Chương 1: TỔNG QUAN VỀ PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

1

NỘI DUNG

- ĐỊNH NGHĨA VỀ QUY TRÌNH PHÁT TRIỂN HỆ THỐNG
- MỘT SỐ QUY TRÌNH PHÁT TRIỂN HỆ THỐNG THÔNG DỤNG
- QUY TRÌNH RUP

2

Định nghĩa quy trình PTHT

- Quy trình PTHT (Software development process)
  - Một tập có cấu trúc (có trật tự) các hoạt động cần thiết để phát triển một hệ thống phần mềm
- Có nhiều quy trình PTHT
  - Vd: Thác nước (Waterfall), Nguyên mẫu (Prototyping), Xoắn ốc (Spiral), Scrum agile….
  - Không tồn tại một quy trình PTHT lý tưởng duy nhất phù hợp cho mọi bài toán, yêu cầu thực tế

3

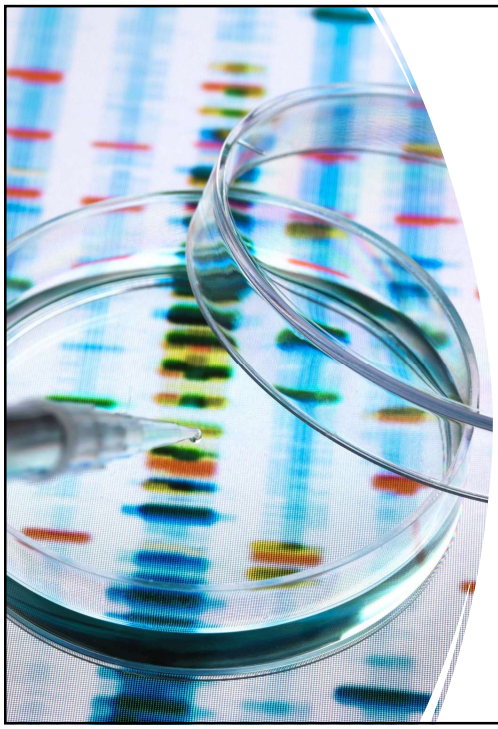
3

Các yếu tố để lựa chọn Quy trình PTHT

- Kiểu của hệ thống phần mềm cần được xây dựng
  - Xây dựng mới từ đầu >< Nâng cấp, chỉnh sửa hệ thống có sẵn
  - Kiểu thông thường, phổ biến >< Kiểu tùy biến, đặc thù
  - Các yêu cầu phần mềm xác định >< Các yêu cầu phần mềm thay đổi (nhanh chóng)
  - Hệ thống trọng yếu (critical) >< Hệ thống nghiệp vụ, kinh doanh
- Quy mô của dự án PTHT, Quy mô (nguồn lực) của nhóm PTHT, thời gian thực hiện dự án PTHT
- Các đặc điểm của nhóm PTHT
  - Kinh nghiệm, Động cơ (+ sự khuyến khích), thái độ làm việc (nỗ lực)
- Kinh phí thực hiện dự án PTHT

4

4



Các hoạt động cơ bản của Quy trình PTHT

- Phân tích tính khả thi (Feasibility study)
- Phân tích và đặc tả yêu cầu (Requirements analysis)
- Thiết kế (Design)
- Thực hiện, cài đặt (Implementation)
- Kiểm thử (Testing)
- Triển khai (Deployment)
- Bảo trì (Maintenance)

5

5

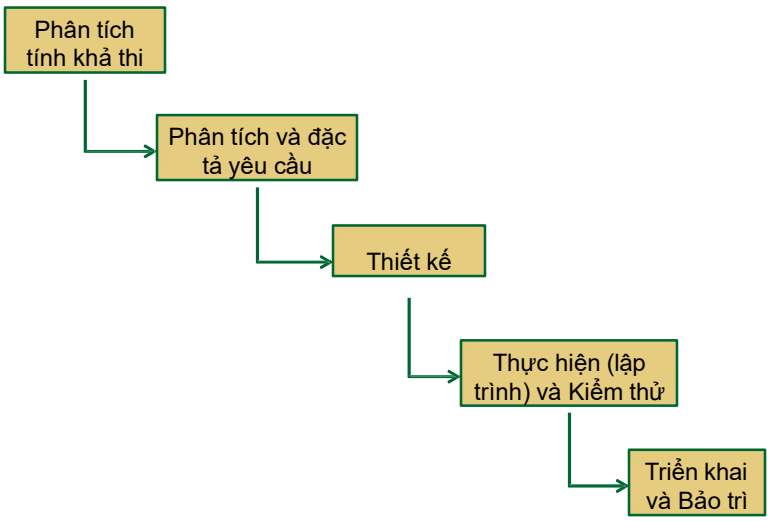
Một số quy trình PTHT thông dụng

- Mô hình thác nước (Waterfall model)
- Mô hình nguyên mẫu (Prototyping model)
- Mô hình xoắn ốc (Spiral model)
- Mô hình nhanh lẹ (Agile model)
- Mô hình hợp nhất (Unified model)

