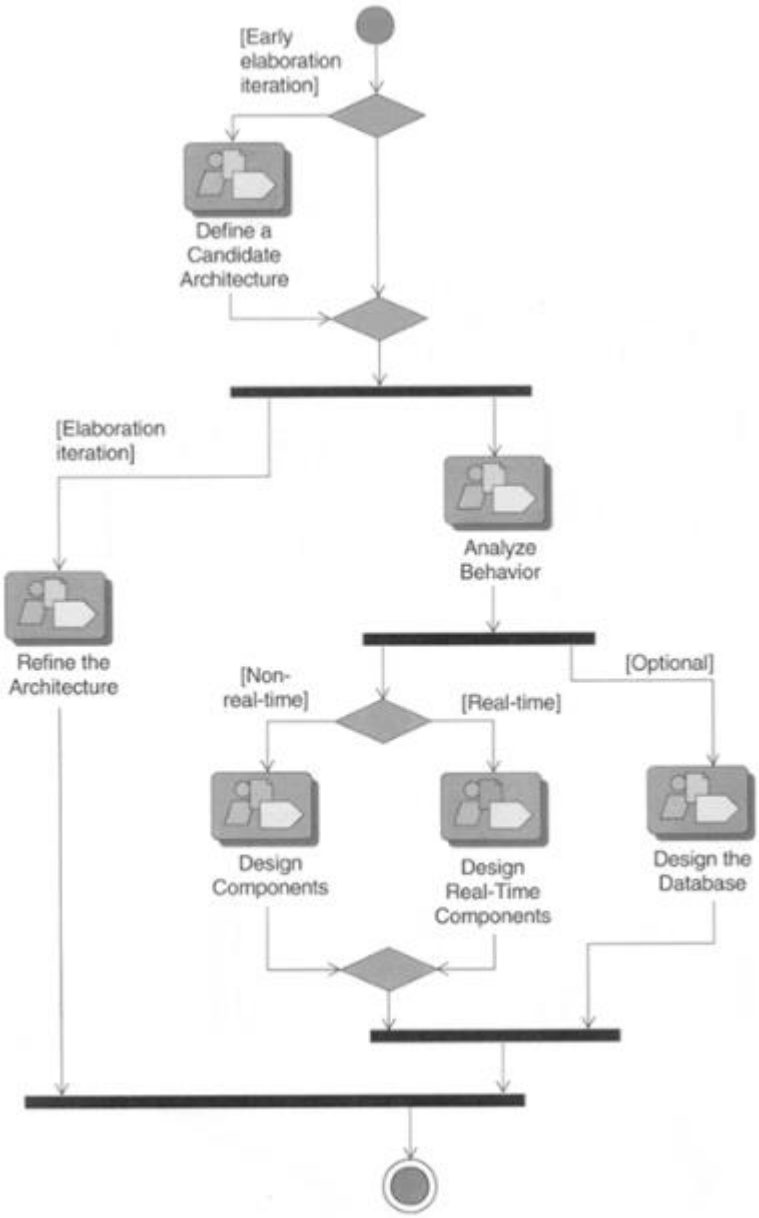
Các yêu cầu của hệ thống bao gồm yêu cầu về chức năng và yêu cầu ngoài chức năng (độ tin cậy, hiêu suất, sự hỗ trợ, ...).

4.3.3. Phân tích và thiết kế (Analysis and design)

Mô tả kiến trúc hệ thống thông qua các sơ đồ phân tích thiết kế. Mục đích của luồng công việc này là chuyển các yêu cầu sang đặc tả để mô tả cách thực cài đặt hệ thống.



Những người thực hiện và các tài liệu trong luồng công việc



Biểu đồ luồng công việc Phân tích và thiết kế

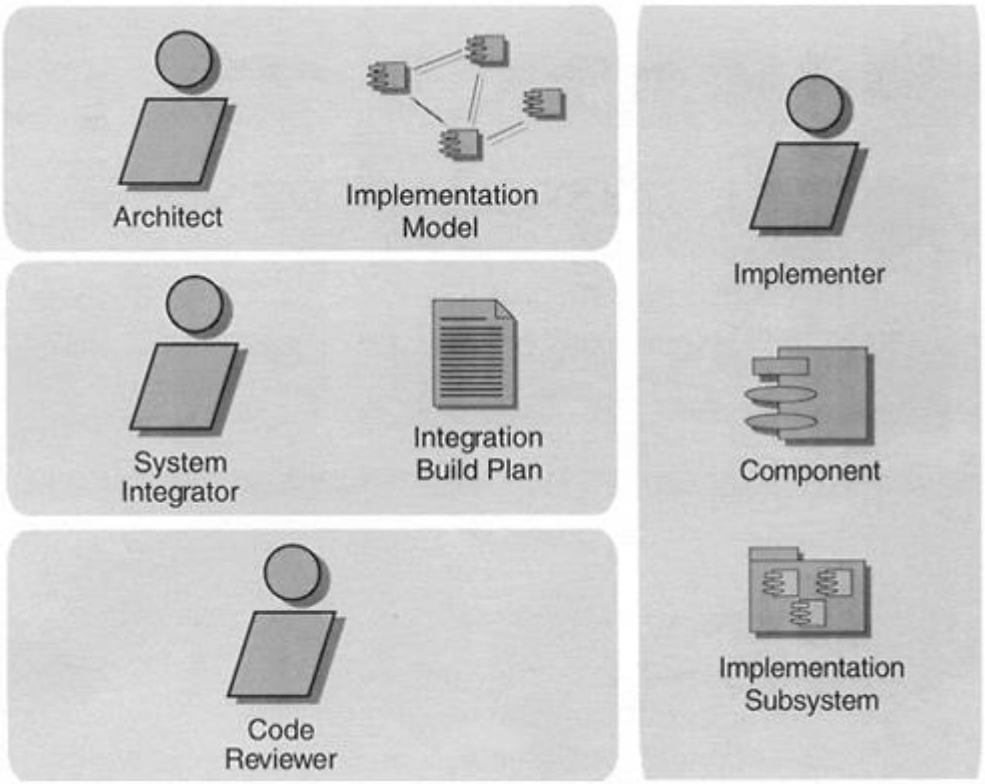
4.3.4. Cài đặt (Implementation)

Thực hiện các việc xây dựng chương trình bằng ngôn ngữ lập trình. Mục đích của luồng công việc này là:

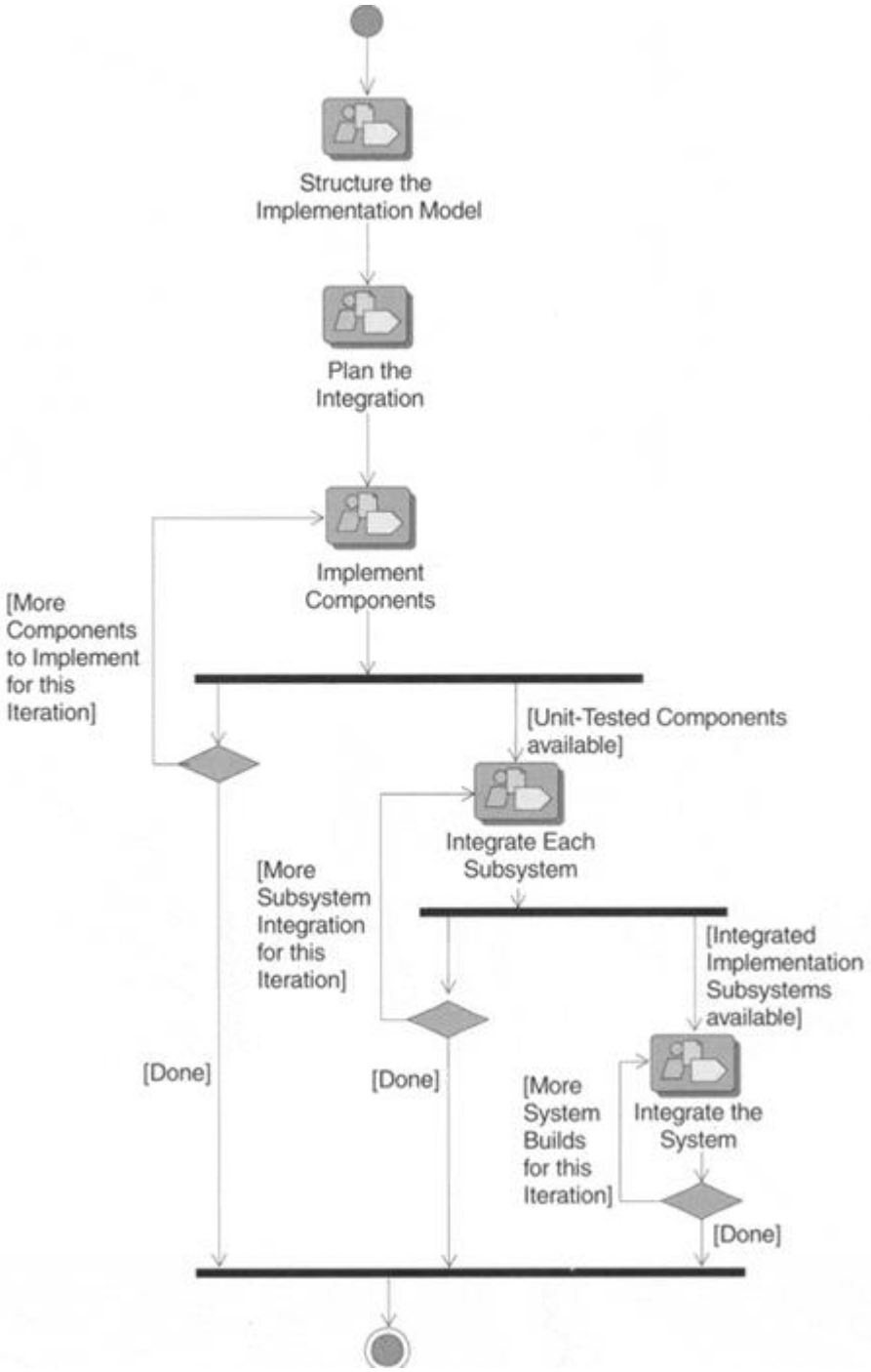
- Xác định cách thức viết mã cài đặt

- Cài đặt các lớp và đối tượng như là các thành phần

- Tích hợp vào trong một hệ thống có thể thực thi được



Những người thực hiện và các tài liệu



Biểu đồ luồng công việc Cài đặt

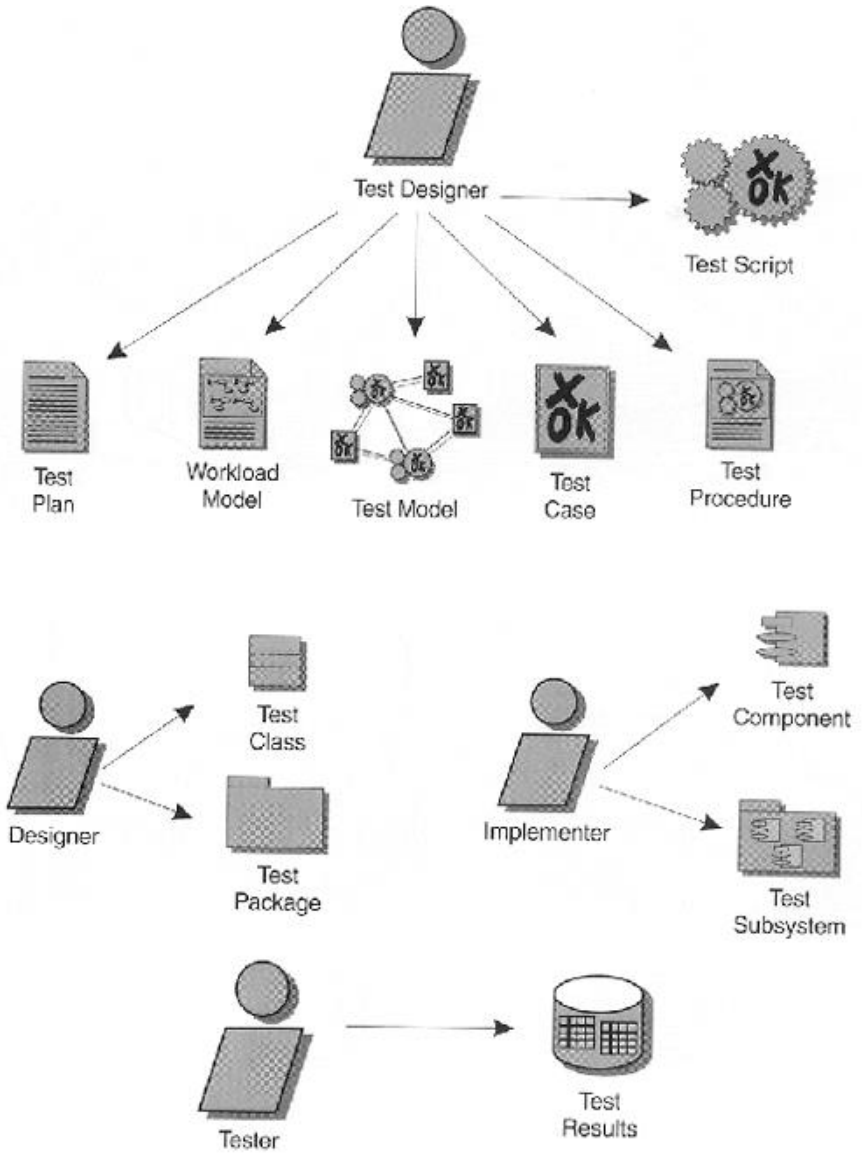
4.3.5. Kiểm thử (Test)

Mô tả các tình huống và kịch bản thử nghiệm, tiến hành thử nghiệm hệ thống phần mềm. Mục đích của kiểm thử là để đảm bảo chất lượng. Luồng công việc này liên quan đến:

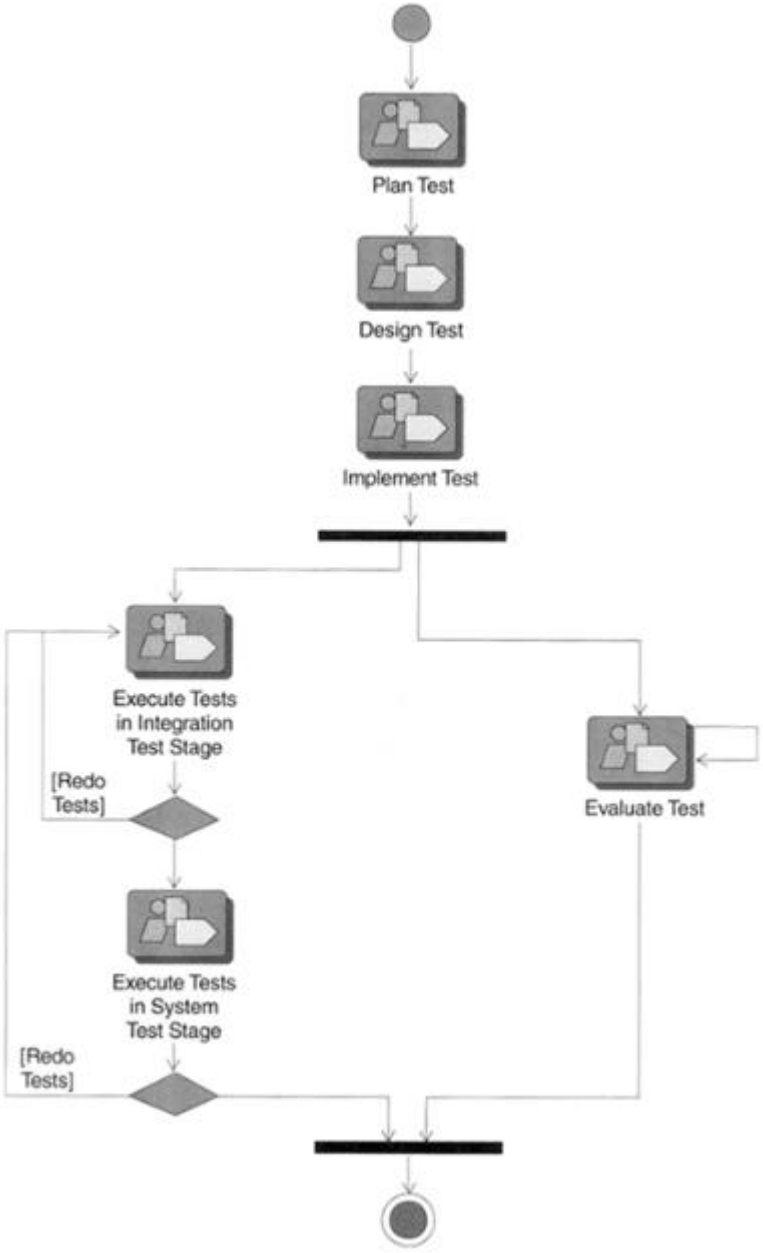
- Xét duyệt sự tương tác giữa các thành phần trong hệ thống
- Xét duyệt sự tích hợp đúng đắn các thành phần
- Xét duyệt tất cả các yêu cầu đã được cài đặt
- Đảm bảo rằng phát hiện các lỗi trước khi triển khai hệ thống

Các bước kiểm thử:

- Kiểm thử đơn vị
- Kiểm thử tích hợp
- Kiểm thử sự chấp nhận
- Kiểm thử hệ thống



Những người thực hiện và tài liệu trong luồng công việc



Biểu đồ luồng công việc Kiểm thử

4.3.6. Triển khai ứng dụng (Deployment)

Mục đích của Sự triển khai là đưa sản phẩm phần mềm đến người sử dụng. Luồng này bao gồm các hoạt động:

- Kiểm thử phần mềm trong môi trường sử dụng cuối cùng

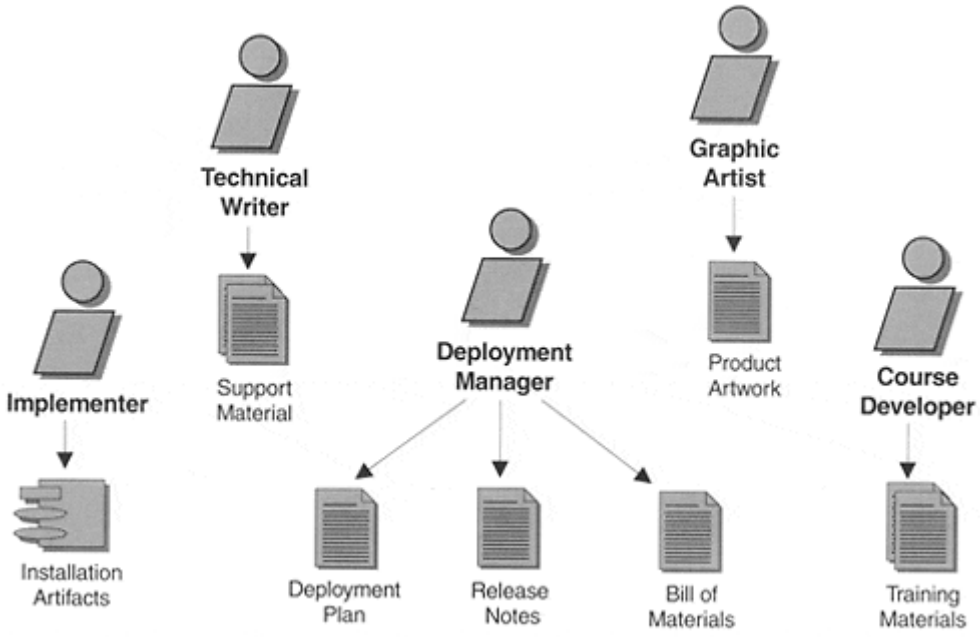
- Đóng gói phần mềm để chuyển giao

- Cài đặt phần mềm

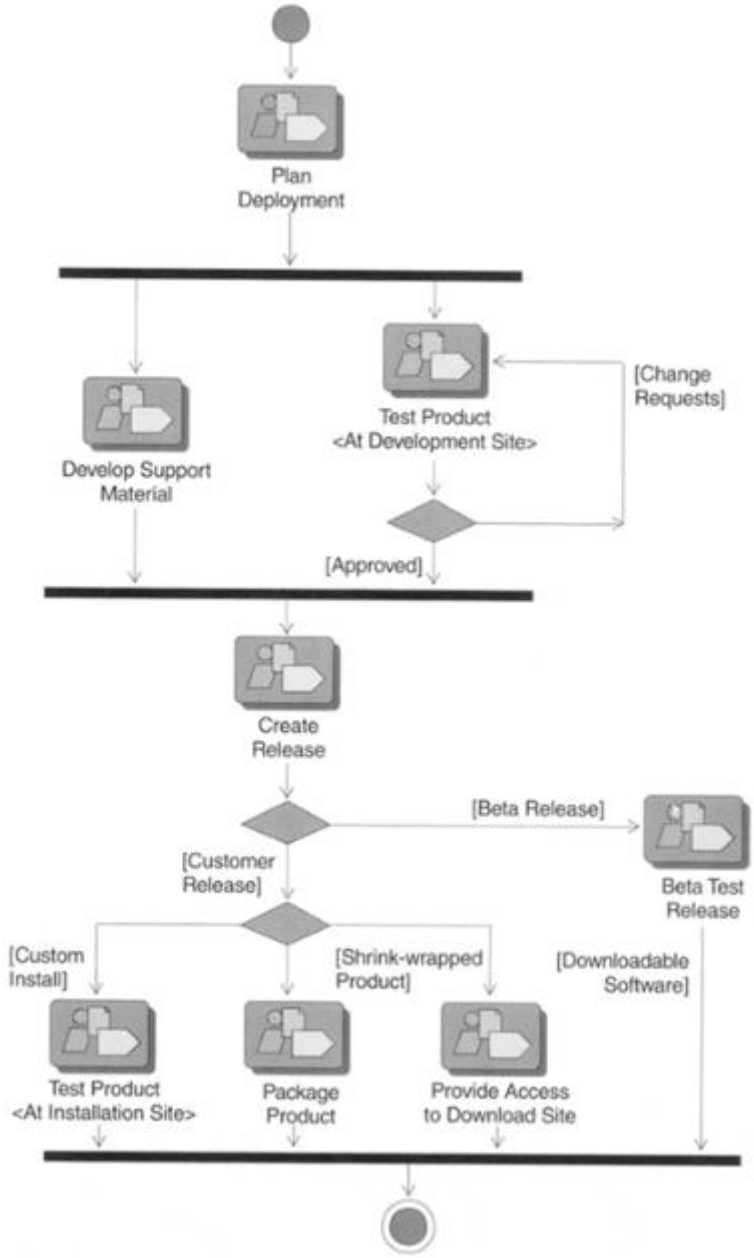
- Huấn luyện người sử dụng

Những người thực hiện: Người quản lý triển khai, Người quản lý dự án, Người viết tài liệu kỹ thuật, Người phát triển, Người cài đặt, Người kiểm thử.

Các tài liệu: Phần mềm có thể thực thi, Hướng dẫn cài đặt, Các lưu ý, Tài liệu huấn luyện sử dụng.



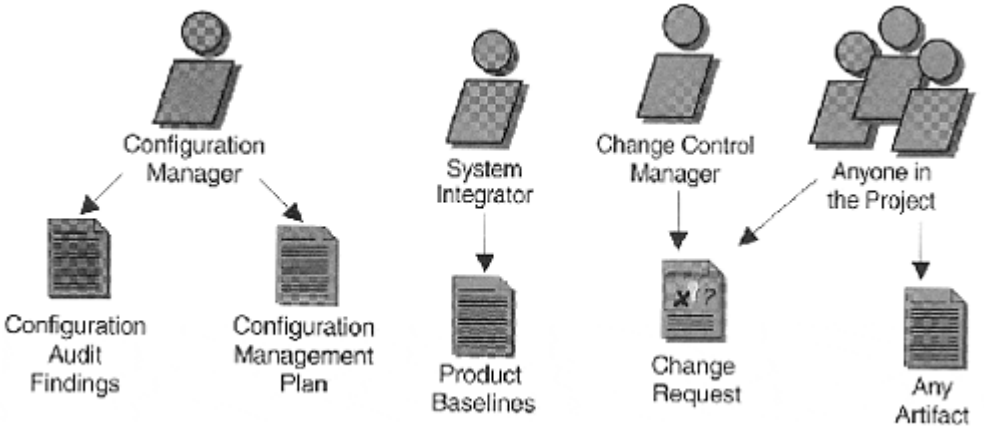
Người thực hiện và tài liệu trong luồng công việc



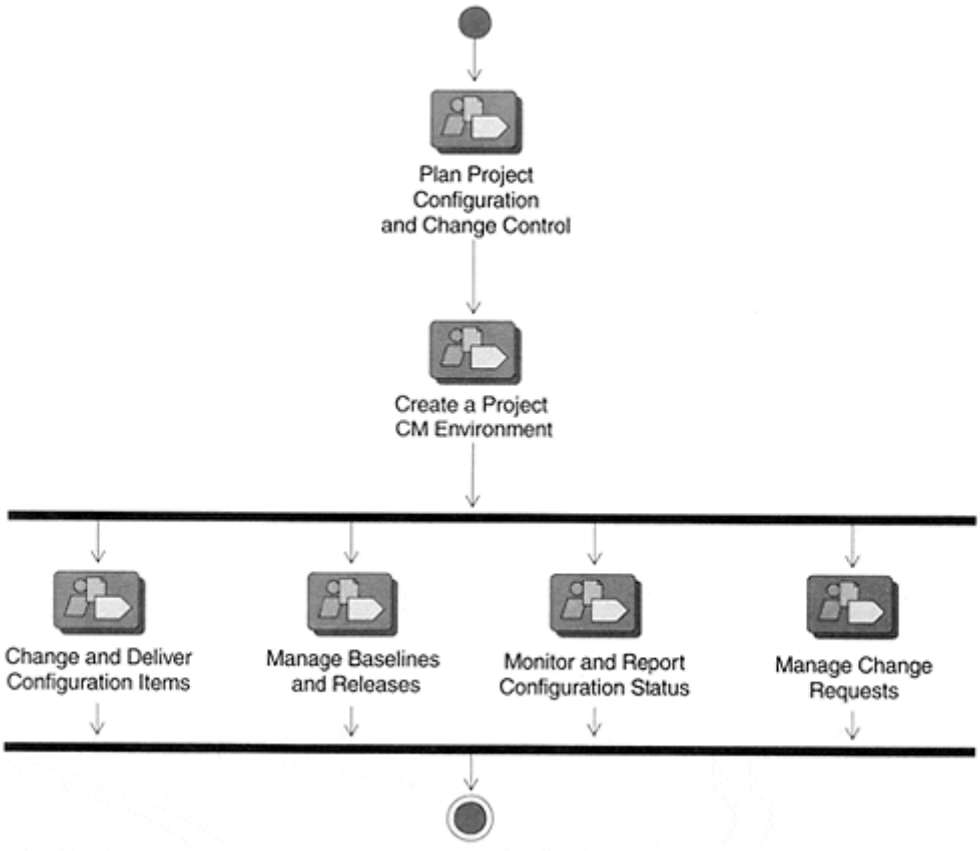
Biểu đồ luồng công việc Triển khai ứng dụng

4.3.7. Quản lý cấu hình và sự thay đổi (Change management)

Kiểm soát các thay đổi và duy trì sự hợp nhất của các thành phần hệ thống. Mục đích của luồng công việc này là theo dõi và duy trì tính toàn vẹn của hệ thống trong quá trình phát triển.



