Chương 1: Bài tập I

Đề bài:

Phân biệt các hướng tiếp cận: Process-Oriented, Data-Oriented, Architecture-Oriented, các điểm mạnh và yếu của từng hướng tiếp cận

1. Process-Oriented Approach
2. Data-Oriented Approach
3. Architecture-Oriented Approach
4. Điểm mạnh yếu của các phương pháp tiếp cận

1) Process-Oriented Approach

Bản chất của việc phân tích và thiết kế đặt trọng tâm vào các chức năng do phần mềm thực hiện.

- Tập trung vào các giải thuật và thao tác xử lý dữ liệu
- Quá trình phát triển phần mềm tập trung vào thể hiện các phương pháp xử lý dữ liệu
- Cấu trúc dữ liệu thông thường không thể hiện rõ

2) Data-Oriented Approach

Dữ liệu không thay đổi bởi các yêu cầu hay đòi hỏi của người dùng về các thao tác nghiệp vụ. Trong thiết kế hướng dữ liệu, hệ thống được thiết kế dựa trên cấu trúc tiến trìn dữ liệu. Việc phân tích thiết kế được tiến hành cho dữ liệu một cách tách bạch với yêu cầu hay đòi hỏi của người dùng về thao tác.

Nghiên cứu và phát triển cơ sở dữ liệu tập trung vào các thực thể và các mối quan hệ của một hệ thống thông tin và dẫn đến sự phát triển của khái niệm, hợp lý và vật lý mô hình dữ liệu, hệ thống chung và các công cụ cũng như phần mềm phát triển các quy trình để hỗ trợ việc phân tích, thiết kế.

Mô tả tổ chức của dữ liệu ,mô tả dữ liệu lấy ra ở đâu và sử dụng như thế nào.

Mô hình dữ liệu được thành lập và được mô tả mối quan hệ giữa các dữ liệu tương ứng này và các quy định về mối quan hệ.

Sử dụng các Business rules để chỉ ra phương pháp xử lí dữ liệu.

3) Architecture-Oriented Approach

Là phương pháp phân tích và thiết kế có cấu trúc. Các yêu cầu của hệ thống đích được phát triển được phân tích bằng việc đặc biệt chú ý tới chức năng của hệ thống và luồng dữ liệu giữa các chức năng. Mục đích của phương pháp này là chuyển các tiến trình trong biểu đồ thành các modules chương trình và tiến hành phân chia các modules bằng cách tiếp cận từ trên xuống.

• Lựa chọn kiến trúc và công nghệ phần mềm để thực hiện bài toán.

• Áp dụng các phương pháp Prototyping để nhanh chóng xây dựng được phần mềm.

Phương pháp Prototyping có vai trò trong duyệt và kiểm soát yêu cầu phần mềm giúp hoàn thiện hơn về yêu cầu, đáp ứng được yêu cầu của người dùng.

Cho người dùng dùng thử mẫu thử như là mẫu thử bản beta từ đó người dùng sẽ dùng thử và đưa ra những điểm tốt, điểm không tốt của mẫu thử, cái nào không cần thiết của mẫu thử từ đó người phân tích viên có thể duyệt yêu cầu nào đã đạt được yêu cầu nào hay những yêu cầu nào rườm rà có thể bỏ qua hay nên bổ sung những yêu cầu gì để thỏa mãn được yêu cầu của người dùng.

Ví dụ: khi cho người dùng delete một bản ghi thì đòi hỏi phải có yêu cầu là có nên delete hay không?

Mẫu thử thẩmđịnh yêu cầu giải thích các yêu cầu và giúp các bên liên quan khám phá được vấn đề.

Thẩm định mẫu thử sẽ hoàn thành có hiệu quả cao và thiết thực. nó có thể dụng chúng trong cách giống nhau như là yêu cầu hệ thống.

Tài liệu đào tạo cho người sử dụng sẽ được cung cấp.

• Sử dụng các Pattern kiến trúc mẫu để chỉ ra phương pháp xử lý dữ liệu

4) Điểm mạnh yếu của các phương pháp tiếp cận

Hướng tiếp cận	Process-Oriented Approach	Data-Oriented Approach	Architecture-Oriented Approach
Điểm mạnh	- Thích hợp với các bài toán phức tạp. - Giảm thời gian đáp ứng của phần mềm do tập trung vào giải thuật và xử lí dữ liệu - Tránh được sự trùng lặp trong cơ sở dữ liệu	- Thích hợp với hệ thống quản lí cơ sở dữ liệu. - Không phụ thuộc vào chức năng và yêu cầu người sử dụng do thiết kế dữ liệu tách bạch. - Biểu diễn đươc các mối quan hệ trong các bảng và giữa các dữ liệu với nhau	- Việc thiết kế phần mềm nhanh do áp dụng các bản mẫu có sẵn. Từ đó thưa kế được những ưu điểm sẵn có. - Áp dụng các kiến trúc công nghệ tốt nhất tăng chất lượng phần mềm.
Điểm yếu	- Khó điều chỉnh các yêu cầu cho nhiều người dùng. - Sử dụng các chức năng chồng chéo nhau là khó tránh khỏi. Kết quả là hệ thống có nhiều chức năng chồng chéo nhau là một trong những nhân tố làm cho việc bảo trì trở nên khó khăn. - Các tệp dữ liệu được xây rất khó để thỏa mãn phần mềm	- Việc xử lí dữ liệu không được linh hoạt do phụ thuộc vào các Business rules - Các chức năng của phần mềm phụ thuộc vào cách tổ chức cơ sở dữ liệu.	- Dữ liệu được xử lí phụ thuộc cao vào các bản mẫu sẵn có ⇨ Bị động trong thiết kế - Phụ thuộc vào công nghệ hiện tại

Table 1: Điểm mạnh yếu của các phương pháp tiếp cận

Chương 2: Bài tập II

Đề bài:

Tổng kết và hệ thống lại các mô hình SDLC: Mô hình thác nước, Mô hình sử dụng lại, Mô hình Spiral, Mô hình Evolutionary, Mô hình RUP. Tìm các tài liệu phân tích các điểm mạnh yếu của các mô hình.

