Thiết kế (6)

Thiết kế ?

- phân tích bài toán/vấn đề
  - xuất phát từ yêu cầu
- mô tả một hoặc nhiều giải pháp
  - đánh giá các giải pháp, chọn giải pháp tốt nhất
- ở một mức trừu tượng nhất định
  - sử dụng các mô hình

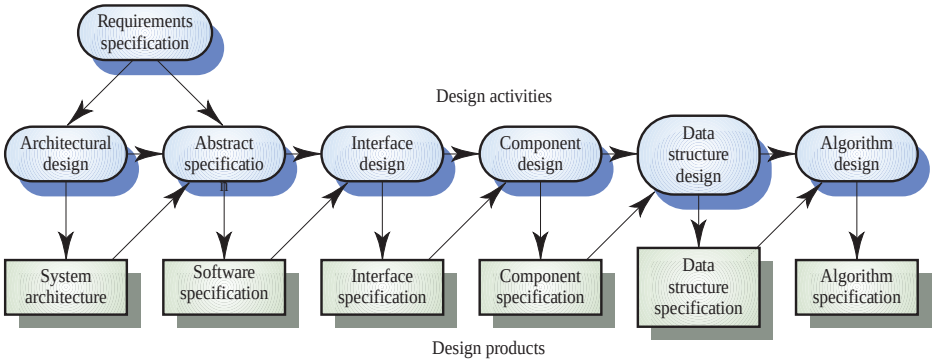
- 3 tính chất
  - trả lời câu hỏi “như thế nào”
  - mô tả chủ yếu là cấu trúc
  - bỏ qua các chi tiết cài đặt
    - giải pháp trừu tượng ≠ giải pháp cụ thể

Các giai đoạn thiết kế

- Hoạt động thiết kế xuất hiện trong các mô hình phát triển khác nhau
- Hai giai đoạn thiết kế chính
  - Thiết kế kiến trúc
    - phân tích giải pháp thành các thành phần
    - định nghĩa giao diện giữa các thành phần
    - định nghĩa phần vấn đề được giải quyết bởi mỗi thành phần
    - có thể được thực hiện bởi nhiều mức trừu tượng
  - Thiết kế chi tiết
    - thiết kế thuật toán, cấu trúc dữ liệu...

3

Các giai đoạn thiết kế



4

Các giai đoạn thiết kế

- ***Architectural design***
  - xác định các hệ thống con
- ***Abstract specification***
  - đặc tả các hệ thống con
- ***Interface design***
  - mô tả giao diện các hệ thống con
- ***Component design***
  - phân tích hệ thống con thành các thành phần
- ***Data structure design***
  - các cấu trúc dữ liệu lưu trữ dữ liệu của bài toán
- ***Algorithm design***
  - thiết kế thuật toán cho các hàm/mô-đun

5

Tại sao phải thiết kế ?

- có một kiến trúc tốt
  - làm chủ được cấu trúc hệ thống
  - “chia để trị”
- đạt được các tiêu chuẩn chất lượng
  - tái sử dụng / dễ keỉem thử / dễ bảo trì...
- thiết kế hướng đến sự thay đổi (design for change)

6

Thiết kế và sự thay đổi

- Thay đổi = tích chất đặc trưng của phần mềm
- Dự báo thay đổi là cần thiết
  - giảm chi phí bảo trì
- Dự báo thay đổi là khó khăn
  - sự thay đổi thường không được xác định trước
  - nhiều yếu tố thay đổi cùng lúc
  - thời điểm thay đổi là khó có thể biết trước

7

Thiết kế và sự thay đổi

- Các yếu tố có thể thay đổi
  - thuật toán
  - cấu trúc dữ liệu
  - biểu diễn dữ liệu bên ngoài
  - thiết bị ngoại vi
  - môi trường xã hội
  - yêu cầu khách hàng

8

Thiết kế hướng mô-đun

- Phần mềm là tập hợp gồm các mô-đun tương tác với nhau
- Mô-đun hóa đóng vai trò quan trọng để có được phần mềm chất lượng với chi phí thấp
- Mục đích thiết kế hệ thống
  - xác định các mô-đun có thể
  - xác định tương tác giữa các mô-đun