Người thực hiện và các tài liệu trong luồng công việc



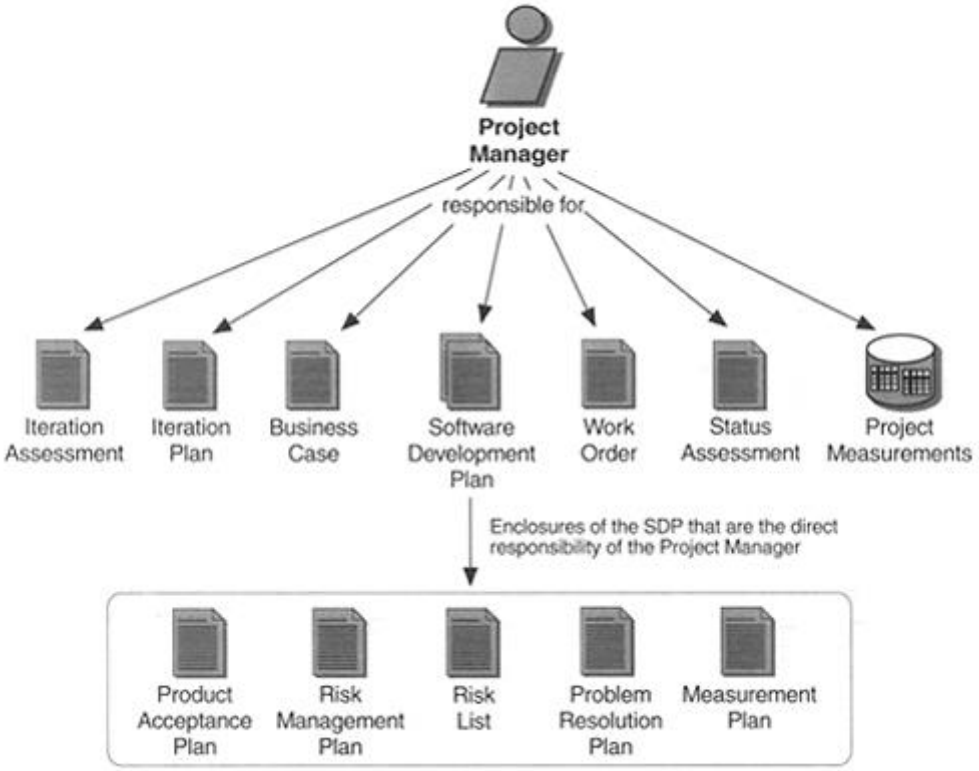
Biểu đồ luồng công việc Quản lý cấu hình và sự thay đổi

4.3.8. Quản lý dự án (Project management)

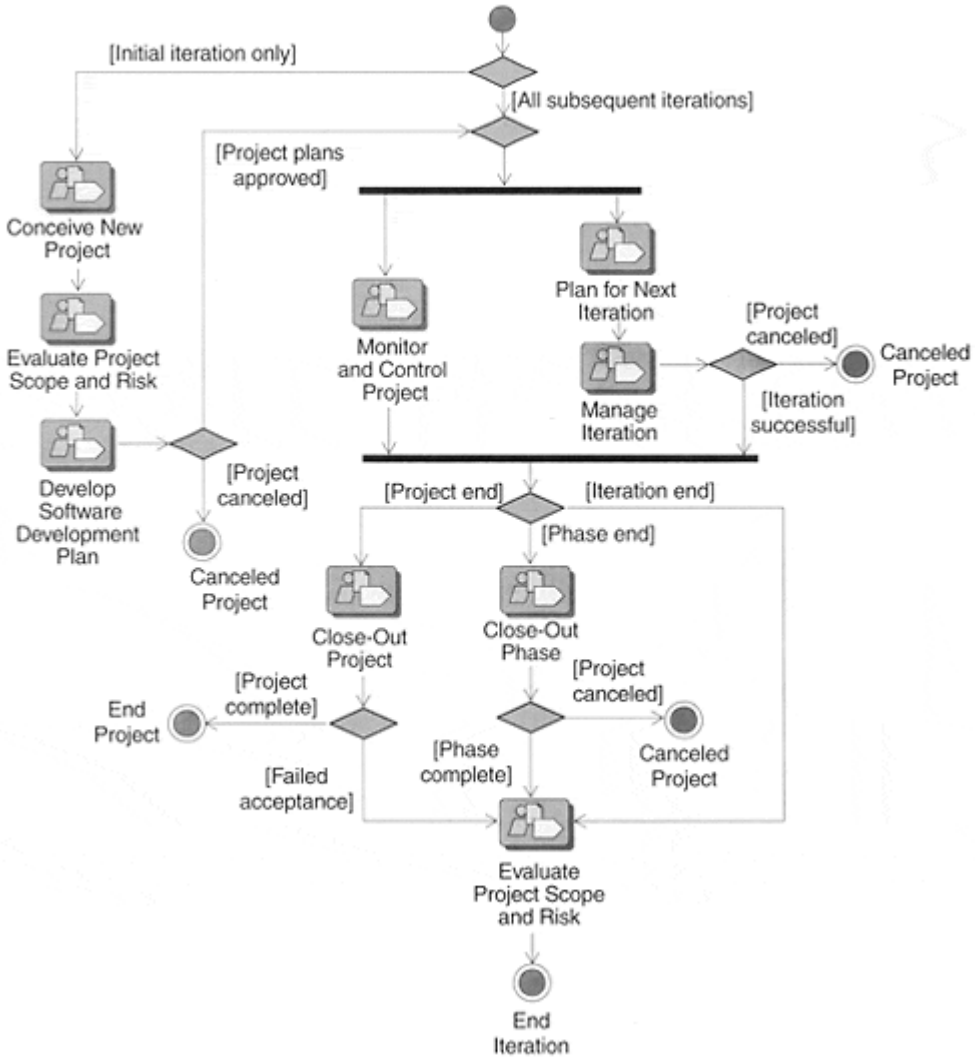
Mục đích của luồng công việc này là:

- Cung cấp một khung để quản lý dự án, quản lý rủi ro

- Cung cấp các hướng dẫn thực hành để lập kế hoạch, thực thi và theo dõi



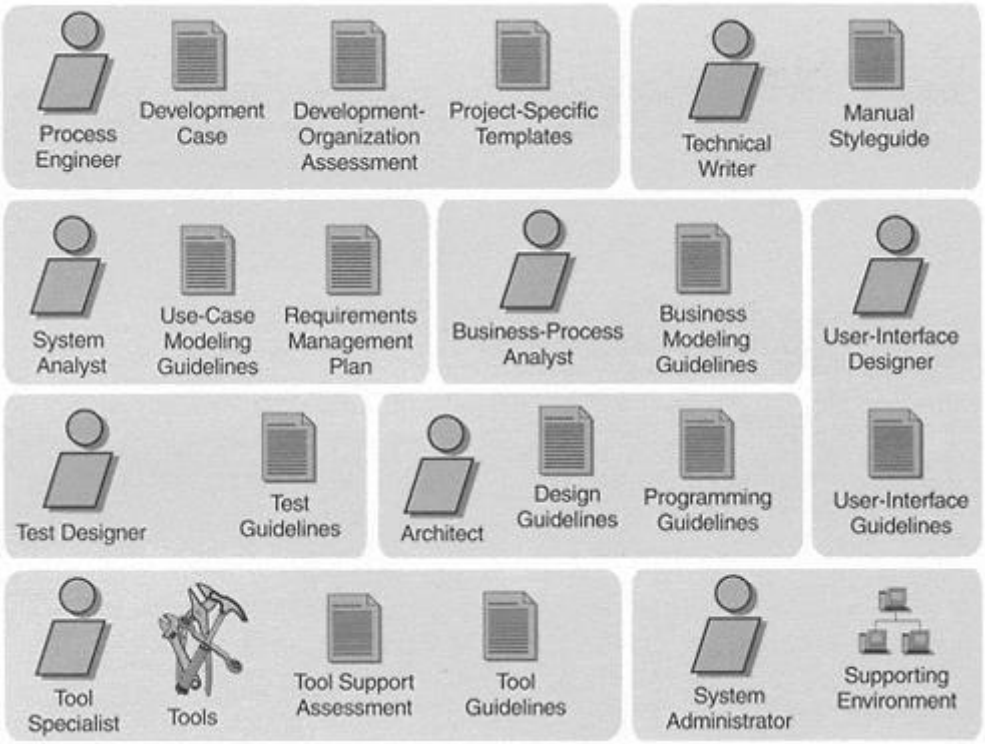
Người thực hiện và các tài liệu trong luồng công việc



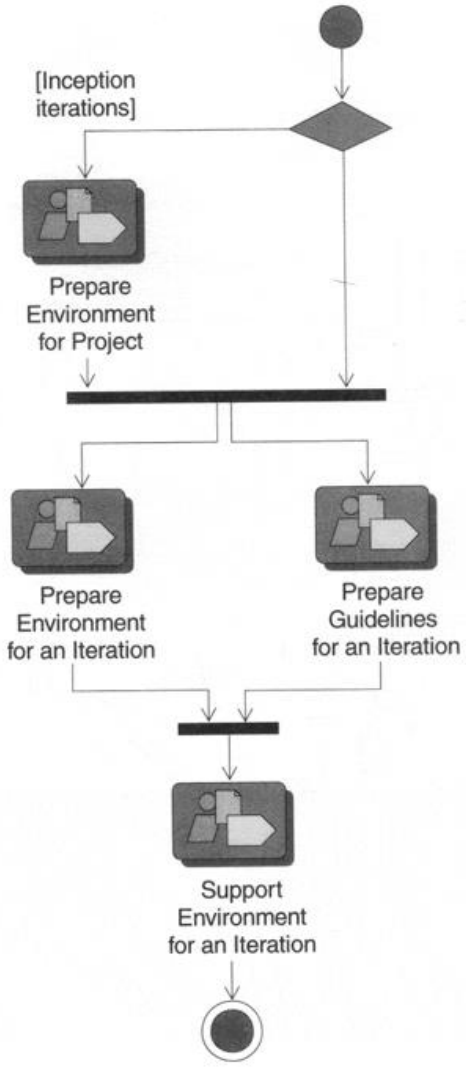
Biểu đồ luồng công việc Quản lý dự án

4.3.9. Quản lý môi trường ứng dụng (Environment)

Đảm bảo các hạ tầng cần thiết để có thể phát triển được hệ thống. Mục đích của luồng công việc này là hỗ trợ sự phát triển hệ thống bằng các quy trình và công cụ.



Người thực hiện và các tài liệu trong luồng công việc



Biểu đồ các luồng công việc Quản lý môi trường ứng dụng

Bài tập

1) RUP là gì? So sánh RUP với XP?
2) Kiến trúc của RUP
3) Các luồng công việc trong RUP;

Chương 5. Khảo sát và phân tích yêu cầu

5.1. Thu thập yêu cầu (Requirements elicitation)

5.1.1. Tổng quan

Mỗi giai đoạn phát triển hệ thống đòi hỏi sự trao đổi giữa nhà phát triển và người dùng để nhận được thông tin có ích. Mỗi giai đoạn cần tìm kiếm một dải rộng các câu hỏi về ứng dụng. Ví dụ: Khi phân tích tính khả thi, các câu hỏi tương đối rộng và tổng quát:

- Đâu là phạm vi của vấn đề?
- Cách tốt nhất để tự động hoá là gì?
- Công ty có cố gắng để phát triển ứng dụng này hay không?
- Công ty có thể hỗ trợ việc phát triển ứng dụng không?

Khi phân tích yêu cầu chúng ta tìm hiểu các thông tin có liên quan đến ứng dụng ***là gì***. Ví dụ:

- Các dữ liệu cần thiết là gì?
- Các xử lý nào được tiến hành và các thông tin chi tiết liên quan?

Khi thiết kế chúng ta phát triển thêm: ***Làm thế nào*** thông tin có liên quan tới ứng dụng:

- Làm thế nào chuyển ứng dụng vào môi trường đã chọn?
- Làm thế nào thiết kế dữ liệu logic được chuyển vào thiết kế dữ liệu vật lý?
- Các module chương trình được phối hợp với nhau như thế nào?

Các thông tin đó không xuất phát từ đâu khác ngoài chính từ yêu cầu của người dùng. Nhiệm vụ của nhà phát triển là phải nắm bắt được các thông tin trên. Có nhiều cách để thu thập dữ liệu: *Phỏng vấn - họp nhóm - quan sát - giới thiệu trước chương trình sau đó xin ý kiến - ấn định công việc tạm thời - làm việc chung - xem xét tài liệu nội bộ, tài liệu ngoài…* Mỗi phương pháp có ưu, nhược điểm riêng (chúng ta sẽ thảo luận sau). Nhà phát triển phần mềm phải biết vận dụng linh hoạt các phương pháp trên để thu được thông tin một cách hiệu quả nhất.

Các tính chất của dữ liệu.

Các dữ liệu được phân biệt theo một vài khía cạnh:

- Định hướng thời gian.
- Cấu trúc.

- Nhập nhằng.
- Ngữ nghĩa.
- Độ lớn.

Mỗi yếu tố trên đều quan trọng trong việc xác định các đặc tả của ứng dụng bởi vì chúng hướng dẫn cho công nghệ phần mềm biết số lượng và kiểu thông tin nên được chọn. Cũng vậy, các kiểu dữ liệu khác nhau có liên quan tới các loại ứng dụng khác nhau và đòi hỏi các kỹ thuật khai thác thông tin khác nhau. Không chú ý tới các đặc tính của dữ liệu sẽ dẫn tới lỗi phân tích thiết kế.

Bên cạnh việc thu thập thông tin, chúng ta cũng cần sử dụng các kỹ thuật định lượng thông tin và biên dịch và ứng dụng đề ra.

Tính chất 1: *Hướng thời gian.*

Tính hướng thời gian của dữ liệu đề cập tới *quá khứ, hiện tại* hoặc các *đòi hỏi tương lai* của ứng dụng đề ra.

Các *dữ liệu quá khứ*, ví dụ, có thể mô tả công việc đã được biến đổi thế nào qua thời gian, các quy định ảnh hưởng thế nào tới nhiệm vụ, vị trí của nó trong tổ chức và nhiệm vụ. Các thông tin quá khứ là chính xác, đầy đủ và xác đáng.

Các *thông tin hiện tại* là các thông tin và cái gì đang xảy ra. Ví dụ thông tin ứng dụng hiện tại liên quan tới quá trình hoạt động của công ty, số lượng các lệnh được thực hiện trong ngày hoặc số lượng các hang hoá được sản xuất, các chính sách, sản phẩm, đòi hỏi nghiệp vụ, yêu cầu pháp quy hiện tại hoặc các rang buộc khác cũng rất cần thiết cho việc phát triển ứng dụng. Các thông tin hiện tại nên được chuyển thành các tư liệu cho phù hợp với đội ngũ phát triển để tăng sự hiểu biết của họ về ứng dụng và phạm vi của bài toán

Các *đòi hỏi trong tương lai* liên quan đến các sự thay đổi sẽ diễn ra, chúng không chính xác và rất khó kiểm tra. Các dự đoán kinh tế, khuynh hướng tiếp thị, kinh doanh là các ví dụ.

Tính chất 2: *Tính có cấu trúc.*

Thông tin chúng ta thu thập được là những thông tin được tổ chức theo một cấu trúc (khuôn mẫu) nhất định; có như vậy mới thể hiện một ý nghĩa phản ánh một đối tượng nào đó, điều này là hiển nhiên. Tuy nhiên, trong quá trình thu thập dữ liệu, chúng ta có khi không hiểu được cấu trúc của thông tin phản ánh, mà rất có thể hiểu theo hướng khác (điều này đã được đề cập ở phần *các lỗi có thể mắc phải trong quá trình phát triển hệ thống* - Chương 2).

Cấu trúc của thông tin định hướng về phần mở rộng theo đó thông tin có thể được phân loại theo một cách nào đó. Cấu trúc có thể tham chiếu tới các hàm, môi trường hoặc dạng dữ liệu hạy

hình thức xử lý. Các thông tin thay đổi từ phi cấu trúc cho tới cấu trúc mà phần cấu trúc được xác định bởi công nghệ phần mềm (SE).

Một ví dụ thực tế khi phân tích chức năng của nghiệp vụ. Các chức năng của nghiệp vụ nếu theo người quản lý hệ thống thì không thể kể ra hết vì đó là các công việc của từng bộ phận, của từng nhân viên. Do vậy ta chỉ nắm được những cái tổng quan (có tính trừu tượng cao - không rõ ràng, cụ thể). Còn các chức năng nghiệp vụ của từng bộ phận, từng nhân viên thì rất nhỏ lẻ. Và đứng giữa một danh sách các chức năng như vậy thì khó có thể thấy được tính cấu trúc của nó. Các nhà phân tích lại phải "ngồi lại" với nhau và tổ chức lại các chức năng nghiệp vụ đó. Có như vậy thì khi xây dựng chương trình, ta tránh phải làm đi làm lại các chức năng giống nhau giữa các bộ phận trong thực tế. Mà ta chỉ cần nêu ra một liên kết (link) từ bộ phận (module) này đến bộ phận khác.

Tính "không chuẩn" của dữ liệu thể hiện rõ nhất ở thông tin trong một tờ "hoá đơn". Hoá đơn thanh toán thể hiện rất nhiều thông tin, như: *Số HD, Tên HĐ, Tên khách hàng, Địa chỉ khách hàng*, … và sau đó là một bảng liệt kê chi tiết *tên các mặt hàng, đơn giá, số lượng, thành tiền*... nhưng trong thực tế, không một bảng dữ liệu có khuôn dạng giống như một hoá đơn nào có mặt trong kho dữ liệu của hệ thống. Điều này là do liên kết dữ liệu từ các bảng khác mà thành, tránh lưu trữ trùng lặp quá nhiêu thông tin. Do vậy, các nhà thiết kế dữ liệu đã tổ chức lại cấu trúc của dữ liệu cần lưu trữ.

Tính chất 3: *Đầy đủ.*

Hơn lúc nào hết, khi tìm hiểu về một đối tượng hay lĩnh vực nào đó, ta luôn cần thông tin phản ánh về nó một cách đầy đủ và chính xác nhất có thể có. Về mặt lý thuyết thì không bao giờ ta có được toàn bộ thông tin về đối tượng hay lĩnh vực mà ta xử lý. Trong thực tế cũng như vậy, thông tin mà ta có chỉ là tạm đủ để ta có thể xử lý mà thôi.

Các thông tin có thể xếp theo cấp độ tính đầy đủ mà cao nhất là mọi thông tin cần thiết sẽ được biểu diễn. Mỗi kiểu ứng dụng đòi hỏi một mức độ đầy đủ khác nhau. Các hệ thống xử lý giao dịch luôn tiếp cận các thông tin đầy đủ và chính xác (ví dụ hệ thống bán vé máy bay). Tuy nhiên các hệ thống xây dựng theo kiến trúc hệ chuyên gia hay trí tuệ nhân tạo (AI) là minh hoạ tốt nhất việc xử lý thông tin không đầy đủ.

Tính chất 4: *Nhập nhằng.*

Tính nhập nhằng là một thuộc tính của dữ liệu không trong sáng về nghĩa hoặc có nhiều nghĩa một cách hữu ý (có chủ định). Tính chất này liên quan đến mức độ ngữ nghĩa. Ví dụ, nhìn thấy một cửa hiệu có thể đề biển “Giặt là hấp”, thì một cậu bé có thể hỏi bố một câu hỏi như sau: “Tại sao giặt lại là hấp?”, vào hoàn cảnh này, ông bố sẽ phải mất rất nhiều công sức để giải thích

cho con hiểu. Như vậy có hiện tượng "ông nói gà, bà hoá cuốc". Để giải quyết vấn đề này cần căn cứ vào ngữ cảnh.

Tính chất 5: *Ngữ nghĩa.*

Mọi người trong một tổ chức đều có một tập hợp các định nghĩa được chia sẻ cho biết các thuật ngữ, chính sách hoặc các hành động được biểu hiện như thế nào.

Ngữ nghĩa rất quan trọng với việc phát triển ứng dụng và với chính bản thân ứng dụng đó. Nếu mọi người dùng chung một thuật ngữ mà có cách hiểu khác nhau thì sẽ dẫn đến không thể trao đổi thông tin được. Đối với ứng dụng thì dữ liệu sẽ không bao giờ xử lý được cho đến khi người sử dụng hiểu được ngữ nghĩa của dữ liệu này. Các ứng dụng sẽ có ý nghĩa xác định với mục dữ liệu được định tính thông qua việc đào tạo và sử dụng lâu dài. Khi các cán bộ chủ chốt chuyển công tác, thì khả năng chuyển hoá ngữ nghĩa dễ mất. Việc đánh mất ngữ nghĩa của một công ty có thể gây tổn thất rất lớn cho công ty đó.

Tính chất 6: *Độ lớn (volume).*

Volume là số lượng các sự kiện nghiệp vụ hệ thống phải tiến hành trong một chu kỳ nào đó. Volume của tạo mới hay thay đổi khách hàng được tiến hành theo tháng hoặc năm, trong đó volume của giao dịch được tiến hành theo ngày giờ hoặc là theo peak volume (peak volume là số các giao dịch hoặc các sự kiện được thực hiện trong thời kỳ bận nhất). Thời kỳ cao điểm có thể là cuối năm hoặc cuối các quý, ví dụ chuẩn bị cho báo cáo nộp thuế. Volume của dữ liệu là một nguồn thông tin phức tạp bởi vì số lượng thời gian cần thiết với một giao dịch đơn lẻ có thể trở thành rất quan trọng đối với lượng lớn dữ liệu cần xử lý sau này.

5.1.2. Phương pháp phỏng vấn

Phỏng vấn là việc tập hợp một nhóm người số lượng ít trong một khoảng thời gian cố định với một mục đích cụ thể. Phỏng vấn thường được tiến hành với 1 hoặc 2 người hỏi đối với 1 người được phỏng vấn. Trong quá trình phỏng vấn, các câu hỏi có thể được thay đổi. Bạn có thể đánh giá được cảm nhận của họ, động cơ và thói quen với các bộ phận, quá trình quản lý hoặc các thông tin về thực thể khác đáng chú ý. Kiểu của phỏng vấn là kiểu của thông tin yêu cầu. Phỏng vấn được dẫn dắt sao cho cả 2 bên tham gia đều cảm thấy thoả mãn với kết quả của nó. Cuộc phỏng vấn được chuẩn bị kỹ đồng nghĩa với việc hiểu được về người đang được phỏng vấn. Do đó bạn không là cho họ bối rối và bạn có thể hỏi vài câu ban đầu được chuẩn bị cho dù không phải là tất cả.

Một cuộc phỏng vấn bao giờ cũng có bắt đầu, đoạn giữa và kết thúc.

- Lúc bắt đầu, bạn tự giới thiệu và đặt các câu hỏi đơn giản. Nên bắt đầu với các câu hỏi tổng quát vì không đòi hỏi các trả lời mang tính quan điểm cá nhân. Hãy chú ý đến kết

quả trả lời để tìm ra mối các câu hỏi tiếp theo và tính trung thực, thái độ của người được phỏng vấn.

- Vào giữa buổi, nên tập trung vào chủ đề. Hãy lấy mọi thông tin bạn cần lưu ý, sử dụng các kỹ thuật mà bạn đã chọn ban đầu. Nếu thấy một vài thông tin qua trọng, hãy hỏi xem bạn có thể được thảo luận sau này.
- Vào lúc kết thúc, hãy tóm tắt các thứ mà bạn đã nghe và nói những gì sẽ phỏng vấn tiếp. Bạn có thể ghi chép và đề nghị người được hỏi xem xét lại. Tốt nhất là trong thời gian 48 giờ và có sự chấp nhận của người dùng theo ngày xác định.

Phỏng vấn có thể sử dụng 2 loại câu hỏi:

- *Câu hỏi mở*: Là câu hỏi có nhiều cách trả lời khác nhau, câu hỏi mở thích hợp cho các chức năng ứng dụng hiện tại cũng như đang đề nghị và cho việc xác định cảm nhận ý kiến, và mong đợi về ứng dụng được đề ra. Một ví dụ là: "Ông có thể nói cho tôi về …", "Ông có thể mô tả làm thế nào …".
- *Câu hỏi đóng*: là câu hỏi mà chỉ trả lời "có" hoặc "không" hoặc một câu trả lời cụ thể. Các câu hỏi đóng tốt cho khai thác thông tin thực tế hoặc bắt người dùng tập trung vào phỏng vấn. Ví dụ, câu hỏi có thể là: "Bạn có dùng các báo cáo hàng tháng hay không?". Với các câu trả lời "Có" thì có thể được tiếp nối bằng câu hỏi mở: "Ông có thể giải thích …"

Các bước tiến hành phỏng vấn thành công

Tiến hành đặt cuộc hẹn phù hợp với thời gian của phỏng vấn.

Chuẩn bị tốt, tìm hiểu kỹ về người được phỏng vấn.

Đúng giờ.

Có kế hoạch mở đầu

- Giới thiệu bản thân, mục đích.
- Sử dụng câu hỏi mở để bắt đầu.
- Luôn lưu ý vào câu trả lời.
- Có kế hoạch cho nội dung chính.
- Kết hợp câu hỏi đóng và mở.
- Luôn bám sát các cách trình bày và phát triển chi tiết.

- Luôn cung cấp thông tin phản hồi, ví dụ: “Cho phép tôi trình lại điều ông vừa nói …”.
- Hạn chế ghi chép nếu thấy không tiện.
- Có kế hoạch kết thúc.
- Tóm tắt nội dung, yêu cầu hiệu chỉnh.
- Yêu cầu xác thực lại nội dung, đánh giá lại ghi chép.
- Cho biết ngày tháng họ sẽ nhận được báo cáo.
- Thống nhất ngày tháng lấy bản hiệu chỉnh.
- Xác nhận lại lịch làm việc.

Các câu hỏi có thể đưa ra theo kiểu có cấu trúc hay phi cấu trúc.

- *Phỏng vấn có cấu trúc* là phỏng vấn trong đó người được phỏng vấn đã có danh sách các mục cần duyệt qua, các câu hỏi xác định và các thông tin cần tìm hiểu đã được xác định trước.
- *Phỏng vấn không cấu trúc* là phỏng vấn được định hướng bởi câu trả lời. Các câu hỏi phần lớn là câu hỏi mở, không có một kế hoạch ban đầu. Do vậy người đi phỏng vấn biết các thông tin cần thiết sẽ dùng từ các câu hỏi mở để phát triển chi tiết hơn về chủ đề.

Phỏng vấn có cấu trúc thích hợp khi bạn biết về các thông tin cần thiết trước khi phỏng vấn. Ngược lại, phỏng vấn phi cấu trúc thích hợp khi bạn không thể đoán trước được chủ đề, hay chưa có thông tin gì về người được phỏng vấn. Các trường hợp điển hình của phỏng vấn là người khách hàng bắt đầu với phỏng vấn phi cấu trúc để cho hai bên nhận thức được về miền của bài toán (hiểu sơ lược vấn đề). Sau đó, phỏng vấn dần dần trở thành có cấu trúc và tập trung vào các thông tin bạn cần để hoàn chỉnh phần phân tích.

Các kết quả phỏng vấn người sử dụng lên được trao đổi lại với người được phỏng vấn trong một thời gian ngắn. Người được phỏng vấn phải được báo trước về thời hạn đối với việc phỏng vấn. Tuy nhiên, có thể xin bố trí bổ sung phỏng vấn trong trường hợp còn nhiều điều cần hỏi hoặc nhiều người cần gặp.

Bảng sau so sánh phỏng vấn có cấu trúc và phỏng vấn phi cấu trúc.

	Phỏng vấn có cấu trúc	**Phỏng vấn phi cấu trúc**
	Dùng dạng chuẩn cho nhiều câu hỏi	Có khả năng mềm dẻo nhất Cần chăm chú nghe và có kỹ năng mở rộng

Ưu điểm	Dễ quản lý và đánh giá Đánh giá được nhiều mục đích. Không cần đào tạo nhiều. Có kết quả trong các phỏng vấn.	câu hỏi. Có thể bao được những thông tin chưa biết Đòi hỏi có thực hành.
Nhược điểm	Chi phí chuẩn bị lớn. Tính có cấu trúc có thể không thích hợp cho mọi tình huống. Giảm tính chủ động của người đi phỏng vấn.	Lãng phí thời gian phỏng vấn. Người được phỏng vấn có thể định kiến với các câu hỏi. Tốn thời gian lựa chọn và phân tích thông tin.

Một kỹ năng tốt là phát triển các sơ đồ như là một phần của tài liệu phỏng vấn. Khi bắt đầu một cuộc phỏng vấn mới, nên bàn bạc về các sơ đồ và đưa cho họ bản ghi chép để họ có thể kiểm tra sau này. Bạn sẽ nhận được ngay ý kiến phản hồi về tính chính xác của sơ đồ và hiểu biết của bạn về ứng dụng. Lợi ích của cách tiếp cận này thể hiện cả mặt kỹ năng và tâm lý. Từ khía cạnh kỹ thuật, bạn thường xuyên được kiểm tra lại các vấn đề mà bạn được nghe. Cho tới khi thời gian phân tích kết thúc, cả bạn và khách hàng đều tin chắc rằng quá trình xử lý ứng dụng là đầy đủ. Từ khía cạnh tâm lý, bạn làm tăng niền tin của khách hàng vào khả năng phân tích bằng cách trình bày các hiểu biết của mình. Mỗi khi bạn cải thiện sơ đồ và đi vào phân tích, bạn cũng tăng được niềm tin của người sử dụng rằng bạn có thể xây dựng được ứng dụng đáp ứng được nhu cầu của họ.

Phỏng vấn thích hợp cho việc nhận thông tin đảm bảo cả số lượng lẫn chất lượng:

Các kiểu thông tin định tính là: các ý kiến, niềm tin, thói quen, chính sách và mô tả.

Các kiểu thông tin định lượng bao gồm: tần suất, số lượng, định lượng các mục được dùng trong ứng dụng.

Phỏng vấn là một dạng khác của thu thập dữ liệu có thể làm bạn lạc lối, thiếu chính xác hoặc thông tin không thích hợp. Bạn cần học cách đọc ngôn ngữ bằng cử chỉ, thói quen để quyết đinh được các điều kiện cần thiết cho cùng một thông tin.

Trong khi phỏng vấn, chúng ta cần chú ý đến hành động của người được phỏng vấn để có cách ứng xử thích hợp. Bảng sau liệt kê một vài tình huống và kinh nghiệm xử lý.

Hành vi của người được phỏng vấn.	**Đáp ứng của người đi phỏng vấn.**
Đoán các câu trả lời chứ không thừa nhận là không biết	Sau phỏng vấn, kiểm tra chéo các câu trả lời.
Cố nói những điều lọt tai người đi phỏng vấn, sai sự thật.	Tránh các câu hỏi dễ đoán được câu trả lời, kiểm tra chéo các câu hỏi
Cho thông tin không đầy đủ	Kiên trì hỏi để đạt mục đích.
Dừng trình bày khi người đi phỏng vấn ghi chép	Ghi nhanh nhất có thể, chỉ hỏi các câu quan trọng
Vội vã hay trả lời rời rạc, uể oải	Nhanh chóng kết thúc, đề nghị bố trí buổi khác
Thể hiện sự không quan tâm, trả lời đứt quãng	Nói chuyện vui sau đó chuyển đề tài khác
Không muốn thay đổi môi trường hiện tại	Động viên cải thiện môi trường hiện tại và so sánh 2 khuynh hướng.
Không hợp tác, từ chối trả lời	Lấy nguồn tin khác và hỏi: "Ông có quan tâm về những điều người khác nói về ông hay không?". Nếu câu trả lời là "Không" thì thôi phỏng vấn.
Phàn nàn về vị trí công tác, lương, …	Tìm ra mấu chốt vấn đề. Cố gắng dẫn dắt về chủ đề chính, ví dụ: "Dường như cơ quan ông có rất nhiều vấn đề, có thể ứng dụng mới mà chúng tôi đề xuất sẽ giải quyết được các vấn đề trên".
Là người thích thú về công nghệ	Chọn lọc các thông tin cần thiết, không để bị lôi cuốn vào các vấn đề công nghệ.

Phỏng vấn và gặp gỡ phù hợp với mọi loại kiểu dữ liệu do đó chúng thường xuyên được sử dụng.

