Nhập môn công nghệ phần mềm

Một số mô hình vòng đời phát triển phần mềm

GV: ThS. Ngô Tiến Đức

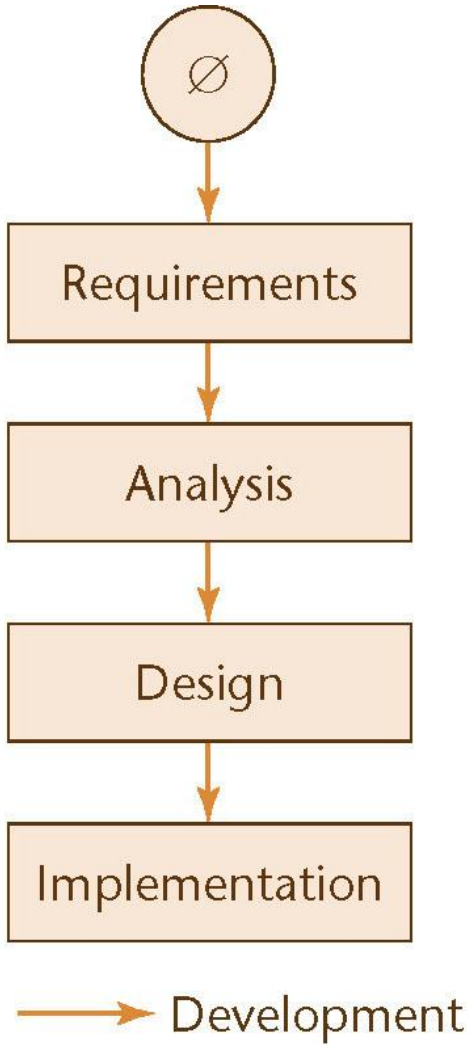
Nội dung chính

- Lý thuyết và thực tế
- Mô hình xây và sửa
- Mô hình thác nước
- Mô hình lặp và tăng trưởng
- Tiến trình linh hoạt
- Mô hình xoắn ốc

Lý thuyết và thực tế (1)

Mô hình trên lý thuyết:

- Các pha được tiến hành tuần tự
- Bắt đầu từ không có gì (from scratch)



Lý thuyết và thực tế (2)

Thực tế hoàn toàn khác:

