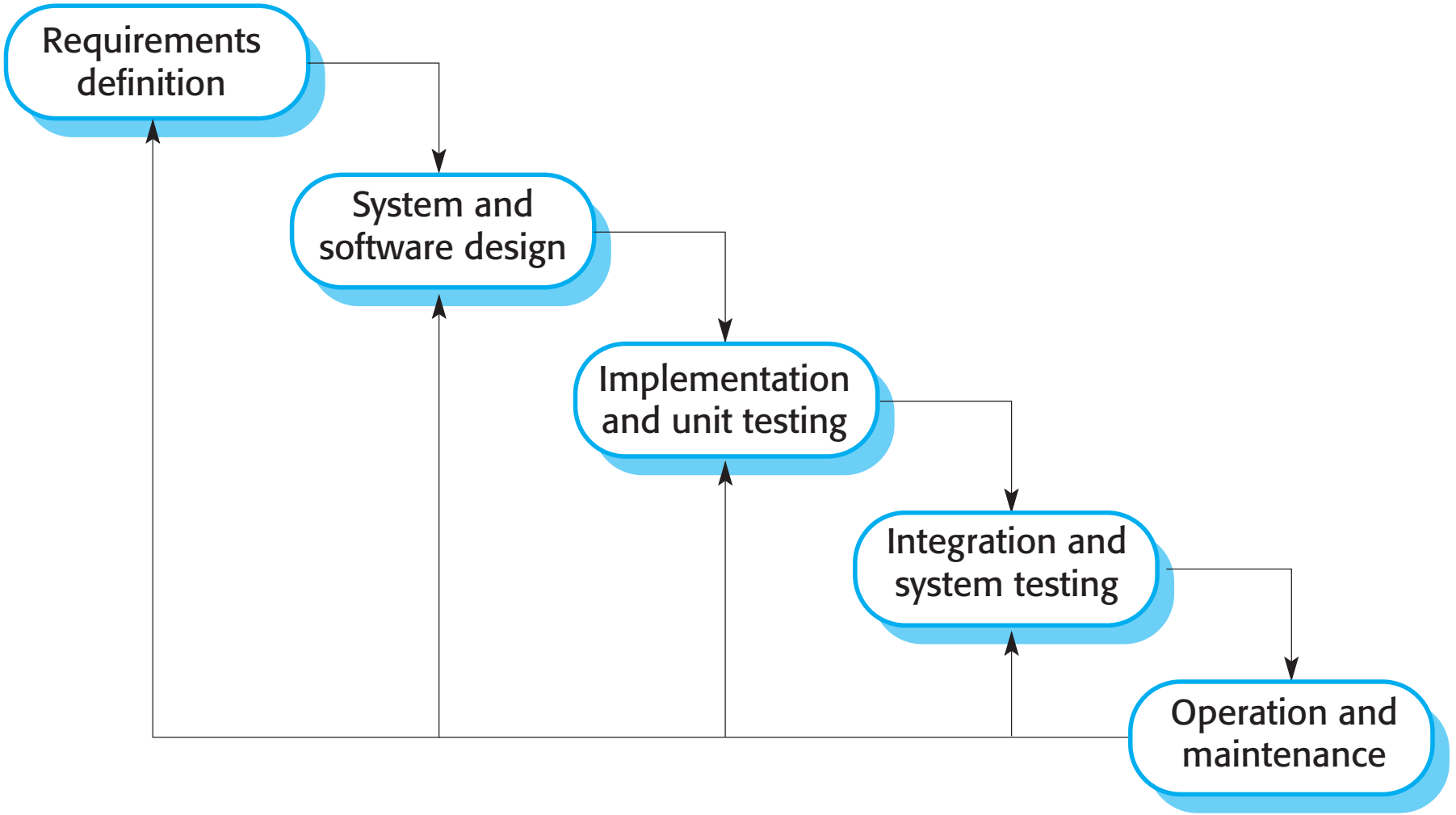
1. Các mô hình quy trình phần mềm

- Mô hình thác nước (water fall)
  - Mô hình kiểu PD (plan-driven). Tách bạch riêng biệt các giai đoạn đặc tả (specification) và phát triển (development).
- Phát triển gia tăng (Incremental development)
  - Đặc tả (Specification), phát triển (development) và xác thực (validation) được xen kẽ. Có thể là plan-driven hoặc agile.