Ưu điểm của phỏng vấn:

- Nhận được cả thông tin chất lượng và số lượng.
- Nhận được cả thông tin đầy đủ và chi tiết.

- Là phương pháp tốt cho các yêu cầu bên ngoài.

Nhược điểm của phỏng vấn:

- Đòi hỏi có kỹ năng giao tiếp.
- Có thể có kết quả thiên vị vì mang tính chủ quan của người được phỏng vấn.
- Có thể dẫn đến các thông tin sai lạc, không liên quan, thiếu chính xác.
- Đòi hỏi phải có 3 người để kiểm tra kết quả.
- Không thích hợp với số lượng lớn người.

5.1.3. Phương pháp quan sát

Quan sát có thể tiến hành thủ công hoặc tự động.

- *Theo cách thủ công*, người quan sát ngồi tại chỗ và ghi chép lại các hoạt động, các bước xử lý công việc. Các băng video đôi khi có thể được dùng. Ghi chép hoặc băng ghi hình được phân tích cho các sự kiện, các mô tả động từ chính, hoặc các hoạt động chỉ rõ lý do, công việc, hoặc các thông tin về công việc.
- *Theo cách tự động*, máy tính sẽ lưu trữ chương trình thường trú, lưu lại vết của các chương trình được sử dụng, email và các hoạt động khác được xử lý bởi máy. Các file nhật ký của máy sẽ được phân tích để mô tả công việc.

Ưu điểm của quan sát:

- Bao trùm được các tiêu chuẩn quyết định, quy trình suy luận, các thủ tục khớp nối (mang tính thực hành).
- Kỹ sư phần mềm sẽ không bị định kiến (không bị ảnh hưởng bởi người khác) mà hoàn toàn tập trung vào vấn đề của mình.
- Quan sát sẽ khắc phục ngăn cách giữa kỹ sư phần mềm và người được phỏng vấn.
- Nhận được các hiểu biết tốt về môi trường công tác hiện tại, vấn đề và quá trình xử lý thông qua quan sát.

Nhược điểm của quan sát:

- Thời gian quan sát có thể không biểu diễn cho các công việc diễn ra thông thường.
- Thói quen dễ thay đổi do biết mình bị quan sát (người bị quan sát sẽ mất tự nhiên, hành động có thể bị ghò ép).
- Mất nhiều thời gian.

Người đi quan sát nên xác định cái gì sẽ được quan sát. Nên xác định thời gian cần thiết cho việc quan sát, hãy xin sự chấp thuận của cả người quản lý và cá nhân trước khi tiến hành quan sát.

Ấn định công việc tạm thời.

Không có gì thay thế được kinh nghiệm. Với một công việc tạm thời, bạn có được nhận thức đầy đủ hơn về các nhiệm vụ. Cũng vậy, đầu tiên bạn học các thuật ngữ hoàn cảnh sử dụng nó. Thời gian kéo dài từ 2 tuần đến 1 tháng đủ dài để bạn có thể quen với phần lớn các công việc thông thường và các tình huống ngoại lệ nhưng không được quá dài để trở thành chuyên gia thực sự đối với công việc.

Công việc tạm thời cho bạn cơ sở hình thức hoá các câu hỏi về chức năng nào của phương pháp hiện thời của công việc sẽ được giữ lại và cái nào sẽ bị loại trừ hoặc thay đổi, nghiên cứu được ngữ cảnh hiện tại. Có thể bằng công việc để thay thế cho các câu hỏi không thực hiện được. Bất lợi của công việc tạm thời là tốn thời gian và sự lựa chọn về thời gian có thể làm tối thiểu hoá vấn đề, không bao hết được các hoạt động hoặc thời gian. Một nhược điểm khác nữa là kỹ sư phần mềm có thể thiên kiến hoá về quá trình xử lý công việc (do tự mình đã làm), nội dung làm ảnh hưởng đến công việc thiết kế sau này.

5.1.4. Phương pháp họp nhóm

Meeting là việc tập trung từ 3 người trở lên trong một khoảng thời gian để thảo luận về một chủ đề nhất định. Meeting có thể vừa bổ sung vừa thay thế phỏng vấn bằng cách cho phép các thành viên kiểm tra lại các kết quả phỏng vấn cá nhân. Nó có thể thay thế phỏng vấn bằng cách cung cấp một diễn đàn cho các thành viên cùng tìm ra các yêu cầu và các giải pháp cho ứng dụng.

Meeting có thể làm lãng phí thời gian. Nói chung nếu meeting càng lớn thì càng ít ý kiến nhất trí và thời gian để đi đến quyết định sẽ kéo dài. Do vậy lên có kế hoạch ban đầu cho meeting. Lịch trình nên cung cấp trước cho các thành viên. Số lượng chủ đề cần thảo luận chỉ nên thấp hơn 5 chủ đề. Meeting nên có thời gian cố định và có địa điểm thống nhất cụ thể với các quyết định cần thiết. Meeting không nên kéo dài quá 2 giờ để có thể đảm bảo được sự tập trung, chú ý của các thành viên.

Ưu điểm của họp nhóm:

- Có thể ra quyết định mà các thành viên đều phải tuân theo (đa số).
- Nhận được cả thông tin tổng hợp và chi tiết.
- Là phương pháp tốt cho các yêu cầu bên ngoài.
- Tập hợp được nhiều người dùng liên quan.

Nhược điểm của họp nhóm:

- Mất nhiều công sức thời gian và tiền bạc để chuẩn bị.
- Nếu số đại biểu nhiều sẽ tốn thời gian để ra được quyết định.
- Các ngắt quãng trong cuộc họp dễ làm mọi người phân tán.
- Dễ chuyển sang các chủ đề ít liên quan như: chính trị, thể thao, thời trang …
- Mời không đúng thành viên dẫn đến chậm có kết quả.

5.1.5. Phương pháp điều tra qua bản câu hỏi

Được ứng dụng khi cần lấy ý kiến của đại đa số người dùng về một số thông tin để có thể tập hợp số liệu thống kê mà không có điều kiện gặp trực tiếp. Với cách này, người thu thập dữ liệu sẽ soạn trước một bản câu hỏi, có thể có sẵn các phương án lựa chọn để người dùng lựa chọn đánh dấu vào, sau đó thu lại và thống kê kết quả.

Ví dụ, các câu hỏi có thể như sau:

Bạn thường ứng dụng máy tính vào các lĩnh vực nào sau đây?

A. Giải trí B. Công việc C. Do ý thích D. Không dùng

Với cách thức này, người thu thập không cần mất thời gian gặp trực tiếp (như phỏng vấn hoặc họp nhóm) mà vẫn thu được thông tin, không đòi hỏi kỹ năng giao tiếp. Các câu hỏi trong danh sách có thể là dạng phỏng vấn trên giấy hoặc máy tính. Ưu điểm chính của câu hỏi là nếu như không cần phải chỉ rõ tên của người trả lời thì thông tin các câu trả lời sẽ có tính trung thực cao hơn. Cũng vậy, các câu hỏi chuẩn xác cung cấp các dữ liệu thực mà theo đó các quyết định có thể được dựa vào. Các mục câu hỏi, như là phỏng vấn có thể là câu hỏi mở hoặc đóng.

Ưu điểm của bản câu hỏi:

- Người cho ý kiến có thể không cần biết tên do vậy cho quan điểm và cảm nhận có tính trung thực cao, có thể dựa vào đó để ra quyết định.
- Có thể tiến hành với nhiều người.
- Thích hợp với các câu hỏi đóng và hữu hạn.
- Phù hợp với công ty đa chức năng và có thể tuỳ biến theo địa phương.

Nhược điểm của bản câu hỏi:

- Khó thực hiện lại được.
- Các câu hỏi không được trả lời không có nghĩa là không có thông tin.
- Các câu hỏi có thể khó hiểu do yêu cầu cần phải ngắn gọn.

- Thực hiện đánh giá có thể chậm.
- Người dùng ít có khả năng đưa ra ý kiến khác (do tính đóng của các câu hỏi).
- Không thể bổ xung thêm thông tin khi đã tiến hành công bố các bản câu hỏi.

5.1.6. Phương pháp thu thập tài liệu

Xem xét tài liệu

Khái niệm tài liệu ám chỉ các cẩm nang, quy định, các thao tác chuẩn mà tổ chức cung cấp như là hướng dẫn cho các nhà quản lý và nhân viên.

Các tài liệu không phải luôn nằm trong đơn vị đó. Tài liệu có thể là tài liệu nội bộ, có thể là các ấn phẩm kỹ thuật, các báo cáo nghiên cứu, … Các tài liệu thực sự có ý nghĩa với kỹ sư phần mềm để tìm hiểu các lĩnh vực mà họ chưa từng có kinh nghiệm. Nó hữu ích cho việc xác định các câu hỏi về quá trình thao tác và sản xuất. Tài liệu đưa ra các thông tin mang tính khách quan.

Tài liệu nội bộ mô tả được ngữ cảnh hiện thời; phù hợp với việc nghiên cứu có tính lịch sử (quá trình hoạt động lâu dài). Tuy nhiên việc phải cung cấp tài liệu nội bộ làm cho người dùng e ngại, gây thành kiến; khó có thể nhận biết được quan điểm, động cơ tiến hành công việc.

Tài liệu ngoài cho ta xác định được các khuynh hướng công nghiệp, ý kiến các chuyên gia, các kinh nghiệm của các công ty khác về thông tin, kỹ thuật. Tuy nhiên thông tin có thể không xác đáng, thiếu chính xác và có thể gây thành kiến.

Xem xét phần mềm

Một cách thường xuyên, các ứng dụng phải thay thế các phần mềm cũ. Hệ thống hiện tại có thể đã có phần mềm hỗ trợ từ trước. Nghiên cứu các phần mềm đã tồn tại cung cấp cho chúng ta các thông tin về quá trình xử lý công việc hiện thời và các mở rộng có ràng buộc bởi thiết kế phần mềm.

Khiếm khuyết của việc thu nhận thông tin từ việc xem xét phần mềm là tài liệu có thể không chính xác hoặc kịp thời, mà có thể không đọc được và thời gian có thể lãng phí nếu ứng dụng đã bị xoá bỏ.

Kết luận

Thu thập dữ liệu là bước khởi đầu vô cùng quan trọng trong quá trình phát triển phần mềm cho hệ thống. Những thông tin thu thập được sẽ là căn cứ để xây dựng phần mềm và là bằng chứng xác thực các yêu cầu của người dùng có được đề cập và có được đáp ứng hay không? Thu thập dữ liệu có thể được tiến hành trong mọi giai đoạn của quá trình phát triển ứng dụng nhưng có

các mục đích khác nhau. Các đặc tính cần lưu ý của dữ liệu cần thu thập là: *tính hướng thời gian*; *tính có cấu trúc*; *tính đầy đủ*; *tính không nhầm lẫn*; *ngữ nghĩa* và *độ lớn*.

Thu thập dữ liệu có thể theo nhiều kỹ năng: *phỏng vấn*; *điều tra qua bản câu hỏi*; *quan sát*; *hội họp*; *làm việc chung*; *ấn định công việc tạm thời*; *xem xét tài liệu* và *xem xét phần mềm hiện tại*. Mỗi kỹ năng có ưu điểm và nhược điểm riêng. Tuy nhiên ưu điểm của kỹ năng này có thể khắc phục nhược điểm của kỹ năng kia (ví dụ: các thông tin không thể hỏi được hoặc diễn đạt không rõ khi phỏng vấn thì có thể thìm được trong quá trình làm việc chung). Tuỳ từng điều kiện hoàn cảnh cụ thể mà người đi thu thập tài liệu có thể áp dụng kỹ năng cho phù hợp. Mục đích chính vẫn là thu thập được nhiều thông tin có tính chân thực cao làm căn cứ cho các công việc sau này.

5.2. Phân tích yêu cầu (Requirements analysis)

Phân tích yêu cầu là công việc bao gồm các tác vụ xác định các yêu cầu cho một hệ thống mới hoặc được thay đổi, dựa trên cơ sở là các yêu cầu (có thể mâu thuẫn) mà những người có vai trò quan trọng đối với hệ thống, chẳng hạn người sử dụng, đưa ra. Việc phân tích yêu cầu có ý nghĩa quan trọng đối với thành công của một dự án.

Việc phân tích yêu cầu một cách có hệ thống còn được gọi là kỹ nghệ yêu cầu (requirements engineering). Thuật ngữ "phân tích yêu cầu" còn được áp dụng cụ thể cho công việc thuần túy phân tích (thay vì các việc khác chẳng hạn như làm rõ yêu cầu hay viết tài liệu yêu cầu).

5.2.1. Phân tích phạm vi dự án

Người phân tích hệ thống dùng thật ngữ phạm vi để chỉ trách nhiệm dự án phải thực thi. Ngược lại, phạm vi dự án là nhiệm vụ lớn và phức tạp được thực hiện bởi chương trình.

Để xác định phạm vi dự án, bằng cách xác định quá trình nghiệp vụ ứng dụng sẽ đối đầu. Đó là phạm vi vấn đề của ứng dụng. Nói chung, có hai phần đối với bất kỳ giải pháp nghiệp vụ: phần triển khai ứng dụng và phần thực hiện bởi con người hay chương trình. Định ra ranh giới ứng dụng tức là xác định quy trình trách nhiệm.

- Chia trách nhiệm thành những nhiệm vụ con để đưa ra ý tưởng cho chính mình về bao nhiêu module chương trình khác nhau yêu cầu
- Xác định bao nhiêu vùng địa lý liên quan (chí nhánh văn phòng).
- Ước lượng số người dùng ứng dụng và thời gian ứng dụng được duy trì.
- Tính chính xác.
- Cuối cùng, hiểu khách hàng mong đợi dự án được triển khai.

Tại thời điểm này, chúng ta có ý tưởng phạm vi dự án. Cân nhắn độ lớn của dự án đối với thời gian và ràng buộc ngân sách. Nếu dự án quá lớn về thời gian và tiền bạc cho chi trả thì bàn bạc với khách hàng để đưa ra quyết định thương lượng cho thoả đáng. Có thể cân nhắc chọn lựa nhiều thời gian hơn, hoặc nhiều tiền bạc hơn, hoặc cả hai.

5.2.2. Phân tích mở rộng yêu cầu nghiệp vụ

a) Xác định yêu cầu nghiệp vụ:

Mỗi một dự án có một hay nhiều yêu cầu nhiệp vụ. Mỗi yêu cầu nghiệp vụ là một mô tả tác nhiệm cụ thể trong nghiệp vụ của khách hàng. Ví dụ, lưu vết quá trình đầu tư. Một tác vụ như kiểm soát đầu tư cần chia nhỏ thành những phần chắc chắn cho đến khi mỗi phần đủ để mô tả công việc chính xác.

Khi mức độ của thành phần chia nhỏ dưới mức tối thiểu, xác định lại trình tự thành phần.

Mỗi tác vụ được gọi là yêu cầu nghiệp vụ hay quy tắc nghiệp vụ. Quy tắc doanh nghiệp được viết theo ngôn ngữ được hiểu bởi những người không chuyên máy tính sao cho người dùng có thể kiểm tra luật một cách chính xác.

b) Xác định yêu cầu chất lượng khách hàng:

Mỗi dự án có thể yêu cầu nhanh, bảo mật, phụ thuộc, dễ dùng, ha bug-free. Trong thế giới thực, thời gian và ràng buộc tài chính làm cho không thể tạo ra những chương trình dự án hoàn chỉnh. Thay vào đó, điều quan trọng để quyết định dựa trên mức độ chấp nhận của chất lượng thoả mãn khách hàng.

c) Phân tích hạ tầng cơ sở hiện hành:

Phần quan trọng trong thiết kế giải pháp là phân tích kỹ thuật thay thế. Điển hình, giải pháp phần mềm được đưa vào hơn là thay thế hệ thống hiện hành. Dự án cần làm việc trên phần cứng, phần mềm, loại mạng mà người dùng hiện có hay cần phải nâng cấp, thay thế.

Khi phân tích giải pháp, nhớ rằng cơ sở hạ tầng phải đảm bảo giải pháp của chúng ta có thể tương thích.

d) Phân tích ảnh hưởng kỹ thuật:

Nếu cần mở rộng chức năng cho hệ thống hiện hành, chúng ta mong ước thay đổi hệ thống cũ cả việc cải thiện nó và tích hợp dễ dàng hơn hệ thống mới. Ví dụ, để chuyển đổi toàn bộ dữ liệu từ CSDL Access sang Microsoft SQL Server, cần hiểu sự khác biệt về giao tác, bảo mật, và những chức năng khác giữa kỹ thuật cũ và giải pháp mới.

5.2.3. Phân tích yêu cầu bảo mật

Yếu tố bảo mật dữ liệu là một trong những yêu cầu chất lượng hàng đầu đối với khách hàng. Khi phân tích yêu cầu này cần:

a) *Xác định vai trò:*

Toàn bộ ứng dụng không chỉ có một mức độ bảo mật. Người dùng cuối chỉ cần quyền truy xuất giới hạn vào hệ thống. Quản trị hệ thống, người thao thác viên cập nhật, và người dùng có quyền truy cập cao hơn ở mọi cấp độ. Bảo mật dựa trên vai trò là kỹ thuật dùng để cấp quyền mức độ bảo mật khác nhau tương ứng quyền hạn và độ chuyên nghiệp của mỗi người dùng trong hệ thống.

b) *Xác định môi trường bảo mật ứng dụng*

c) *Xác định ảnh hưởng bảo mật*

d) *Kế hoạch vận hành*

e) *Kế hoạch kiểm soát và đăng nhập*

f) *Xác định mức độ yêu cầu bảo mật*

g) *Rà soát bảo mật hiện tại*

5.2.4. Phân tích yêu cầu tốc độ

Mỗi phút giao dịch: Cung cấp dịch vụ phụ thuộc vào số lượng lớn người dùng, ứng dụng phân tán dùng những giao tác. Số giao tác mỗi phút (TPM) là độ đo tốc độ hệ thống cơ sở dữ liệu.

Băng thông: Ứng dụng phân tán làm nghẽn việc sử dụng mạng. Sự phản hồi của ứng dụng xác định băng thông mạng (độ rộng của đường truyền mạng). Băng thông thường đo bằng megabit/giây.

Khả năng chứa: Lượng lưu trữ cả chính lẫn phụ sẵn sàng đối với ứng dụng là vấn đề lưu tâm quan trọng cho tốc độ chung của ứng dụng. RAM đòi hỏ của ứng dụng gây ra những khác biệt lớn cho tốc độ của ứng dụng.

Nút thắt: Trong mỗi hệ thống, có phần giới hạn tốc độ hệ thống nói chung. Ví dụ: CPU tốc độ nhanh cũng không cải thiện gì mấy nếu như phải chờ dữ liệu từ một ổ cứng quá chậm. Trong trường hợp này ổ cứng sẽ là nút thắt của toàn bộ hệ thống. Chúng ta có thể nhận biết nút thắt bằng cách sử dụng công cụ báo cáo hệ thống như *Màn hình điều khiển tốc độ trên Window NT* (Windows NT Performance Monitor).

Thiết kế dữ liệu: Yếu tố cấu trúc dữ liệu (thiết kế csdl) cũng có ảnh hưởng rất nhiều đến tốc độ truy xuất dữ liệu. Một thiết kế csdl tối ưu về mặt tốc độ truy xuất là điều mà các nhà phân tích thiết kế cần phải quan tâm rất nhiều.

Truy vấn dữ liệu: Thời gian phản hồi lâu của ứng dụng sau một hành động của người dùng thường tập trung ở các báo cáo. Để báo cáo có thể chạy nhanh thì vấn đề tối ưu câu truy vấn cho báo cáo đó là điều rất quan trọng. Thông thường những câu truy vấn chạy chậm là do dùng quá nhiều phép kết nối các bảng (JOIN) hoặc sử dụng các hàm tính toán thống kê (hàm gộp: MAX, MIN, COUNT, AVG, SUM) trên một lượng lớn các bản ghi. Để cải thiện việc này, chúng ta có thể cắt giảm các bảng không cần thiết trong phép JOIN của truy vấn nếu thông tin trong bảng đó không nhằm mục đích phục vụ cho việc gì, có thể hạn chế việc dùng các hàm tính toán thống kê trong truy vấn (thực hiện trên Server) thay bằng để việc tính toán thống kê đó trên phía Client. Việc này sẽ làm giảm tải gánh nặng trên Server.

Sắp xếp chỉ mục (Index): Việc so sánh dữ liệu trong điều kiện tìm kiếm sẽ nhanh hơn nếu dữ liệu so sánh được sắp xếp. Có 2 dạng sắp xếp chỉ mục (index): Một là index tự động (những cột có ràng buộc kháo chính – primary key, những cột ràng buộc khoá duy nhất (Unique key). Hai là index bằng tay (những cột được tạo index bằng câu lệnh người dùng (Create Index). Dựa vào điều này chúng ta có thể cải thiện tốc độ truy xuất dữ liệu bằng cách điều khiển index đến những cột dữ liệu có mặt trong điều kiện tìm kiếm.

Thuật ngữ tốc độ thường dùng đồng nghĩa với sự phản hồi – số lượng thời gian chiếm giữ để phản hồi lại hành động của người dùng.

5.2.5. Phân tích yêu cầu vận hành

Chúng ta có thể giảm bớt chi phí vận hành bằng nhiều cách. Cách tốt nhất là đảm bảo chương trình được kiểm thử và chạy debug trước khi đưa vào triển khai.

5.2.6. Phân tích khả năng mở rộng yêu cầu

Qua thời gian, những yêu cầu giải pháp sẽ thay đổi. Người dùng cần những chức năng mới, các quy luật đặt ra sẽ bị sửa đổi, và phần cứng, phần mềm nền mới thay đổi theo. Ứng dụng thiết kế tốt là có khả năng mở rộng được – nó có thể uyển chuyển cải thiện mà không cần phải viết lại hoàn toàn.

Cách tốt nhất để đạt được khả năng mở rộng là ngắt ứng dụng thành những đối tượng thành phần, mỗi thành phần hoàn thành một nhiệm vụ riêng lẻ. Nếu những yêu cầu của những nhiệm vụ đặc biệt thay đổi, đối tượng tương ứng có thể thay đổi và biện dịch lại mà không gây ảnh hưởng đến bất kỳ đối tượng khác nào.

5.2.7. Phân tích những yêu cầu sẵn có

Tính sẵn sàng liên quan đến nghiệp vụ. Tính sẵn sàng của ứng dụng càng cao, giá trị ứng dụng càng cao. Chúng ta phải xác định bao nhiêu giờ trong ngày ứng dụng cần được thao tác, giờ nào là giờ quan trọng trong ngày.

5.2.8. Phân tích yếu tố con người

Thiết kế ứng dụng được giám sát bởi nhiều người lập trình là phần quan trọng của yếu tố con người. Chúng ta nên xác định kinh nghiệm gì mà chúng ta muốn người dùng có. Với bất kỳ ứng dụng nào khác, kinh nghiệm người dùng càng tốt thì chi phí càng cao.

Bắt đầu định nghĩa mục tiêu của người dùng. Xác định người dùng với những nhu cầu đặc biệt như thế nào. Chúng ta cần điều tiết người dùng qua việc diều tiết nghe và nhìn, hay người dùng nói tiếng nước ngoài. Phụ thuộc vào vị trí địa lý của người sử dụng, chúng ta cần sửa đổi ứng dụng thích ứng theo vị trí địa lý.

5.2.9. Phân tích yêu cầu tích hợp

Nếu cần giải pháp giao tiếp với ứng dụng kế thừa, việc truy xuất CSDL tồn tại, hay việc chuyển đổi dữ liệu cũ sang khuôn dạng mới, chúng ta cần phải đưa kế hoạch tích hợp ứng dụng với phần mềm cũ. Có nhiều cách để thực hiện việc chuyển đổi dữ liệu cũ sang khuôn dạng mới như: sử dụng việc sao chép bằng tay, dùng các tiện ích chuyển đổi từ hệ quản trị CSDL hay viết mã lệnh để kết nối đến CSDL và thực hiện sao chép…

5.2.10. Phân tích thực tiễn nghiệp vụ tồn tại

Phần định nghĩa trong quy tắc nghiệp vụ liên quan đến sự hiểu biết ngữ cảnh trong những quy tắc thao tác. Hiểu được những thực tế nghiệp vụ của doanh nghiệp có thể giúp chúng ta tránh được sai sót thậm chí giúp tìm cách tốt hơn, hiệu quả hơn của tự động hoá tiến trình nghiệp vụ. Không hiểu rõ ràng sơ đồ tổ chức, không thể đem lại sự chấp thuận phù hợp cho thiết kế ứng dụng của chúng ta.

5.2.11. Phân tích yêu cầu khả năng quy mô

Nếu ứng dụng thành công sẽ hấp dẫn nhiều người dùng hơn. Đặc biệt, nếu ứng dụng chạy trên môi trường web thì sự thành công đồng nghĩa với tăng nhu cầu. Ứng dụng phải được thiết kế có quy mô, nó phải hỗ trợ nâng cấp cho phép phục vụ nhiều người hơn.

5.3. Xác định yêu cầu

- Mục tiêu của việc xác định yêu cầu:

Xác định thật chính xác và đầy đủ các yêu cầu đặt ra cho phần mềm sẽ được xây dựng.

- Kết quả nhận được sau khi xác định yêu cầu:
  - Danh sách các công việc sẽ được thực hiện trên máy tính.
  - Những mô tả chi tiết về những công việc này khi thực hiện trong thế giới thực.

5.3.1. Yêu cầu và mô tả yêu cầu

- Yêu cầu phần mềm là công việc muốn thực hiện trên máy tính. Những công việc này phải xuất phát từ thực tế chứ không thuần tuý tin học.
- Mô tả yêu cầu là mô tả đầy đủ các thông tin liên quan đến công việc tương ứng. Các mô tả dùng làm cơ sở để nghiệm thu và đánh giá phần mềm khi được chuyển giao.

Những loại thông tin được quan tâm khi xác định yêu cầu:

a) Tên công việc:

Cần xác định tên công việc một cách cụ thể, tránh chung chung, mơ hồ

b) Người thực hiện:

Cần xác định chính xác người hay bộ phận thực hiện công việc trên máy tính (còn gọi là người dùng). Những người có những công việc tương tự như nhau được xếp vào cùng một loại người dùng.

Cùng một công việc có thể có nhiều loại người dùng khác nhau thực hiện và ngược lại, một loại người dùng có thể thực hiện nhiều công việc khác nhau.

c) Thời gian, địa điểm:

Cần xác định chính xác thời gian và địa điểm thực hiện công việc. Các thông tin này sẽ có ý nghĩa nhất định trong một số trường hợp đặc thù.

d) Cách thức tiến hành và các quy định liên quan:

Đây là phần chính yếu khi xác định yêu cầu. Các thông tin cần quan tâm như sau:

- Các công việc cần kiểm tra khi thực hiện công việc ghi nhận thông tin

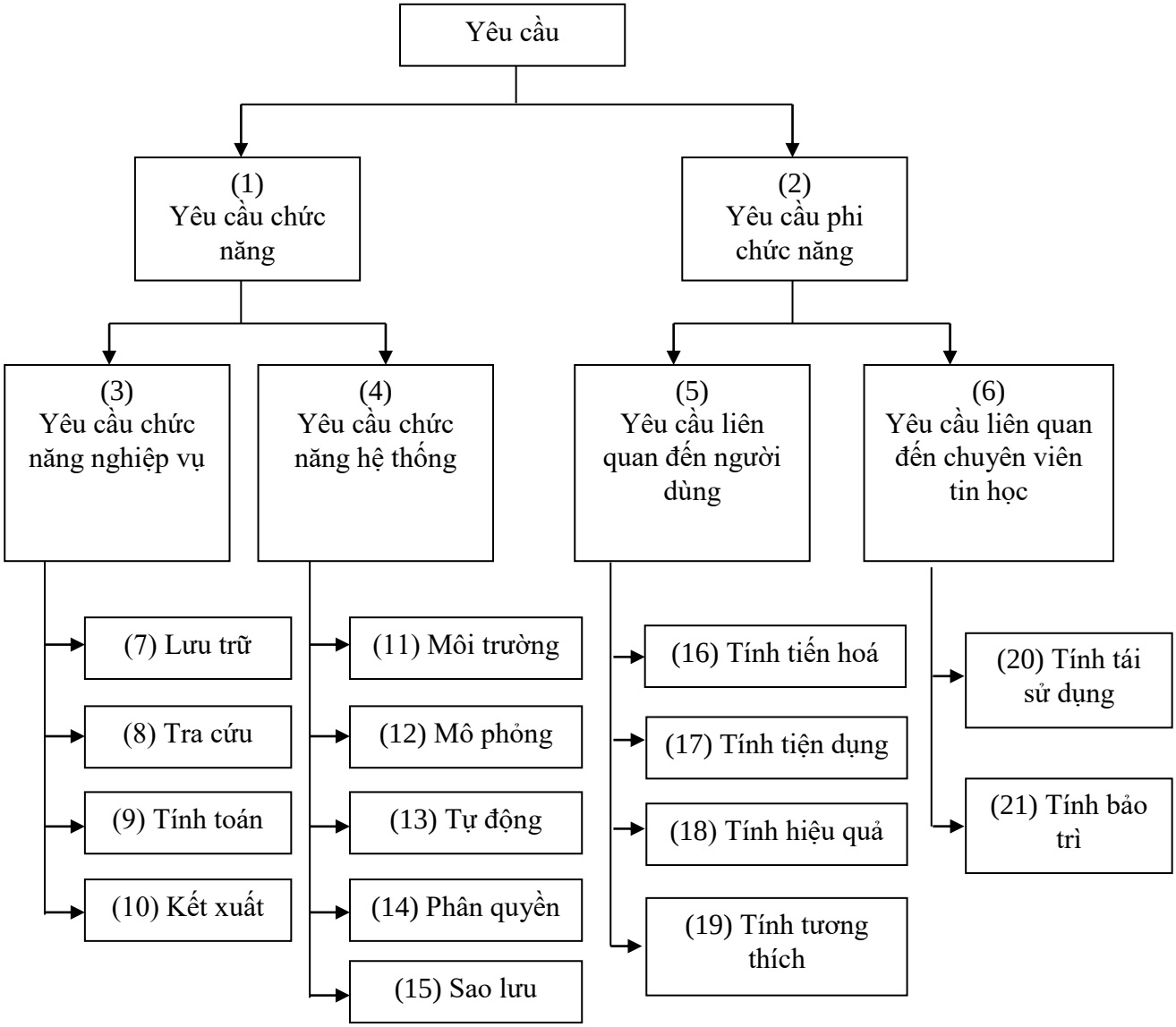
Ví dụ: Quy định về việc cho mượn sách khi độc giả mượn sách là chỉ cho những độc giả mượn sách khi thẻ độc giả còn hạn.