- Lỗi có thể xảy ra
- Khách hàng thay đổi hoặc không nắm rõ yêu cầu

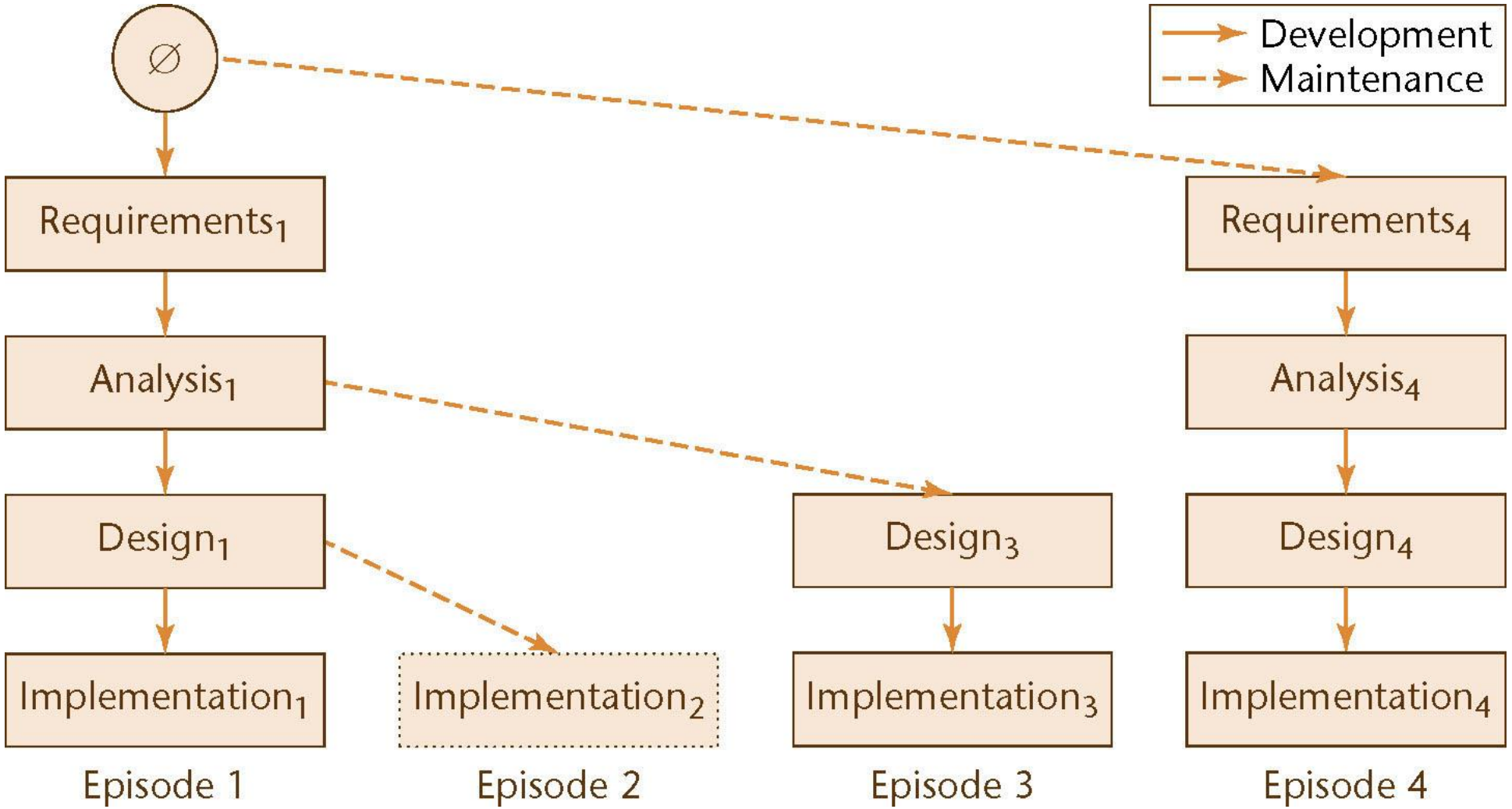
Lý thuyết và thực tế (3)

Moving Target Problem: Khách hàng thay đổi yêu cầu khi phần mềm đang được phát triển

- Ảnh hưởng đến phần mềm
- Có thể dẫn đến lỗi hồi quy (regression fault)
- Có thể dẫn đến phải thiết kế và cài đặt lại