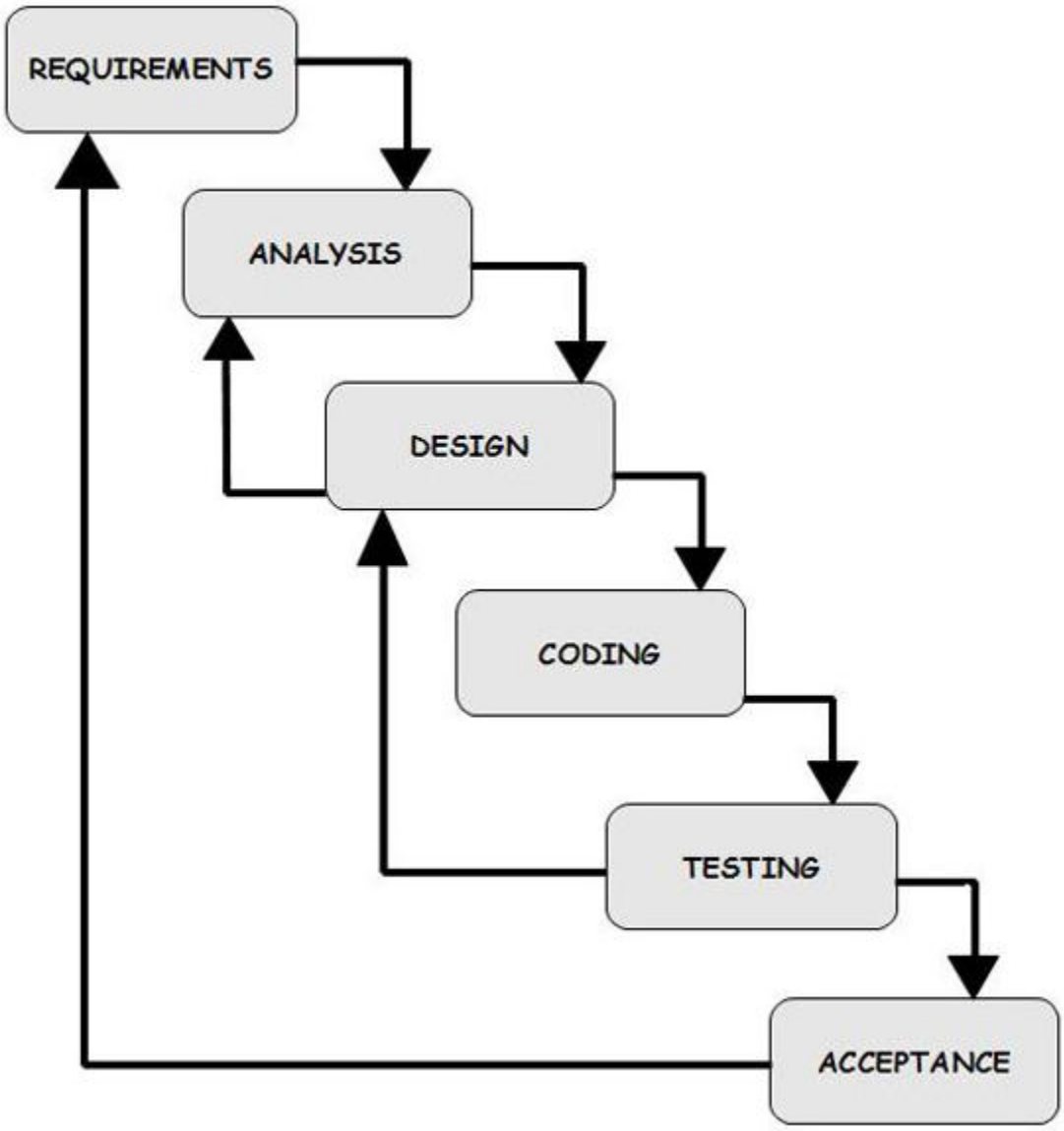
1. Mô hình thác nước
2. Mô hình sử dụng lại
3. Mô hình Spiral
4. Mô hình Evolutionary
5. Mô hình RUP

1) Mô hình thác nước

1.1) Khái niệm và mô hình

Mô hình thác nước (tiếng Anh: *waterfall model*) là một mô hình của quy trình phát triển phần mềm, trong đó quy trình phát triển trông giống như một dòng chảy, với các pha được thực hiện theo trật tự nghiêm ngặt và không có sự quay lui hay nhảy vượt pha là: phân tích yêu cầu, thiết kế, lập trình, kiểm thử, liên kết và bảo trì.

Mô hình thác nước gồm 4 giai đoạn phân tích, thiết kế , lập trình kiểm thử.



Hình 1 : Các giai đoạn của mô hình thác nước

- Phân tích yêu cầu và tài liệu đặc tả (Requirements) là giai đoạn xác định những yêu cầu liên quan đến chức năng và phi chức năng hệ thống phần mềm cần có. Giai đoạn này cần có sự tham gia tích cực của khách hàng và kết thúc bằng một tài liệu "Bản đặc tả yêu câu phần mềm" hay SRS, trong đó bao gồm toàn bộ tài liệu đã được duyệt và nghiệm thu bởi những người có trách nhiệm đối với dự án (từ phía khách hàng). SRS chính là nền tảng cho các hoạt động tiếp theo và cho đến cuối của dự án.

- Phân tích hệ thống (Analysis): giai đoạn xác định các công việc cần làm để hệ thống phần mềm, hiểu lĩnh vực thông tin, chức năng, hành vi, tính năng và giao diện của phần mềm sẽ phát triển. Cần phải tạo tư liệu và bản thảo với khách hang, người dung. giai đoạn xác định các công việc cần làm để hệ thống phần mềm
- Thiết kế (Design): là quá trình nhiều bước với 4 thuộc tính khác nhau của 1 chương trình: cấu trúc dữ liệu, kiến trúc phần mềm, biểu diễn giao diện và chi tiết thủ tục (thuật toán).
- Lập trình (coding): Chuyển thiết kế thành chương trình máy tính bởi ngôn ngữ nào đó. Nếu thiết kế đã được chi tiết hóa thì lập trình có thể thuần túy cơ học.
- Kiểm thử (Testing): Kiểm tra các chương trình và module cả về logic bên trong và chức năng bên ngoài, nhằm phát hiện ra lỗi và đảm bảo với đầu vào xác định thì cho kết quả mong muốn.

- Cài đặt và bảo trì (Acceptance): đây là giai đoạn cài đặt, cấu hình và huấn luyện khách hàng. Giai đoạn này sửa chữa những lỗi của phần mềm nếu có và có thể phát triển thêm những yêu cầu mới mà khách hàng yêu cầu (như sửa đổi, thêm, bớt chức năng/đặc điểm của hệ thống.

1.2) Phân tích ưu nhược điểm

❖ Ưu điểm:

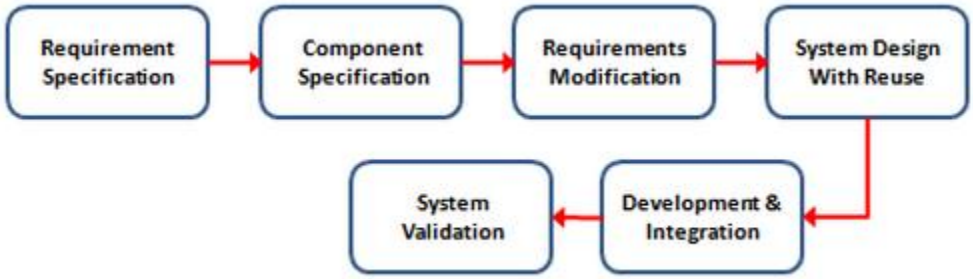
- Chuỗi các hoạt động được thực hiện theo quy trình rõ ràng.
- Thay đổi yêu cầu được giảm tối thiểu khi dự án bắt đầu.
- Dễ phân công công việc, phân bố chi phí, giám sát công việc.