6

6

Mô hình thác nước (Waterfall model)



7

7

Mô hình thác nước (Waterfall model)

- Được giới thiệu bởi Winston Royce vào năm 1970, và hiện tại vẫn là mô hình được sử dụng phổ biến nhất trong các dự án PTHT
- Việc PTHT dựa trên một tập hợp các giai đoạn (phases) có thứ tự liên tiếp
  - Trật tự (thứ tự) của các giai đoạn là xác định, và các kết quả của một giai đoạn trước sẽ được sử dụng làm đầu vào (input) cho các giai đoạn sau
- Một khi tiến trình PTHT kết thúc và hệ thống phần mềm được bàn giao (signed off) cho khách hàng, thì hệ thống phần mềm sẽ không thể được thay đổi, điều chỉnh
  - Tiến trình PTHT chỉ có thể được mở lại (để đáp ứng các điều chỉnh, thay đổi) thông qua một quy trình thực hiện thay đổi chính thức (a formal change process)
- Đặc điểm quan trọng nhất của Quy trình thác nước: **các giai đoạn (phases) không giao nhau, không lặp lại** (trong một tiến trình PTHT)
  - Giai đoạn Thiết kế (Design) không thể bắt đầu cho đến khi giai đoạn Phân tích (Analysis) được hoàn thành, và Giai đoạn Kiểm thử (Testing) không thể bắt đầu cho đến khi giai đoạn Thực hiện, lập trình (Implementation) được hoàn thành

8

8

Mô hình thác nước (Waterfall model)

- Các ưu điểm
  - Là quy trình PTHT đơn giản, dễ hiểu, và dễ sử dụng
  - Các tài liệu được hoàn thành sau mỗi giai đoạn
  - Các yêu cầu phần mềm được cung cấp sớm cho các người kiểm thử (the testers)
  - Cho phép người quản lý dự án (Project Manager – PM) lập kế hoạch và kiểm soát thực hiện một cách chặt chẽ
  - Quy trình này cũng rất nổi tiếng và được biết bởi cả những người không chuyên về PTHT, giúp nó dễ dàng được dùng để trao đổi
- Các nhược điểm
  - Chỉ phù hợp đối với các bài toán thực tế **khi mà các yêu cầu phần mềm được xác định rõ ràng, đầy đủ và cố định từ đầu** (trước giai đoạn Thiết kế)
  - Không phù hợp đối với các dự án kéo dài và tiếp diễn lâu
  - Có thể có nhiều nguy cơ (risk) và không chắc chắn (uncertainty)
  - Khó (không thể) sớm có các kết quả (phiên bản) ban đầu của phần mềm

9

9

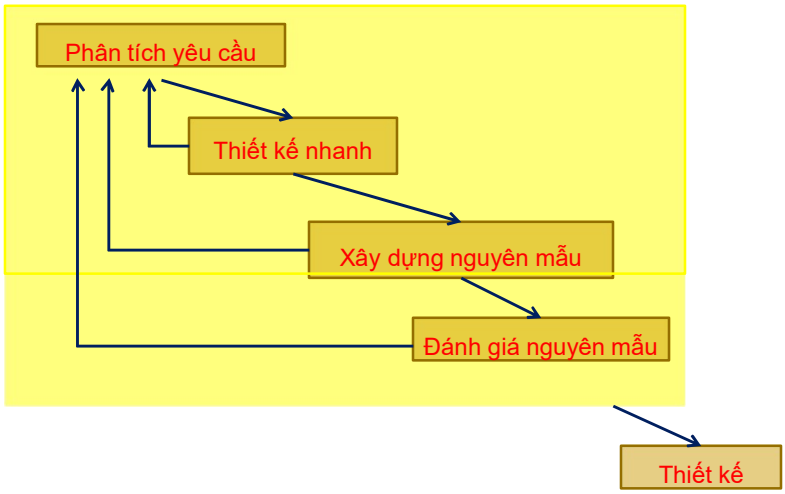
Mô hình thác nước (Waterfall model)

- Khi nào nên sử dụng mô hình thác nước?
  - Khi các yêu cầu phần mềm được xác định rõ ràng, đầy đủ và cố định
  - Định nghĩa về sản phẩm (hệ thống phần mềm) không thay đổi
  - Các công nghệ liên quan cần thiết được nắm vững
  - Các nguồn lực và kinh nghiệm của nhóm PTHT đủ đáp ứng
  - Thời gian thực hiện dự án ngắn (không kéo dài)