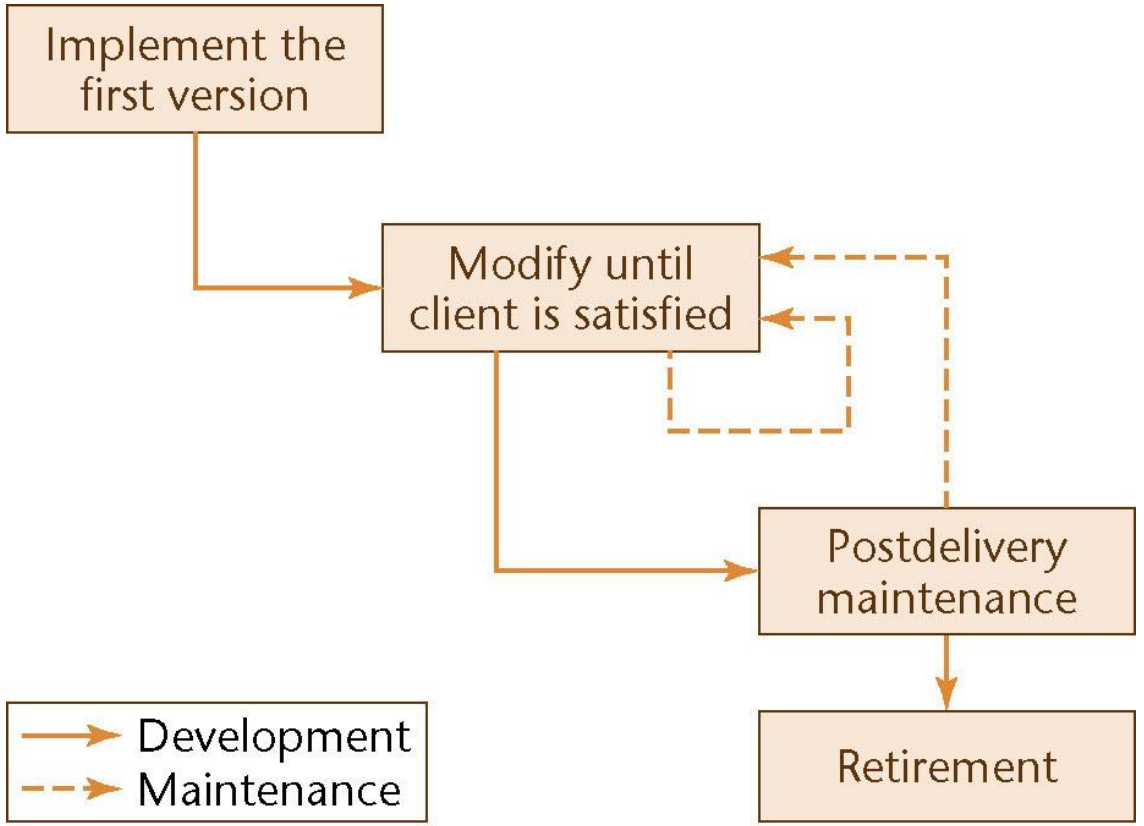
-> Chưa có giải pháp triệt để

Lý thuyết và thực tế (4)