❖ Nhược điểm:

- Thực tế các dự án ít khi tuân theo dòng tuần tự của mô hình, mà thường có sự lặp lại.
- Mối quan hệ giữa các giai đoạn không được thể hiện
- Khách hàng ít khi tuyên bố rõ ràng khi nào xong hết các yêu cầu
- Khách hàng phải có lòng kiên nhẫn chờ đợi thời gian nhất định mới có sản phẩm. Nếu phát hiện ra lỗi nặng thì rất khó khắc phục.
- Khả năng thất bại cao

2) Mô hình sử dụng lại

2.1) Tổng quan



Hình 2: Mô hình sử dụng lại

Mô hình sử dụng lại : tái sử dụng thông tin được tạo ra trong các dự án phát triển phần mềm trước đó nhằm giảm các chi phí, tài nguyên cho việc phát triển dự án mới. Việc sử dụng lại cho phép xây dựng hệ thống phần mềm mới với chất lượng và độ tin cậy cao hơn.

Mô hình gồm 6 giai đoạn:

1. Requirements specification (Yêu cầu kỹ thuật)
2. Component analysis (Phân tích thành phần)
3. Requirements modification (Sửa đổi)
4. System design with reuse (Thiết kế hệ thống với các thành phần tái sử dụng)
5. Development and integration (Phát triển)
6. System validation (Xác nhận hệ thống)

2.2) Phân tích ưu khuyết điểm

a. Ưu điểm

- Giảm chi phí phát triển phần mềm so với việc xây dựng một hệ thống phần mềm mới hoàn toàn.
- Tiết kiệm thời gian vì mỗi giai đoạn phát triển lại sử dụng lại các giai đoạn của quá trình phát triển phần mềm trước nhưng được tinh chế
- Giảm thiểu các sai sót, lỗi của sản phẩm cuối cùng so với phần mềm trước đó.

b, nhược điểm

- Việc sử dụng lại có thể không khả thi vì các thành phần tái sử dụng có thể không đầy đủ, cần phải thiết kế mới.
- Có thể không đáp ứng được nhu cầu của khách hàng
- không khả thi khi thành phần sử dụng lại chứa nhiều lỗi liên quan đến thiết kế hay không thể khắc phục.

3) Spiral SDLC

3.1) Spiral Model trong SDLC là gì?

Mô hình xoắn ốc là một quá trình phát triển phần mềm (còn gọi với thuật ngữ khác là software development life-cycle, SDLC) với định hướng giải quyết rủi ro (risk-driven). Nó tương đối giống với mô hình lặp và tăng tiến (iterative and incremental development model), nhưng nhấn mạnh vào phân tích và quản lý rủi ro của dự án phần mềm.

Mô hình xoắn ốc phát triển và tiến hóa sau mô hình thác nước, dựa trên kinh nghiệm cũng như những cải tiến của mô hình thác nước. Riêng nó bao chứa các mô hình khác như là các trường hợp đặc biệt, và cung cấp các hướng dẫn làm thế nào để kết hợp các mô hình khác một cách phù hợp nhất với 1 dự án phần mềm.

3.2) Mô hình

Mô hình trực quan của mô hình phát triển này giống như tên gọi của nó:

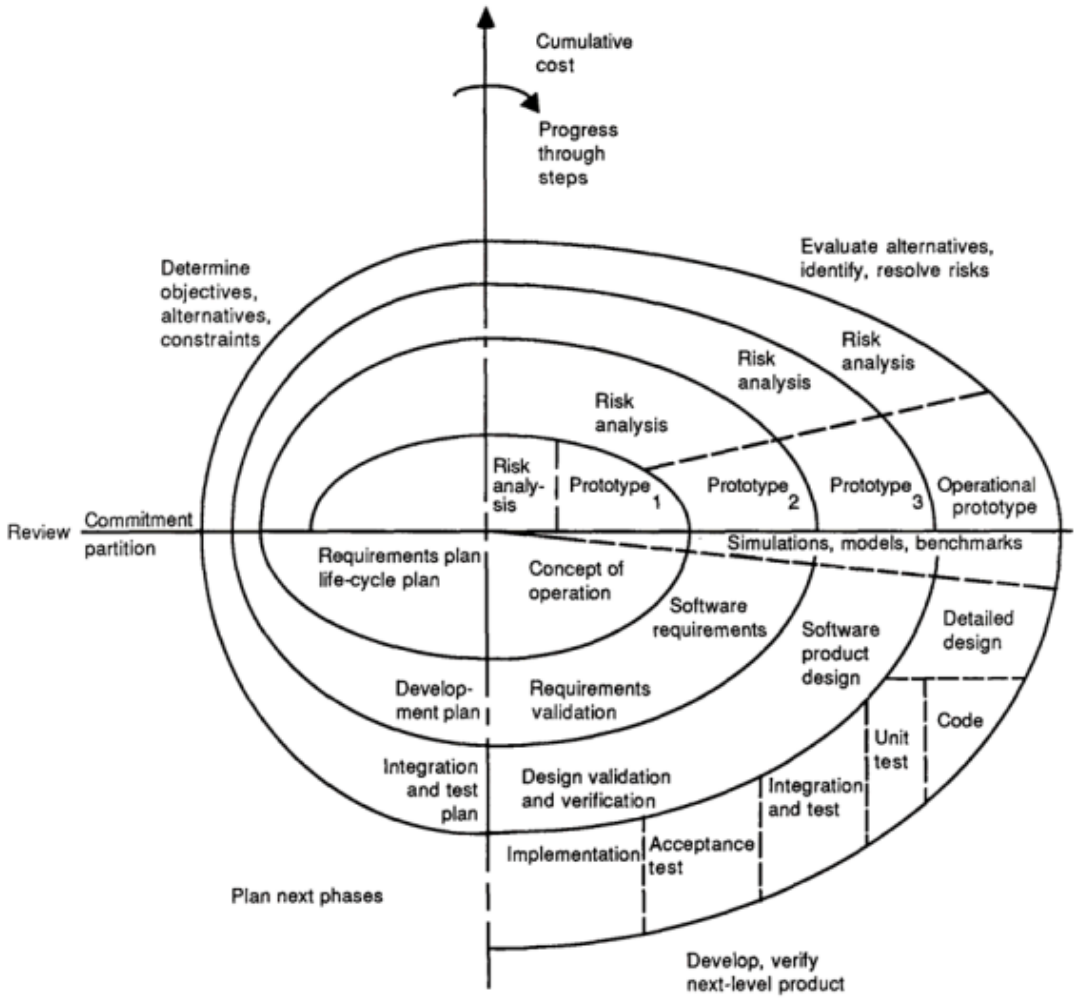


Figure 2. Spiral model of the software process.

Hình 3: Mô hình phát triển phần mềm xoắn ốc (A Spiral Model of Software Development and Enhancement - Boehm, 1988)

Có thể thấy, bán kính của quá trình phát triển tăng dần, nó đại diện cho chi phí phát sinh trong quá trình hoàn thành từng bước phát triển phần mềm, còn góc của quá trình (Boehm gọi là angle dimension) biểu diễn quá trình hoàn thành mỗi bước phát triển.

Các bước triển khai

Mô hình xoắn ốc gồm nhiều vòng lặp khác nhau qua 4 pha: Lên kế hoạch, Phân tích rủi ro, Thực hiện và Đánh giá.