- Các quy định, công thức tính toán khi thực hiện công việc tính toán.

Ví dụ: Quy định về cách tính lương của một công ty

Lương = Ngày công * Mức lương / 22 + Thưởng – Phạt

5.3.2. Phân loại yêu cầu



(1) Yêu cầu chức năng là danh sách các công việc sẽ được thực hiện trên máy tính cùng với các thông tin mô tả tương ứng.

(2) Yêu cầu phi chức năng là các yêu cầu liên quan đến chất lượng phần mềm.

(3) Yêu cầu chức năng nghiệp vụ là các chức năng của phần mềm tương ứng với công việc có thực trong thế giới thực.

(4) Yêu cầu chức năng hệ thống là các chức năng phần mềm được phát sinh thêm khi thực hiện các công việc trên máy tính thay vì trong thế giới thực hoặc các chức năng không tương ứng với bất kỳ công việc nào trong thế giới thực.

(5) Yêu cầu liên quan đến người dùng là yêu cầu chất lượng phần mềm mà người dùng cần.

(6) Yêu cầu liên quan đến chuyên viên tin học là yêu cầu chất lượng phần mềm mà chuyên viên tin cần.

(7) Chức năng lưu trữ: tương ứng với công việc ghi chép thông tin trên sổ sách (kèm theo các quy định khi ghi chép).

(8) Chức năng tra cứu: tương ứng với công việc tìm kiếm, theo dõi hoạt động và xem thông tin về đối tượng.

(9) Chức năng tính toán: tương ứng với công việc tính toán (theo quy định và công thức cho trước).

(10) Chức năng kết xuất: tương ứng với công việc lập báo cáo (theo biểu mẫu cho trước).

(11) Chức năng môi trường: định cấu hình thiết bị, ngày giờ, số người thực hiện…

(12) Chức năng mô phỏng: mô phỏng hoạt động của thế giới thực.

(13) Chức năng tự động: tự động thông báo, nhắc nhỏ người dùng.

(14) Chức năng phân quyền: phân quyền sử dụng giữa các loại người dùng.

(15) Chức năng sao lưu: sao lưu, phục hồi dữ liệu.

5.3.3. Các bước xác định yêu cầu

Đối tượng tham gia xác định yêu cầu:

- *Chuyên viên tin học:* những người hiểu rõ về khả năng của máy tính. Họ phải tìm hiểu rất kỹ về công việc của hệ thống nhằm tránh sự hiểu nhầm cho những bước phân tích sau này.
- *Nhà chuyên môn:* những người hiểu rõ về công việc của mình. Họ cần lắng nghe ý kiến của các chuyên viên tin học để đảm bảo các yêu cầu của họ là có thể thực hiện được với chi phí và thời gian hợp lý.

Hai nhóm người này cần phải phối hợp thật chặt chẽ với nhau để có thể xác định đầy đủ và chính xác các yêu cầu.

a) Khảo sát hiện trạng:

- Tìm hiểu tổng quan về thế giới thực:

- Quy mô hoạt động
- Các hoạt động mà đơn vị có tham gia

- Tìm hiểu hiện trạng tổ chức (cơ cấu tổ chức):

- Đối nội
- Đối ngoại
- Các chức danh

- Tìm hiểu hiện trạng nghiệp vụ:

- Thông tin đầu vào
- Quá trình xử lý
- Thông tin kết xuất

Sau đó tiến hành xếp loại các nghiệp vụ vào bốn loại sau nhằm tránh thiếu sót khi tim hiểu thông tin:

- Nghiệp vụ lưu trữ
- Nghiệp vụ tra cứu
- Nghiệp vụ tính toán
- Nghiệp vụ tổng hợp, thống kê

b) Lập danh sách các yêu cầu:

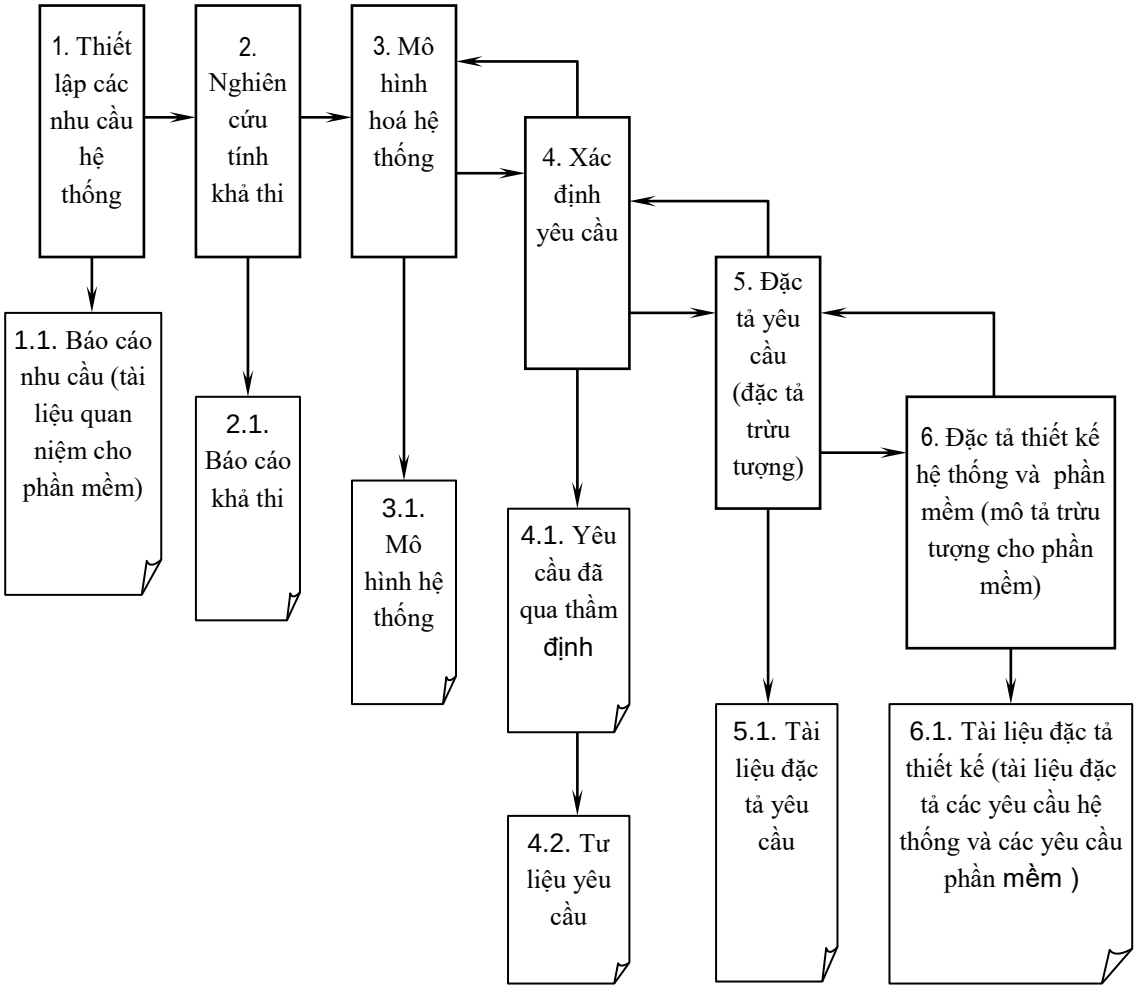
- Xác định yêu cầu chức năng nghiệp vụ
- Xác định yêu cầu chức năng hệ thống
- Xác định yêu cầu phi chức năng

5.4. Đặc tả yêu cầu (Requirements specification)

Đặc tả một vấn đề là mô tả (một cách rất riêng nhờ các kỹ thuật thể hiện) các đặc trưng của vấn đề đó. Vấn đề đó có thể là đối tượng, khái niệm, một thủ tục nào đó, …

Yêu cầu đầu tiên của đặc tả là phải mang tính chính xác.

Phân tích và định rõ yêu cầu là bước kỹ thuật đầu tiên trong tiến trình kỹ nghệ phần mềm. Hoạt *động phân tích* và *định rõ yêu cầu* hướng tới đặc tả yêu cầu phần mềm được thể hiện trong các khuôn cảnh như sau:



Các đặc tả thường mang tính trừu tượng hoá cao. Do vậy người ta phân chia thành nhiều mức đặc tả. Càng ở mức cao (những mức đầu tiên của quá trình làm mịn hoặc chính xác

hoá) đặc tả càng trừu tượng. Càng xuống các mức thấp hơn, đặc tả càng tiến dần tới cụ thể - tức là một thể hiện trên một máy tính cụ thể với một ngôn ngữ lập trình cụ thể - đây chính là quá trình làm mịn dần.

Các loại hình đặc tả.

Có hai kiểu đặc tả đó là *đặc tả hình thức* và *đặc tả phi hình thức*.

Đặc tả hình thức: Là các đặc tả chính xác tức là không thể dẫn tới những cách hiểu khác nhau. Đặc tả hình thức sử dụng công cụ chủ yếu là đại số và logic.

Ví dụ: Đặc tả một ma trận:

- Cấp của ma trận n x n (n là số tự nhiên lẻ).
- Phần tử cuối của hàng 1 bằng phần tử đầu của hàng cuối.
- Phần tử trung tâm bằng trung bình cộng của các phần tử ở 4 góc.

Hoặc có thể diễn đạt như sau:

- A n x n = (a[i, j])n x n; n = 2k + 1, k $\in$ Z.
- a[1, n] = a[n, 1].
- $a\left[\frac{n+1}{2},\frac{n+1}{2}\right]=\left(a[1,1]+a[1,n]+a[n,1]+a[n,n]\right)/4$

Đặc tả phi hình thức: Diễn đạt bằng những ngôn ngữ, tuy không chặt chẽ nhưng được nhiều người biết và có thể trao đổi với nhau để chính xác hoá các điểm chưa rõ ràng, những khái niệm còn mơ hồ.

Ví dụ: Có hai con hậu trên bàn cờ. Hai con hậu sẽ đụng độ nếu chúng nằm trên cùng hàng, cùng cột hoặc trên cùng một đường chéo song song với đường chéo chính hay đường chéo phụ. => Rõ ràng ở đây có một số khái niệm mơ hồ.

Đặc tả hỗn hợp: Phối hợp cả hai kiểu đặc tả trên.

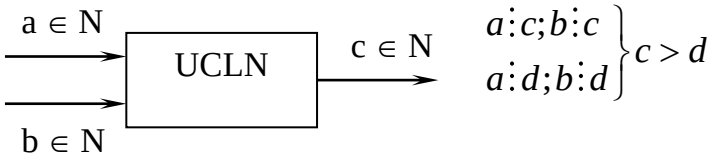
Trong thực tế, có nhiều loại hình đặc tả, ví dụ như:

- *Đặc tả cấu trúc dữ liệu:* Nêu các thành phần của dữ liệu

  Ví dụ: Đặc tả một phân số: Phân_số = { x/y, x $\in$ Z, y $\in$ N }

  Số_phức = { a + b.i | a, b $\in$ R }

- *Đặc tả chức năng:* Mô tả thông qua việc nêu lên các tính chất hay thuộc tính của tên vào và tên ra.



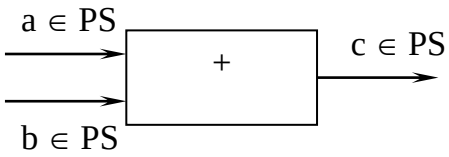
Ví dụ:

- *Đặc tả đối tượng:* Bao gồm đặc tả cấu trúc dữ liệu và mô tả các chức năng.

Ví dụ: đặc tả đối tượng phân số.

PS = { x/y, x ∈ Z, y ∈ N }

Phép cộng: +: PS x PS → PS



- *Đặc tả thao tác:* Nêu lên trình tự tiến hành công việc.

Ví dụ 1: x, y, z ∈ PS. Các bước cần thực hiện đối với phép cộng (+) 2 phân số.

```
z = x + y       {       Quy_đồng_mẫu_số(x, y);
                        z.tử_số = x.tử_số + y.tử_số;
                        z.mẫu_số = x.mẫu_số;
                };
```

Ví dụ 2: Quy trình **Bán hàng**:

1. Khách hàng yêu cầu được mua hàng.
2. Hướng dẫn khách xem và lựa chọn hàng hoá.
3. Thoả thuận hình thức thanh toán: Tiền mặt, séc, chuyển khoản, …
4. Ghi hoá đơn cho khách.
5. Nhận tiền và giao hàng hoá cho khách.

- *Đặc tả cú pháp:* Thực chất là các định nghĩa có tính truy hồi từ tổng thể đến cơ sở. Mô tả cách lắp ghép các ký hiệu, các từ với nhau lại để tạo thành chương trình. Ví dụ: Trong ngôn ngữ lập trình PASCAL, tên (định danh - identify) được khái quát như sau: Là dãy các ký tự bắt đầu bằng chữ cái hoặc dấu gạch nối dưới, sau đó có thể là chữ số, chữ cái hoặc dấu gạch nối dưới.

  <định danh> = <chữ cái> ∪ <định danh> ∪ <ký tự>

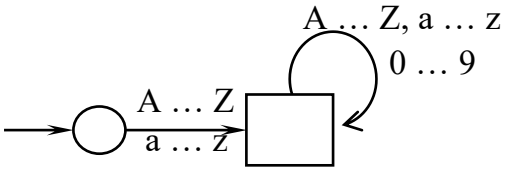
  <ký tự> = <chữ cái> ∪ <chữ số>

  <chữ cái> = { A, B, C, …, Z } ∪ { a, b, …, z }

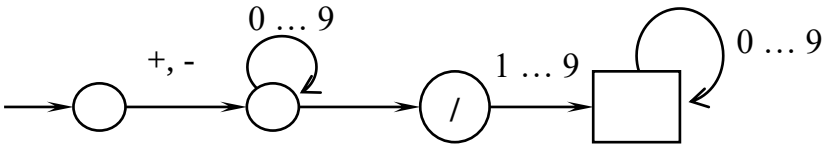
  <chữ số> = { 0, 1, 2, …, 9 }

- *Đặc tả qua sơ đồ:*

  Ví dụ: Đặc tả định danh



  Đặc tả phân số



g. Đặc tả thuật toán: Các bước thao tác để giải quyết bài toán.

Kiểu đặc tả phải phù hợp với giải pháp. Các yêu cầu của phần mềm có thể được phân tích theo một số cách khác nhau. Các ký thuật phân tích có thể dẫn tới những đặc tả trên giấy hay trên máy tính (được xây dựng nhờ CASE) có chứa các mô tả ngôn ngữ đồ hoạ và tự nhiên cho yêu cầu phần mềm. Việc làm bản mẫu giúp đặc tả có thể được triển khai, tức là bản mẫu sẽ thể hiện những công việc thực hiện các yêu cầu. Các ngôn ngữ đặc tả hình thức dẫn đến biểu diễn hình thức.

Các nguyên lý đặc tả (Tự đọc).

Đặc tả có thể xem như một tiến trình biểu diễn. Mục đích cuối cùng của đặc tả là các yêu cầu được biểu thị sao cho dẫn tới việc cài đặt phần mềm thành công. Balzer và Goldman đề nghị 8 nguyên lý đặc tả tốt.

***Nguyên lý 1**: Phân tách chức năng với cài đặt.*

Trước hết, theo định nghĩa, đặc tả là một mô tả về điều mong muốn, chứ không phải là cách thực hiện nó (cài đặt). Đặc tả có thể chấp nhận 2 dạng hoàn toàn khác nhau. *Dạng thứ nhất* là dạng của các *hàm toán học*: Với một tập *dữ liệu đầu vào* đã cho, tạo ra một *tập dữ liệu đầu ra đặc biệt*. Dạng tổng quát của đặc tả như thế là tìm ra (một hoặc tất cả những) kết quả ứng với P (đầu vào), với P biểu thị một tân từ bất kỳ. Trong đặc tả như thế, kết quả thu được phải được diễn đạt một cách đầy đủ, toàn vẹn, theo dạng đó là cái gì (không phải đó là như thế nào). Một phần điều này là vì kết quả của một hàm (toán học) của đầu vào (phép toán có điểm bắt đầu và điểm kết thúc đã xác định rõ) không bị ảnh hưởng bởi môi trường bao quanh.

***Nguyên lý 2**: Cần ngôn ngữ đặc tả hệ thống hướng tiến trình.*

Xét tình huống trong đó môi trường là động và sự thay đổi của nó ảnh hưởng tới hành vi của thực thể nào đó tương tác với môi trường đó (như trong "hệ thống máy tính nhúng"). Hành vi của nó không thể biểu diễn được ở dạng hàm (toán học) của đầu vào. Thay vì thế, cần phải sử dụng cách biểu diễn khác - cách mô tả hướng tiến trình, trong đó đặc tả cái gì đã đạt được bằng cách xác định một mô hình các thao tác mong muốn đạt được của hệ thống dưới dạng các công việc đáp ứng chức năng đối với kích thích khác nhau từ môi trường.

Những đặc tả hướng tiến trình như vậy, trình bày một mô hình về hành vi hệ thống, thông thường đã bị loại ra khỏi các ngôn ngữ đặc tả hình thức, nhưng chúng lại là bản chất nếu nhiều tình huống động phức tạp hơn cần phải được đặc tả. Trong thực tế, cần phải thừa nhận rằng trong những tình huống như vậy cả tiến trình cần tự động hoá lẫn môi trường tồn tại của nó đều phải được mô tả một cách hình thức. Tức là, toàn bộ hệ thống các bộ phận tương tác phải được đặc tả chứ không chỉ một thành phần được đặc tả.

***Nguyên lý 3**: Đặc tả phải bao gồm hệ thống có phần mềm là một thành phần trong đó*

Một hệ thống bao gồm các thành phần tương tác nhau. Chỉ bên trong hoàn cảnh của hệ thống toàn bộ và tương tác giữa các thành phần của nó thì hành vi của một thành phần riêng mới có thể được xác định. Nói chung, một hệ thống có thể được mô hình hoá như một tập hợp các sự vật tích cực và thụ động. Những sự vật này có liên quan lẫn nhau và qua thời gian thì mối quan hệ giữa các sự vật thay đổi. Mối quan hệ động này đưa ra sự kích thích cho các sự vật tích cực, còn gọi là các *tác nhân*, đáp ứng. Sự đáp ứng có thể gây ra những thay đổi thêm nữa, và do đó, tạo ra thêm kích thích để cho các tác nhân có thể đáp ứng lại.

***Nguyên lý 4**: Đặc tả phải bao gồm cả môi trường mà hệ thống vận hành.*

Tương tự, môi trường mà trong đó hệ thống vận hành và tương tác với cũng phải được xác định.

May mắn là điều này đơn thuần chỉ cần sự thừa nhận rằng bản thân môi trường cũng là một hệ thống bao gồm các sự vật tương tác, cả tích cực lẫn thụ động, mà trong đó hệ thống chỉ là một tác nhân. Các tác nhân khác, theo định nghĩa là không thay đổi bởi vì chúng là một phần của môi trường, giới hạn phạm vi của việc thiết kế và cài đặt về sau. Trong thực tế, sự khác nhau duy nhất giữa hệ thống và môi trường của nó là ở chỗ nỗ lực thiết kế và cài đặt về sau sẽ vận hành chỉ trong đặc tả cho hệ thống. Đặc tả môi trường làm cho "giao diện" của hệ thống được xác định theo cùng cách như bản thân hệ thống chứ không đưa vào cách hình thức khác.

Cần phải chú ý rằng bức tranh đặc tả hệ thống được trình bày ở đây chính là bức tranh của tập hợp các tác nhân xoắn xuýt nhau cao độ phản ứng với những kích thích trong môi trường (thay đổi các sự vật) do các tác nhân đó tạo ra. Chỉ có thông qua những hành động điều phối của tác nhân mà hệ thống mới đạt tới mục tiêu của nó. Sự phụ thuộc lẫn nhau vi phạm vào nguyên lí phân tách (cô lập với các phần khác của hệ thống và môi trường). Nhưng đây là một nguyên lí *thiết kế*, không phải là nguyên lí đặc tả. Thiết kế tuân theo đặc tả, và quan tâm tới việc phân rã một đặc tả thành các mẩu gần tách biệt để chuẩn bị cho cài đặt. Tuy nhiên đặc tả phải vẽ lại chính xác bức chân dung của hệ thống và môi trường của nó như cộng đồng người dùng cảm nhận theo một cách thức nhiều chi tiết như các giai đoạn cài đặt và thiết kế cần tới. Vì mức độ chi tiết cần thiết này là khó thấy trước, nếu không nói là không thể, nên đặc tả, thiết kế và cài đặt phải được thừa nhận như một hoạt động tương tác. Do đó điều mấu chốt là công nghệ cần có để bao quát thật nhiều cho hoạt động này khi bản đặc tả được soạn thảo và thay đổi (trong cả hai giai đoạn phát triển khởi đầu và bảo trì về sau).

***Nguyên lý 5**: Đặc tả hệ thống phải là một mô hình nhận thức.*

Đặc tả hệ thống phải là một mô hình nhận thức chứ không phải là một mô hình thiết kế hay cài đặt. Nó phải mô tả một hệ thống như cộng đồng người sử dụng cảm nhận thấy. Các sự vật mà nó thao tác phải tương ứng với các sự vật của lĩnh vực đó; các tác nhân phải mô hình cho các cá nhân, tổ chức và trang thiết bị trong lĩnh vực đó; còn các hành động họ thực hiện thì phải mô hình cho những hoạt động thực tế xuất hiện trong lĩnh vực.

Đặc tả phải có khả năng tổ hợp vào trong nó những qui tắc hay luật bao trùm các sự vật thuộc lĩnh vực. Một số trong những trường hợp là luật *bài trừ những trạng thái nào đó của hệ thống* (như "hai sự vật không thể đồng thời ở cùng một chỗ và vào cùng một lúc"), và do đó giới hạn hành vi của các tác nhân hay chỉ ra nhu cầu soạn thảo thêm để ngăn cản những trạng thái này khỏi nảy sinh. Các luật khác *mô tả cách các sự vật đáp ứng lại khi bị kích thích*

(như luật chuyển động của Newton). Những luật này, biểu thị cho "tính vật lí" của lĩnh vực, là phần cố hữu của đặc tả hệ thống.

***Nguyên lý 6**: Đặc tả phải thể hiện tính vận hành.*

Đặc tả phải đủ đầy đủ và hình thức để có thể được dùng trong việc xác định liệu một cài đặt được đề nghị có thoả mãn đặc tả cho những trường hợp kiểm thử tuỳ ý không. Tức là, *với kết quả của việc cài đặt trên một tập dữ liệu được chọn một cách tuỳ ý, phải có thể dùng đặc tả để xác định tính hợp lệ cho những kết quả đó*. Điều này kéo theo rằng đặc tả, mặc dầu không phải là một đặc tả hoàn toàn về cách thức, vẫn có thể hành động như một bộ sinh các hành vi có thể trong số những hành vi phải có của cài đặt được đề nghị. Do đó, theo một nghĩa mở rộng, đặc tả này phải là vận hành...

***Nguyên lý 7**: Đặc tả chấp nhập dung sai về tính không đầy đủ.*

Không đặc tả nào có thể là đầy đủ hoàn toàn. Môi trường trong đó nó tồn tại thường quá phức tạp cho điều đó. Một đặc tả bao giờ cũng là một mô hình - một sự trừu tượng hoá - của một tình huống thực (hay được mường tượng) nào đó. Do đó, nó sẽ không đầy đủ. Hơn thế nữa, như đã được phát biểu nó sẽ tồn tại tại ở nhiều mức chi tiết. Tính vận hành được yêu cầu ở trên không nhất thiết là cần thiết. Các công cụ phân tích được sử dụng để giúp cho người đặc tả và để kiểm thử đặc tả phải có khả năng xử lí với tính không đầy đủ. Một cách tự nhiên điều này làm cho việc phân tích bị yếu đi, khi có thể được thực hiện bằng cách mở rộng phạm vi các hành vi chấp nhận được thỏa mãn cho đặc tả, nhưng một sự suy giảm như vậy phải phản ánh các mức độ bất trắc còn lại.

***Nguyên lý 8**: Đặc tả phải được cục bộ hoá và được ghép lỏng lẻo.*

Các nguyên lí trước xử lí đặc tả như một thực thể tĩnh. Thực thể này nảy sinh từ cái động của đặc tả. Cần phải thừa nhận rằng mặc dầu mục tiêu chính của một đặc tả là để dùng làm cơ sở cho thiết kế và cài đặt một hệ thống nào đó, nó không phải là một sự vật tĩnh dựng sẵn mà là một sự vật động đang trải qua thay đổi đáng kể. Việc thay đổi như thế xuất hiện trong ba hoạt động chính: phát biểu, khi một đặc tả ban đầu đang đươc tạo ra, phát triển, khi đặc tả được soạn thảo trong quá trình thiết kế lặp để phản ánh môi trường đã thay đổi và / hoặc các yêu cầu chức năng phụ.

Với nhiều thay đổi xuất hiện cho đặc tả, điều mấu chốt là nội dung và cấu trúc của nó được chọn để làm phù hợp hoạt động này. Yêu cầu chính cho sự phù hợp đó là ở chỗ thông tin bên trong đặc tả phải được cục bộ hoá sao cho chỉ một phần nhỏ (một cách lí tưởng) cần phải sửa đổi khi thông tin thay đổi, và ở chỗ đặc tả cần được cấu trúc (ghép) một cách lỏng

lẻo để cho từng phần có thể được thêm vào hay loại bỏ một cách dễ dàng, và cấu trúc được điều chỉnh một cách tự động.

Mặc dầu các nguyên lí được Balzer và Goldman tán thành tập trung vào tác động của đặc tả trên định nghĩa về ngôn ngữ hình thức, những lời bình luận của họ áp dụng được cho cả mọi dạng đặc tả. Tuy nhiên, các nguyên lí cần phải được dịch thành sự thực hiện. Trong mục sau chúng ta sẽ xem xét một tập các hướng dẫn để tạo ra một đặc tả các yêu cầu.

Các mức trừu tượng của đặc tả.

Các đặc tả được thể hiện ở một vài mức trừu tượng khác nhau cùng với mối tương liên giữa các mức ấy. Mỗi mức nhắm đến các đối tượng đọc khác nhau mà họ có quyền quyết định về việc dựa vào đó mà thực hiện đánh giá bản thiết kế của các nhà phát triển phần mềm. Các mức đó là:

***Mức 1**: Định ra yêu cầu.*

Được thể hiện bằng ngôn ngữ tự nhiên về các dịch vụ mà hệ thống sẽ phải cung cấp. Phần này phải được viết sao cho dễ hiểu đối với khách hàng và người quản lý hợp đồng, người sẽ mua sản phẩm phần mềm và người sẽ sử dụng nó. Kỹ thuật đặc tả phi hình thức là thích hợp cho mức đặc tả này.

***Mức 2**: Đặc tả yêu cầu.*

Tài liệu nêu ra các dịch vụ một cách chi tiết hơn. Tài liệu này đôi khi còn được gọi là tài liệu đặc tả chức năng. Yêu cầu đối với đặc tả ở mức này là phải chính xác đến mức có thể làm cơ sở cho hợp đồng giữa nhà phát triển phần mềm và khách hàng. Đồng thời cũng cần được viết sao cho dễ hiểu đối với nhân viên kỹ thuật của cả nơi mua phần mềm và nơi phát triển hệ thống. Kỹ thuật đặc tả hình thức hẳn là thích hợp cho mức đặc tả như vậy, tuy nhiên cũng còn tuỳ thuộc vào trình độ kiến thức cơ bản của khách hàng. Tốt hơn cả là ta có thể dùng loại hình hỗn hợp để đặc tả.

***Mức 3**: Đặc tả phần mềm / đặc tả thiết kế (đây là mô tả trừu tượng cho phần mềm).*

Dùng làm cơ sở cho việc thiết kế và thực thi. Cần thể hiện một quan hệ rõ ràng giữa tư liệu này và đặc tả yêu cầu. Ta phải xác định rằng: đối tượng đọc ở đây chủ yếu là các kỹ sư phần mềm chứ không phải là người sử dụng hoặc người quản lý. Kỹ thuật đặc tả hình hình thức là hoàn toàn phù hợp cho mức đặc tả này.

5.5. Xét duyệt yêu cầu (Requirements validation)

Việc xét duyệt bản *Đặc tả yêu cầu phần mềm* (và/ hoặc bản mẫu) do cả người phát triển phần mềm và khác hàng cùng tiến hành. Bởi vì đặc tả tạo nên nền tảng cho giai đoạn phát triển nên cần phải cực kì cẩn thận trong khi tiến hành cuộc họp xét duyệt.