Mô hình xây và sửa

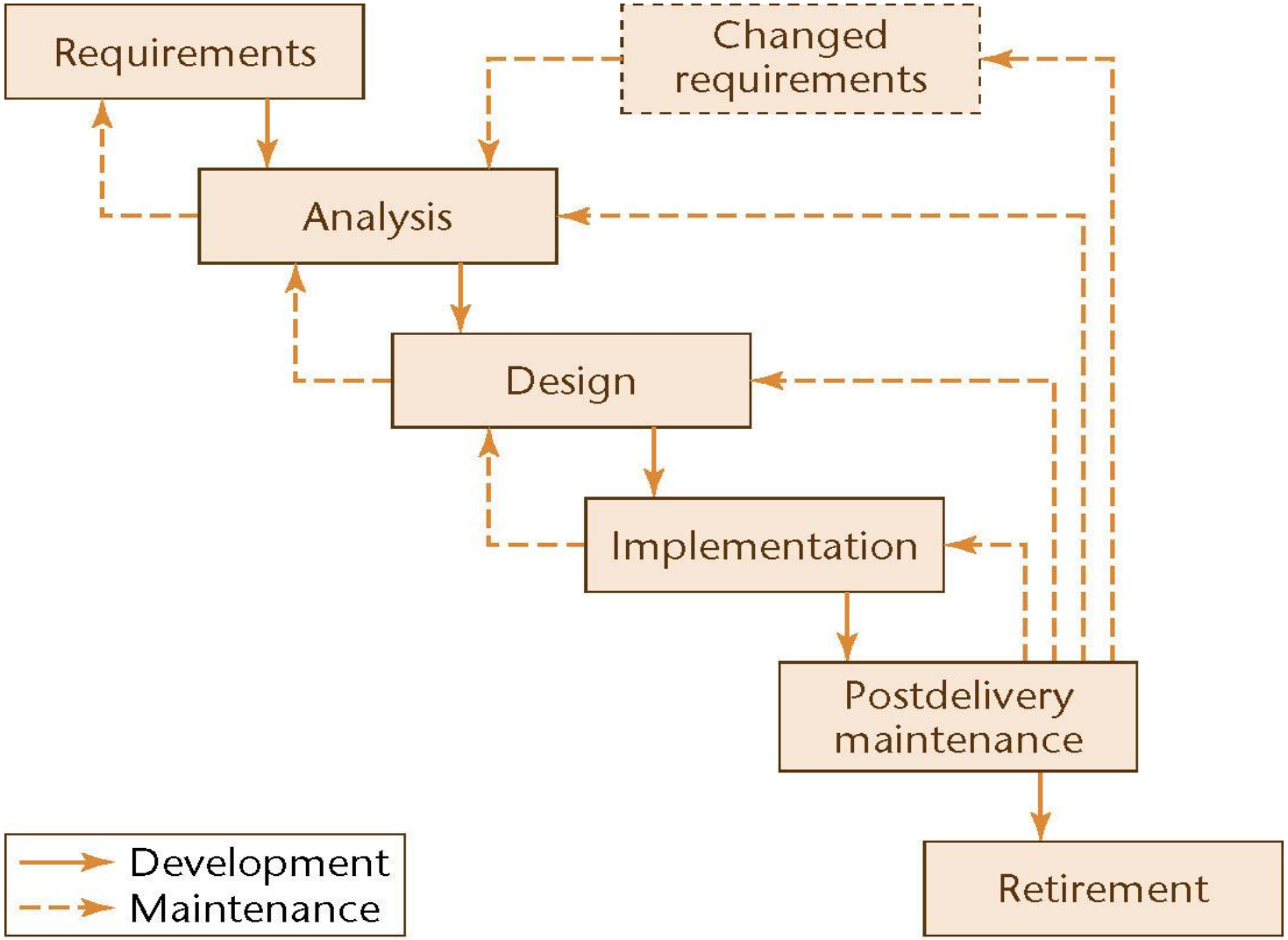
Code-and-fix: Không thiết kế, không đặc tả



Mô hình thác nước (1)

- Mô hình cổ điển
- Làm tài liệu cuối mỗi pha
- Pha sau dựa vào tài liệu pha trước
- Vòng lặp phản hồi sau mỗi pha

Mô hình thác nước (2)