10

10

Mô hình nguyên mẫu (Prototyping model)



11

11

Mô hình nguyên mẫu (Prototyping model)

- Thay vì cố định các yêu cầu trước khi tiến hành thiết kế hoặc thực hiện (lập trình), **một (hoặc một số các) nguyên mẫu (prototype) được xây dựng để hiểu chính xác các yêu cầu phần mềm**
- Mỗi nguyên mẫu (prototype) được xây dựng dựa trên các yêu cầu phần mềm hiện thời (thu được từ đánh giá các nguyên mẫu trước)
- Nhờ việc sử dụng thử nguyên mẫu, khách hàng có thể có được "cảm nhận thực tế" về hệ thống phần mềm, bởi vì các tương tác với nguyên mẫu cho phép khách hàng hiểu rõ hơn, chính xác hơn về các yêu cầu của hệ thống phần mềm mong muốn
- Sử dụng nguyên mẫu là hợp lý đối với việc phát triển các hệ thống phần mềm lớn và phức tạp (khi không có quy trình thu thập yêu cầu hoặc hệ thống sẵn có nào giúp xác định các yêu cầu phần mềm)
- Một nguyên mẫu thường không phải là một hệ thống phần mềm hoàn chỉnh/hoàn thiện, và rất nhiều các chi tiết không được xây dựng trong nguyên mẫu

12

12

Mô hình nguyên mẫu (Prototyping model)

- Các ưu điểm
  - Người sử dụng được tham gia tích cực vào trong quá trình PTHT
  - Sử dụng nguyên mẫu là một mô hình hoạt động của hệ thống, những người sử dụng hiểu rõ hơn về hệ thống đang được xây dựng
  - Các lỗi, vấn đề có thể được phát hiện từ (rất) sớm
  - Sớm có được các phản hồi đánh giá từ người sử dụng, giúp có được các giải pháp PTHT tốt hơn
  - Các chức năng còn thiếu có thể được phát hiện sớm
  - Các chức năng không rõ ràng hoặc khó thao tác có thể được phát hiện
- Các nhược điểm
  - Người sử dụng có thể nghĩ rằng việc phát triển phần mềm là dễ dàng, và vì vậy trở nên không nhất quán trong việc diễn đạt các yêu cầu
  - Không có việc lập kế hoạch ngay từ đầu, có thể dẫn đến các vấn đề về quản lý dự án: không xác định được thời hạn hoàn thành, ngân sách và các kết quả bàn giao
  - Mô hình này thường dẫn đến kéo dài quá trình PTHT
  - Các người phát triển có xu hướng bàn giao một nguyên mẫu hoạt động cơ bản, thay vì bàn giao một sản phẩm hoàn thiện thực sự

13

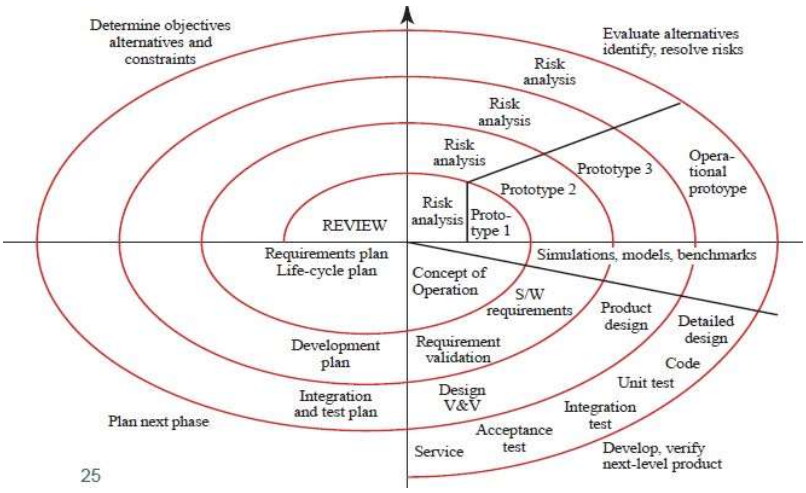
13

Mô hình nguyên mẫu (Prototyping model)

- Khi nào nên dùng mô hình nguyên mẫu?
  - Khi các yêu cầu phần mềm không thể được xác định tại thời điểm bắt đầu dự án
  - **Khi những người sử dụng (vì các lý do khác nhau) không thể diễn đạt các yêu cầu của họ một cách rõ ràng**
  - Mô hình PTHT này rất phù hợp để phát triển "cảm nhận" (look and feel) hoặc giao diện sử dụng của hệ thống, bởi vì các đặc điểm này *rất khó để miêu tả bằng tài liệu*, mà thường thu được thông qua việc dùng thử nghiệm
  - Khi khách hàng yêu cầu chứng minh tính khả thi
  - Khi cần có các demos cho các cấp quản lý ở mức cao
  - Khi các vấn đề về công nghệ cần được thử nghiệm, kiểm tra