Việc xét duyệt trước hết được tiến hành ở mức *vĩ mô*. Tại mức này, người xét duyệt cố gắng đảm bảo rằng bản đặc tả được đầy đủ, nhất quán và chính xác. Cần đề cập tới các câu hỏi sau:

1. Các mục tiêu và mục đích đã được thiết lập cho phần mềm có nhất quán với mục tiêu và mục đích của hệ thống hay không?
2. Những giao diện quan trọng với mọi phần tử hệ thống đã được mô tả chưa?
3. Luồng và cấu trúc thông tin đã được mô tả thích hợp cho lĩnh vực vấn đề chưa?
4. Các biểu đồ có rõ ràng không? Liệu mỗi biểu đồ có thể đứng riêng không lời giải thích không?
5. Các chức năng chính có còn bên trong phạm vi và đã được mô tả thích hợp chưa?
6. Liệu hành vi của phần mềm có nhất quán với thông tin nó phải xử lí và chức năng nó phải thực hiện hay không?
7. Các ràng buộc thiết kế có hiện thực không?
8. Rủi ro công nghệ phát triển là gì?
9. Các yêu cầu phần mềm khác đã được xem xét đến chưa?
10. Các tiêu chuẩn hợp lệ đã được phát biểu chi tiết chưa? Chúng có thích hợp để mô tả một hệ thống thành công không?
11. Liệu có sự không nhất quán, bỏ sót hay dư thừa nào không?
12. Việc tiếp xúc với khách hàng có đầy đủ không?
13. Người dùng đã xét duyệt bản *Tài liệu sơ bộ của người dùng* hay bản mẫu chưa?
14. Các ước lượng về *Kế hoạch dự án phần mềm* bị ảnh hưởng thế nào?

Để đưa ra câu trả lời cho nhiều câu hỏi trên, việc xét duyệt có thể tập trung vào mức *chi tiết*. Tại đây, mối quan tâm của chúng ta là vào từ ngữ của bản đặc tả. Chúng ta cố gắng làm lộ ra vấn đề có thể ẩn náu bên trong nội dung đặc tả. Những hướng dẫn sau đây là gợi ý về việc xét duyệt chi tiết bản đặc tả:

- Phải quan sát các mối nối có sức thuyết phục (như "chắc chắn", "do đó", "rõ ràng", "hiển nhiên", "từ đó suy ra rằng") và hỏi "Tại sao chúng lại có đó?"
- Theo dõi những thuật ngữ mông lung (như "một số", "đôi khi", "thường", "thông thường", "bình thường", "phần lớn", "đa số"); yêu cầu làm sáng tỏ.
- Khi có nêu danh sách, nhưng không đầy đủ, thì phải đảm bảo mọi khoản mục đều được hiểu rõ. Chú ý vào các từ như "vân vân", "cứ như thế", "cứ tiếp tục như thế", "sao cho".
- Phải chắc chắn phát biểu phạm vi không chứa những giả thiết không được nói rõ (như *mã hợp lệ trong khoảng 10 tới 100*. Đó là số nguyên, số thực hay số hệ 16?
- Phải nhận biết về các động từ mơ hồ như "xử lí", "loại bỏ", "nhảy qua", "xoá bỏ" Có thể có nhiều cách hiểu về nó.
- Phải nhận biết các đại từ "vu vơ" (như "mô đun vào/ra liên lạc với mô đun kiểm tra tính hợp lệ dữ liệu và đặt cờ báo kiểm soát *của nó*." Cờ kiểm soát của ai?).
- Tìm các câu có chứa sự chắc chắn (như "bao giờ", "mọi", "tất cả", "không một", "không bao giờ") rồi yêu cầu bằng chứng.
- Khi một thuật ngữ được định nghĩa tường minh tại một chỗ thì hãy thử thay thế định nghĩa này vào chỗ xuất hiện của nó.
- Khi một cấu trúc được mô tả theo lời thì hãy vẽ ra bức tranh để giúp hiểu được nó.
- Khi một tính toán được xác định thì hãy thử với ít nhất hai thí dụ.

Một khi việc xét duyệt đã hoàn tất thì bản bản *đặc tả yêu cầu phần mềm* sẽ được cả khách hàng lẫn người phát triển "ký tắt". Bản đặc tả trở thành một "hợp đồng" cho việc phát triển phần mềm. Những thay đổi trong yêu cầu được nêu ra sau khi bản đặc tả đã hoàn thành sẽ không bị huỷ bỏ. Nhưng khách hàng phải lưu ý rằng từng thay đổi sau khi kí đều là một mở rộng của phạm vi phần mềm và do đó có thể làm tăng thêm chi phí và / hoặc kéo dài lịch biểu (thời gian thực hiện).

Ngay cả với những thủ tục xét duyệt tốt nhất tại chỗ thì một số vấn đề đặc tả thông thường vẫn còn lại. Bản đặc tả rất khó "kiểm thử" theo mọi cách có ý nghĩa, và do đó sự không nhất quán hay bỏ sót có thể bị bỏ qua không để ý tới. Trong khi xét duyệt, người ta có thể khuyến cáo những thay đổi cho bản đặc tả. Có thể sẽ cực kì khó khăn để lượng định tác động toàn cục của thay đổi; tức là, làm sao việc thay đổi trong một chức năng lại ảnh hưởng tới các yêu cầu cho chức năng khác?

Bài tập:

1. Trình bày các kỹ thuật thu thập yêu cầu
2. Trình bày mô hình phân tích yêu cầu
3. Trình bày các tài liệu đặc tả yêu cầu

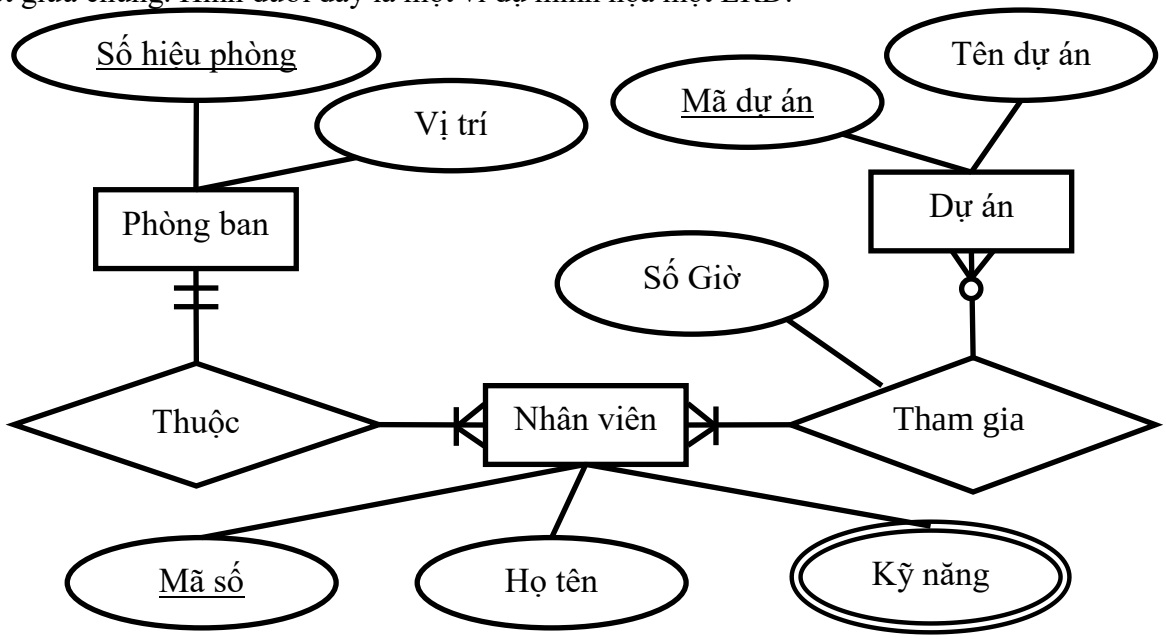
Chương 6. Mô hình hóa hệ thống

6.1. Mô hình hóa miền thông tin (Information Domain Modeling)

Mô hình hoá miền thông tin cần thực hiện:

- Định danh dữ liệu: Tên dữ liệu (thực thể, đối tượng)
- Định nghĩa các thuộc tính: Tên thuộc tính, kiểu dữ liệu của thuộc tính, miền trị của thuộc tính, các ràng buộc trên thuộc tính.
- Thiết lập mối quan hệ giữa các dữ liệu: Mối quan hệ giữa các dữ liệu (giữa các đối tượng hay thực thể).

Mô hình thực thể - liên kết (Entity Relationship Model), sau đây viết tắt là ERM, là một mô hình dữ liệu mức khái niệm. ERM như một công cụ để giao tiếp giữa nhà phân tích thiết kế và người dùng cuối trong giai đoạn thu thập và phân tích yêu cầu. Có ba thần cơ bản là: Thực thể, thuộc tính và mối liên kết trong ERM. Sơ đồ thực thể - liên kết (Entity Relationship Diagram), sau đây viết tắt là ERD, là một thể hiện đồ họa về các thực thể và mối liên kết giữa chúng. Hình dưới đây là một ví dụ minh họa một ERD:



Hình 6.1: Ví dụ về mô hình thực thể - liên kết

6.1.1. Thực thể:

Thực thể (Entity) là một khái niệm để chỉ ***một lớp các đối tượng*** cụ thể hay các ***khái niệm độc lập*** có cùng những đặc trưng chung người dùng cuối (end user) ***quan tâm***.

Thực thể vốn tồn tại trong thế giới thực là các lớp đối tượng như: Nhân viên, phòng ban, sinh viên, … hay các khái niệm như: Tài khoản, điểm thi, … Nếu như thực thể dùng để chỉ một lớp các đối tượng thì một đối tượng cụ thể trong lớp đó được gọi là một bản thể.

Chẳng hạn thực thể nhân viên có các đặc trưng như: Mã nhân viên, họ tên, ngày sinh, địa chỉ. Khi đó mỗi nhân viên cụ thể được gọi là một bản thể của thực thể nhân viên. Ví dụ:

Các đặc trưng của thực thể	Một bản thể
Mã nhân viên	20254
Họ tên	Lê Mai Dung
Ngày sinh	11/5/1948
Địa chỉ	Hồng Việt – Đông Hưng – Thái Bình

Bảng sau cho thấy sự khác nhau giữ thực thể và bản thể:

	Thực thể (Entity)	Bản thể (Instance)
Khái niệm	Chỉ một lớp các đối tượng	Chỉ một đối tượng cụ thể
Số lượng	Một	Nhiều
Bản chất	"Khung" chứa dữ liệu	Dữ liệu
Thể hiện	Tên thực thể và tên các đặc trưng	Bộ các giá trị tương ứng với các đặc trưng

Trong ERM, mỗi thực thể được gán một tên duy nhất và được biểu diễn bằng hình chữ nhật trong đó ghi tên thực thể.

- **Thuộc tính của kiểu thực thể**

Thuộc tính (Attribute) là các đặc trưng của thực thể. Mỗi thực thể có một tập các thuộc tính gắn kết với nó. Ví dụ: Thực thể sinh viên có các thuộc tính: mã sinh viên, họ tên, địa chỉ, điện thoại mà các trường đại học quan tâm để quản lý sinh viên.

Trong ERM, các thuộc tính được mô tả bằng hình elip trong đó ghi tên thuộc tính và được nối với thực thể bằng một đoạn thẳng.

Thuộc tính định danh (thuộc tính khóa) là một hay một số thuộc tính của một thực thể mà giá trị của nó cho phép phân biệt được các bản thể khác nhau của một thực thể. Ví dụ: Trong thực thể sinh viên, thuộc tính định danh có thể là mã sinh viên. Trong ERM, tên của thuộc tính định danh được gạch chân phía dưới.

Thuộc tính đa trị là thuộc tính có thể nhận nhiều hơn một giá trị đối với mỗi bản thể. Ví dụ: Trong thực thể nhân viên, kỹ năng là thuộc tính đa trị. Một nhân viên có thể có nhiều kỹ năng như soạn thảo văn bản, lái xe ô tô, … Trong ERM, thuộc tính đa trị được biểu diễn bởi hình elip kép.

- **Mối liên kết của kiểu thực thể**

Các mối liên kết (Relationships) gắn kết các thực thể trong ERM. Một mối liên kết có thể kết nối một thực thể với một hoặc nhiều thực thể khác. Nó phản ánh mối quan hệ vốn có giữa các bản thể của thực thể đó.

Mối liên kết giữa các thực thể chia làm hai loại: mối quan hệ tương tác (Đại lý "mua" hàng, giảng viên "dạy" sinh viên, …) và mối quan hệ sở hữu hay phụ thuộc (Nhân viên "có" kỹ năng, hóa đơn "của" khách hàng, …). Do vậy, người ta thường dùng một động từ để đặt tên cho mối liên kết.

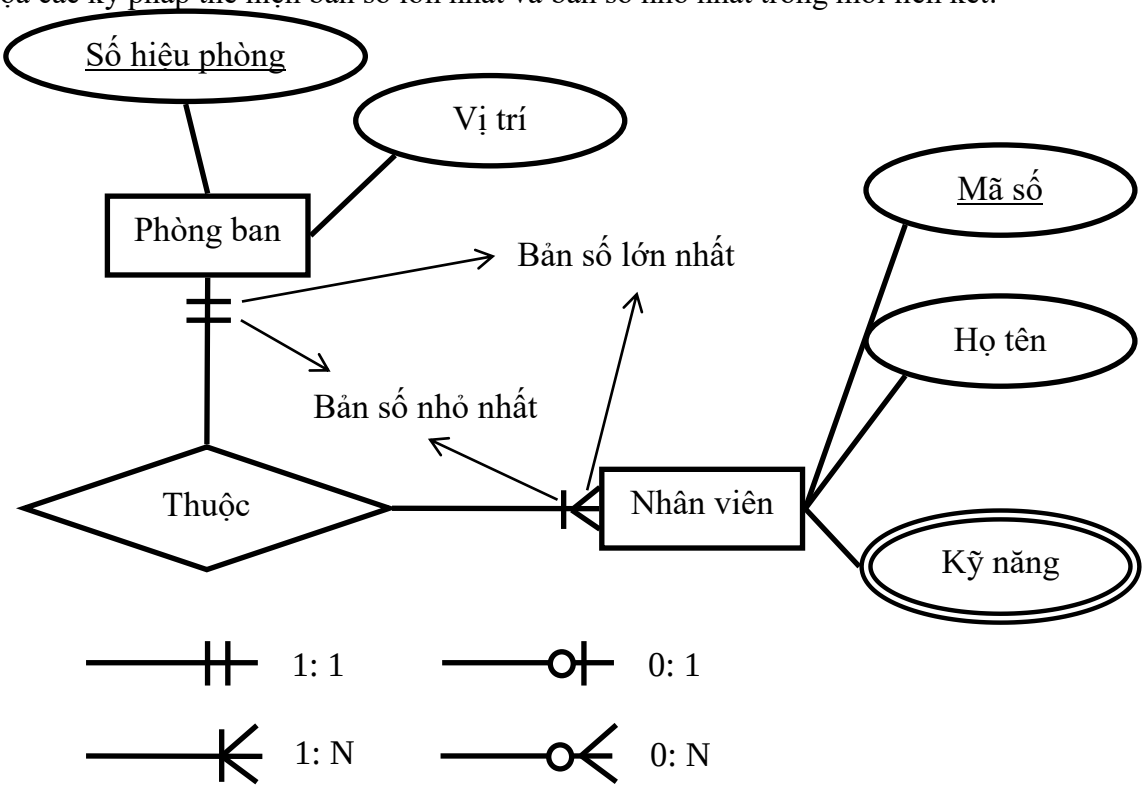
Mối liên kết cũng có thể có thuộc tính. Ví dụ mối liên kết "tham gia" gắn kết thực thể "Nhân viên" với "Dự án" sẽ có thuộc tính là "số giờ" để chỉ ra số giờ mà một nhân viên tham gia vào dự án.

- ***Các bản số của thực thể trong mối liên kết***

Giả sử có hai thực thể A và B được liên kết với nhau bằng một mối quan hệ, ta xét các trường hợp sau:

- Nếu một bản thể của thực thể A có thể có quan hệ với chỉ một bản thể của thực thể B và ngược lại thì ta nói rằng mối liên kết giữa thực thể A và thực thể B là mối liên kết một – một (1:1).
- Nếu một bản thể của thực thể B có thể có quan hệ với nhiều bản thể của thực thể A và một bản thể của thực thể A chỉ có thể có quan hệ với một bản thể của thực thể B thì ta nói rằng mối liên kết giữa thực thể A và thực thể B là mối liên kết một – nhiều (1:N).
- Nếu một bản thể của thực thể A có thể có quan hệ với nhiều bản thể của thực thể B và ngược lại thì ta nói rằng mối liên kết giữa thực thể A và thực thể B là mối liên kết nhiều nhiều (M:N).

Bản số (Cardinality) của thực thể B trong mối liên kết với thực thể A là số các bản thể của thực thể B có thể liên kết với một bản thể của thực thể A trong mối liên kết đó. Hình sau minh họa các ký pháp thể hiện bản số lớn nhất và bản số nhỏ nhất trong mối liên kết:



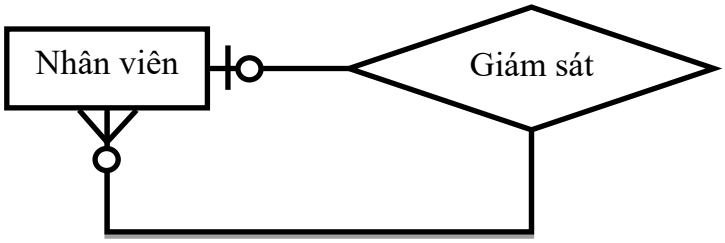
Hình 6.2: Các ký hiệu sử dụng trong mô hình ER

Ta sử dụng thuật ngữ ***"sự phụ thuộc tồn tại" (existence dependency)*** để chỉ rằng một bản thể của một thực thể không thể tồn tại nếu không tồn tại một bản thể của thực thể kia. Khi đó một ***thực thể yếu (weak entity)*** là một thực thể có sự tồn tại phụ thuộc. Ví dụ thực thể "Bản sao" là thực thể yếu trong mối liên kết với thực thể "Phim", hay thực thể "Người thân" là thực thể yếu trong mối liên kết với thực thể "Nhân viên".

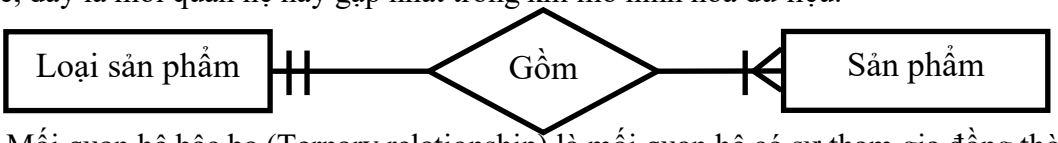
- *Bậc của mối liên kết*

Bậc (Degree) của mối liên kết là số lượng các thực thể tham gia vào mối liên kết đó. Có ba loại thường gặp là: Mối liên kết bậc một, bậc hai và bậc ba.

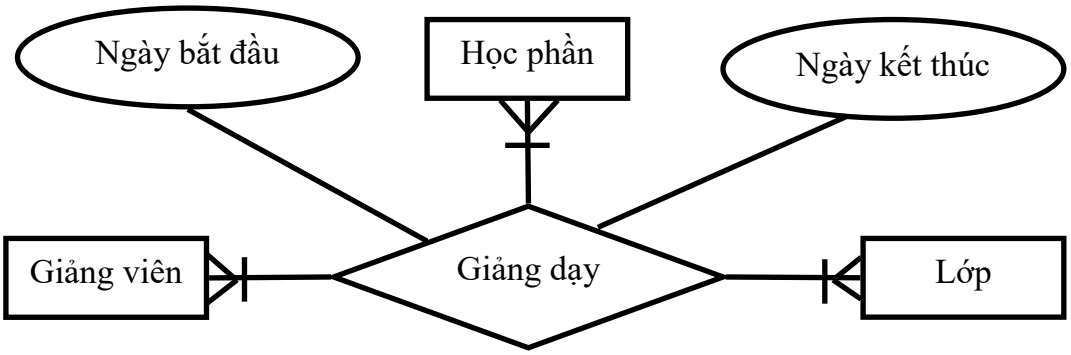
Mối quan hệ bậc một còn được gọi là mối quan hệ đệ quy (Recursive relationship) là mối quan hệ giữa các bản thể của cùng một thực thể.



Mối quan hệ bậc hai (Binary relationship) là mối quan hệ giữa các bản thể của hai thực thể, đây là mối quan hệ hay gặp nhất trong khi mô hình hóa dữ liệu.



Mối quan hệ bậc ba (Ternary relationship) là mối quan hệ có sự tham gia đồng thời của các bản thể thuộc ba thực thể khác nhau.



- ***Ví dụ thiết kế ERM***

Trong phần này ta xét một ví dụ tổng quát về việc xây dựng ERM cho cơ sở dữ liệu công ty. Giả sử sau khi tập hợp và phân tích các yêu cầu ta thu được các ghi chép súc tích về yêu cầu người dùng về dữ liệu như sau:

Công ty được tổ chức thành các đơn vị, mỗi đơn vị có tên đơn vị, mã số đơn vị, vị trí và một nhân viên quản lý, công ty cần ghi lại thông tin về ngày bắt đầu lên làm quản lý của nhân viên. Tại một thời điểm một nhân viên nếu có chỉ được quản lý một đơn vị.

Mỗi đơn vị được giao phụ trách một số dự án. Mỗi dự án cần lưu thông tin về tên dự án, mã số dự án, địa điểm, ngày bắt đầu và ngày hoàn thành dự án.

Công ty có một kho vật tư để cấp phát cho các dự án. Mỗi vật tư có một tên, đơn vị tính và được đánh một mã số duy nhất. Mỗi dự án được cung cấp một số loại vật tư với số lượng nhất định. Công ty cần lưu trữ thông tin về các lần cấp vật tư cho từng dự án như: Ngày cấp phát, người nhận. Với mỗi lần cấp vật tư cần ghi rõ cấp vật tư nào với số lượng bao nhiêu.

Với mỗi nhân viên cần lưu thông tin về họ tên nhân viên, mã số nhân viên, địa chỉ, giới tính, ngày sinh. Một nhân viên chỉ làm việc tại một đơn vị nhưng có thể tham gia vào các dự án của những đơn vị khác. Công ty cần lưu lại thông tin về số giờ nhân viên tham gia vào dự án, mỗi nhân viên có thể được giám sát trực tiếp bởi một nhân viên khác.

Đi kèm với mỗi nhân viên, công ty cần lưu giữ thông tin về thân nhân của họ như: họ tên, ngày sinh, địa chỉ.

- **Xác định các thực thể và thuộc tính**

Dựa vào các ghi chép ở trên ta xác định được các thực thể và thuộc tính của chúng như bảng dưới đây:

Thực thể	Thuộc tính
Đơn vị	Mã đơn vị, tên đơn vị, vị trí

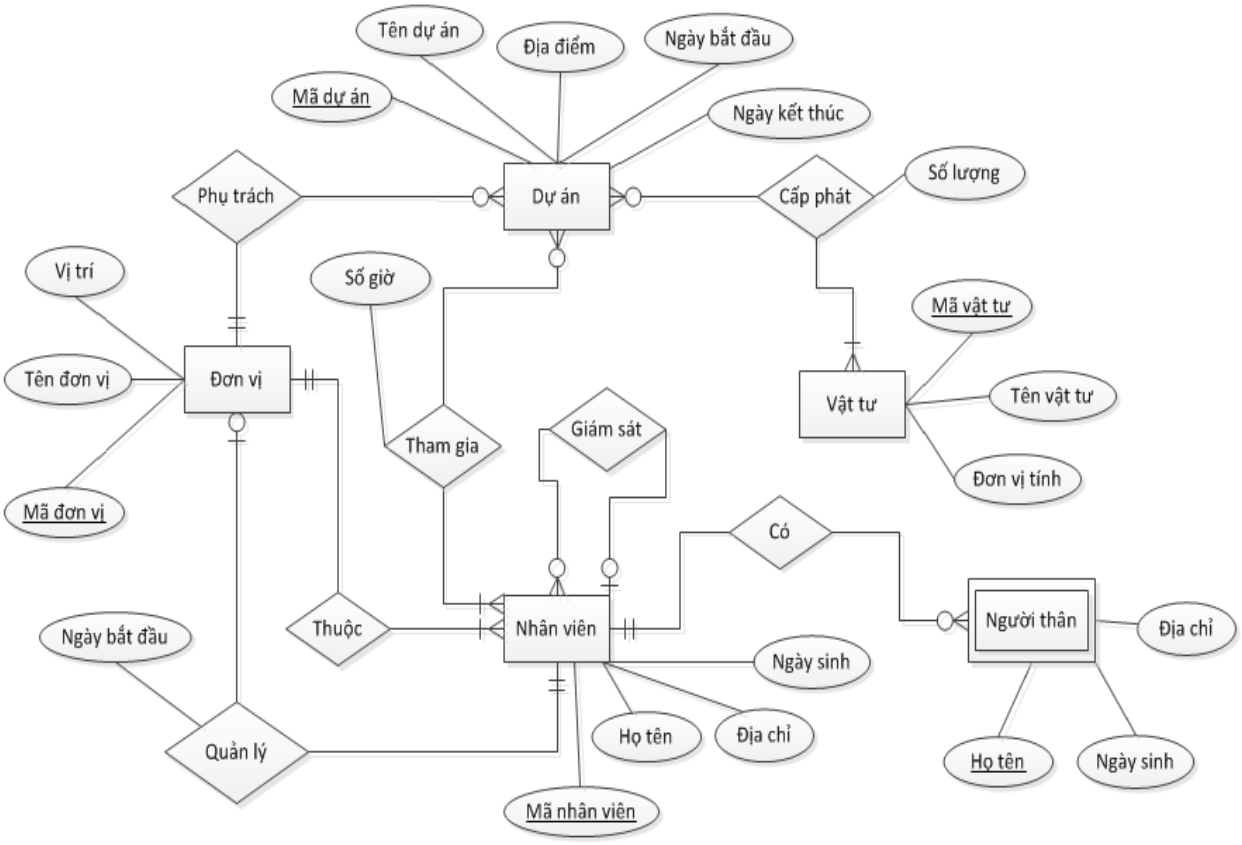
Dự án	Mã số dự án, tên dự án, địa điểm, ngày bắt đầu, ngày hoàn thành
Vật tư	Mã vật tư, tên vật tư, đơn vị tính
Nhân viên	Mã nhân viên, họ tên, ngày sinh, địa chỉ
Người thân	Họ tên, ngày sinh, địa chỉ

- **Xác định mối liên kết giữa các thực thể**

Khi xác định được các thực thể và thuộc tính của chúng, dựa vào các thông tin ở trên ta xác định được mối liên kết giữa các thực thể như trong bảng sau:

Mối liên kết	Loại liên kết	Thuộc tính
Nhân viên ***"thuộc"*** đơn vị	1 : N	
Nhân viên ***"quản lý"*** đơn vị	1 : 1	Ngày bắt đầu
Nhân viên ***"giám sát"*** nhân viên	1 : N	
Đơn vị ***"phụ trách"*** dự án	1 : N	
Nhân viên ***"tham gia"*** dự án	N: M	Số giờ
Vật tư ***"cấp phát"*** cho dự án	N: M	Số lượng
Nhân viên ***"có"*** người thân	1: N	

Kết quả ta có ERD về công ty như sau:



Hình 6.3: Ví dụ về mô hình ER

6.1.2. Quy tắc chuyển đổi mô hình thực thể quan hệ sang mô hình các bảng quan hệ:

✓ ***Quy tắc 1:***

Mỗi thực thể (Trừ thực thể yếu) chuyển thành một quan hệ có cùng tên và tập thuộc tính. Thuộc tính khóa trong thực thể tạo thành khóa chính trong quan hệ.

✓ ***Quy tắc 2:***

Mỗi thuộc tính đa trị tạo thành một quan hệ mới có thuộc tính là thuộc tính đa trị và thêm thuộc tính khóa của quan hệ biểu diễn thực thể hoặc mối liên kết chứa thuộc tính đa trị. Nếu thuộc tính đa trị là phức hợp thì ta chỉ đưa vào quan hệ mới các thành phần đơn của nó.

✓ ***Quy tắc 3:***

Mỗi thực thể yếu chuyển thành một quan hệ có cùng tên và tập thuộc tính với thực thể yếu và thêm vào thuộc tính khóa của quan hệ liên quan. Khóa chính của quan hệ là sự kết hợp của thuộc tính khóa bộ phận trong thực thể yếu và thuộc tính khóa của quan hệ liên quan.

✓ ***Quy tắc 4:***

Với mối liên kết hai ngôi 1: 1, thêm những thuộc tính khóa của một quan hệ vào quan hệ còn lại.

✓ ***Quy tắc 5:***

Với mối liên kết hai ngôi 1: N, thêm khóa của quan hệ 1 vào làm khóa ngoài trong quan hệ nhiều.

✓ ***Quy tắc 6:***

Mỗi mối liên kết hai ngôi M: N tạo thành một quan hệ có tên là tên của mối liên kết, tập thuộc tính là các thuộc tính của mối liên kết (nếu có) và thêm vào các thuộc tính khóa của các quan hệ liên quan.

✓ ***Quy tắc 7:***

Mỗi mối liên kết n ngôi R (n > 2) tạo thành một quan hệ S có tập thuộc tính là các thuộc tính trong mối liên kết và thêm vào các thuộc tính khóa của các quan hệ liên quan. Tuy nhiên, nếu một thực thể E tham gia vào mối liên kết là 1 thì khóa chính của quan hệ S không được chứa thuộc tính khóa ngoài tham gia chiếu tới quan hệ E tương ứng với kiểu thực thể E.

Bài tập: Áp dụng bài toán trên mục 6.1.1, xác định các bảng cho hệ thống.

6.2. Mô hình hóa chức năng (Functional Modeling)

Bản chất của phần mềm là biến đổi thông tin. Mô hình hoá chức năng cần xác định:

- Định danh các chức năng: Tên chức năng (công việc cần thực hiện trên phần mềm).
- Xác định cách thức dữ liệu (thông tin) di chuyển trong hệ thống.
- Xác định các tác nhân tạo dữ liệu và tác nhân tiêu thụ dữ liệu: Đối tượng tạo dữ liệu và đối tượng tiêu thụ dữ liệu trong chức năng.