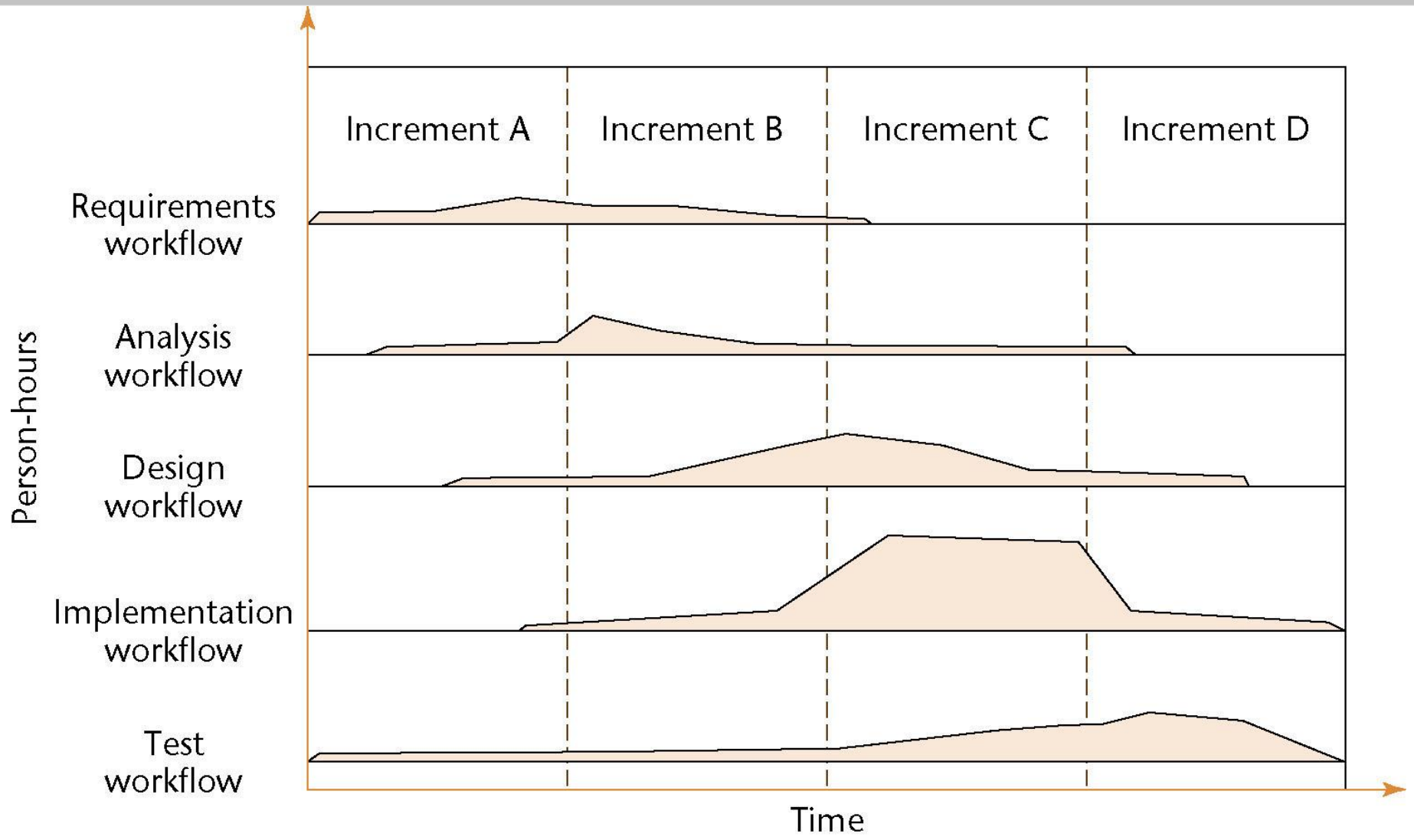


Mô hình lặp và tăng trưởng (1)

Iteration and Incrementation:

- Bản chất: Lặp lại các bước nhiều lần, kết quả lần sau tốt hơn lần trước
- Các pha phát triển:
  - Không kết thúc khi chuyển sang pha khác
  - Kéo dài liên tục trong suốt vòng đời -> gọi là các workflow

Mô hình lặp và tăng trưởng (2)



Mô hình lặp và tăng trưởng (3)

Luật Miller: *Tại mỗi thời điểm, người ta chỉ có thể tập trung vào tối đa khoảng 7 vấn đề.*

Phương pháp làm mịn từng bước để xử lý các vấn đề lớn:

- Tập trung xử lý các việc quan trọng trước
- Các việc ít quan trọng hơn xử lý sau

Mô hình lặp và tăng trưởng (4)

Chia dự án là tập các dự án nhỏ:

- Thực hiện một phần dự án lớn -> tương ứng với 1 lần tăng trưởng
- Có artifact cho mỗi workflow
    - Thay đổi các artifact (tăng trưởng)
    - Kiểm thử các artifact
    - Nếu cần, thay đổi các artifact (lặp)

-> Tập artifact cuối cùng sẽ là sản phẩm hoàn thiện

Mô hình lặp và tăng trưởng (5)

Ưu điểm:

- Mỗi bước lặp đều có test workflow giúp phát hiện và sửa lỗi sớm
- Có phiên bản dùng được của sản phẩm ngay từ đầu
- Bàn giao từng phần
- Khách hàng có thể dựa vào phần đã hoàn thành để yêu cầu cho những modul sau