14

14



Mô hình xoắn ốc (Spiral model)

15

15

Mô hình xoắn ốc (Spiral model)

- Được đề xuất bởi Barry Boehm
- Là một mô hình phát triển tiến hóa, dựa trên sự kết hợp lai ghép của đặc điểm phát triển lặp (iterative) của Mô hình nguyên mẫu (Prototyping model) và phát triển theo các bước tuần tự (sequential) của Mô hình thác nước (Waterfall model)
  - **Chú trọng vào việc phân tích nguy cơ (risk analysis)**
- Trong mô hình xoắn ốc, hệ thống phần mềm được phát triển qua một chuỗi các phiên bản tăng cường (incremental releases)
  - Trong các bước phát triển lặp ban đầu, thì các phiên bản của hệ thống phần mềm có thể chỉ là các mô hình được phác thảo trên giấy hoặc là các nguyên mẫu (prototypes)
  - Trong các bước phát triển lặp về sau, các phiên bản ngày càng hoàn thiện của hệ thống phần mềm sẽ được tạo ra

16

16

Mô hình xoắn ốc (Spiral model)

- Các ưu điểm
  - Chú trọng vào phân tích rủi ro (risk analysis), nhờ đó giúp giảm thiểu rủi ro trong dự án PTHT
  - Phù hợp đối với các dự án lớn và quan trọng đặc biệt
  - Các chức năng mới có thể được bổ sung vào sau
  - Các phiên bản đầu của hệ thống phần mềm được tạo ra sớm
- Các nhược điểm
  - Chi phí cao (thời gian, nguồn lực, tiền bạc) để áp dụng
  - Việc phân tích rủi ro (risk analysis) đòi hỏi kỹ năng và kinh nghiệm cao
  - Thành công của dự án phụ thuộc rất nhiều vào giai đoạn phân tích rủi ro (risk analysis)
  - Không phù hợp cho các dự án nhỏ

17

17

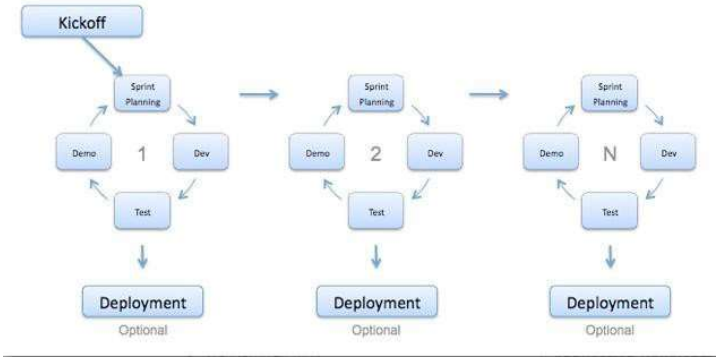
Mô hình xoắn ốc (Spiral model)

- Khi nào nên dùng mô hình xoắn ốc?
  - **Khi việc đánh giá (phân tích) các chi phí và các rủi ro là quan** trọng
  - Đối với các dự án có độ rủi ro trung bình đến cao
  - Các người sử dụng không chắc chắn về các nhu cầu của họ
  - Các yêu cầu phần mềm phức tạp và lớn
  - Cần phát triển một dòng sản phẩm mới (New product line)
  - Mong muốn có các thay đổi quan trọng (cần nghiên cứu và khảo sát cẩn thận)

18

18

Mô hình nhanh lẹ (Agile model)



19

19

Mô hình nhanh lẹ (Agile model)

- Là một kiểu mô hình tăng cường (incremental) và lặp lại (iterative)
- Hệ thống phần mềm được phát triển thông qua các vòng phát triển nhanh, tăng cường (incremental and rapid cycles)
- Giúp tạo ra các phiên bản tăng cường ở mức nhỏ, trong đó mỗi phiên bản tiếp theo được xây dựng dựa trên các tính năng của phiên bản liền trước đó
- Mỗi phiên bản tăng cường được kiểm thử cẩn thận để đảm bảo chất lượng phần mềm
- **Được sử dụng cho các dự án PTHT đòi hỏi thời gian hoàn thành nhanh chóng**
  - Lập trình nhanh (Extreme Programming – XP) là một trong những phương pháp phát triển phần mềm nổi tiếng thuộc nhóm mô hình nhanh lẹ

20

20

Mô hình nhanh lẹ (Agile model)