Mô hình chức năng là biểu diễn có cấu trúc về các chức năng, các hoạt động, và các quá trình bên trong hệ thống.

Phân rã chức năng là quá trình phân rã các mối quan hệ có tính chức năng thành các phần nhỏ hơn mà dựa vào đó có thể xây dựng lại hệ thống.

Phương pháp:

- Sử dụng biểu đồ khối luồng chức năng
- Sử dụng kỹ thuật phân tích và thiết kế hướng chức năng

Ví dụ: Xác định chức năng của hệ thống cho vay cầm đồ:

1. **Chức năng hệ thống:**

   - Khai báo, cấu hình hệ thống
   - Quản trị người dùng và phân quyền
   - Sao lưu (backup) dữ liệu
   - Khôi phục dữ liệu
   - Thay đổi mật khẩu người dùng

2. **Chức năng khai báo danh mục:**

   - Danh mục Loại hàng
   - Danh mục khách hàng
   - Danh mục bảng giá
   - Danh mục nhân viên
   - Danh mục loại thu
   - Danh mục loại chi

3. **Chức năng giao dịch:**

   - Quản lý cho vay cầm đồ, họ nằm
   - Quản lý cho vay họ góp
   - Phiếu thu
   - Phiếu chi
   - Quản lý tồn quỹ

4. **Báo cáo thống kê:**

   - Báo cáo cho vay cầm đồ, họ nằm
   - Báo cáo cho vay họ góp

- Báo cáo sổ thu

- Báo cáo sổ chi

- Báo cáo tổng hợp thu chi

6.3. Mô hình hóa luồng thông tin (Information Flow Modeling)

Biểu đồ luồng dữ liệu là biểu diễn đồ hoạ của luồng dữ liệu trong một hệ thống thông tin.

Mỗi một sơ đồ dòng dữ liệu thường gồm các thành phần chức năng hoặc tiến trình, dòng dữ liệu, kho dữ liệu và các đối tượng.

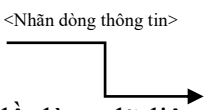
Mô hình hoá luồng thông tin cần xác định các thành phần trong sơ đồ:

- Tiến trình (Chức năng): trong sơ đồ dòng dữ liệu, chức năng hay tiến trình là một quá trình biến đổi thông tin. Từ thông tin đầu vào nó biến đổi, tổ chức lại thông tin, bổ sung thông tin hoặc tạo ra thông tin mới, tổ chức thành thông tin đầu ra, phục vụ cho hoạt động của hệ thống như lưu vào kho dữ liệu hoặc gửi cho các tiến trình hay đối tượng khác.

  Ký hiệu:

- Dòng dữ liệu: Dòng dữ liệu là dòng chuyển dời thông tin vào hoặc ra khỏi một tiến trình, một chức năng, một kho dữ liệu hoặc một đối tượng nào đó. Các thành phần của dòng dữ liệu bao gồm đường biểu diễn dòng, mũi tên chỉ hướng dịch chuyển thông tin và tên của dòng. Cần chú ý là các dòng dữ liệu khác nhau phải mang tên khác nhau, và các thông tin trải qua thay đổi thì phải có tên mới cho phù hợp.

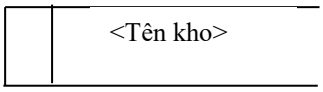
  Ký hiệu: dùng mũi tên



- Kho dữ liệu: Trong sơ đồ dòng dữ liệu, kho dữ liệu thể hiện các thông tin cần lưu trữ. Dưới dàng vật lý, kho dữ liệu này có thể là tập tài liệu, cặp hồ sơ hoặc tệp thông tin

trên đĩa. Trong sơ đồ dòng dữ liệu, dưới tên kho dữ liệu chúng ta sẽ chỉ quan tâm tới các thông tin được chứa trong đó.

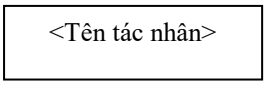
Ký hiệu:



Trong một trang sơ đồ dòng dữ liệu ta có thể đặt một kho dữ liệu ở nhiều chỗ, nhằm giúp việc thể hiện các dòng dữ liệu trở nên dễ dàng hơn.

- Tác nhân ngoài: Tác nhân ngoài có thể là một người, một nhóm người hoặc một tổ chức bên ngoài hệ thống, nhưng có mối liên hệ với hệ thống.
- Tác nhân trong: Tác nhân trong là một chức năng hoặc một tiến trình bên trong hệ thống, được miêu tả ở trang khác của sơ đồ.

Ký hiệu: Hình chữ nhật

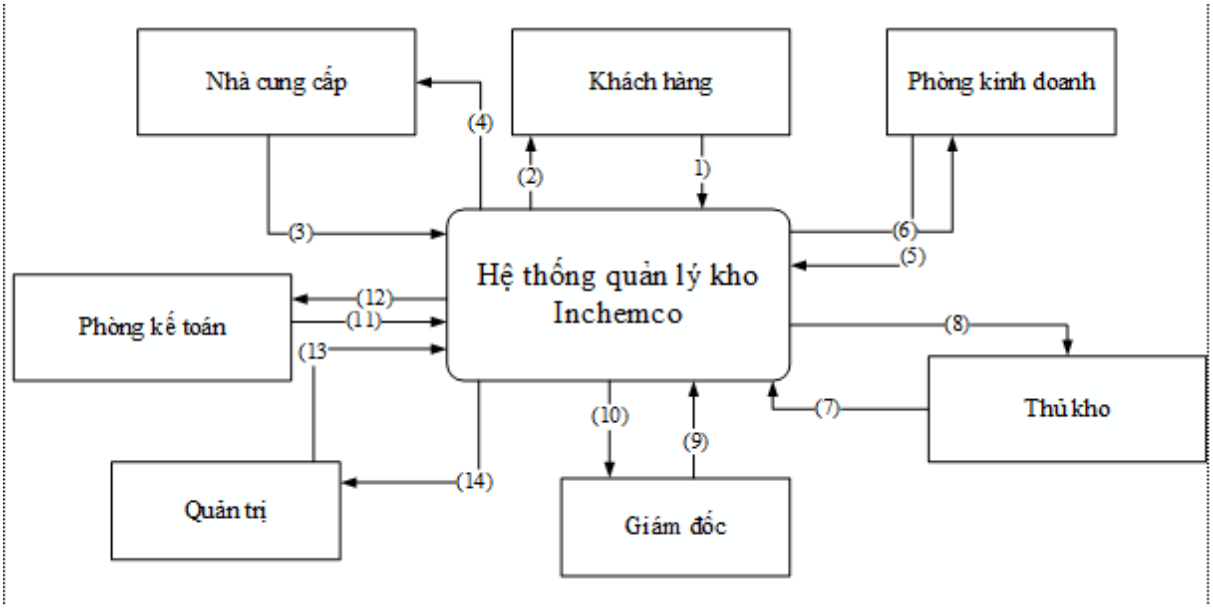


Sơ đồ ngữ cảnh:

Sơ đồ ngữ cảnh (Context Diagrams) bao gồm ba nhóm thành phần:

- Thành phần chính là một vòng tròn nằm ở vị trí trung tâm của sơ đồ, biểu thị cho toàn bộ hệ thống đang được nghiên cứu.
- Xung quanh vòng tròn trung tâm này là tất cả các phần tử bên ngoài, có quan hệ với hệ thống (tác nhân ngoài).
- Tất cả các đường truyền thông tin vào và ra khỏi hệ thống (nghĩa là nối hệ thống với mọi tác nhân ngoài của nó).

Ví dụ về biểu đồ luồng dữ liệu:



Hình 2.2.1. Sơ đồ luồng dữ liệu mức ngữ cảnh.

❖ *Chú thích*

1. Thông tin dữ liệu khách hàng, đơn đặt hàng.
2. Thông tin dữ liệu hàng hóa/báo giá, phiếu xuất, phiếu thu, công nợ…
3. Thông tin dữ liệu nhà cung cấp, hóa chất.
4. Thông tin hóa đơn, đơn hàng, phiếu chi, công nợ...
5. Thông tin dữ liệu yêu cầu cập nhật đơn hàng, phiếu xuất, phiếu nhập, thông tin khách hàng, chương trình khuyến mại…
6. Thông tin dữ liệu các phiếu nhập hàng, phiếu xuất hàng, đơn đặt hàng, báo cáo doanh số….
7. Thông tin yêu cầu dữ liệu phiếu điều hàng, phiếu hàng khách trả, phiếu hủy hàng, khai báo hàng tồn đầu, phiếu nhập, phiếu xuất, phiếu xuất chuyển kho …
8. Thông tin dữ liệu phiếu điều hàng, phiếu hàng khách trả, phiếu hủy hàng, báo cáo hàng tồn, báo cáo nhập xuất tồn, báo cáo hàng nhập, báo cáo hàng xuất …
9. Thông tin yêu cầu cung cấp thống kê, báo cáo nhập xuất tồn kho, báo cáo doanh số bán hàng, báo cáo doanh thu lợi nhuận, báo cáo công nợ….
10. Thông tin cung cấp thống kê, báo cáo nhập xuất tồn kho, báo cáo doanh số bán hàng….
11.Yêu cầu cập nhật dữ liệu công nợ đối với khách hàng, nhà cung cấp, doanh số, doanh thu, Phiếu thu, Phiếu chi…
12. Dữ liệu công nợ đối với khách hàng, nhà cung cấp, doanh số, doanh thu, Báo cáo thu, Báo cáo chi, Báo cáo tổng hợp thu chi…
13. Dữ liệu yêu cầu cập nhật thông tin quản trị hệ thống, phân quyền người dùng
14. Thông tin quản trị hệ thống, sao lưu (backup), phục hồi dữ liệu.

Quy tắc vẽ luồng thông tin:

- Luồng thông tin không đi từ kho đến kho
- Luồng thông tin không đi từ kho đến tác nhân
- Luồng thông tin không đi từ tác nhân đến kho
- Luồng thông tin không đi từ tác nhân đến tác nhân
- Luồng thông tin không đi từ tiến trình đến chính nó
- Mỗi luồng thông tin chỉ có một chiều (một mũi tên)

Bài tập: Vẽ sơ đồ mức ngữ cảnh của hệ thống cho vay cầm đồ.

Sơ đồ mức đỉnh:

Biểu diễn thông tin về các tiến trình chính, có cả tác nhân ngoài, kho dữ dữ liệu và dòng thông tin.

VD: Cho vay cầm đồ:

- **Khai báo hệ thống**

  + Cấu hình hệ thống

  + Quản trị người dùng và phân quyền

  + Sao lưu dữ liệu (backup)

  + Khôi phục dữ liệu (retore)

  + Thay đổi mật khẩu người dùng

- **Khai báo danh mục:**

  + Quản lý danh mục nhân viên

  + Quản lý danh mục loại hàng

  + Quản lý danh mục khách hàng

  + Quản lý danh mục loại thu

  + Quản lý danh mục loại chi

- **Quản lý Giao dịch:**

  + Quản lý cho vay cầm đồ

  + Quản lý cho vay họ góp

+ Phiếu thu

+ Phiếu chi

- **Báo cáo thống kê:**

+ Báo cáo cho vay họ góp

+ Báo cáo cho vay cầm đồ

+ Báo cáo sổ thu

+ Báo cáo sổ chi

+ Báo cáo tổng hợp thu chi

Sơ đồ mức dưới đỉnh:

Biểu diễn thông tin về các tiến trình con của từng tiến trình chính, có cả tác nhân ngoài, kho dữ dữ liệu và dòng thông tin.

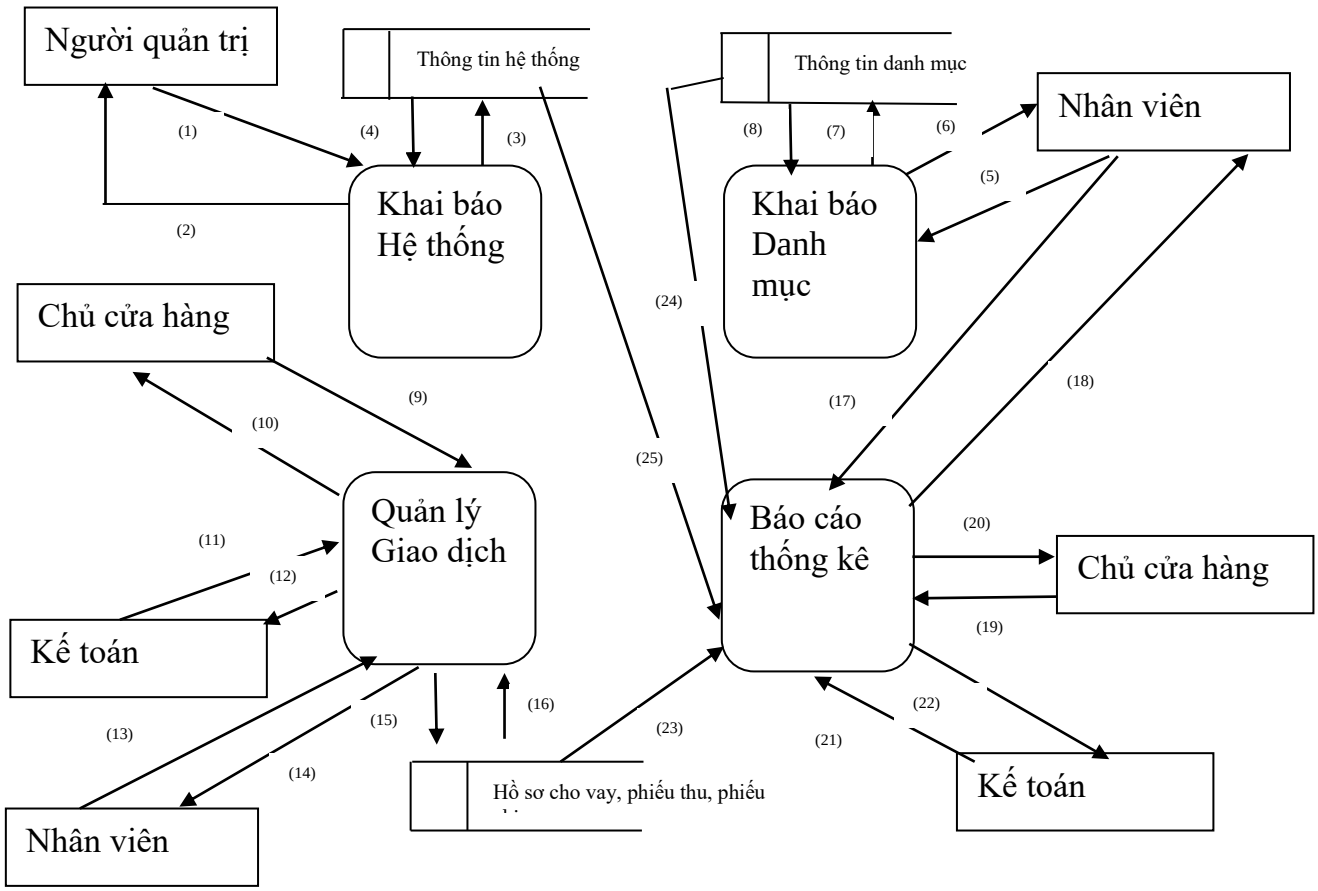
VD:

- Mức dưới đỉnh của Quản lý khai báo hệ thống
- Mức dưới đỉnh của Quản lý khai báo danh mục
- Mức dưới đỉnh của Quản lý giao dịch
- Mức dưới đỉnh của Quản lý báo cáo thống kê

Bài tập:

1. Trình bày Biểu đồ Phân rã chức năng của một hệ thống nào đó
2. Trình bày biểu đồ luồng dữ liệu của một hệ thống nào đó

Sơ đồ mức đỉnh của Hệ thống cho vay cầm đồ:



Sơ đồ mức đỉnh của Hệ thống cho vay cầm đồ:

Ghi chú:

(1): Thông tin tài khoản người dùng, thông tin cấu hình hệ thống

(2): Thông tin tra cứu tài khoản người dùng, dữ liệu backup

(3): Thông tin tài khoản người dùng, thông tin cấu hình hệ thống

(4): Thông tin tài khoản người dùng, lịch sử truy cập

(5): Thông tin nhân viên, khách hàng, loại hàng, loại thu, loại chi

(6): Thông tin tra cứu về nhân viên, khách hàng, loại hàng, loại thu, loại chi

(7): Thông tin danh mục nhân viên, khách hàng, loại hàng, loại thu, loại chi

(8): Thông tin tra cứu về nhân viên, khách hàng, loại hàng, loại thu, loại chi

(9): Thông tin phiếu cho vay, phiếu thu, phiếu chi

(9): Phiếu cho vay cầm đồ, Phiếu cho họ góp, Phiếu phiếu thu, Phiếu phiếu chi

(10): Phiếu cho vay cầm đồ, Phiếu cho họ góp, Phiếu phiếu thu, Phiếu phiếu chi, thông tin tra cứu về cho vay, phiếu thu, phiếu chi

(11): Phiếu phiếu thu, Phiếu phiếu chi

(12): Thông tin tra cứu phiếu thu, phiếu chi

(13): Phiếu cho vay cầm đồ, Phiếu cho họ góp

(14): Thông tin tra cứu Phiếu cho vay cầm đồ, Phiếu cho họ góp

(15): Phiếu cho vay cầm đồ, Phiếu cho họ góp, Phiếu phiếu thu, Phiếu phiếu chi

(16): Thông tin tra cứu về cho vay, phiếu thu, phiếu chi

(17): Thông tin yêu cầu tra cứu về cho vay cầm đồ, cho vay họ góp

(18): Báo cáo, tra cứu về cho vay cầm đồ, cho vay họ góp

(19): Thông tin yêu cầu tra cứu về cho vay cầm đồ, cho vay họ góp, phiếu thu, phiếu chi

(20): Báo cáo, tra cứu về cho vay cầm đồ, cho vay họ góp, báo cáo sổ thu, báo cáo sổ chi, báo cáo tổng hợp thu chi

(21): Thông tin yêu cầu tra cứu về phiếu thu, phiếu chi

(22): Báo cáo về sổ thu, báo cáo sổ chi, báo cáo tổng hợp thu chi

(23): Thông tin về cho vay, phiếu thu, phiếu chi

(23): Thông tin về nhân viên, khách hàng, loại thu, loại chi

(24): Thông tin về danh mục khách hàng, loại hàng, nhân viên

(25): Thông tin về người dùng

Chương 7. Thiết kế hệ thống

7.1. Tổng quan về thiết kế

Trong đời sống hàng ngày, khi một người nào đó cần xây dựng một ngôi nhà, người đó mời một kỹ sư xây dựng đến, yêu cầu thiết kế cho họ ngôi nhà. Với các số liệu về căn nhà cần xây dựng. Căn cứ vào đó, người kỹ sư sẽ thiết kế ra mô hình ngôi nhà. Đây không phải là ngôi nhà được đã được xây dựng trong thực tế, mà chỉ là trên bản vẽ. Nhưng thông qua mô hình đó, cùng với sự mô tả chi tiết của người kỹ sư, chủ nhà cũng có thể hình dung ra ngôi nhà của mình. Bản thiết kế này rất quan trọng, nó giúp cho chủ nhà cùng với kỹ sư xây dựng hiểu về công việc mình cần làm, nếu có yêu cầu chỉnh sửa thì thực hiện chỉ trên bản vẽ. Còn khi đã bắt tay vào xây dựng thực tế thì việc chỉnh sửa lúc này sẽ rất khó khăn và tốn kém.

Khi sản xuất phần mềm cũng vậy. Rõ ràng, yêu cầu của khách hàng cũng không khác gì yêu cầu cần xây ngôi nhà của chủ nhà nọ. Công việc của kỹ sư xây dựng và kỹ sư phầm mềm theo từng giai đoạn cũng có nhiều điểm chung. Ta hãy xem xét bảng so sánh sau:

Kỹ sư xây dựng	**Kỹ sư phần mềm**
• Khảo sát địa hình, tìm hiểu nhu cầu của chủ nhà: cần xây nhà bao nhiêu tầng, kích thước bao nhiêu, trang trí như thế nào, …	• Tìm hiểu nhu cầu khách hàng, khảo sát hệ thống, lấy số liệu, …
• Thiết kế ngôi nhà trên bản vẽ	• Thiết kế phần mềm, đưa ra mô hình
• Tìm hiểu ý kiến chủ nhà về bản thiết kế	• Duyệt lại với khách hàng
• Thực hiện các chỉnh sửa nếu cần	• Thực hiện các chỉnh sửa nếu cần
• Cho thi công ngôi nhà	• Tiến hành cài đặt chương trình

Thiết kế là bước đầu tiên trong giai đoạn phát triển cho bất kỳ sản phẩm hay hệ thống công nghệ nào. Nó có thể được định nghĩa là *"... tiến trình áp dụng nhiều kỹ thuật và nguyên lý với mục đích xác định ra một thiết bị, một tiến trình hay một hệ thống đủ chi tiết để cho phép thực hiện nó về mặt vật lý."*.

Mục tiêu của thiết kế là tạo ra một mô hình hay biểu diễn của một thực thể (sự vật: ngôi nhà, chiếc xe hơi, cái cầu, …) mà sau này được xây dựng.

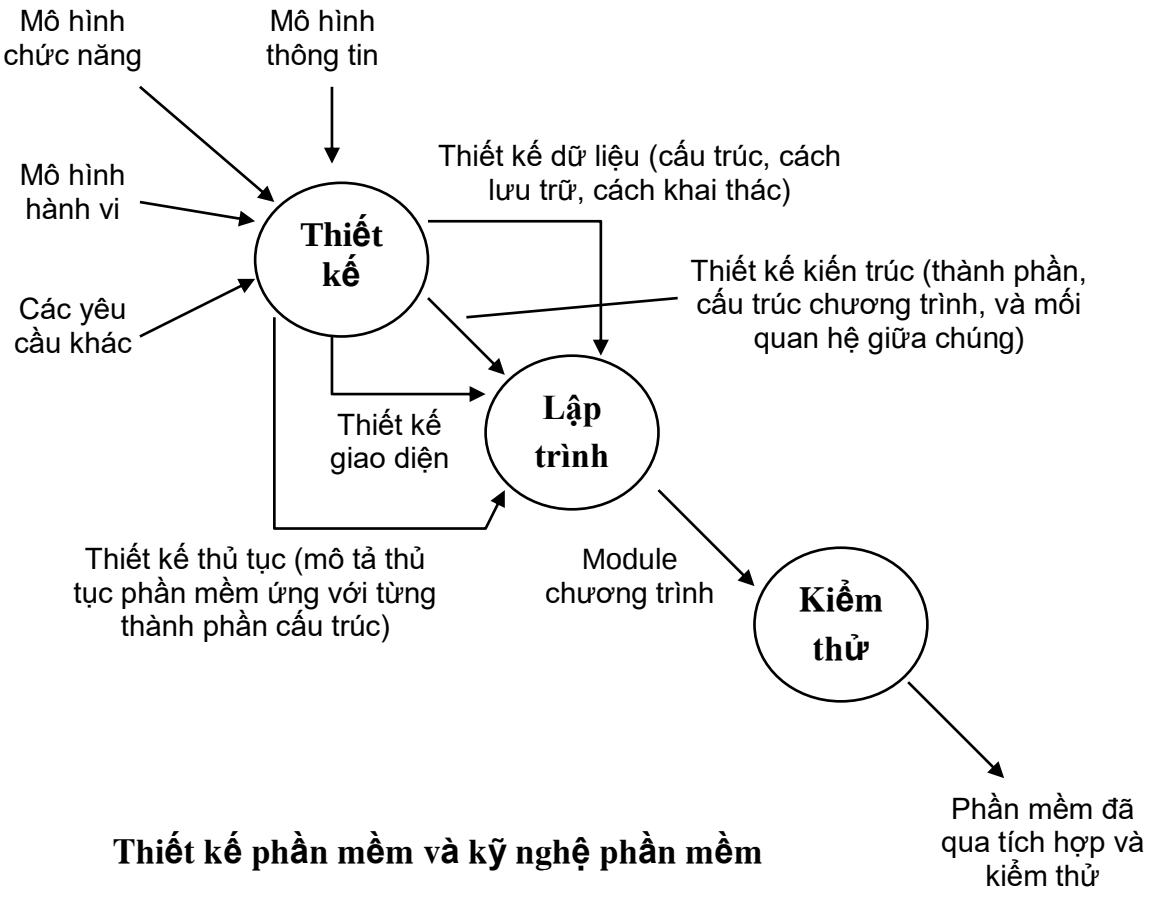
Thiết kế là một quá trình sáng tạo, đòi hỏi kinh nghiệm và sự tinh nhanh của người thiết kế.

Thiết kế phải được thực hành và học bằng kinh nghiệm, bằng khảo sát các hệ thống đang tồn tại, không thể học bằng sách vở (nói đúng ra là không đủ).

Thiết kế phần mềm là một quá trình chuyển hoá các yêu cầu thành một biểu diễn phần mềm. Bước đầu, biểu diễn mô tả toàn bộ về phần mềm. Việc làm mịn tiếp theo sau dẫn tới một biểu diễn thiết kế gần với chương trình gốc.

Thiết kế phần mềm nằm ở trung tâm kỹ thuật của tiến trình kỹ nghệ phần mềm và được áp dụng bất kể khuôn cảnh kỹ nghệ được sử dụng (thác nước, xoáy ốc, bản mẫu, thế hệ thứ 4 - 4GT, …). Một khi các yêu cầu về phần mềm đã được phân tích và đặc tả thì thiết kế phần mềm là một trong ba hoạt động kỹ thuật - *thiết kế, lập trình, kiểm thử* - những hoạt động cần để xây dựng và kiểm chứng phần mềm. Từng hoạt động này biến đổi thông tin theo cách cuối cùng tạo ra phần mềm máy tính hợp lệ.

Luồng thông tin trong giai đoạn kỹ thuật này của tiến trình kỹ nghệ phần mềm được minh hoạ trong sơ đồ sau:



Thiết kế phần mềm và kỹ nghệ phần mềm

Các yêu cầu phần mềm, được biểu thị bởi các mô hình thông tin, chức năng và hành vi là cái vào cho bước thiết kế. Bằng việc sử dụng một trong số các phương pháp thiết kế, bước thiết kế tạo ra thiết kế dữ liệu, thiết kế kiến trúc và thiết kế thủ tục.

- ***Thiết kế dữ liệu*:** Chuyển mô hình lĩnh vực thông tin đã tạo ra trong bước phân tích thành cấu trúc dữ liệu sẽ cần cho việc cài đặt phần mềm.

- ***Thiết kế kiến trúc*:** Định nghĩa ra mối quan hệ giữa các thành phần cấu trúc chính của chương trình.

"Hình mẫu thiết kế" có thể được dùng để đạt tới các yêu cầu đã được xác định cho hệ thống, và những ràng buộc ảnh hưởng tới cách mà các hình mẫu thiết kế kiến trúc này có thể được áp dụng. Biểu diễn thiết kế kiến trúc - khuôn khổ của hệ thống dựa trên máy tính - có thể được suy ra từ đặc tả hệ thống, mô hình phân tích và tương tác của các hệ con được định nghĩa bên trong mô hình phân tích.

• ***Thiết kế giao diện:*** Mô tả cho cách phần mềm trao đổi với chính nó, với hệ thống liên tác với nó, và với người dùng nó. Giao diện bao gồm một luồng thông tin (như dữ liệu và / hoặc điều khiển) và các kiểu hành vi đặc biệt. Do đó, các biểu đồ luồng dữ liệu và điều khiển cung cấp nhiều thông tin cần cho thiết kế giao diện.

• ***Thiết kế thủ tục***: Biến đổi các thành phần cấu trúc của kiến trúc phần mềm thành mô tả thủ tục cho các cấu phần phần mềm. Chương trình gốc được sinh ra rồi việc kiểm thử được tiến hành để tích hợp và làm hợp lệ.

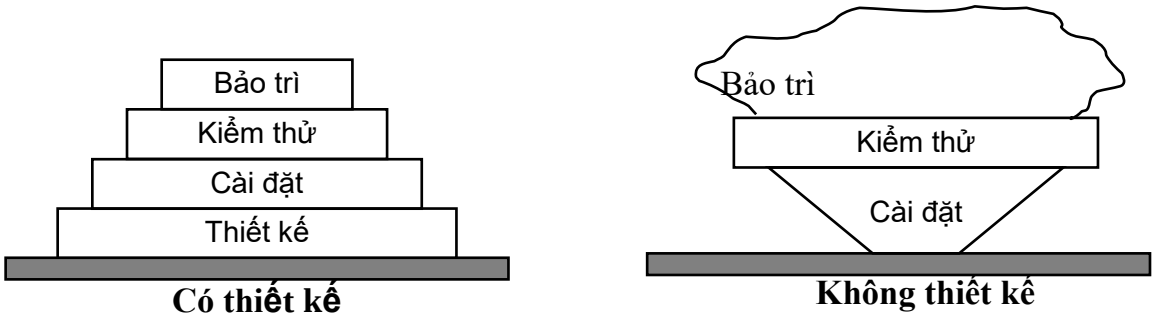
Trong khi thiết kế chúng ta ra các quyết định mà cuối cùng sẽ ảnh hưởng tới sự thành công của việc xây dựng phần mềm và điều quan trọng là ảnh hưởng tới sự dễ dàng bảo trì nó. Nhưng tại sao thiết kế lại quan trọng?