Tiến trình linh hoạt (1)

Agile: A methodology/approach

- Ít chú trọng phân tích và thiết kế
- Cài đặt sớm hơn: Phần mềm chạy được quan trọng hơn tài liệu
- Thích ứng với thay đổi
- Hợp tác chặt chẽ với khách hàng

Tiến trình linh hoạt (2)

- Tuyên ngôn: “Deliver working software frequently”
  - Lý tưởng nhất là sau 2 hoặc 3 tuần
- Kỹ thuật timeboxing:
  - Thời gian cố định
  - Các thành viên trong nhóm cố gắng hoàn thành công việc trong khoảng thời gian đó

Tiến trình linh hoạt (3)

- Khách hàng: "Tôi sẽ nhận được phiên bản mới với những tính năng bổ sung sau từng đó thời gian"
- Đội phát triển: "Chúng tôi có từng đó thời gian để bổ sung những tính năng mới"
  - Khách hàng không can thiệp trong khoảng thời gian này
  - Nếu không thể hoàn thành: Giảm tải công việc

-> Cố định **thời gian** thay vì **các tính năng**

Tiến trình linh hoạt (4)

Công việc được chia thành các (user) story: "Là ..., tôi muốn ..."

- Story được chia nhỏ thành các task
- Viết test case cho các task trước khi cài đặt
- Ước lượng thời gian cho các story
- Chọn story để phát triển

Tiến trình linh hoạt (5)

Scrum team - Các cuộc họp:

- Refinement and planning
- Retrospective
- Demo

Tiến trình linh hoạt (6)

Phương pháp stand-up meeting:

- Giờ cố định hàng ngày
- Yêu cầu tất cả thành viên tham gia
- Đứng vòng tròn
- Không quá lâu (15 phút)

Tiến trình linh hoạt (7)

Phương pháp stand-up meeting - Trả lời các câu hỏi:

- Tôi đã làm được gì từ buổi meeting ngày hôm qua?
- Hôm nay tôi sẽ làm công việc gì?
- Tôi đang gặp vấn đề gì với công việc đang làm?
- Chúng ta có quên gì không?
- Tôi đã học được gì để có thể chia sẻ với mọi người?

Tiến trình linh hoạt (8)

Phương pháp stand-up meeting:

- Mục đích: Báo cáo tiến độ
    - Nêu ra vấn đề chứ không giải quyết vấn đề
    - Giải pháp cho các vấn đề gặp phải?