- Tích hợp (Integration) và cấu hình (configuration)
  - Hệ thống được lắp ráp từ những thành phần có thể cấu hình hiện có. Có thể là PD hoặc agile.
- Trong thực tế, hầu hết các **hệ thống lớn** được phát triển bằng cách sử dụng một **quy trình kết hợp tất cả các mô hình trên.**

1.1 Mô hình thác nước



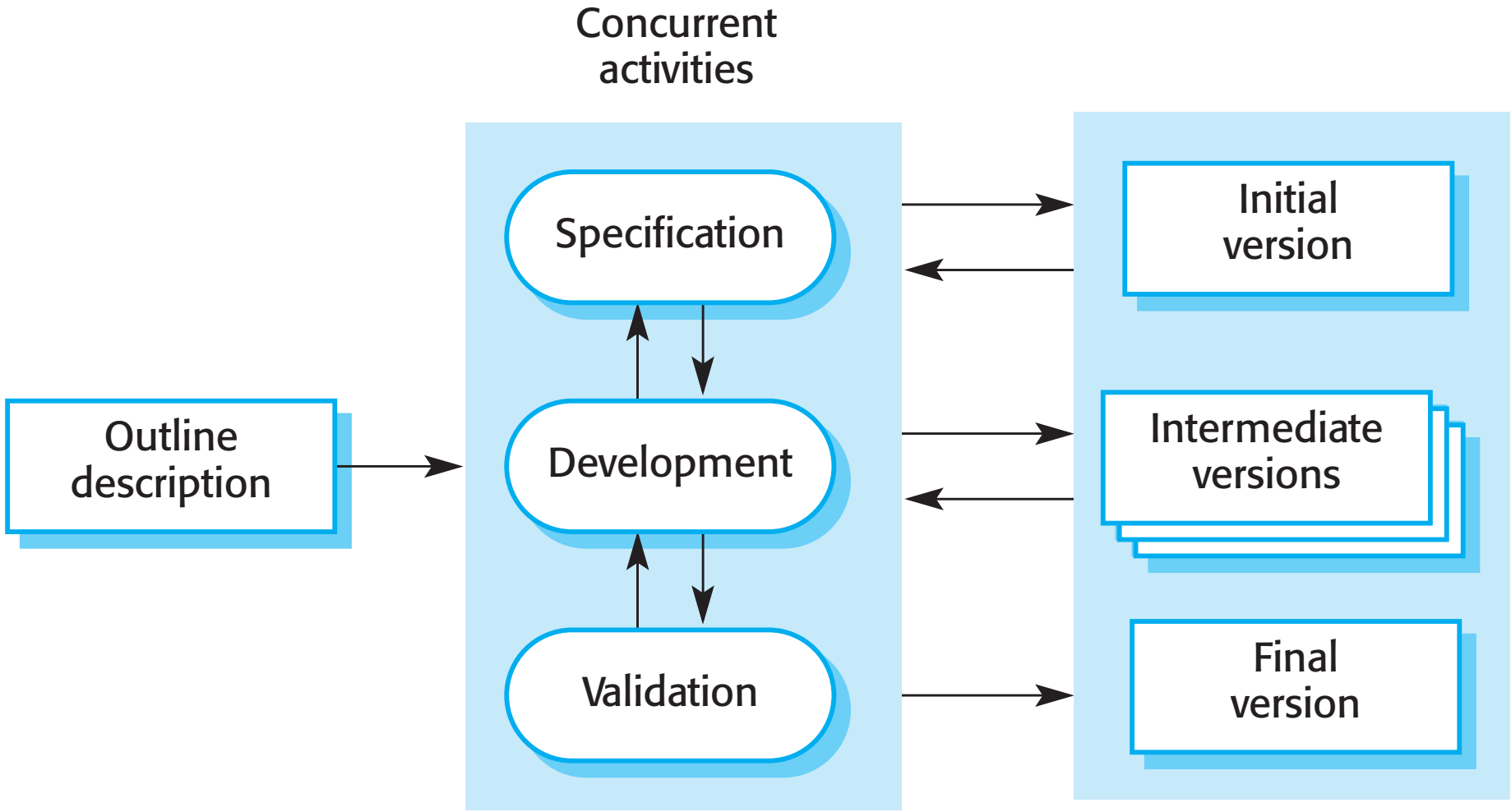
1.1.1 Các giai đoạn của mô hình thác nước