Tầm quan trọng của thiết kế phần mềm có thể được phát biểu bằng một từ - ***chất lượng***. Thiết kế là nơi chất lượng được nuôi dưỡng trong việc phát triển phần mềm: cung cấp cách biểu diễn phần mềm có thể được xác nhận về chất lượng, là cách duy nhất mà chúng ta có thể chuyển hoá một cách chính xác các yêu cầu của khách hàng thành sản phẩm hay hệ thống phần mềm cuối cùng. Thiết kế phần mềm phục vụ như một nền tảng cho mọi bước kỹ nghệ phần mềm và bảo trì:

Tầm quan trọng của thiết kế:

• Không có thiết kế, ta có nguy cơ dựng lên một hệ thống không ổn định - một hệ thống sẽ thất bại khi có một thay đổi nhỏ; một hệ thống khó có thể mà thử được; một hệ thống không thể nào xác định được chất lượng chừng nào chưa đến cuối tiến trình kiểm thử, khi thời gian còn rất ngắn mà không ít tiền đã phải chi ra.

• Thiết kế tốt là chìa khoá cho công trình hữu hiệu, không thể hình thức hoá quá trình thiết kế trong bất kỳ một công trình nào. Chú ý rằng RAISE chỉ là một phương pháp nghiêm ngặt để viết ra thiết kế, phát triển nó, kiểm tra nó chứ tuyệt nhiên không phải là một phương pháp hình thức để phát triển thiết kế.



Thiết kế phần mềm trải qua một số giai đoạn sau:

Giai đoạn 1: Nghiên cứu và hiểu ra vấn đề. Không hiểu rõ vấn đề thì không có thể thiết kế được phần mềm hữu hiệu.

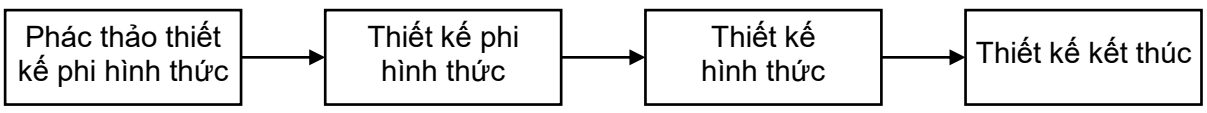
Giai đoạn 2: Làm sáng tỏ các đặc điểm lớn của một hoặc một vài giải pháp có thể. Việc chọn giải pháp phụ thuộc vào kinh nghiệm của người thiết kế; phụ thuộc vào các thành phần có thể tái sử dụng và phụ thuộc vào sự đơn giản của các giải pháp trước đó. Kinh nghiệm cho thấy, nếu các nhân tố là tương tự thì nên chọn giải pháp đơn giản nhất.

Giai đoạn 3: Mô tả từng điều trừu tượng (chưa rõ ràng) trong giải pháp. Trước khi tạo ra các tư liệu chính thức, người thiết kế nên thấy rằng cần phải xây dựng một mô tả ban đầu sơ khai rồi chi tiết hoá nó. Các sai sót và khiếm khuyết trong mức thiết kế ban đầu sẽ được phát hiện và được điều chỉnh cho phù hợp tại các mức chi tiết thiết kế tiếp theo.

Quá trình khắc phục khiếm khuyết này sẽ được lặp lại cho từng phần trừu tượng từ mức thiết kế ban đầu cho đến khi một đặc tả thiết kế chi tiết cho từng phần trừu tượng kết thúc. *Nên phân chia ra các phần nhỏ ứng với thiết kế rồi tổ hợp lại, sao cho việc mô tả chi tiết các phần nhỏ đó chỉ trong khoảng một trang giấy*.

7.2. Quá trình thiết kế (Design process)

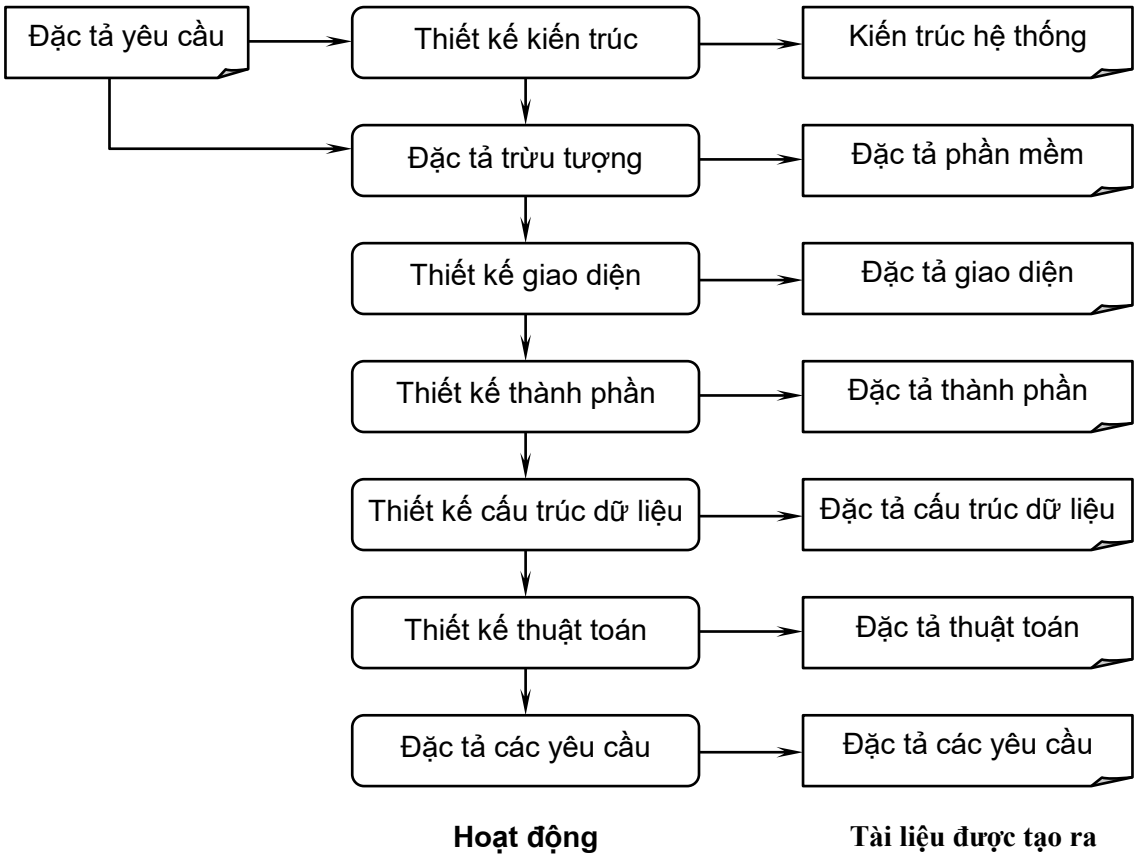
Quá trình thiết kế là quá trình tăng cường hình thức hoá trong sự tiến triển của thiết kế và phải luôn quay trở lại các thiết kế đúng đắn ít hình thức (từ "hình thức" ở đây có nghĩa là mang tính mô tả được hệ thống trong thực tế) có trước đây của quá trình đó. Nhà thiết kế phải bắt đầu với một bản phác thảo hết sức không hình thức rồi sau đó tinh chế nó, thêm vào đó các thông tin để là cho thiết kế trở nên hình thức hơn. Quá trình thiết kế thể hiện như sau:



Quan hệ giữa thiết kế và đặc tả là rất chặt chẽ. Mặc dầu quá trình đưa ra một đặc tả yêu cầu được xem như là một phần tử cơ bản của hợp đồng là một hoạt động riêng biệt, song việc hình

thức hoá đặc tả yêu cầu hẳn là một phần của quá trình thiết kế. Thực tế, người làm thiết kế sẽ lặp đi lặp lại giữa đặc tả và thiết kế.

Quá trình thiết kế liên quan mật thiết đến việc mô tả hệ thống ở một số mức trừu tượng khác nhau. Khi một thiết kế được phân chia thành nhiều thành phần thì người ta thường phát hiện ra được những sai xót ở giai đoạn trước. Do đó phải quay trở lại để tinh chế. Thông thường thì người ta bắt đầu giai đoạn sau ngay trước khi giai đoạn trước kết thúc đơn giản là để lui quá trình tinh chế. Hình vẽ dưới đây nêu các hoạt động của quá trình thiết kế và các sản phẩm của nó. Các giai đoạn là khá tuỳ ý nhưng nó làm cho quá trình thiết kế trở nên nhìn thấy được và từ đó dễ quản lý được.



Các hoạt động thiết kế và sản phẩm của thiết kế.

Thành quả của mỗi hoạt động thiết kế là một bản đặc tả. Đặc tả này có thể là một đặc tả trừu tượng, hình thức và được tạo ra để làm rõ các yêu cầu, nó cũng có thể là một đặc tả về một thành phần nào đó của hệ thống phải được thực hiện như thế nào. khi quá trình thiết kế tiến triển thì các yêu cầu ngày càng được bổ sung vào bản đặc tả đó. Các kết quả cuối cùng là các đặc tả về thuật toán và các cấu trúc dữ liệu được dùng làm cơ sở cho việc thực hiện hệ thống.

Thực tế, các hoạt động thiết kế diễn ra song song với các sản phẩm thiết kế khác nhau. Các sản phẩm này lại được triển khai ở các mức chi tiết khác nhau trong diễn biến của quá trình thiết kế.

Các hoạt động cốt yếu trong việc thiết kế một hệ thống phần mềm lớn

1. Thiết kế kiến trúc: Các hệ con tạo nên hệ tổng thể và các quan hệ của chúng là được phân hoạch rõ ràng và ghi thành tài liệu.

2. Đặc tả trừu tượng: Đối với mỗi hệ con, một đặc tả trừu tượng các dịch vụ mà nó cung cấp và các ràng buộc phải tuân theo cũng được hỗ trợ.

3. Thiết kế giao diện: ở đây bạn đọc không nên hiểu "giao diện" chỉ là những gì hiển thị trên màn hình, mà phải hiểu rằng đó có thể là tương tác giữa các thành phần trong hệ thống với

nhau. Giao diện với từng hệ con khác cũng được thiết kế và ghi thành tài liệu. Đặc tả giao diện không được mơ hồ và cho phép sử dụng hệ con đó mà không cần biết đến những gì được diễn ra bên trong của hệ con đó (theo kiểu "hộp đen").

4. Thiết kế các thành phần: Các dịch vụ được cung cấp bởi hệ con được phần chia thành các thành phần hợp thành của hệ con đó.

5. Thiết kế cấu trúc dữ liệu: Các cấu trúc dữ liệu được dùng trong việc thực hiện hệ thống được thiết kế chi tiết và được đặc tả ở đây.

6. Thiết kế thuật toán: Các cách thức (phương pháp xử lý) được dùng để cung cấp cho các dịch vụ được thiết kế chi tiết và được đặc tả.

Quá trình này được lặp lại cho mỗi hệ con sao cho đến khi các thành phần hợp thành được xác định một cách rõ ràng và đều có thể chuyển đổi (ánh xạ) một cách trực tiếp vào các thành phần của ngôn ngữ lập trình, chẳng hạn như các gói (packets), các thủ tục (procedures) và các hàm (functions).

Phương pháp tiếp cận thường xuyên được khuyến khích sử dụng là phương pháp tiếp cận *từ trên xuống* (top down): Vấn đề lớn được phân chia một cách đệ quy thành các vấn đề con cho đến khi các vấn đề dễ giải quyết được xác định rõ ràng. Trong quá trình này người thiết kế không nhất thiết phải phân rã tất cả các thành phần trừu tượng (nghĩa là vấn đề này còn phức tạp mà cách giải quyết là chưa xác định rõ) khi mà bằng kinh nghiệm họ đã biết chắc chắn rằng có thể hoàn toàn xây dựng được. Do đó họ có thể tập trung sức lực vào các thành phần đáng xét nhất.

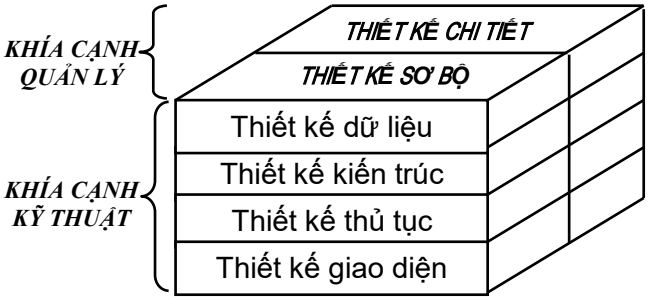
Chú ý rằng khi mà phương pháp hướng đối tượng được chấp nhận thì phương pháp từ trên xuống sẽ ít hiệu quả. Khi đó người thiết kế sử dụng các đối tượng sẵn có để làm khung thiết kế.

Theo quan điểm quản lý dự án, thiết kế phần mềm được tiến hành theo 2 bước:

Bước 1- Thiết kế sơ bộ: Quan tâm tới việc chuyển hoá các yêu cầu thành kiến trúc dữ liệu và các thành phần phần mềm.

Bước 2- Thiết kế chi tiết: Tập trung vào việc làm mịn biểu diễn kiến trúc để dẫn tới cấu trúc dữ liệu chi tiết và biểu diễn các quy trình tính toán và xử lý của phần mềm.

Trong phạm vi thiết kế sơ bộ và chi tiết, có xuất hiện một số hoạt động thiết kế khác nhau. Bên cạnh việc thiết kế dữ liệu, kiến trúc và thủ tục, nhiều ứng dụng hiện đại có hoạt động thiết kế giao diện phân biệt. Thiết kế giao diện lập ra cách bố trí và cơ chế tương tác người-máy (HCI – humen computer interface). Mối quan hệ giữa các khía cạnh kỹ thuật và quản lý của thiết kế được minh hoạ trong hình vẽ dưới đây.



Quan hệ giữa khía cạnh kỹ thuật và khía cạnh quản lý trong thiết kế

Việc mô tả thiết kế.

Thiết kế phần mềm là một mô hình của thế giới thực mô tả các thực thể và các mối quan hệ của chúng với nhau.

Thiết kế cần được mô tả sao cho đạt được ở mức độ sau:

- Làm cơ sở cho việc thực hiện chi tiết.
- Làm phương tiện liên lạc giữa các nhóm thiết kế các hệ con.
- Cung cấp đầy đủ thông tin cho người bảo trì hệ thống.

Người ta thường dùng các khái niệm đồ thị, các ngôn ngữ mô tả chương trình hoặc văn bản không hình thức để tạo dựng tài liệu thiết kế.

7.3. Các nguyên tắc thiết kế (Design Principles)

Phương pháp cấu trúc được dùng rộng rãi trong những năm đầu của những năm 1980. Nó đã được dùng thành công trong nhiều dự án lớn, nó làm giảm giá thành một cách đáng kể, sử dụng được các khái niệm chuẩn và đảm bảo rằng việc thiết kế tuân theo một chuẩn nhất định. Các công cụ CASE (Computer Aided Software Engineering – thiết kế phần mềm có máy tính hỗ trợ) đã được dùng để trợ giúp cho phương pháp này.

Các phương pháp thiết kế thường trợ giúp một vài cách nhìn nhận hệ thống như sau:

- *Nhìn nhận cấu trúc*: Cho cái nhìn cấu trúc thông qua lược đồ cấu trúc.
- *Nhìn nhận quan hệ thực thể:* Mô tả cấu trúc dữ liệu logic thường dùng, đề cập đến đặc tả dữ liệu quan hệ thực thể.
- *Nhìn nhận dòng dữ liệu:* Về lược đồ dòng dữ liệu.

Người ta còn dùng lược đồ chuyển trạng thái để bổ sung cho phương pháp trên.

Để đảm bảo chất lượng cho một biểu diễn thiết kế, cần có các tiêu chuẩn cho thiết kế tốt. Song về mặt phương pháp, chúng ta đưa ra các hướng dẫn sau:

1. Thiết kế nên đưa ra cách tổ chức theo cấp bậc để dùng cách kiểm soát thông minh trong số các thành phần phần mềm.
2. Thiết kế nên theo các module, tức là phần mềm nên được phân hoạch một cách logic thành các thành phần thực hiện chức năng hay các chức năng con xác định.
3. Thiết kế nên chứa cách biểu diễn phân biệt và tách biệt giữa dữ liệu và thủ tục.
4. Thiết kế nên dẫn tới các module (như chương trình con hay thủ tục) nêu ra các đặc trưng chức năng đặc biệt.
5. Thiết kế nên dẫn đến giao diện là rút gọn độ phức tạp của việc nối ghép lại giữa các module và với môi trường bên ngoài.
6. Thiết kế nên được hướng theo cách dùng một phương pháp lặp lại được điều khiển bởi thông tin có trong phân tích các yêu cầu phần mềm.

Các đặc trưng trên của một thiết kế tốt có được khi thực hiện đúng tiến trình thiết kế kỹ nghệ phần mềm thông qua việc áp dụng các nguyên lý thiết kế cơ bản, phương pháp luận hệ thống và xét duyệt thấu đáo.

Như vậy, mỗi phương pháp thiết kế phần mềm đều đưa vào những phương pháp trực cảm và lý pháp duy nhất, cũng như một cách nhìn thiển cận thế nào đó về cái gì đặc trưng cho chất lượng thiết kế

Tuy vậy mỗi phương pháp đều có những đặc trưng sau:

1. Một cơ chế để chuyển hoá từ biểu diễn miền thông tin thành biểu diễn thiết kế
2. Một kí pháp để biểu diễn các thành phần chức năng và giao diện của chúng
3. Các trực cảm để làm mịn và phân hoạch
4. Các hướng dẫn về định giá chất lượng

Bất kể phương pháp luận thiết kế nào được dùng, công trình sư phần mềm phải áp dụng một tập các khái niệm nền tảng cho thiết kế dữ liệu, kiến trúc và thủ tục:

- Trừu tượng
- Modul
- Kiến trúc phần mềm.
- Cấp bậc điều khiển

- Cấu trúc dữ liệu
- Thủ tục phần mềm
- Che dấu thông tin

Thiết kế hướng chức năng

Hệ thống được thiết kế theo quan điểm chức năng, bắt đầu ở mức cao nhất, sau đó tinh chế dần dần để thành thiết kế chi tiết hơn. Trạng thái của hệ thống là tập trung và được chia sẻ cho các chức năng thao tác trên trạng thái đó.

Ban đầu, ta coi yêu cầu mức cao nhất của hệ thống là một chức năng duy nhất cần phải thực hiện. Sau đó, ta trả lời cho câu hỏi "Để thực hiện chức năng trên thì cần phải làm các công việc gì?" – từ *công việc* trong câu hỏi trên được coi là chức năng con của chức năng trên. Thực hiện xong các chức năng con cũng là thực hiện xong chức năng cha. Hệ thống được phân rã dần dần, và được làm mịn. Hình ảnh của hệ thống sẽ được xây dựng theo các bước trên.

Thiết kế hướng đối tượng

Hệ thống được nhìn nhận như một bộ các đối tượng (chứ không phải là một tập hợp các chức năng). Hệ thống được phân tán, mỗi đối tượng có thông tin và trạng thái của riêng nó. Đối tượng là một bộ các thuộc tính xác định trạng thái của đối tượng đó và các phép toán thực hiện trên đó. Mỗi đối tượng là một khách thể của một lớp mà *lớp được xác định bởi thuộc tính và các phép toán* của nó. Nó được thừa kế từ một vài lớp đối tượng cấp cao hơn, sao cho định nghĩa nó chỉ cần nêu đủ các khác nhau giữa nó và các lớp cao hơn nó. Các đối tượng liên lạc với nhau chỉ bằng trao đổi các *thông báo.* Trong thực tế, hầu hết các liên lạc được thực hiện giữa các đối tượng bằng cách nối đối tượng này với một thủ tục, mà thủ tục này kết hợp với một đối tượng khác.

Thiết kế hướng đối tượng dựa trên ý tưởng che dấu thông tin. Gần đây theo cách thiết kế này, người ta đã phát triển nhiều hệ thống cấu tạo bởi nhiều thành phần độc lập và có tương tác với nhau.

Sự thật, các hệ phần mềm lớn phức tạp đến mức mà người ta đã dùng các phương pháp tiếp cận khác nhau trong việc thiết kế các thành phần khác nhau trong hệ thống. Chẳng có một chiến lược tốt nhất nào cho các dự án lớn. Các cách tiếp cận hướng chức năng và hướng đối tượng là bổ sung hỗ trợ cho nhau chứ không phải là loại bỏ nhau. Kỹ sư phần mềm sẽ chọn ra cách tiếp cận thích hợp nhất trong từng giai đoạn thiết kế. Nhìn ở mức tổng thể thì hệ thống như là một bộ các đối tượng (chứ không phải là một bộ các chức năng), cho nên ở mức trừu tượng cao thì cách tiếp cận hướng đối tượng là thích hợp hơn. Đến mức chi tiết thì một cách tự nhiên hơn nên xem chúng là các chức năng tương tác giữa các đối tượng. Sau đó mỗi đối tượng lại được phân giải thành các thành phần, tức là có thể xem nó như là một hệ con.

Rất nhiều hệ thống, đặc biệt là hệ thống thời gian thực được nhúng (vào một hệ thiết bị vật chất có thực) được cấu tạo như một hệ gồm một bộ các quá trình hoạt động song song và có liên lạc với nhau. Các hệ này thường phải tuân theo các ràng buộc nghiêm ngặt về thời gian, mà các phần cứng thường phản ứng tương đối chậm, chỉ có cách tiếp cận nhiều bộ xử lý hoạt động song song mới có thể hoàn thành được yêu cầu về thời gian.

Các chương trình tuần tự là dễ thiết kế, thực hiện và kiểm tra và thử nghiệm hơn là các hệ thống song song. Sự phụ thuộc về thời gian giữa các quá trình là khó hình thức hoá, khó khống chế và thử nghiệm.

Do đó, quá trình thiết kế nên được xem như là một hoạt động gồm 2 giai đoạn:

Giai đoạn 1: Minh định cấu trúc thiết kế logic, cụ thể là các thành phần của hệ thống và các mối quan hệ giữa chúng. Có thể dùng cách nhìn hướng chức năng hoặc cách nhìn hướng đối tượng.

Giai đoạn 2: Thực hiện cấu trúc đó trong dạng có thể thực hiện được. Giai đoạn này đôi khi được gọi là thiết kế chi tiết và đôi khi là lập trình. Chắc rằng sự quyết định về tính song song nên là ở giai đoạn này chứ không phải là các giai đoạn sớm hơn trong quá trình thiết kế.

7.4. Thiết kế dữ liệu

7.4.1. Tổng quan

Mục tiêu chính của thiết kế dữ liệu là mô tả cách thức tổ chức lưu trữ các dữ liệu của phần mềm. Có hai dạng lưu trữ chính mà người thiết kế cần phải cân nhắc và lựa chọn:

+ Lưu trữ dưới dạng tệp tin: chỉ thích hợp với một số phần mềm đặc thù (cờ tướng, trò chơi…). Đặc điểm chung của các phần mềm này là chú trọng nhiều vào xử lý, hình thức giao diện mà không chú trọng đến việc lưu trữ lại các thông tin được tiếp nhận trong quá trình sử dụng phần mềm

+ Lưu trữ dưới dạng cơ sở dữ liệu: đây là cách tiếp cận thông dụng, quan tâm nhiều đến việc lưu trữ thông tin, cách thức xử lý dữ liệu.

7.4.2. Kết quả thiết kế

- Thông tin tổng quát

+ Danh sách các bảng dữ liệu

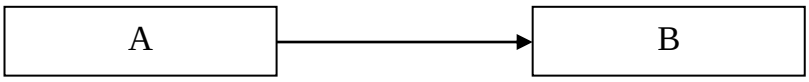
+ Danh sách các liên kết: mối quan hệ giữa các bảng dữ liệu

- Thông tin chi tiết

+ Danh sách các thuộc tính của từng thành phần (bảng dữ liệu)

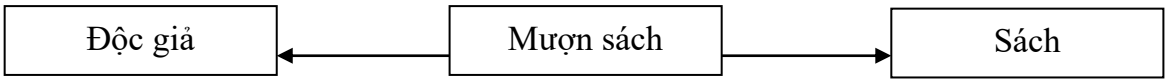
+ Danh sách các miền giá trị toàn vẹn của các thành phần: các quy định về tính hợp lệ của các thông tin được lưu trữ.

Dùng sơ đồ logic để biểu diễn sơ đồ liên kết giữa các bảng dữ liệu:



Với mỗi liên kết xác định duy nhất một mũi tên, như với sơ đồ trên, mỗi phần tử của A sẽ xác định duy nhất một phần tử của B, mỗi phần tử của B có thể tương ứng với nhiều phần tử của A.

Ví dụ:

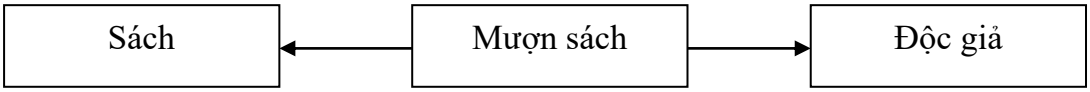


7.4.3. Quá trình thiết kế

- **Phương pháp:**
  + Phương pháp trực tiếp
  + Phương pháp gián tiếp
- **Các bước thiết kế:**
  + Thiết kế với tính đúng đắn
  + Thiết kế với tính tiến hoá
  + Thiết kế với tính hiệu quả
  + Thiết kế với yêu cầu hệ thống
- **Thiết kế với tính đúng đắn:**
  + Đảm bảo đầy đủ và chính xác về mặt ngữ nghĩa các thông tin liên quan đến các công việc trong phần mềm
  + Các thống tin phục vụ cho các yêu cầu chất lượng khác của phần mềm không được xét ở đây.

    Ví dụ:

    Sơ đồ logic các bảng quan hệ cho hệ thống mượn sách:



Chi tiết các bảng:

1. ***DOCGIA*** *(MaDocGia, LoaiDocGia, TenDocGia, NgaySinh, DiaChi, DienThoai, SoCMND, SoThe, NgayLapThe, NgayHetHan, TienDatCoc, TrangThai).*
2. ***SACH*** *(MaSach, TenSach, NhaXuatBan, NgonNgu, LoaiSach, TacGia, SoTrang, NamXuatBan, NgayNhap).*
3. ***MUONSACH*** *(SoPhieu, NgayMuon, NgayTra, MaSach, MaDocGia, TienPhat, TinhTrang, GhiChu).*

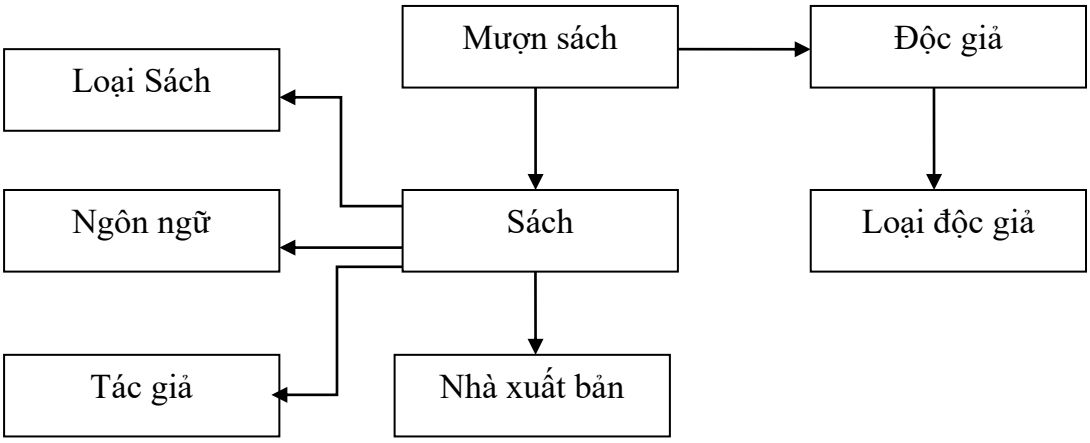
- **Thiết kế với tính tiến hoá:**

  \+ Vẫn đảm bảo tính đúng đắn nhưng thoả mãn thêm yêu cầu tính tiến hoá

  \+ Cần chú ý đảm bảo tính đúng đắn khi cải tiến sơ đồ logic

  Ví dụ:

  Sơ đồ logic các bảng quan hệ cho hệ thống mượn sách:



Chi tiết các bảng:

1. ***LOAIDOCGIA*** *(MaLDG, TenLDG).*
2. ***DOCGIA*** *(MaDocGia, MaLDG, TenDocGia, NgaySinh, DiaChi, DienThoai, SoCMND, SoThe, NgayLapThe, NgayHetHan, TienDatCoc, TrangThai).*
3. ***LOAISACH*** *(MaLS, TenLS).*
4. ***NHAXUATBAN*** *(MaNXB, TenNXB, DiaChi).*
5. ***NGONNGU*** *(MaNN, TenNN).*

6. ***TACGIA*** *(MaTG, TenTG).*
7. ***SACH*** *(MaSach, TenSach, MaNXB, MaNN, MaLS, MaTG, SoTrang, NamXuatBan, NgayNhap).*
8. ***MUONSACH*** *(SoPhieu, NgayMuon, NgayTra, MaSach, MaDocGia, TienPhat, TinhTrang, GhiChu).*

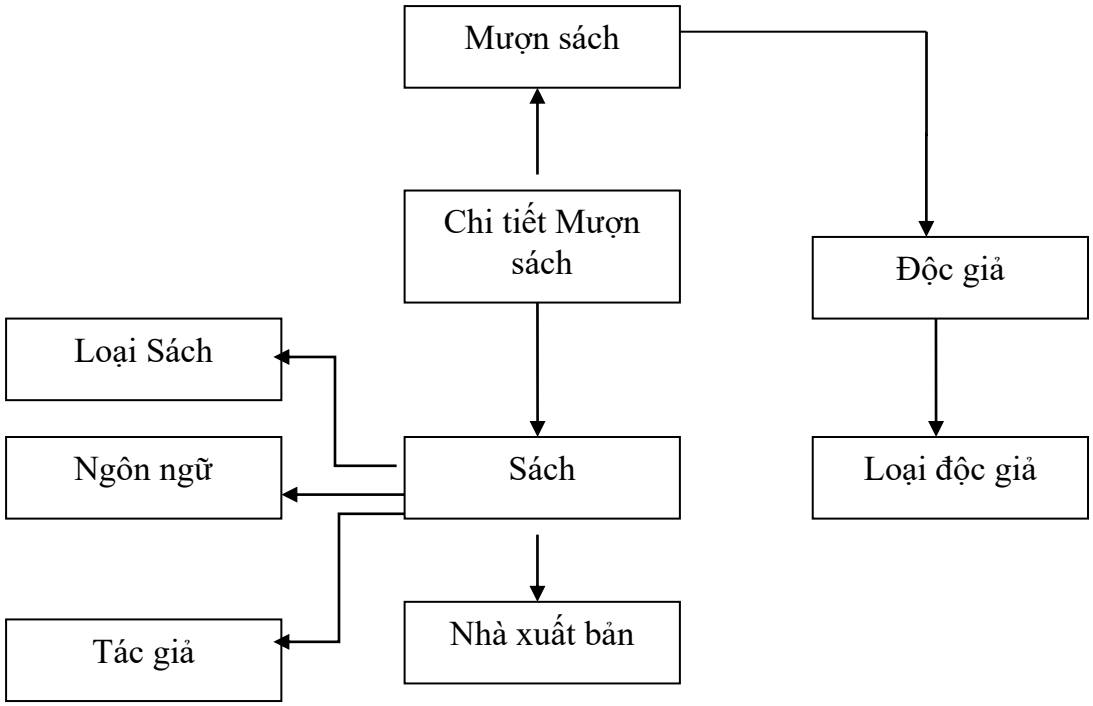
- **Thiết kế với tính hiệu quả:**

  + Vẫn đảm bảo tính đúng đắn, tính tiến hoá nhưng thoả mãn thêm yêu cầu tính hiệu quả (truy xuất nhanh, lưu trữ tối ưu)

  + Cần chú ý đảm bảo tính đúng đắn và tiến hoá khi cải tiến sơ đồ logic

  Ví dụ:

  Sơ đồ logic các bảng quan hệ cho hệ thống mượn sách:



  Chi tiết các bảng:

1. ***LOAIDOCGIA*** *(MaLDG, TenLDG).*
2. ***DOCGIA*** *(MaDocGia, MaLDG, TenDocGia, NgaySinh, DiaChi, DienThoai, SoCMND, SoThe, NgayLapThe, NgayHetHan, TienDatCoc, TrangThai).*
3. ***LOAISACH*** *(MaLS, TenLS).*
4. ***NHAXUATBAN*** *(MaNXB, TenNXB, DiaChi).*
5. ***NGONNGU*** *(MaNN, TenNN).*

6. ***TACGIA*** *(MaTG, TenTG).*
7. ***SACH*** *(MaSach, TenSach, MaNXB, MaNN, MaLS, MaTG, SoTrang, NamXuatBan, NgayNhap).*
8. ***MUONSACH*** *(SoPhieu, NgayMuon, MaDocGia).*
9. ***CHITIET_MUONSACH*** *(SoPhieu, MaSach, NgayTra, SoTienPhat, TinhTrang, GhiChu).*

- **Thiết kế với yêu cầu hệ thống:**

  + Vẫn đảm bảo tính đúng đắn, tính tiến hoá, tính hiệu quả nhưng thoả mãn thêm yêu cầu hệ thống (phân quyền, cấu hình phần cứng, môi trường phần mềm…).