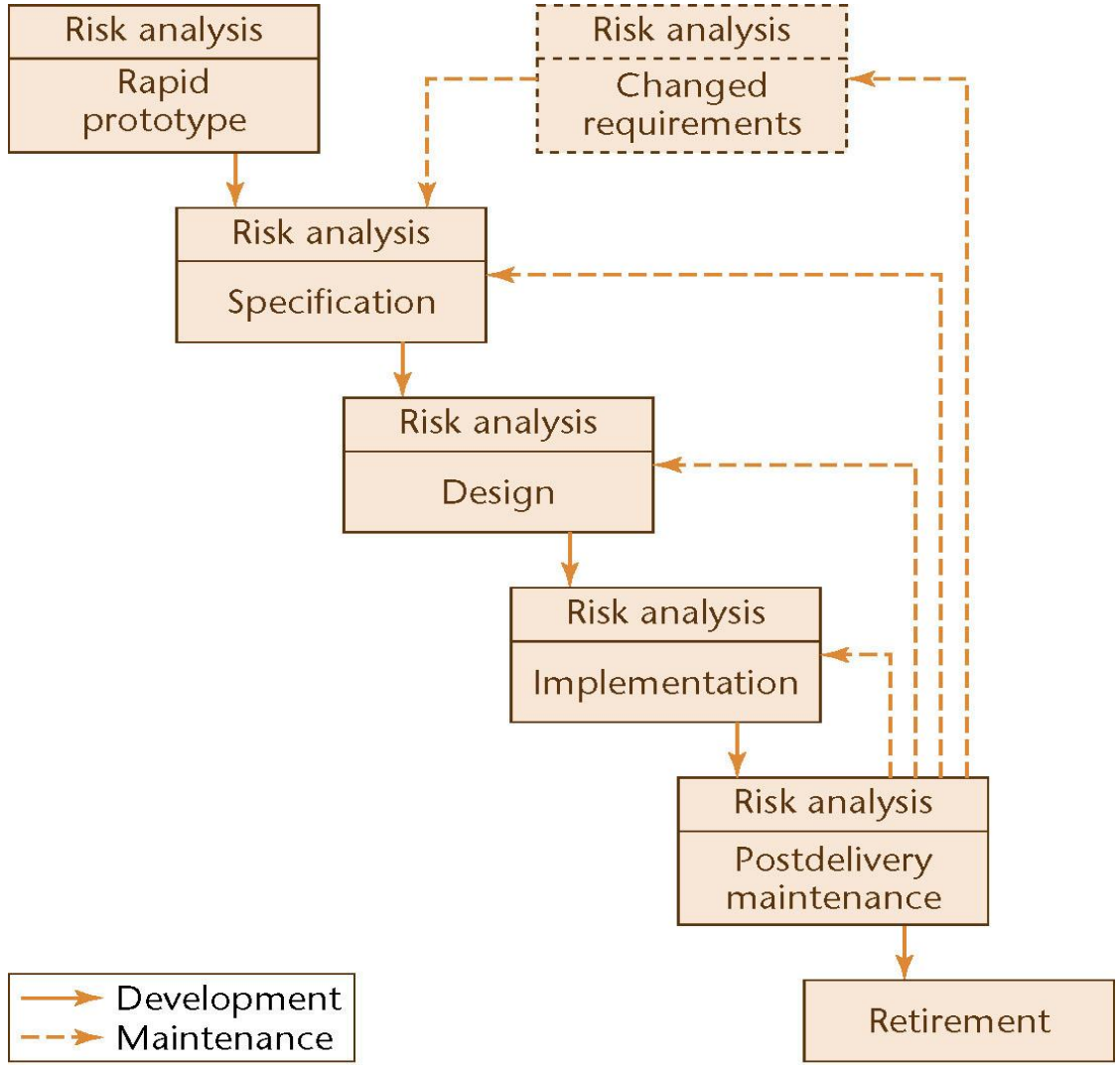
-> Vai trò của giao tiếp và ưu tiên đáp ứng nhu cầu khách hàng một cách nhanh nhất

Mô hình xoắn ốc (1)