- Các giai đoạn tách biệt trong mô hình thác nước gồm:
  - Phân tích và định nghĩa yêu cầu
  - Thiết kế hệ thống và phần mềm
  - Thực hiện và kiểm thử đơn vị (unit test)
  - Tích hợp và kiểm tra hệ thống
  - Vận hành và bảo trì
- Hạn chế chính của mô hình thác nước là khó đáp ứng sự thay đổi sau khi quá trình này đang được tiến hành.
- Về nguyên tắc, một giai đoạn phải được hoàn thành trước khi chuyển sang giai đoạn tiếp theo.

1.1.2 Các vấn đề của mô hình thác nước

- Việc phân chia dự án thành các giai đoạn riêng biệt, không linh hoạt gây ra khó khăn cho việc đáp ứng các yêu cầu thay đổi của khách hàng.
  - Do đó, mô hình này chỉ thích hợp khi các yêu cầu được hiểu rõ và các thay đổi sẽ khá hạn chế trong quá trình thiết kế.
  - Có rất ít hệ thống kinh doanh có yêu cầu ổn định.
- Mô hình thác nước chủ yếu được sử dụng cho các dự án kỹ thuật hệ thống lớn, nơi một hệ thống được phát triển bởi nhiều bên.
  - Bản chất theo kế hoạch của mô hình thác nước giúp điều phối công việc.

1.2 Phát triển gia tăng (theo thời gian)



1.2.1 Lợi ích của phát triển gia tăng

- Giảm chi phí đáp ứng các yêu cầu của khách hàng.
  - Số lượng phân tích và tài liệu phải được thực hiện lại sẽ ít hơn nhiều so với yêu cầu của mô hình thác nước.
- Dễ dàng nhận được phản hồi của khách hàng về công việc phát triển đã thực hiện.
  - Khách hàng có thể nhận xét về các demo của phần mềm và xem mức độ đã thực hiện.
- Tốc độ phân phối và phát triển phần mềm đưa tới khách hàng hoàn toàn nhanh hơn.
  - Khách hàng có thể sử dụng và nhận được giá trị từ phần mềm sớm hơn so với quy trình thác nước.

1.2.2 Các vấn đề của phát triển gia tăng

- Quy trình không được nhìn thấy.
  - Người quản lý cần được giao kết quả thường xuyên để đo lường tiến độ. Nếu các hệ thống được phát triển một cách nhanh chóng, thì việc tạo ra các tài liệu ghi lại mọi phiên bản của hệ thống sẽ gây tốn kém.
- Cấu trúc hệ thống có xu hướng suy giảm khi các bước tăng mới được thêm vào.
  - Trừ khi dành thời gian và tiền bạc cho việc tái cấu trúc để cải thiện phần mềm, việc thay đổi thường xuyên có xu hướng làm hỏng cấu trúc của nó. Việc kết hợp các thay đổi tăng thêm của phần mềm ngày càng trở nên khó khăn và tốn kém.