- Các ưu điểm
  - Thỏa mãn khách hàng với các phiên bản phần mềm tăng cường mau lẹ
  - Nhấn mạnh đến sự tương tác giữa các tác nhân hơn là quá trình và các công cụ (Khách hàng, người phát triển, và các người kiểm thử liên tục tương tác trao đổi với nhau)
  - Trao đổi thường xuyên giữa nhóm phân tích nghiệp vụ và nhóm lập trình
  - **Thích nghi (đáp ứng) nhanh với các yêu cầu thay đổi**
  - Thậm chí cho phép các thay đổi yêu cầu phần mềm được bổ sung sau
- Các nhược điểm
  - Đối với các hệ thống phần mềm lớn, mô hình này khó đánh giá được các chi phí (effort) cần thiết tại thời điểm bắt đầu tiến trình PTHT
  - Ít chú trọng đến các thiết kế và tài liệu cần thiết
  - Thường chỉ các người lập trình có kinh nghiệm (senior programmers) mới có khả năng đưa ra các quyết định cần thiết trong quá trình phát triển. (Không phù hợp cho các người lập trình ít kinh nghiệm, trừ khi phải làm việc kết hợp với các người có kinh nghiệm)

21

21

Mô hình nhanh lẹ (Agile model)

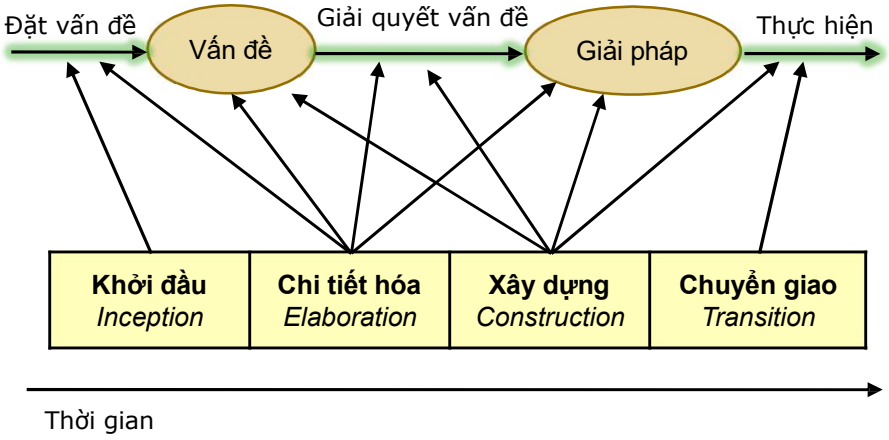
- Khi nào nên dùng mô hình nhanh lẹ?
  - Khi các thay đổi cần được bổ sung và thực hiện. Các thay đổi mới có thể được thực hiện với chi phí thấp, nhờ việc thường xuyên tạo ra các phiên bản tăng cường
  - Để thực hiện một tính năng mới (a new feature), những người lập trình chỉ cần tốn thời gian vài ngày, hoặc thậm chí là chỉ vài giờ để thực hiện
  - Không như mô hình thác nước, trong mô hình nhanh lẹ, thì việc lập kế hoạch tốn chi phí thấp hơn (nhiều). **Mô hình nhanh lẹ giả sử rằng các nhu cầu của người dùng sẽ có thể (thường xuyên) thay đổi. Các thay đổi luôn luôn có thể được yêu cầu và các tính năng luôn luôn có thể được bổ sung hoặc loại bỏ dựa trên các phản hồi**. Điều này giúp đem lại cho khách hàng hệ thống phần mềm mà họ cần và muốn dùng
  - Cả người phát triển và người sử dụng hệ thống đều thấy họ được tự do về thời gian và các lựa chọn hơn là so với các mô hình tuân thủ chặt chẽ theo trình tự các bước (vd: mô hình thác nước)

22

22

Mô hình hợp nhất (Unified model)

- Góc nhìn quản lý dự án

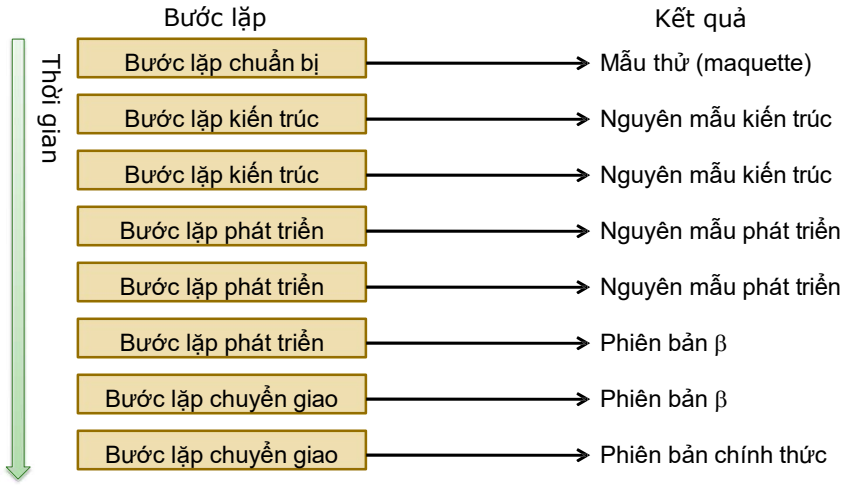


23

23

Mô hình hợp nhất (Unified model)