Vòng xoắn ốc cơ sở: bắt đầu từ pha lên kế hoạch, các yêu cầu phần mềm được thu thập và xác định, các rủi ro được đánh giá.

Các vòng xoắn ốc sau đó được xây dựng dựa trên vòng xoắn ốc cơ sở:

1. Các yêu cầu được thu thập suốt pha lên kế hoạch.

2. Trong pha phân tích rủi ro, các rủi ro được nhận diện và đưa ra các giải pháp đối với các rủi ro này. Cuối pha này, nhóm phát triển cho ra một nguyên mẫu của phần mềm cần phát triển.

3. Phần mềm được xây dựng trong pha thực hiện (engineering), cùng với khâu kiểm thử ở cuối pha.

4. Pha đánh giá cho phép các khách hàng đánh giá sản phẩm của dự án tại thời điểm hiện tại trước khi dự án tiếp tục chuyển sang vòng xoắn ốc tiếp theo.

3.3) Áp dụng khi nào?

Khi việc đánh gia chi phí và rủi ro dự án là rất quan trọng.

Cho dự án có rủi ro từ mức trung bình trở lên.

Các dự án dài hạn mà không thể biết trước những thay đổi lớn tiềm ẩn.

Khi khách hàng không chắc chắn họ cần gì trong phần mềm mà mình yêu cầu.

Khi các yêu cầu phần mềm rất phức tạp.

Khi cần cho ra 1 dòng sản phẩm mới (các tính năng được bổ sung dần tùy theo yêu cầu thị trường).

Khi mong đợi ở đầu ra của dự án là những sự thay đổi lớn (như nghiên cứu, khám phá).

3.4) Các ưu điểm/nhược điểm?

Ưu điểm:

– Quá trình phá triển lặp đi lặp lại và liên tục rất hữu ích cho quản lý rủi ro. Các nhà phát triển chỉ cần mô tả các đặc tính cốt yếu trước rồi sau đó phát

triển các nguyên mẫu, sản phẩm dựa trên mô tả này. Những nguyên mẫu này được kiểm thử và nếu có các thay đổi mong muốn, những thay đổi này sẽ được thực hiện trên hệ thống mới (của vòng xoắn ốc sau). Phương pháp tiếp cận liên tục và đều đặn này sẽ tối thiểu hóa mọi rủi ro hay thất bại phát sinh từ các thay đổi của hệ thống.

– Như vậy mô hình xoắn ốc có khả năng đáp ứng cao các thay đổi có thể xảy ra trong bất kỳ pha nào của dự án phần mềm (nhất là thay đổi trong yêu cầu phần mềm).

– Việc xây dựng nguyên mẫu khá nhanh và ít tốn kém, việc ước lượng chi phí phát triển trở nên dễ dàng hơn; đồng thời khách hàng sẽ hiểu sâu hơn và giành được nhiều quyền quản trị trên hệ thống mới hơn (họ đều có thể tham gia tích cực vào mỗi vòng xoắn ốc).

Nhược điểm:

– Chỉ phát huy hiệu quả thực sự so với các mô hình khác trên các dự án lớn (với chi phí liên quan và độ phức tạp cao hơn rất nhiều). Do đó, đây là một mô hình khá tốn kém, cả về mặt tài chính lẫn con người.

– Để áp dụng mô hình này, cần có những chuyên gia nhiều kinh nghiệm, kỹ năng trong việc đánh giá sự bất định và rủi ro của dự án.

– Việc thực hiện dự án cần có kỹ luật chặt chẽ, từng bước của dự án cần tuân theo nghiêm ngặt.

– Chỉ riêng chi phí cho việc đánh giá rủi ro của 1 hệ thống còn có thể cao hơn cả chi phí để xây dựng lên hệ thống đó.

– Thành công của 1 dự án phụ thuộc khá nhiều vào pha phân tích rủi ro.

4) Evolutionary SDLC

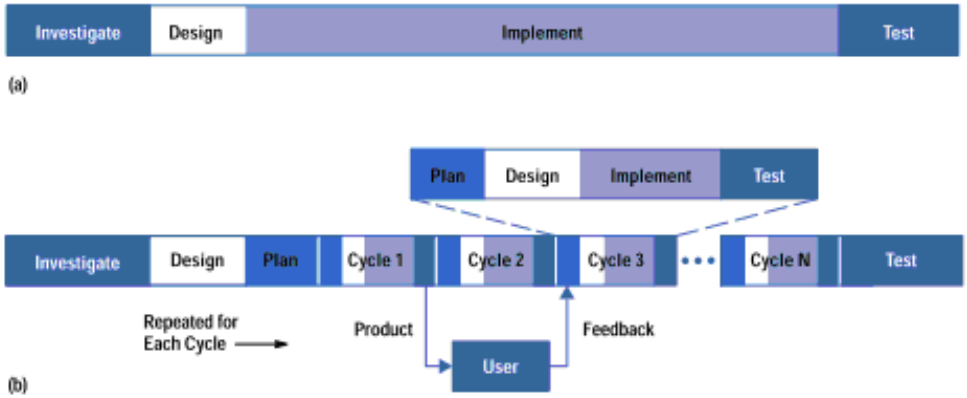
4.1) Khái niệm

Mô hình tiến hóa dựa trên ý tưởng là nhanh chóng phát triển một phiên bản đầu tiên của phần mềm từ những đặc tả rất trừu tượng và thay đổi, cải tiến phiên bản này theo đánh giá và yêu cầu của khách hàng. Mỗi phiên bản phần

mềm sau sẽ kế thừa những đặc tính tốt nhất từ phiên bản trước đó. Các phiên bản sau được cải tiến dựa trên phản hồi của khách hàng để tạo ra một hệ thống thỏa mãn nhu cầu của khách hàng. Và khi phần mềm thỏa mãn yêu cầu của khách hàng, nó có thể được bàn giao hoàn toàn cho khách hàng.

4.2) Mô hình

– Để hiểu về mô hình tiến hóa, ta sẽ tìm sự khác nhau giữa mô hình tiến hóa và mô hình thác nước truyền thống:



Hình 4: Sự khác nhau giữa mô hình thác nước và mô hình tiến hóa trong phát triển phần mềm (The Evolutionary Development Model for Software - Elaine L. May and Barbara A. Zimmer)

– Mô hình thác nước được áp dụng rất phổ biến. Tuy nhiên, một nhược điểm lớn của mô hình này, đó là nó chỉ hiệu quả đối với các dự án phần mềm mà các đặc tả đã rất rõ ràng ngay từ đầu, yêu cầu nhóm phát triển phải hiểu rõ về phần mềm mà mình đang xây dựng, lượng giá được những khó khăn có thể gặp phải trong suốt quá trình phát triển. Tuy nhiên thực tế thì các yêu cầu phần mềm luôn luôn thay đổi trong suốt quá trình dự án được thực hiện, gây nhiều khó khăn cho việc thực hiện dự án (giai đoạn implement). Hơn nữa, rất nhiều yêu cầu khách hàng lại không rõ ràng ngay từ đầu, cũng như hiểu biết của nhóm phát triển không toàn diện về phần mềm mà mình đang xây dựng.