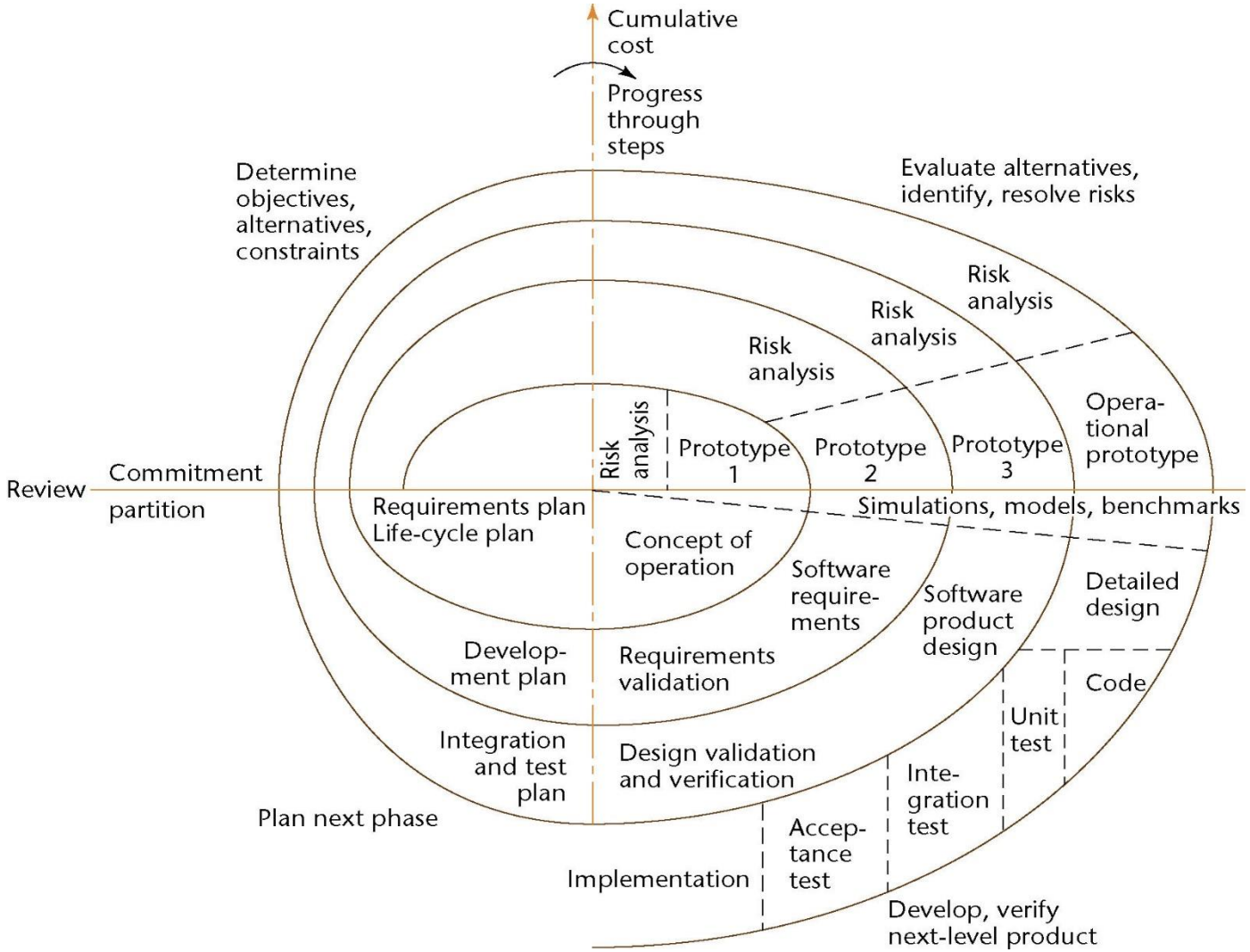
- Hoạt động theo nguyên tắc vòng lặp, vòng sau rộng hơn vòng trước
- Mỗi pha của các lần lặp:
  - Bắt đầu bằng việc phân tích rủi ro
  - Kết thúc bằng đánh giá lỗi và lập kế hoạch pha tiếp theo
  - Nếu rủi ro không xử lý được thì dừng lại ngay lập tức

Mô hình xoắn ốc (2)

- Tập trung vào phân tích rủi ro
- Ưu điểm và nhược điểm?
- Phù hợp với các phần mềm quy mô lớn



Mô hình xoắn ốc (3)



Bài tập tại lớp

Thảo luận nhóm:

- Mô hình bản mẫu nhanh (Rapid Prototype)
- Mô hình mã nguồn mở (Open-source)
- Mô hình xoắn ốc (Spiral)

Bài tập về nhà

Trả lời câu hỏi từ 11 đến 20 trong ngân hàng câu hỏi thi

Tài liệu tham khảo

- Stephen R. Schach. *Object-Oriented and Classical Software Engineering*. 8th Edition, WCB/McGraw-Hill, 2010
- T. Đ. Quế, N. M. Hùng. *Bài giảng Nhập môn công nghệ phần mềm*. HVCNBCVT, 2020