9

Các tiêu chuẩn của một phương pháp thiết kế

- Các tiêu chuẩn để đánh giá một phương pháp thiết kế hướng mô-đun
  - tính phân rã (modular decomposability)
  - tính tổng hợp (modular composability)
  - tính dễ hiểu (modular understandability)
  - tính liên tục (modular continuity)
  - tính bảo vệ (modular protection)

10

Các tiêu chuẩn của một phương pháp thiết kế

- tính phân rã (modular decomposability)
  - phân rã vấn đề thành các vấn đề con nhỏ hơn
  - có thể giải quyết các vấn đề con một cách độc lập
  - các phương pháp thiết kế từ trên xuống (to-down design) thỏa mãn tiêu chuẩn này

11

Các tiêu chuẩn của một phương pháp thiết kế

- tính tổng hợp (modular composability)
  - các mô-đun dễ dàng được kết hợp với nhau để tạo nên các hệ thống mới
  - có mối quan hệ chặt chẽ với tính tái sử dụng
  - tính tổng hợp có thể xung đột với tính phân rã
    - phân rã thành các mô-đun chuyên biệt thay vì các mô-đun tổng quát

12

Các tiêu chuẩn của một phương pháp thiết kế

- tính dễ hiểu (modular understandability)
  - thiết kế các mô-đun một cách dễ hiểu
  - tính chất mỗi mô-đun
    - mỗi mô-đun có dễ hiểu ?
    - các tên sử dụng có ý nghĩa ?
    - cso sử dụng thuật toán phức tạp ?
  - Ví dụ
    - ☹ sử dụng “goto”
    - ☹ chương trình vài nghìn dòng lệnh, nhưng không sử dụng hàm/thủ tục

13

Các tiêu chuẩn của một phương pháp thiết kế

- tính liên tục (modular continuity)
  - một sự thay đổi trong đặc tả yêu cầu chỉ dẫn đến sự thay đổi trong một (hoặc một số ít) mô-đun
  - Ví dụ
    - ☺ không sử dụng số hoặc chuỗi ký tự trong chương trình, chỉ được sử dụng các hằng đã định nghĩa
    - ☹ sử dụng mảng

14

Các tiêu chuẩn của một phương pháp thiết kế

- tính bảo vệ (modular protection)
  - kiến trúc đươc thiết kế sao cho nếu một điều kiện bất thường xảy ra, chỉ một (hoặc một số ít) mô-đun bị ảnh hưởng

15

Thiết kế kiến trúc

- Kiến trúc = tập hợp các thành phần/mô-đun và quan hệ giữa chúng
  - các thành phần/mô-đun
    - hàm / nhóm các hàm / lớp ...
  - quan hệ
    - sử dụng / gọi / thừa kế ...

16

Chất lượng của kiến trúc

- mỗi mô-đun có tính kết cố cao (high cohesion)
  - một mô-đun là một đơn vị lô-gíc
  - toàn bộ mô-đun cùng đóng góp thực hiện một mục tiêu
- liên kết lỏng lẽo (low coupling) giữa các mô-đun
  - ít ràng buộc, phụ thuộc lẫn nhau
- dễ hiểu
- định nghĩa rỏ ràng
  - các mô-đun và quan hệ giữa chúng

17

Các loại kiến trúc

- Ba loại mô hình kiến trúc thường được sử dụng
  - chia sẽ dữ liệu: mô hình “Repository”
  - chia sẽ dịch vụ, servers: mô hình “Client-Server”
  - mô hình lớp (layered model)

18

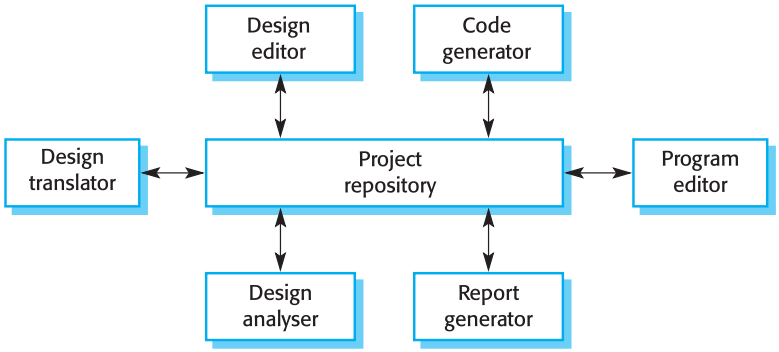
Mô hình “Repository”