– Mô hình tiến hóa giúp giải quyết được nhược điểm này của mô hình thác nước. Nó chi giai đoạn thực hiện (implement) ra thành nhiều chu kỳ. Mỗi

chu kỳ cũng có 4 bước như 1 mô hình thác nước con (incremental waterfalls): Lên kế hoạch, Thiết kế, Thực hiện, Kiểm thử. Nhờ đó, người dùng có cơ hội tiếp cận tới phần mềm cuối mỗi chu kỳ con và đưa ra được đánh giá, phản hồi lại cho nhóm phát triển. Bằng cách này, các thay đổi trong yêu cầu người dùng có thể được giải quyết trong các chu kỳ con sau đó.

– Vì người dùng được tham gia và có ảnh hưởng quan trọng trong quá trình thực hiện (implement), do đó sản phẩm cuối cùng đáp ứng rất sát yêu cầu của người dùng.

4.3) Các bước triển khai

1. Khảo sát yêu cầu, đưa ra các đặc tả yêu cầu người dùng.

2. Thiết kế ban đầu dựa trên các đặc trưng quan trọng nhất của hệ thống.

3. Lên kế hoạch thực hiện.

4. Thực hiện các chu kỳ phát triển thác nước con (Plan – Design – Implement – Test). Sau mỗi bước test sẽ giao cho khách hàng 1 phiên bản trung gian để khách hàng đánh giá và nhận phản hồi. Bước kế hoạch của mỗi chu kỳ sẽ dựa trên các phản hồi của khách hàng trong các chu kỳ trước đó.

5. Kiểm thử phiên bản cuối cùng.

Áp dụng khi nào?

Giống với mô hình xoắn ốc. Được áp dụng nhiều cho các dự án lớn mà yêu cầu ban đầu còn chưa rõ ràng, nhưng cần có sản phẩm sớm.

4.4) Các ưu điểm/nhược điểm?

Ưu điểm:

– Nhờ chia nhỏ quá trịnh thực hiện ra thành các phần nhỏ hơn và quản lý được, kế hoạch dự án sẽ trở nên rõ ràng và trực quan hơn. Nó làm giảm sự phức tạp của dự án, giảm thiểu được nhiều rủi ro cũng như giải quyết được kịp thời các rủi ro, thay đổi (nhờ liên tục nhận được các phản hồi cuối mỗi chu kỳ con).

– Tầm nhìn dài hạn của hệ thống được chia thành các bước ngắn hạn.

– Các bước thực hiện được xác định độ ưu tiên rõ ràng.

– Có kết quả sớm – đây là "công cụ" giao tiếp rất tốt giữa nhóm phát triển và khách hàng (thảo luận và phản hồi quanh sản phẩm hiện thời)

– Có phản hồi của khách hàng từ bên ngoài nhóm phát triển.

– Có sự cải tiến dần dần cho sản phẩm hiện tại.

– Có thể dễ dàng nhận ra những yêu cầu nào và công việc nào có thể được hoàn thành.

– Có thể xác định trước các thành phần chức năng chung.

Nhược điểm:

– Không phù hợp với các dự án nhỏ.

– Độ phức tạp quản lý tăng.

– Không biết rõ ràng khi nào thì kết thúc dự án → bản than điều này là 1 rủi ro.

– Chi phí tốn kém

– Yêu cầu nhiều tài nguyên và kỹ năng cho phân tích rủi ro (giống spiral model)

– Các tiến trình dự án phụ thuộc nhiều vào bước phân tích rủi ro.

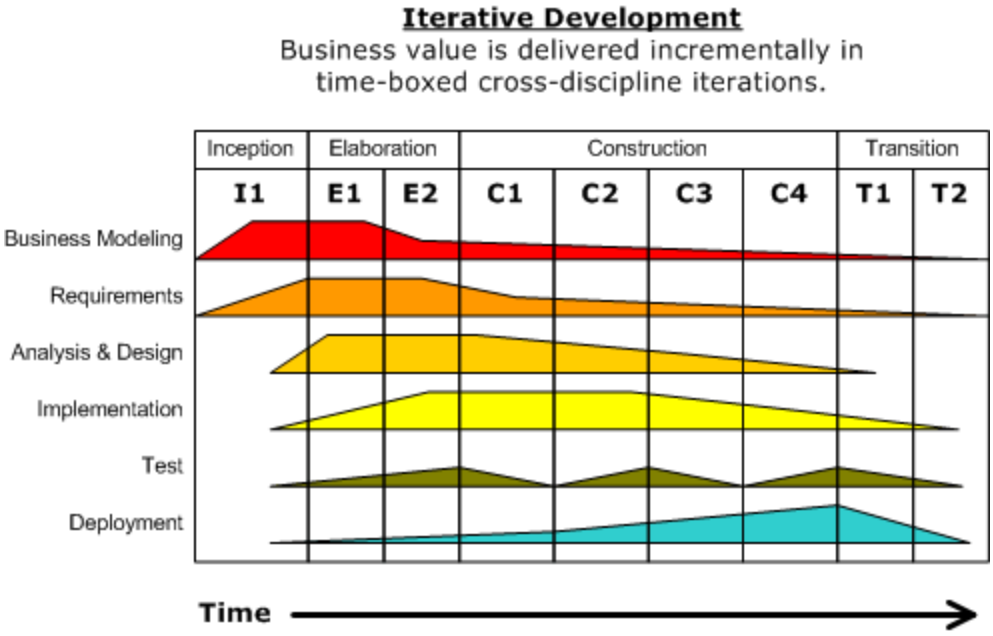
5) RUP (Rational Unified Process) SDLC

5.1) Khái niệm

RUP (Rational Unified Process) là một quá trình phát triển phần mềm theo mô hình lặp (iterative) được sáng tạo bởi Rational Software Corporation từ 2003. Nó là quá trình phát triển mang tính thích nghi hơn là mang tính khuôn mẫu như các mô hình khác. RUP sẽ được các đội phát triển thiết kế, lựa chọn những đặc tính phù hợp mà họ thấy cần cho dự án phần mềm của mình.

5.2) Mô hình

Các pha và khung phát triển của RUP có dạng:



Hình 5: Mô hình lặp của RUP (Wikipedia)

Các pha : Bắt đầu, Cộng tác – chuẩn bị, Xây dựng và Chuyển dịch.

Các công việc chính cần thực hiện: Tìm hiểu mô hình dịch vụ, Xác định các yêu cầu, Phân tích – Thiết kế, Thực hiện, Kiểm thử và Triển khai.

Ngoài các công việc chính trên, còn có các công việc hỗ trợ khác: Quản lý cấu hình và thay đổi, Quản lý dự án, Môi trường dự án.