- Góc nhìn kỹ thuật

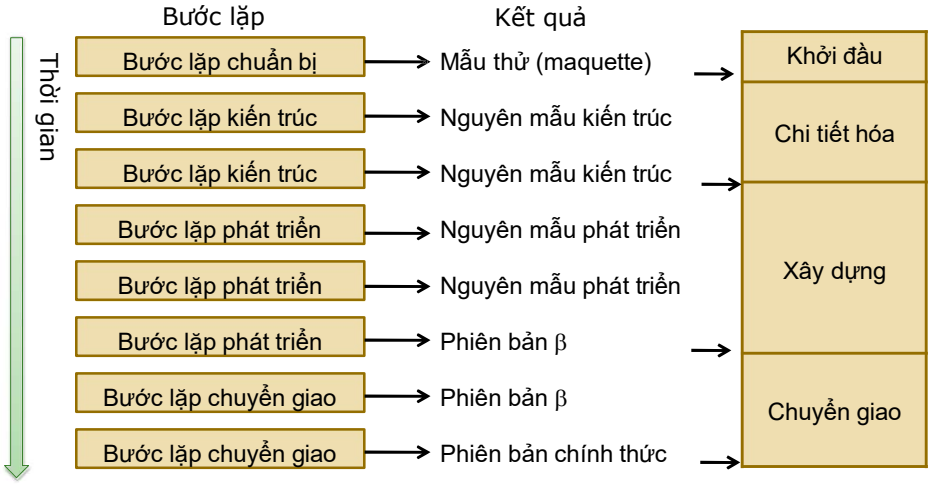


24

24

Mô hình hợp nhất (Unified model)

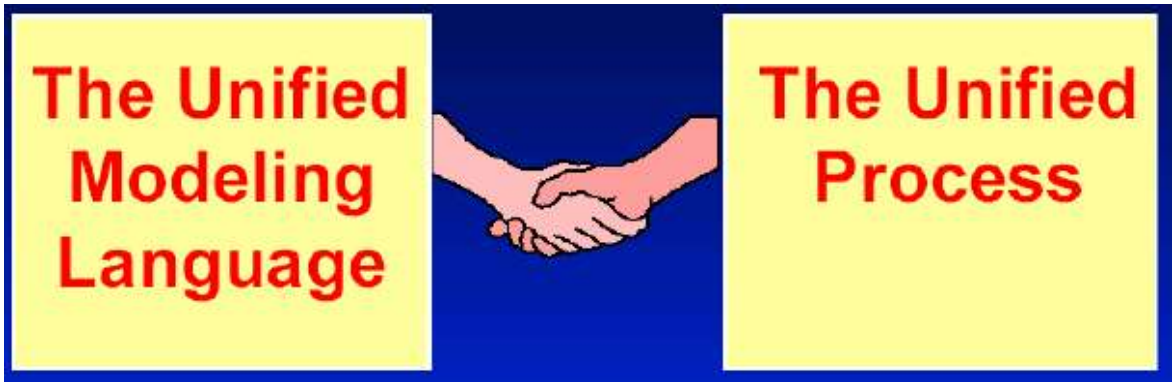
- Kết hợp 2 góc nhìn



25

25

Mô hình hợp nhất và UML



26

26

Quy trình RUP

- RUP (Rational Unified Process) là một quy trình mô hình hóa với UML:
  - Các nguyên tắc cơ bản
  - Các giai đoạn chính (phases)
  - Các bước chính (steps)

27

27

Các nguyên tắc cơ bản (1)

- Lặp và tăng trưởng
  - Dự án được chia thành những vòng lặp hoặc giai đoạn ngắn để dễ dàng kiểm soát
  - Cuối mỗi vòng lặp, phần thi hành được của hệ thống phần mềm được sản sinh theo cách thêm vào dần dần
- Tập trung vào kiến trúc
  - Hệ thống phức tạp được chia thành các mô-đun để dễ dàng triển khai và bảo trì
  - Kiến trúc này được trình bày theo 5 góc nhìn khác nhau

28

28

Các nguyên tắc cơ bản (2)