- Nguyên tắc
  - dữ liệu chia sẽ được tập trung trong một CSDL
  - các hệ thống con đều truy cập vào CSDL chung
- Khi một lượng dữ liệu lớn cần chia sẽ giữa các hệ thống con
  - mô hình “Repository” thường được sử dụng

19

Mô hình “Repository”

- Ví dụ kiến trúc một công cụ CASE



20

Mô hình “Repository”

- Ưu diểm
  - đơn giản
  - hiệu quả khi chia sẽ lượng dữ liệu lớn
  - sự độc lập của các hệ thống con
- Hạn chế
  - các hệ thống con phải thống nhất trên mô hình dữ liệu “repository”
  - khó khăn khi phân tán dữ liệu

21

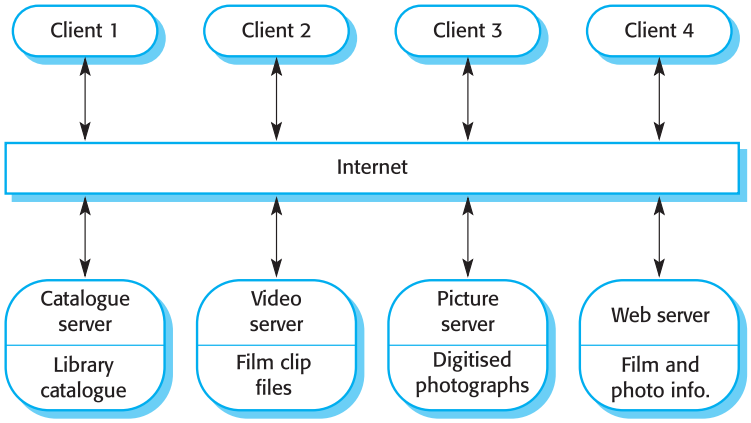
Mô hình “Client-Server”

- Nguyên tắc
  - mô hình phân tán: dữ liệu và xử lý được phân tán trên nhiều thành phần khác nhau
- Hệ thống bao gồm
  - các *servers* cung cấp các dịch vụ
    - có thể có nhiều servers
  - các *clients* yêu cầu các dịch vụ
  - phương thức trao đổi
    - mạng hay trên một máy tính

22

Mô hình “Client-Server”

- Ví dụ



23

Mô hình “Client-Server”

- Ưu điểm
  - sử dụng hiệu quả mạng
  - dễ dàng thêm server mới hoặc nâng cấp server hiện tại
  - phân tán dữ liệu dễ dàng
- Hạn chế
  - mỗi hệ thống con quan lý dữ liệu riêng của nó
    - có thể dẫn đến dư thừa
  - không có kiến trúc tập trung ghi nhận các dich vụ
    - khó khăn để xác định dữ liệu hay dịch vụ sử dụng

24

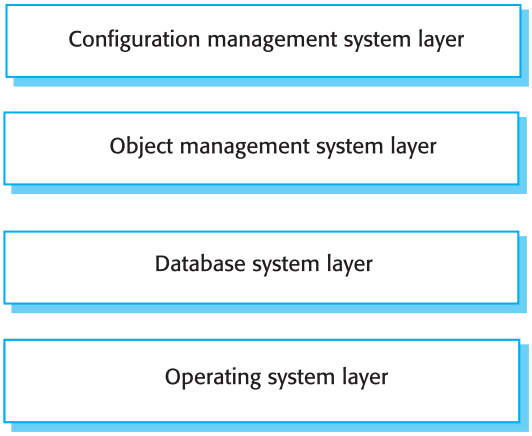
Mô hình lớp

- Nguyên tắc
  - tổ chức hệ thống thành tập hợp các lớp
  - mỗi lớp cung cấp tập hợp các dịch vụ
- được sử dụng để mô tả quan hệ giữa các hệ thống con
- khi giao diện của một lớp thay đổi, chỉ lớp kế cận bị ảnh hưởng
- hỗ trợ mô hình phát triển tăng trưởng

25

Mô hình lớp

- Ví dụ: hệ thống quản lý phiên bản



26