5.3) Các bước triển khai

– Pha bắt đầu (inception phase). Mục đích của pha này là xác định phạm vi của hệ thống để làm cơ sở cho việc tính toán chi phí ban đầu và ngân sách. Trong pha này, các yếu tố liên quan đến kinh doanh được xác định (như bối cảnh kinh doanh, yếu tố thành công, dự báo tài chính…). Pha này sinh ra các mô hình ca sử dụng, kế hoạch dự án, thẩm định rủi ro ban đầu và mô tả dự án. Các thành phần này và các yếu tốc kinh doanh được để kiểm tra và xem xét xem có tiếp tục thực hiện dự án hay không (phải vượt qua LifeCyle Objective).

– Pha chuẩn bị (elaboration). Mục đích của pha này là giảm thiểu các rủi ro chính. Các rủi ro này được phát hiện và đề xuất giải pháp bằng cách phân

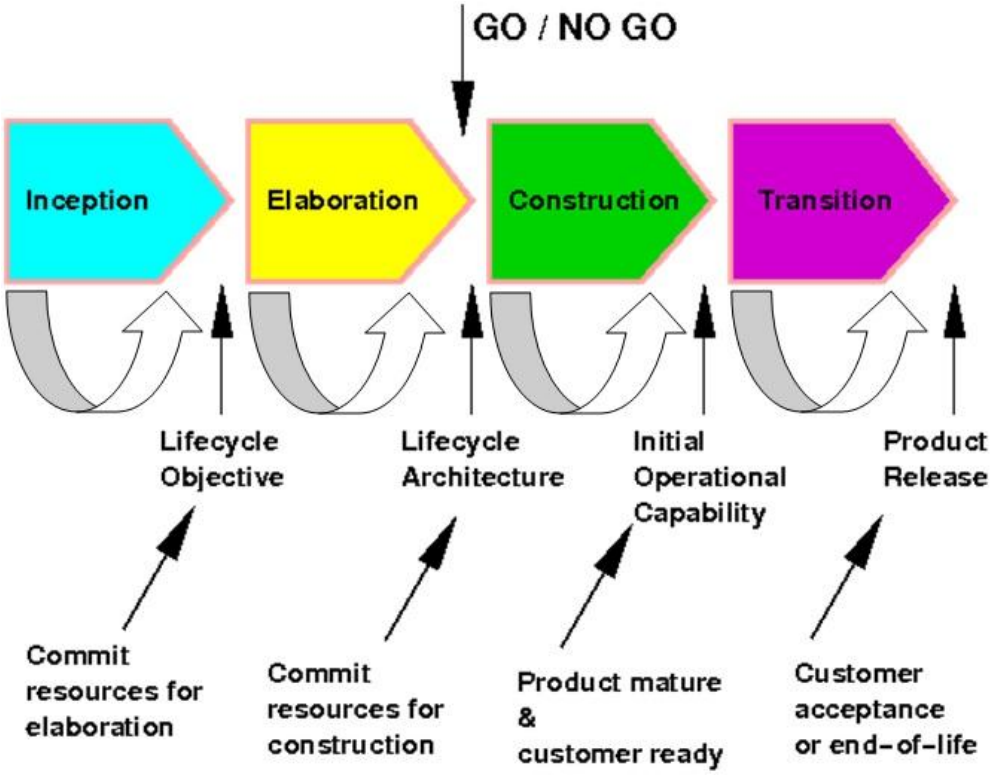
tích các yếu tố liên quan đến dự án. Đến pha này, dự án bắt đầu được định hình: Vấn đề được phân tích và kiến trúc của dự án ban đầu được xác định.

– Pha này phải vượt qua được cột mốc kiến trúc vòng đời (LifeCycle Architecture Milestone). Nếu không vượt qua được cột mốc này, vẫn còn thời gian để thiết kế lại hoặc quyết định hủy dự án.

– Pha xây dựng. Mục đích chính của pha này là xây dựng hệ thống phần mềm. Trọng tâm của pha này đặt vào việc phát triển các thành phần và chức năng của hệ thống. Việc lập trình được thực hiện chủ yếu ở pha này. Việc xây dựng hệ thống phần mềm ở pha này có thể được chia theo chức năng, thành phần (component) thành các bộ phận quản lý được và xây dựng nguyên mẫu được (tức là phần đó tương đối hoàn chỉnh về chức năng). Mỗi bộ phận được thực hiện trong 1 vòng lặp riêng.

– Pha chuyển dịch. Pha này thực hiện việc chuyển dịch hệ thống từ quá trình phát triển sang sản phẩm hoàn thiện, đưa nó tới người sử dụng và giúp họ sử dụng nó hiệu quả. Các hoạt động của pha này gồm có đào tạo người dùng cuối và nhân viên bảo trì; thực hiện kiểm thử hệ thống để kiểm tra tính đúng đắn của nó so với mong muốn của người dùng cuối. Sản phẩm cũng được đối chiếu với yêu cầu chất lượng đề ra trong pha bắt đầu.

– Một lưu ý quan trọng về các pha trong RUP: bản thân mỗi pha là một vòng lặp, có thể được thực hiện cho đến khi đạt được mục tiêu mỗi pha.



Hình 6: Các pha và mục tiêu mỗi pha trong RUP (Introduction to Software Development - Ma'am Marium Nosheen)

5.4) Áp dụng khi nào?

Mô hình RUP được áp dụng rộng rãi, nhờ tính thích nghi (adaptive) của nó. Các nhóm phát triển có thể chọn lựa các yếu tố cảm thấy cần thiết cho dự án phần mềm của mình để áp dụng RUP.

5.5) Các ưu điểm/nhược điểm?

(An overview of the Rational Unified Process (RUP) - Eric Villagomez)

Ưu điểm:

– Thường xuyên nhận được phản hồi từ các cổ đông
– Sử dụng các tài nguyên dự án 1 cách hiệu quả
– Cung cấp được chính xác cái mà khách hàng muốn
– Các vấn đề được phát hiện sớm trong dự án
– Hỗ trợ mô hình phát triển lặp
– Cải thiện quản lý rủi ro

Nhược điểm:

– Các tiến trình dự án rất phức tạp để thực hiện

– Quá trình phát triển có thể vượt quá tầm kiểm soát (do đánh gia ban đầu sai về chi phí, tài nguyên và rủi ro cũng như do các yếu tố bất định)

– Cần các chuyên gia để có thể đáp ứng được các mục tiêu của mô hình phát triển này.

– Tiến trình nặng

Chương 3: Bài tập III

Đề bài:
Tìm các KPA cơ bản của Requirement Engineering và vẽ sơ đồ biểu diễn mối quan hệ này. Mô tả ngắn gọn nội dung của từng KPA.