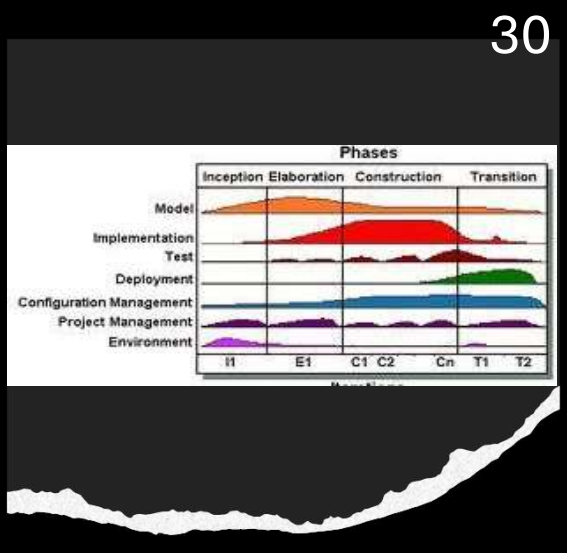
- Dẫn dắt theo các ca sử dụng (use cases)
  - Nhu cầu người dùng thể hiện bởi các ca sử dụng. Các ca sử dụng ảnh hưởng xuyên suốt cho mọi giai đoạn phát triển hệ thống, là cơ sở xác định vòng lặp và tăng trưởng, là căn cứ để phân chia công việc trong nhóm
  - **Nắm bắt nhu cầu**: Phát hiện các ca sử dụng
  - **Phân tích**: Đi sâu vào mô tả các ca sử dụng
  - **Thiết kế và cài đặt**: Xây dựng hệ thống theo các ca sử dụng
  - **Kiểm thử và nghiệm thu hệ thống**: Thực hiện theo các ca sử dụng
- Khống chế các nguy cơ (risks)
  - Phát hiện sớm và loại bỏ các nguy cơ đối với dự án PTHT

29

29

Các giai đoạn của RUP (1)

- RUP được tổ chức thành 4 giai đoạn: Khởi đầu (Inception), Chi tiết hóa (Elaboration), Xây dựng (Construction) và Chuyển giao (Transition)



30

Các giai đoạn của RUP (2)

31

- Giai đoạn khởi đầu (Inception)
  - Cho một cái nhìn tổng quát về hệ thống phần mềm (chức năng, hiệu năng, công nghệ,...) và về dự án PTHT sẽ triển khai (phạm vi, mục tiêu, tính khả thi,...) => Đưa ra kết luận nên phát triển dự án hay loại bỏ?
- Giai đoạn chi tiết hóa (Elaboration)
  - Phân tích chi tiết hơn về hệ thống (chức năng và cấu trúc tĩnh)
  - Đề xuất một kiến trúc hệ thống (nguyên mẫu)

31

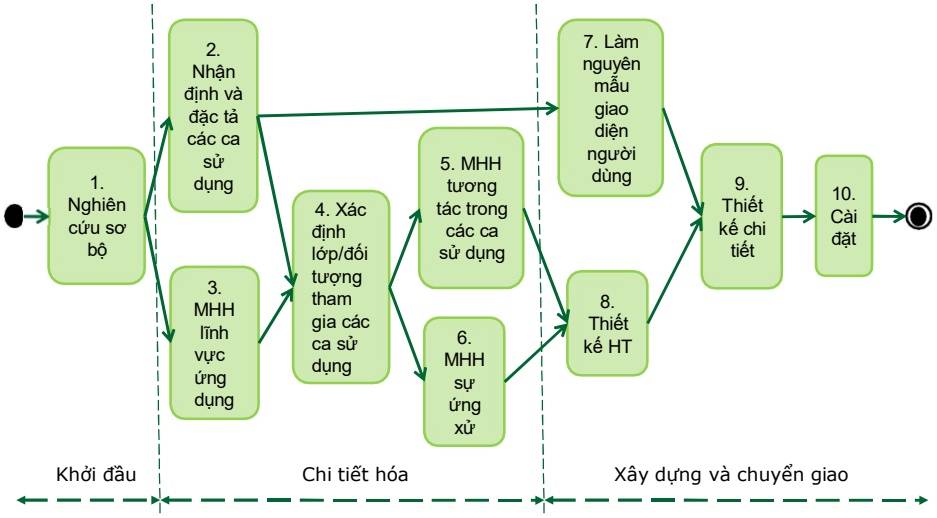
Các giai đoạn của RUP (3)

32

- Giai đoạn xây dựng (Construction)
  - Tập trung vào thiết kế và cài đặt hệ thống
  - Kiến trúc hệ thống được chi tiết hóa, chỉnh sửa
  - Kết thúc khi đã cho ra được 1 hệ thống hoàn chỉnh với tài liệu kỹ thuật đi kèm
  - Là giai đoạn tốn nhiều thời gian, công sức nhất
- Giai đoạn chuyển giao (Transition)
  - Chuyển giao hệ thống cho người dùng cuối: chuyển đổi dữ liệu, lắp đặt, kiểm tra, đào tạo,...