  + Cần chú ý đảm bảo tính đúng đắn, tính tiến hoá, tính hiệu quả khi cải tiến sơ đồ logic.

7.5. Thiết kế giao diện

7.5.1. Tổng quan

Thiết kế giao diện cần nắm bắt các yếu tố chính sau:

1. Hồ sơ cá nhân người dùng: biết họ là ai, mục đích của người dùng là gì; kỹ năng và kinh nghiệm của người dùng, nhu cầu của họ.
2. Mượn những ứng xử từ những hệ thống quen thuộc đối với người dùng.
3. Cho người dùng thấy rõ các chức năng một cách sẵn sàng.
4. Ứng xử của chương trình từ trong ra ngoài phải kết dính, gắn kết.
5. Thay đổi trong ứng xử nên tương phản với diện mạo cua chương trình.
6. Shortcut: cung cấp cả cách thức cụ thể và tóm tắt các tác vụ được làm.
7. Tính tập trung: một số giao diện (GUI) được tập trung chú ý nhiều hơn.
8. Ngữ pháp: thông qua giao diện biết số luật tao tác.
9. Trợ giúp, độ an toàn, hạn chế ngữ cảnh người dùng, giao diện đẹp,…

7.5.2. Kết quả thiết kế

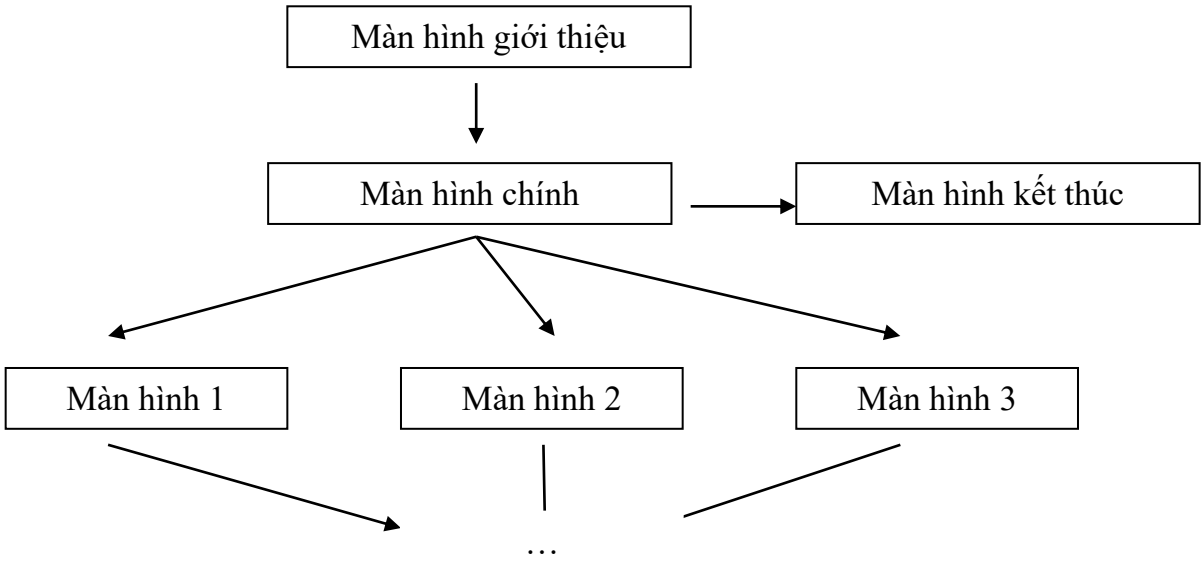
- Màn hình giao diện:

Màn hình giao diện là một trong các hình thức giao tiếp giữa người sử dụng và phần mềm khi họ thực hiện các công việc của mình trên máy tính. Mục tiêu chính của thiết kế giao diện là mô tả hệ thống các màn hình giao diện này.

- Kết quả thiết kế giao diện:

+ Sơ đồ màn hình: mô tả các thông tin tổng quát về hệ thống các màn hình cùng với quan hệ về việc chuyển điều khiển giữa chúng.

+ Mô tả chi tiết từng màn hình: mô tả chi tiết nội dung, hình hức trình bày và các thao tác mà người dùng có thể thực hiện trên từng màn hình.



- Mô tả màn hình giao diện:

Các thông tin cần mô tả một màn hình giao diện bao gồm:

+ Tên màn hình: tên công việc tương ứng muốn thực hiện trên máy tính

+ Nội dung: cấu trúc các thành phần bên trong màn hình. Cấu trúc thành phần bên trong màn hình có thể chia làm 3 loại:

✓ Thành phần dữ liệu:

- Thông tin nhập liệu
- Thông tin kết xuất

✓ Thành phần xử lý: các nút điều khiển cho phép người dùng yêu cầu phần mềm thực hiện một xử lý nào đó.

✓ Hình thức trình bày: cách thức bố trí, sắp xếp các thành phần tron màn hình (vị trí, màu sắc, kích thước…).

7.5.3. Phân loại màn hình giao diện

- Màn hình chính: cho phép người dùng lựa chọn công việc mong muốn thực hiện trên máy ính từ danh sách các công việc.
- Màn hình nhập liệu: cho phép người dùng lưu trữ, xử lý các thông tin từ thế giới thực.

- Màn hình kết quả: trình bày cho người dùng kết quả thực hiện một công việc nào đó
- Màn hình thông báo: thông báo, nhắn nhở người dùng trong quá trình thực hiện một công việc nào đó.
- Màn hình tra cứu: cho phép tìm kiếm thông tin đã được lưu trữ với các tiêu chí tìm kiểm khác nhau.

7.5.4. Thiết kế màn hình chính

- Ý nghĩa sử dụng: Màn hình chính là màn hình cho phép người dùng chọn được công việc mà họ muốn thực hiện với phần mềm. Thông thường mỗi phần mềm có một màn hình chính.
- Nội dung: Danh sách các công viêc có thể thực hiện với phần mềm
- Hình thức trình bày:

  Phím nóng: cho phép chọn danh sách công việc với người dùng chuyên nghiệp. Thông thường nó không được sử dụng riêng rẽ mà phải kết hợp với các hình thức khác.

  Thực đơn: nhóm từng công việc theo chức năng, đây là dạng sử dụng thông dụng nhất.

  Biểu tưởng: công việc thể hiện trực quan thông qua biểu tưởng (ký hiệu hay hình ảnh tượng trưng cho công việc). Hình thức này cũng thường được kết hợp với các hình thức khác.

  Sơ đồ: dùng sơ đồ để thể hiện sự trực quan các đối tượng chính, được thể hiện qua các thao tác trực tiếp trên sơ đồ.

  Tích hợp: sự kết hợp của nhiều hình thức với nhau.
- Thao tác người dùng: lựa chọn công việc cần thực hiện trên màn hình chính

7.5.5. Thiết kế màn hình tra cứu

- Ý nghĩa sử dụng: Màn hình tra cứa là màn hình cho phép người dùng tìm kiếm và xem các thông tin về các đối tượng.
- Nội dung:

  Tiêu chí tra cứu: Các thông tin sử dụng cho việc tìm kiếm.

  Kết quả tra cứu: Cho biết tìm thấy hay không.

  + Các thông tin cơ bản về đối tượng tìm kiếm.

  + Các thông tin về quá trình hoạt động của đối tượng (quan hệ với các đối tượng khác).

- Hình thức trình bày:

  Tiêu chí tra cứu: Biểu thức logic, cây, tích hợp.

  Kết quả tra cứu: Thông báo, danh sách đơn, xâu các danh sách, cây danh sách.

- Thao tác người dùng:

  + Nhập các giá trị cho tiêu chí tra cứu,

  + Yêu cầu bắt đầu tra cứu,

  + Xem kết quả tra cứu.

7.5.6. Thiết kế màn hình nhập liệu

- Ý nghĩa sử dụng: Màn hình chính là màn hình cho phép người dùng thực hiện các công việc có liên quan đến ghi chép trong thế giới thực.

- Nội dung:

  Thông tin nhập liệu: Người dùng chịu trách nhiệm nhập trức tiếp các giá trị, phần mềm sẽ tiến hành kiểm tra tính hợp lệ của giá trị nhập và các quy định có liên quan.

  Thông tin tính toán: Phần mềm có trách nhiệm tính toán và xuất thông tin này trên màn hình. Thông tin này sẽ giúp việc nhập liệu thuận tiện hơn.

- Hình thức trình bày:

  Danh sách: Có dạng một danh sách trong thế giới thực (danh sách loại sách, danh sách độc giả…).

  Hồ sơ: Có dạng một hồ sơ với nhiều thông tin chi tiết (hồ sơ học sinh…).

  Phiếu: Có dạng một phiếu với nhiều dòng chi tiết (phiếu mượn, phiếu nhập, phiếu xuất…).

  Tích hợp: Sử dụng đồng thời nhiều hình thức trên.

- Thao tác người dùng:

  + Thêm mới,

  + Sửa,

  + Ghi,

  + Xoá,

  + Huỷ bỏ

  + Tìm kiếm,

+ Các nút chuyển điều khiển sang màn hình khác.

Có thể kết hợp các phím nóng để tăng tốc độ thao tác

Bài tập:

1. Thiết kế dữ liệu cho một bài toán cụ thể
2. Thiết kế màn hình cho một bài toán cụ thể

Chương 8. Kiểm thử phần mềm

8.1. Tổng quan

Kiểm thử phần mềm là tiến hành thử nghiệm để so sánh kết quả thực tế với lý thuyết.

Mục đích của kiểm thử phần mềm là tìm ra các lỗi hay khiếm khuyết phần mềm nhằm đảm bảo hiệu quả hoạt động tối ưu của phần mềm.

Bộ kiểm thử (test cases) là dữ liệu dùng để kiểm tra hoạt động của chương trình. Một bộ kiểm thử tốt là bộ có khả năng phát hiện ra lỗi của chương trình. Khi tiến hành kiểm thử, chúng ta chỉ có thể chứng minh được sự tồn tại của lỗi nhưng không chứng minh được rằng trong chương trình không có lỗi.

Nội dung của bộ kiểm thử:

+ Tên module/chức năng muốn kiểm thử
+ Dữ liệu vào:
    - Dữ liệu của chương trình: số, xâu ký tự, tệp tin,…
    - Môi trường thử nghiệm: phần cứng, hệ điều hành
    - Thứ tự thao tác (kiểm thử giao diện)
+ Kết quả mong muốn:
    - Thông thường: số, xâu ký tự, tệp tin,..
    - Màn hình, thời gian phản hồi
+ Kết quả thực tế

Không gian thử nghiệm là tập các bộ số thử nghiệm. Không gian này nói chung là rất lớn. Nếu có thể vét cạn được không gian thử nghiệm thì chắc chắn qua phép kiểm tra đơn vị sẽ không còn lỗi. Tuy nhiên điều này không khả thi trong thực tế.

Phương pháp kiểm thử là cách chọn bộ số thử nghiệm để tăng cường độ tin cậy của đơn vị cần kiểm tra. Hay nói cách khác, phương pháp kiểm thử là phân hoạch không gian thử nghiệm thành nhiều miền rồi chọn bộ số thử nghiệm đại diện cho miền đó. Như vậy cần tránh mọi bộ số thử nghiệm đều rơi vào một miền kiểm tra.

8.2. Yêu cầu đối với kiểm thử

– Tính lặp lại:

+ Kiểm thử phải lặp lại được (kiểm tra xem lỗi đã được sửa hay chưa)

+ Dữ liệu/trạng thái phải mô tả được

- Tính hệ thống: phải đảm bảo đã kiểm tra hết các trường hợp.
- Được lập tài liệu: phải kiểm soát được tiến trình/kết quả.

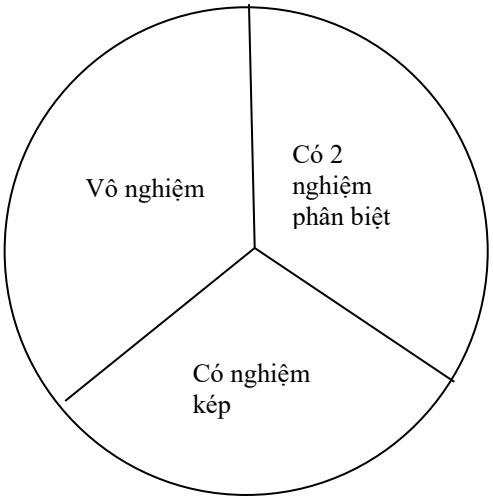
8.3. Các kỹ thuật kiểm thử

8.3.1. Phương pháp hộp đen

Phương pháp này chỉ dựa trên bản đặc tả các chức năng. Do đó, chúng chỉ chú tâm đến phát hiện các sai sót về chức năng mà không quan tâm đến cách cài đặt cụ thể. Với phương pháp này, chúng ta có thể phát hiện các sai sót, thiếu sót về chức năng; sai sót về giao diện của module, kiểm tra tính hiệu quả; phát hiện lỗi khởi tạo, lỗi kết thúc.

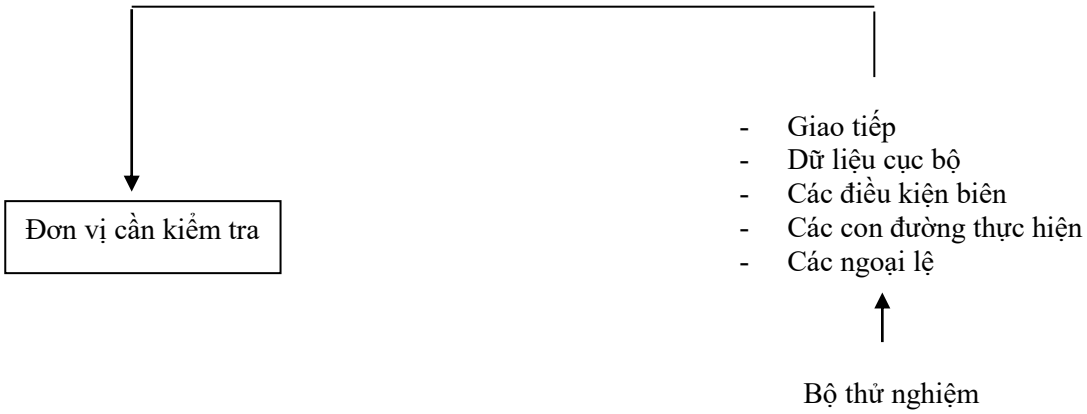
Do không thể kiểm thử mọi trường hợp trên thực tế, chúng ta chia không gian thử nghiệm dựa vào giá trị nhập xuất của đơn vị cần kiểm tra. Ứng với mỗi vùng dữ liệu chúng ta sẽ thiết kế những bộ thử nghiệm tương ứng và đặc biệt là các bộ thử nghiệm tại các giá trị biên của vùng dữ liệu.

Ví dụ: Kiểm thử chương trình giải phương trình bậc 2, ta chia không gian thử nghiệm thành 3 vùng khác nhau:



8.3.2. Phương pháp hộp trắng

Theo phương pháp này, chúng ta chia không gian thử nghiệm dựa vào cấu trúc của đơn vị cần kiểm tra.



- Kiểm tra giao tiếp của đơn vị là để đảm bảo dòng thông tin vào ra đơn vị luôn đúng (đúng giá trị, khớp kiểu…)
- Kiểm tra dữ liệu cục bộ để bảo bảo dữ liệu được lưu trữ trong đơn vị toàn vẹn trong suốt quá trình thuật giải được thực hiện.
- Kiểm tra điều kiện biên của các câu lệnh điều kiện (IF, CASE, …), vòng lặp (WHILE, FOR, …) để đảm bảo đon vị luôn chạy đúng tại các biên này.
- Kiểm tra để đảm bảo mọi con đường thực hiện phải được đi qua ít nhất một lần. Con đường thực hiện của một đơn vị là một dãy có thứ tự các câu lệnh bên trong của đơn vị đó sẽ được thực hiện khi kích hoạt đơn vị.

8.3.3. Phương pháp hộp xám

Kiểm thử hộp xám liên quan đến hiểu biết về cấu trúc dữ liệu bên trong và các thuật toán cho mục đích của các bài kiểm thử thiết kế. Khi thực hiện những bài kiểm thử với User hoặc mức độ hộp đen, Tester không nhất thiết phải truy cập vào mã nguồn của phần mềm. Ta có thể thao tác với dữ liệu đầu vào và định dạng đầu ra không xác định như hộp xám bởi vì đầu vào và đầu ra rõ ràng ở bên ngoài của "hộp đen" mà chúng được hệ thống gọi ra trong quá trình kiểm thử. Sự phân biệt này là đặc biệt quan trọng khi tiến hành kiểm thử tích hợp giữa hai Module được viết mã bởi hai nhà phát triển khác nhau, mà ở đó chỉ có các giao diện được bộc lộ ra để kiểm thử.

Tuy nhiên, các kiểm thử mà yêu cầu thay thế một kho lưu trữ dữ liệu back-end như một cơ sở dữ liệu hoặc một tập tin đăng nhập không xác định như hộp xám, người dùng sẽ không thể thay đổi các kho lưu trữ dữ liệu trong khi sản phẩm vẫn đang hoạt động bình thường.

Kiểm thử hộp xám cũng có thể bao gồm kỹ thuật đảo ngược để xác định đối tượng, giá trị biên hoặc các thông báo lỗi.

Khi biết được những khái niệm cơ bản về cách thức các phần mềm hoạt động như thế nào, Tester thực hiện kiểm thử phần mềm từ bên trong tốt hơn so với bên ngoài. thường, một Tester hộp xám sẽ được phép thiết lập một môi trường kiểm thử bị cô lập với các hoạt động như gieo một cơ sở dữ liệu. Các kiểm thử có thể quan sát trạng thái của sản phẩm được kiểm thử sau khi thực hiện hành động nhất định giống như việc thực hiện các câu lệnh SQL đối với cơ sở dữ liệu và sau đó thực hiện truy vấn để đảm bảo rằng những thay đổi dự kiến đã được phản ánh. Kiểm thử hộp xám thực hiện kịch bản kiểm thử thông minh, dựa trên thông tin hạn chế. Điều này sẽ đặc biệt áp dụng cho các kiểu xử lý dữ liệu, kể cả xử lý ngoại lệ, và cứ thế.

8.4. Các giai đoạn và chiến lược kiểm thử

Đối với các dự án phần mềm lớn, những người tham gia kiểm thử chia làm 2 nhóm:

- Nhóm thứ nhất: gồm những người tham gia trong dự án phát triển phần mềm. Nhóm này có trách nhiệm kiểm tra các đơn vị của chương trình để chắc chắn chúng thực hiện theo đúng thiết kế.
- Nhóm thứ hai: độc lập gồm các chuyên gia tin học nhưng không thuộc nhóm thứ nhất. Nhóm này có nhiệm vụ phát hiện các lỗi do nhóm thứ nhất chủ quan còn để lại.

8.4.1. Kiểm thử đơn vị

Kiểm thử đơn vị hay còn được gọi là kiểm thử thành phần, đề cập đến việc kiểm thử chức năng từng phần của mã, thường ở mức độ chức năng. Trong một môi trường hướng về đối tượng thì điều này thường là cấp độ lớp, và các kiểm thử đơn vị tối thiểu bao gồm hàm dựng và hàm hủy. Nhiều loại kiểm thử được viết bởi các nhà phát triển như họ làm việc trong mã (kiểu hộp trắng) để đảm bảo rằng từng hàm riêng biệt hoạt động đúng như kỳ vọng. Một hàm có thể có nhiều kiểm thử từ đó giúp nắm bắt được các trường hợp góc hoặc các nhánh trong Code. Kiểm thử đơn vị một mình không thể đảm bảo hết được từng chức năng của từng bộ phận trong phần phềm nhưng nó được sử dụng để đảm bảo rằng các khối kiến trúc của phần mềm hoạt động độc lập với nhau.

Kiểm thử đơn vị là một quá trình phát triển phần mềm có liên quan đến ứng dụng đồng bộ của một loạt các chiến lược phòng ngừa phát hiện lỗi và để giảm thiểu rủi ro, thời gian và chi phí. Nó được thực hiện bởi kỹ sư hay nhà phát triển trong suốt giai đoạn xây dựng của vòng đời phát triển phần mềm. Không chỉ tập trung vào việc đảm bảo chất lượng truyền thống mà phải gia tăng nó lên vì thế kiểm thử đơn vị có mục đích loại bỏ những lỗi cấu trúc trước khi mã hóa rồi mới

thúc đẩy việc quản lý chất lượng. Chiến lược này nhằm nâng cao chất lượng cũng như hiệu quả của phần mềm trong tiến trình quản lý và phát triển chung.

Tùy thuộc vào kỳ vọng của tổ chức phát triển phần mềm, kiểm thử đơn vị có thể bao gồm phân tích mã tĩnh, phân tích luồng dữ liệu, phân tích dữ liệu, đánh giá mã cân bằng, phân tích mã bao phủ và các thực hành xác nhận phần mềm khác.

Kiểm thử đơn vị sử dụng kỹ thuật hộp trắng và dựa vào hồ sơ thiết kế để xây dựng các bộ thử nghiệm sao cho khả năng phát hiện lỗi là lớn nhất.

Vì đơn vị kiểm tra không là một chương trình đầy đủ, hơn nữa đơn vị này có thể được gọi từ các đơn vị khác hoặc gọi đến các đơn vị khác nên dù chương trình đã được hoàn tất đầy đủ các đơn vị, chúng ta cũng không nên giả thiết sự tồn tại hoặc tính đúng đắn của các đơn vị khác mà phải xây dựng các module giả lập đơn vị gọi tên là **driver** và đơn vị bị gọi là **stub.**

Driver đóng vai trò như một chương trình chính nhập các bộ số thử nghiệm và gởi chúng đến đơn vị cần kiểm tra đồng thời nhận kết quả trả về của đơn vị cần kiểm tra.

Stub là chương trình giả lập thay thế các đơn vị được gọi bởi đơn vị cần kiểm tra. **Stub** thực hiện các thao tác xử lý dữ liệu đơn giản như in ấn, kiểm tra dữ liệu nhập và trả kết quả ra.

8.4.2. Kiểm thử tích hợp

Kiểm thử tích hợp là một hình thức kiểm thử phần mềm nhằm tìm cách xác minh các giao diện giữa các thành phần xung đột của một thiết kế. Các thành phần này có thể tích hợp theo cách lặp đi lặp lại hoặc tất cả cùng nhau ("Big Bang"). Thông thường cách thức này được coi là một thực hành tốt hơn vì nó cho phép các vấn đề về giao diện được định vị một cách nhanh chóng và cố định hơn.

Kiểm thử tích hợp làm lộ ra các khiếm khuyết trong các giao diện và tương tác giữa các thành phần tích hợp (Modules). Các nhóm thành phần đã được kiểm thử lớn dần từng bước tương ứng với các thuộc tính của cấu trúc thiết kế đã được tích hợp và kiểm thử cho đến khi phần mềm hoạt động như một hệ thống.

Giai đoạn này được tiến hành sau khi đã hoàn tất công việc kiểm thử từng module riêng lẻ bằng cách tích hợp các module này lại với nhau. Mục đích của giai đoạn này là kiểm tra giao diện của đơn vị, kiểm tra tính đúng đắn so với đặc tả, kiểm tra tính hiệu quả.

Phương pháp thực hiện chủ yếu sử dụng kiểm tra chức năng. Các đơn vị có thể được tích hợp theo một trong 2 chiến lược sau:

- **Phương pháp kiểm thử trên xuống:**

- Thuật giải của hướng tiếp cận này gồm những bước sau:
  - Sử dụng module chính như một driver và các stub được thay cho tất cả các module là con trực tiếp của module chính.
  - Lần lượt thay thế các stub mỗi lần một cái bởi các module thực sự.
  - Tiến hành kiểm tra tính đúng đắn.
  - Một tập hợp các bộ thử nghiệm được hoàn tất khi hết stub.
  - Kiểm tra lùi có thể được tiến hành để đảm bảo rằng không phát sinh lỗi mới.
- Ưu điểm:
  - Kiểm thử trên xuống kết hợp với phát triển trên xuống sẽ giúp phát hiện sớm các lỗi thiết kế và làm giảm giá thành sửa đổi.
  - Nhanh chóng có các phiên bản thực hiện với các chức năng chính.
  - Có thể thẩm định tính dùng được của sản phẩm sớm.
- Nhược điểm:
  - Nhiều module cấp thấp rất khó mô phỏng: thao tác với cấu trúc dữ liệu phức tạp, kết quả trả về phức tạp…

- **Phương pháp kiểm thử dưới lên:**

  Kiểm tra module lá trước do đó không cần phải viết stub.

  - Thuật giải của hướng này gồm:
    - Các module cấp thấp được nhóm thành từng nhóm (thực hiện cùng chức năng)
    - Viết driver điều khiển tham số nhập xuất.
    - Bỏ driver và gắn chùm vào module cao hơn
  - Ưu điểm:
    - Tránh xây dựng các module tạm thời phức tạp.
    - Tránh sinh các kết quả nhân tạo (nhập từ bàn phím).
    - Thuận tiện cho phát triển các module để dùng lại.
  - Nhược điểm:
    - Chậm phát hiện các lỗi kiến trúc.
    - Chậm có các phiên bản thực hiện.

8.4.3. Kiểm thử chấp nhận

Kiểm thử chấp nhận được tiến hành bởi khách hàng, còn được gọi là alpha testing. Mục đích là nhằm thẩm định lại xem phần mềm có những sai sót, thiếu sót với yêu cầu của người sử dụng không.

Trong giai đoạn này, dữ liệu dùng cho kiểm thử do người sử dụng cung cấp.

8.4.4. Kiểm thử hệ thống

Kiểm thử hệ thống giúp xác minh rằng một hệ thống được tích hợp có đáp ứng đầy đủ các yêu cầu hay không. Ngoài ra, kiểm thử phần mềm phải đảm bảo rằng các chương trình hoạt động như kỳ vọng, không còn bị phá hủy hay lỗi phần nào đó trong môi trường hoạt động của nó hoặc không gặp sự cố khi hoạt động với tiến trình khác (điều này bao gồm bộ nhớ chia sẻ không bị hỏng, nguồn tài nguyên không bị dư thừa hay chiếm dụng quá mức và không bị đẩy ra khi hoạt động song song các tiến trình).

Đến giai đoạn này, công việc kiểm thử được tiến hành với nhìn nhận phần mềm như một yếu tố trong một hệ thống thông tin phức tạp hoàn chỉnh.

Công việc kiểm thử nhằm kiểm tra khả năng phục hồi sau lỗi, độ an toàn, hiệu năng và giới hạn của phần mềm.