1) Các KPA cơ bản của Requirement Engineering

1.1) Định nghĩa Requirement Engineering

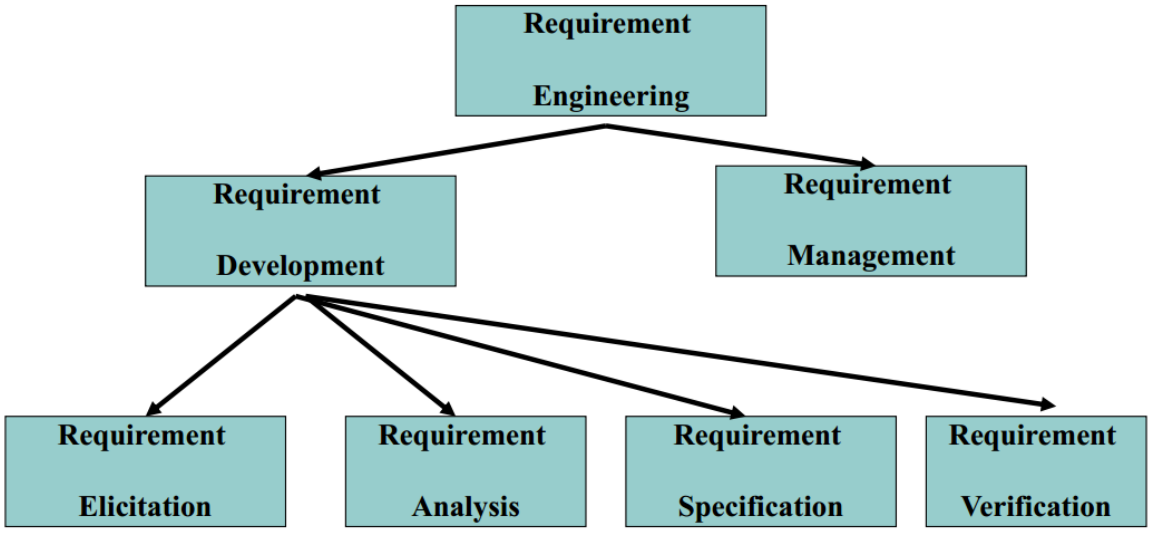
Requirement Engineering (quy trình yêu cầu phần mềm) là cơ chế thích hợp để hiểu khách hàng muốn gì, phân tích yêu cầu, đánh giá tính khả thi, đàm phán một giải pháp hợp lí, xác định giải pháp rõ ràng, xác nhận yêu cầu kĩ thuật và quản lí yêu cầu khi chúng được chuyển thành hệ thống hoạt động. Sản phẩm của quá trình là tài liệu lớn bằng ngôn ngữ tự nhiên, mô tả những gì hệ thống làm, không mô tả chúng được làm như thế nào.

1.2) Các KPA (key process area – vùng xử lí quan trọng) cơ bản

– Requirement Development (Phát triển yêu cầu)
  + Requirement Elicitation (Phát hiện yêu cầu)
  + Requirement Analysis (Phân tích yêu cầu)
  + Requirement Specification (Đặc tả yêu cầu)
  + Requirement Verification (Kiểm thử yêu cầu)
– Requirement Management (Quản lí yêu cầu)

1.3) Sơ đồ mối quan hệ giữa các KPA:

Về mặt cấu trúc

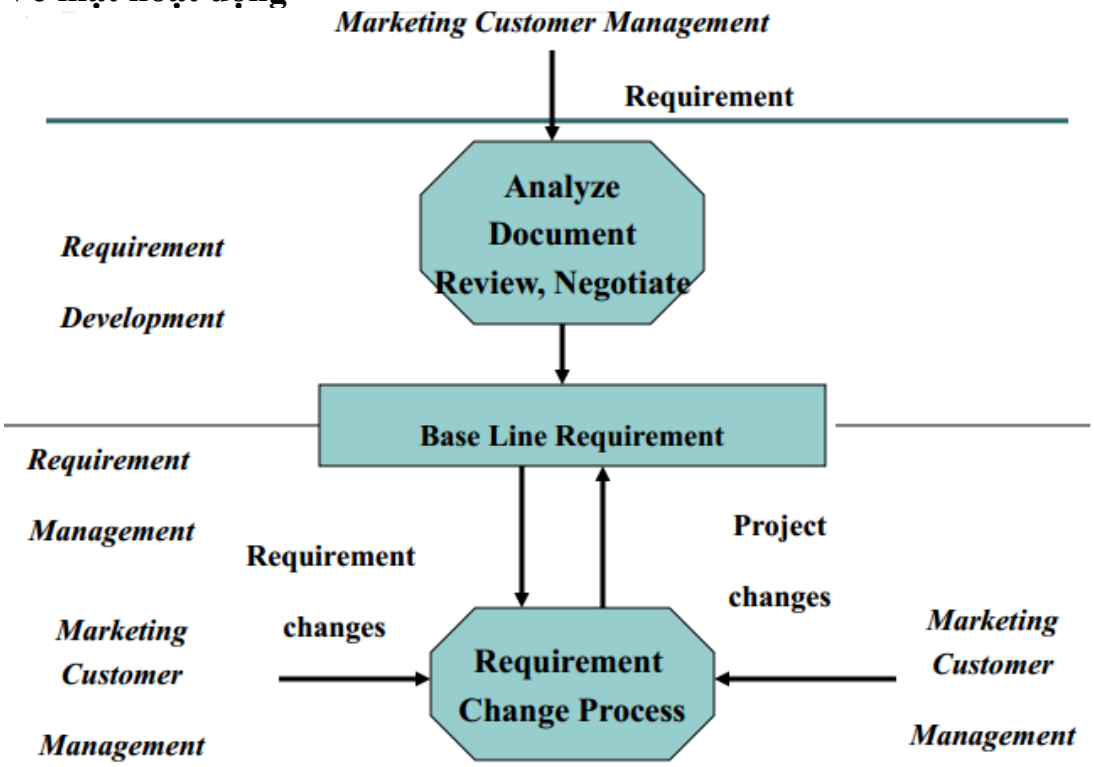


Hình 7: Sơ đồ quan hệ cấu trúc các KPA

(Slide bài giảng Phân tích yêu cầu phần mềm – Thầy Huỳnh Quyết Thắng)

Quy trình yêu cầu phần mềm có 2 KPA cơ bản là Phát triển yêu cầu và Quản lí yêu cầu, trong đó Phát triển yêu cầu bao gồm 4 KPA thành phần được chỉ ra trong sơ đồ.

Về mặt hoạt động



Hình 8: Sơ đồ quan hệ trong hoạt động của các KPA

(Slide bài giảng Phân tích yêu cầu phần mềm – Thầy Huỳnh Quyết Thắng)

Ban quản lí tiếp thị khách hàng (Marketing Customer Management) sẽ tiến hành các hoạt động với khách hàng để xác định yêu cầu đối với hệ thống đang xây dựng.

Giai đoạn Requirement Development sẽ tạo ra bản tài liệu yêu cầu cơ sở - Base Line Requirement (thông qua các hoạt động như phân tích tài liệu, phỏng vấn, đàm phán…) để gửi cho khách hàng và làm bước khởi đầu cho giai đoạn Requirement Management.

Sau khi đã có bản yêu cầu cơ sở, ban quản lí tiếp thị khách hàng sẽ tiếp tục làm việc với khách hàng để thu nhận, xử lí các yêu cầu thay đổi từ khách hàng và tạo nên phiên bản tiếp theo của tài liệu yêu cầu. Sau đó bản yêu cầu mới lại cập nhật vào Base Line Requirement và được gửi cho khách hàng, để tiếp tục lấy ý kiến của khách hàng. Quá trình tiếp tục cho tới khi các bên liên quan đạt được sự đồng thuận về các yêu cầu của hệ thống.