32

Các bước chính của RUP (1)



33

33

Các bước chính của RUP (2)

- Nghiên cứu sơ bộ
  - Đưa ra cái nhìn khái quát về hệ thống phần mềm và dự án PTHT
  - Đưa ra kết luận: nên/không nên triển khai dự án?
- Nhận định và đặc tả các ca sử dụng
  - Nắm bắt nhu cầu của người dùng, phát hiện các ca sử dụng
  - Mỗi ca sử dụng phải được đặc tả (được mô tả) dưới dạng kịch bản và/hoặc một biểu đồ trình tự
- Mô hình hóa lĩnh vực ứng dụng
  - Đưa ra mô hình biểu đồ lớp phản ánh mọi khái niệm, nghiệp vụ
  - Các lớp ở đây là các lớp lĩnh vực (không phải là các lớp đối tượng)

34

34

Các bước chính của RUP (3)

- **Xác định các đối tượng/lớp tham gia ca sử dụng**
  - Với mỗi ca sử dụng, phát hiện các lớp lĩnh vực, lớp điều khiển, lớp biên
- **Mô hình hóa tương tác trong các ca sử dụng**
  - Các đối tượng tương tác bằng cách trao đổi thông điệp
  - Tạo kịch bản của ca sử dụng: biểu đồ trình tự, biểu đồ giao tiếp
- **Mô hình hóa sự ứng xử**
  - Các đối tượng điều khiển có khả năng ứng xử đối với các sự kiện đến từ bên ngoài để điều khiển
  - Sử dụng biểu đồ máy trạng thái để mô tả hành vi ứng xử của các đối tượng điều khiển

35

35

Các bước chính của RUP (4)

- **Làm nguyên mẫu giao diện người dùng**
  - Sử dụng các bộ tạo lập giao diện người dùng (graphical user interface – GUI) để làm sớm nguyên mẫu giao diện, giúp cho việc mô hình hóa và cài đặt hệ thống dễ dàng hơn
- **Thiết kế hệ thống**
  - Thiết kế kiến trúc tổng thể của hệ thống
  - Chia thành các hệ thống con, chọn lựa loại hình điều khiển thích hợp
  - Dùng biểu đồ thành phần mô tả các thành phần vật lý
  - Dùng biểu đồ triển khai mô tả cách bố trí, triển khai các thành phần thực thi của hệ thống vào các phần cứng và nền tảng hạ tầng
  - Kiến trúc khách/chủ (client/server) là một kiến trúc hệ thống hay được sử dụng

36

36

Các bước chính của RUP (5)

- **Thiết kế chi tiết**
  - Thiết kế chi tiết đối với các lớp, các liên kết, các thuộc tính, các phương thức
  - Xác định các giải pháp cài đặt hệ thống
- **Cài đặt**
  - Lập trình và kiểm thử
  - Hệ thống được nghiệm thu dựa theo các ca sử dụng

37

37

Các công cụ trợ giúp (1)

- Hỗ trợ việc lập trình phát triển hệ thống (Integrated Development Environment – IDE)
  - Soạn thảo, biên dịch
  - Gỡ lỗi, kiểm thử
  - Xây dựng nguyên mẫu giao diện
- Hỗ trợ việc mô hình hóa (Modeling tools)
  - Sản sinh, biến đổi, điều chỉnh các mô hình và biểu đồ
  - Kiểm tra cú pháp của các mô hình
  - Lưu trữ và quản lý phiên bản các mô hình
  - Kiểm thử và đánh giá các mô hình
  - Mô phỏng và thực hiện mô hình
  - Sinh ngược mô hình từ phần mềm có sẵn

38

38

Các công cụ trợ giúp (2)

- Hỗ trợ quy trình phát triển hệ thống và quản lý dự án
  - Dẫn dắt và hỗ trợ trực tuyến, chỉ rõ công việc và sản phẩm của các giai đoạn
  - Hỗ trợ tiến trình lặp
  - Hỗ trợ làm việc nhóm
  - Tích hợp được với các công cụ (tools) khác
  - Trợ giúp quản lý, lên kế hoạch, theo dõi quá trình thực hiện dự án

39

39

Các công cụ trợ giúp (3)

- Rational rose
- Enterprise Architect
- StarUML
- Visio
- Visual Studio
- Visual Studio Code
- Word

40

40

41