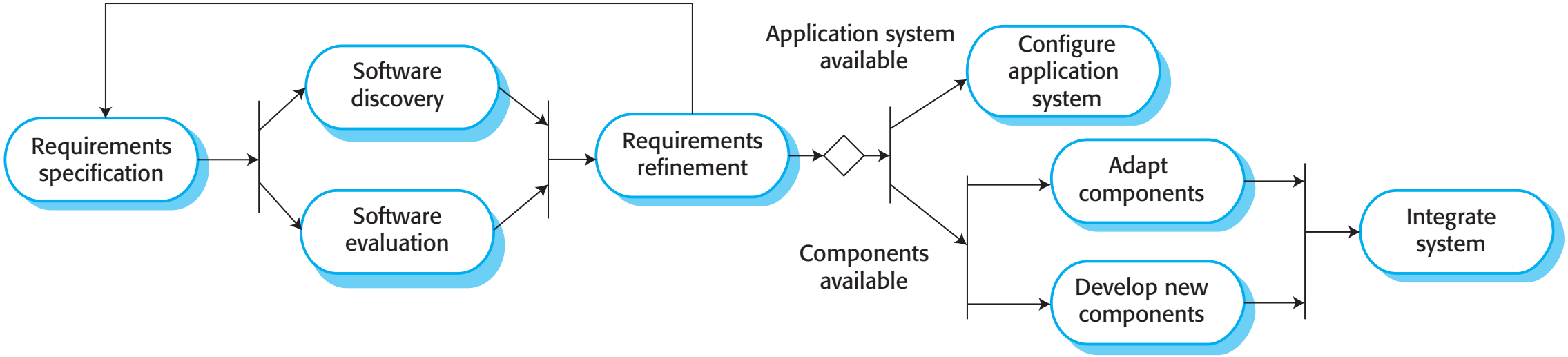
1.3 Tích hợp và cấu hình

- Dựa trên việc tái sử dụng phần mềm trong đó các hệ thống được tích hợp từ các thành phần hiện có hoặc hệ thống ứng dụng (đôi khi được gọi là hệ thống COTS - Commercial-off-the-shelf).
- Các thành phần được sử dụng lại có thể được định cấu hình để điều chỉnh hành vi và chức năng của chúng theo yêu cầu của người dùng.
- Tái sử dụng hiện là cách tiếp cận tiêu chuẩn để xây dựng nhiều loại hệ thống kinh doanh.
  - *Tái sử dụng được đề cập sâu hơn ở chương 15.*

1.3.1 Các loại phần mềm có thể tái sử dụng

- Hệ thống ứng dụng độc lập (đôi khi gọi là COTS) được cấu hình để sử dụng trong một môi trường cụ thể.
- Tập hợp các đối tượng được phát triển dưới dạng package để tích hợp với các thành phần trong framework như .NET hoặc J2EE.
- Các dịch vụ web, những dịch vụ này được phát triển theo các tiêu chuẩn của dịch vụ và có sẵn để gọi từ xa.

1.3.2 Kỹ thuật phát triển phần mềm theo định hướng tái sử dụng



1.3.3 Các giai đoạn chính

- Đặc tả yêu cầu
- Khám phá và đánh giá phần mềm
- Sàng lọc yêu cầu
- Cấu hình hệ thống ứng dụng
- Tích hợp thành phần thích hợp

1.3.4 Ưu điểm và nhược điểm

- Giảm chi phí và rủi ro do ít phần mềm hơn được phát triển từ đầu.
- Phân phối và triển khai hệ thống nhanh hơn.
- **Nhưng** các yêu cầu thỏa hiệp là không thể tránh khỏi, do đó hệ thống có thể không đáp ứng được nhu cầu thực sự của người dùng.
- Mất quyền kiểm soát sự phát triển của các phần tử hệ thống được sử dụng lại.