2) Mô tả ngắn gọn các KPA

2.1) Requirement Development (Phát triển yêu cầu)

Phát triển yêu cầu là giai đoạn xác định các yêu cầu của khách hàng đối với hệ thống, sản phẩm cho ra là bản yêu cầu cơ sở. Bốn giai đoạn nhỏ của KPA này đảm nhận các công việc cụ thể của quá trình yêu cầu phần mềm

+ Requirement Elicitation (Phát hiện yêu cầu)

Phát hiện yêu cầu là quá trình thu thập và tài liệu hóa các nhu cầu của các bên liên quan, xác định các yêu cầu tài nguyên và thu thập các thông tin, tài liệu cần thiết khác.

Đây là bước đầu tiên trong quá trình tìm hiểu các vấn đề được yêu cầu giải quyết. Hoạt động cơ bản thuộc về con người, các bên liên quan sẽ thiết lập mối quan hệ giữa đội phát triển và khách hàng. Giai đoạn này còn được gọi là 'requirements capture', 'requirements discovery', và 'requirements acquisition' (Guide to the Software Engineering Body of Knowledge, 2-4).

Các hoạt động cụ thể:

- Phỏng vấn khách hàng
- Quan sát người dùng thực hiện công việc của họ
- Nghiên cứu các kịch bản làm việc
- Tổ chức các hội thảo
- Kiểm tra các báo cáo vấn đề
- Tái sử dụng yêu cầu

→ **Mục tiêu:**

- Xác định các yêu cầu quá trình phát triển
- Xác định tầm nhìn và phạm vi
- Xác định các lớp người dùng
- Xác định các tiêu chí sản phẩm
- Xác định các trường hợp ca sử dụng
- Xác định các sự kiện hệ thống và đáp ứng

+ Requirement Analysis (Phân tích yêu cầu)

Phân tích yêu cầu là quá trình phân tích các dữ liệu thu được trong Phát hiện yêu cầu, giải quyết xung đột, phân tích luật thương mại, tài liệu hóa các giả định, các ràng buộc và các sự phụ thuộc, đồng thời làm việc với các bên liên quan để tạo lập các ưu tiên ban đầu.

Các hoạt động cụ thể:

- Vẽ sơ đồ ngữ cảnh
- Tạo mẫu
- Phân tích tính khả thi
- Gán độ ưu tiên các yêu cầu
- Mô hình hóa các yêu cầu
- Tạo một từ điển dữ liệu
- Phân bổ các yêu cầu tới hệ thống con
- Áp dụng việc triển khai hàm đánh giá chất lượng

→ **Mục tiêu:**

- Phát hiện và giải quyết xung đột giữa các yêu cầu
- Tìm ra phạm vi của phần mềm và cách mà nó tác động tới môi trường xung quanh
- Nghiên cứu các yêu cầu hệ thống để tìm ra yêu cầu phần mềm

+ Requirement Specification (Đặc tả yêu cầu)

Đặc tả yêu cầu là quá trình xác định các văn bản chức năng và bổ trợ dựa trên các yêu cầu và hỗ trợ bằng các công nghệ trực quan đa dạng như mô hình hóa tiến trình, biểu đồ UML, các bảng khung…

Các hoạt động cụ thể:

- Áp dụng các mẫu đặc tả yêu cầu phần mềm
- Xác định các nguồn yêu cầu
- Gán nhãn riêng cho mỗi yêu cầu
- Ghi lại các quy tắc kinh doanh
- Xác định các thuộc tính chất lượng

→ **Mục tiêu:**

- Tạo tài liệu định nghĩa hệ thống
- Đặc tả được yêu cầu hệ thống
- Đặc tả được yêu cầu phần mềm

+ Requirement Verification (Kiểm thử yêu cầu)

Kiểm thử yêu cầu là quá trình xem xét lại các đặc tả yêu cầu và các minh họa trực quan đi kèm với các bên liên quan để xác định các thuộc tính chất lượng như sự hoàn thiện, sự phù hợp, sự rõ ràng, tính thực tiễn…

Các hoạt động cụ thể:

- Kiểm tra các tài liệu yêu cầu
- Kiểm tra các yêu cầu
- Xác định các tiêu chí chấp nhận
- Tạo mẫu thử

• Kiểm tra sự chấp nhận

→ **Mục tiêu:**

• Đảm bảo các kĩ sư phần mềm hiểu rõ tất cả yêu cầu

• Đảm bảo các yêu cầu đưa ra được khách hàng chấp nhận

2.2) Requirement Management (Quản lí yêu cầu)

2.2.1) Định nghĩa Requirement Management

Quản lý các yêu cầu phần mềm thực hiện các giao tiếp và thoả thuận với người sử dụng về các yêu cầu của phần mềm cần thực hiện (CMU/SEI 1995)

• Xác định giới hạn của phần mềm (Requirement baseline)
• Duyệt lại các giới hạn của phần mềm
• Quản lý các thay đổi trong yêu cầu phần mềm (Requirement Changes)

2.2.2) Các bước chính

+ Xác định một quá trình kiểm soát thay đổi yêu cầu: Thiết lập một quá trình mà qua đó yêu cầu thay đổi được đề xuất, phân tích và giải quyết. Quản lý tất cả những thay đổi được đề xuất thông qua quá trình này. Công cụ có khiếm khuyết theo dõi thương mại có thể hỗ trợ quá trình thay đổi kiểm soát.
+ Thành lập ban kiểm soát thay đổi: Một nhóm nhỏ của các bên liên quan được thành lập để tiếp nhận các đề xuất thay đổi yêu cầu, xử lí chúng, xem xét đề xuất nào được chấp nhận hay loại bỏ và thiết lập các ưu tiên.
+ Phân tích các tác động của sự thay đổi yêu cầu: Đánh giá sự ảnh hưởng của thay đổi tới thiết kế hệ thống, mã nguồn các chương trình, các nhiệm vụ cần tiến hành để đáp ứng sự thay đổi đó.
+ Thiết lập một đường cơ sở và kiểm soát các phiên bản của tài liệu yêu cầu: Đường cơ sở bao gồm các yêu cầu đã được cam kết thực hiện trong một thông cáo cụ thể. Các thay đổi chỉ thực hiện dựa trên những cam kết đã có.
+ Duy trì một lịch sử của yêu cầu thay đổi: Ghi lại những ngày mà yêu cầu đặc điểm kỹ thuật đã được thay đổi, những thay đổi đã được thực hiện, những người thực hiện thay đổi, và tại sao. Một phiên bản điều khiển công cụ hoặc yêu cầu thương mại quản lý công cụ có thể tự động hoá các tác vụ này.

Tài liệu tham khảo

[1] Slide bài giảng Phân tích yêu cầu phần mềm – Thầy Huỳnh Quyết Thắng

[2] IEEE Computer Society Professional Practices Committee – Guide to the Software Engineering Body of Knowledge 2004 Version

[3] Karl E. Wiegers – Software Requirements, Second Edition

[4] Ma'am Marium Nosheen – Introduction to Software Development

[5] Boehm – A Spiral Model of Software Development and Enhancement, 1988

[6] Wikipedia