

Nhập môn Công nghệ phần mềm

Tuần 10: Thiết kế và cài đặt



KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Nội dung của slide này được dịch và hiệu chỉnh dựa vào các slides của Ian Sommerville



Nội dung

1. Thiết kế hướng đối tượng sử dụng UML
2. Thiết kế mẫu
3. Các vấn đề về cài đặt



Nội dung

1. Thiết kế hướng đối tượng sử dụng UML
2. Thiết kế mẫu
3. Các vấn đề về cài đặt



Phát triển hướng đối tượng

- Phân tích, thiết kế và lập trình hướng đối tượng có liên quan với nhau nhưng tách rời nhau.
- **Phân tích hướng đối tượng**
 - ▣ liên quan đến việc phát triển mô hình đối tượng của miền ứng dụng.
- **Thiết kế hướng đối tượng**
 - ▣ liên quan đến việc phát triển mô hình hệ thống hướng đối tượng để cài đặt các yêu cầu.
- **Lập trình hướng đối tượng**
 - ▣ liên quan đến việc hiện thực hóa thiết kế hướng đối tượng sử dụng ngôn ngữ lập trình hướng đối tượng.



Đối tượng và lớp đối tượng

- **Đối tượng** là một thực thể có một trạng thái và một tập các thao tác hoạt động trên trạng thái đó.
- **Lớp đối tượng** được sử dụng như một template cho các đối tượng
 - Gồm việc khai báo tất cả các thuộc tính và dịch vụ liên quan đến một đối tượng trong lớp đó.



Quy trình thiết kế hướng đối tượng

- Thiết kế các lớp đối tượng và quan hệ giữa các lớp này.
- Các hệ thống hướng đối tượng thường dễ thay đổi hơn so với hệ thống được phát triển dựa vào các phương pháp hướng chức năng.
 - ▣ Đối tượng bao gồm cả dữ liệu và các thao tác trên dữ liệu → dễ hiểu và dễ thay đổi hơn các thực thể độc lập.
 - ▣ Việc thay đổi cài đặt của một đối tượng hay việc thêm các dịch vụ không nên gây ảnh hưởng đến các đối tượng khác của hệ thống.



Các giai đoạn của quy trình thiết kế

□ Để phát triển thiết kế hệ thống từ khái niệm đến chi tiết:



- Định nghĩa ngữ cảnh và các tương tác bên ngoài với hệ thống



- Thiết kế kiến trúc hệ thống



- Nhận diện các đối tượng chính




- Phát triển các mô hình thiết kế



- Đặc tả giao diện đối tượng



Các giai đoạn của quy trình thiết kế



- Định nghĩa ngữ cảnh và các tương tác bên ngoài với hệ thống



- Thiết kế kiến trúc hệ thống



- Nhận diện các đối tượng chính



- Phát triển các mô hình thiết kế



- Đặc tả giao diện đối tượng



Ngữ cảnh hệ thống và tương tác

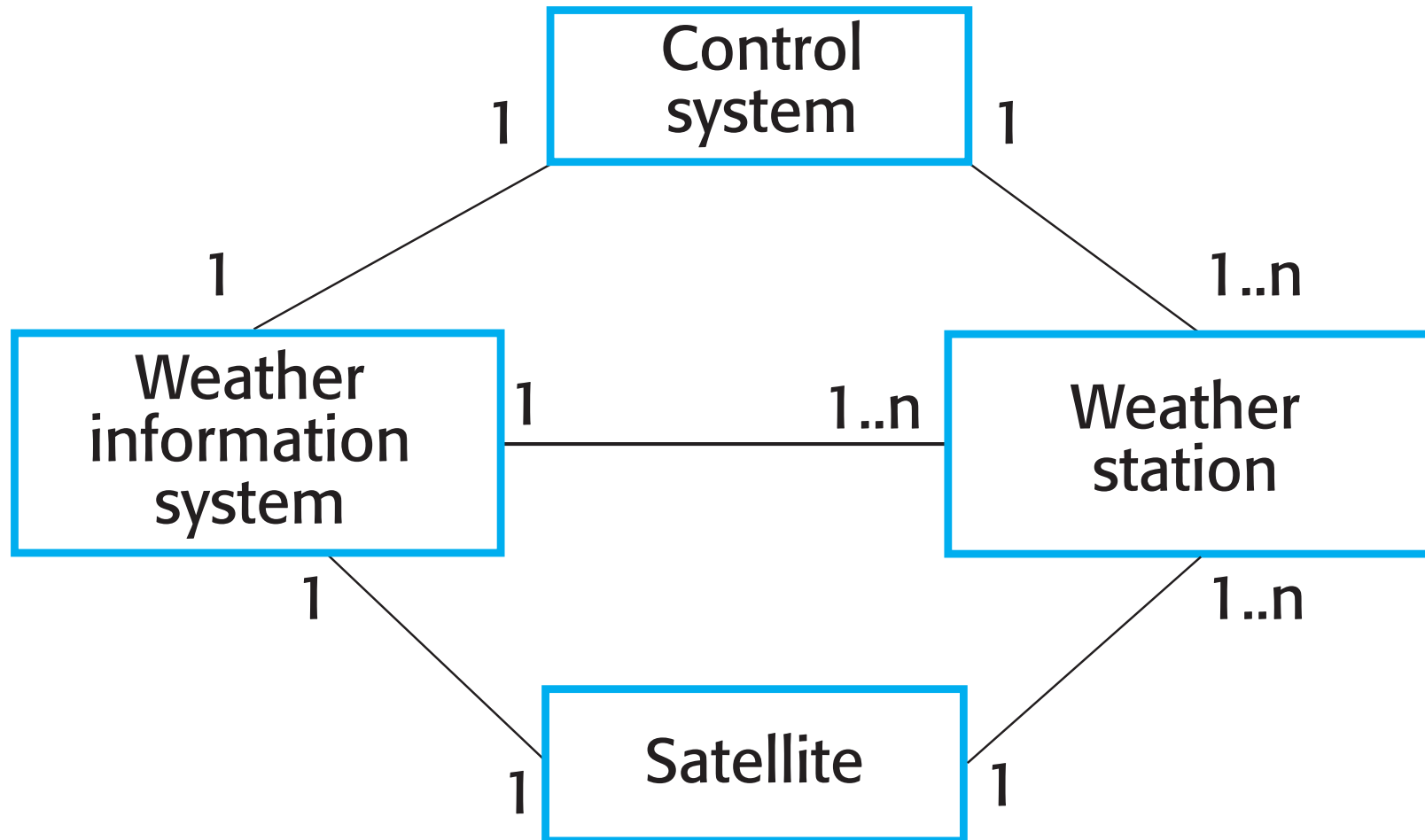
- Việc hiểu mối quan hệ giữa phần mềm đang thiết kế và môi trường bên ngoài là cần thiết
 - ▣ quyết định việc cung cấp các chức năng của hệ thống như thế nào và
 - ▣ cách cấu trúc hoá hệ thống để giao tiếp với môi trường của nó.
- Việc hiểu ngữ cảnh cũng giúp ta
 - ▣ thiết lập ranh giới của hệ thống với môi trường
 - ▣ quyết định xem tính năng nào được cài đặt trong hệ thống đang được thiết kế và tính năng nào nằm trong các hệ thống có liên quan.



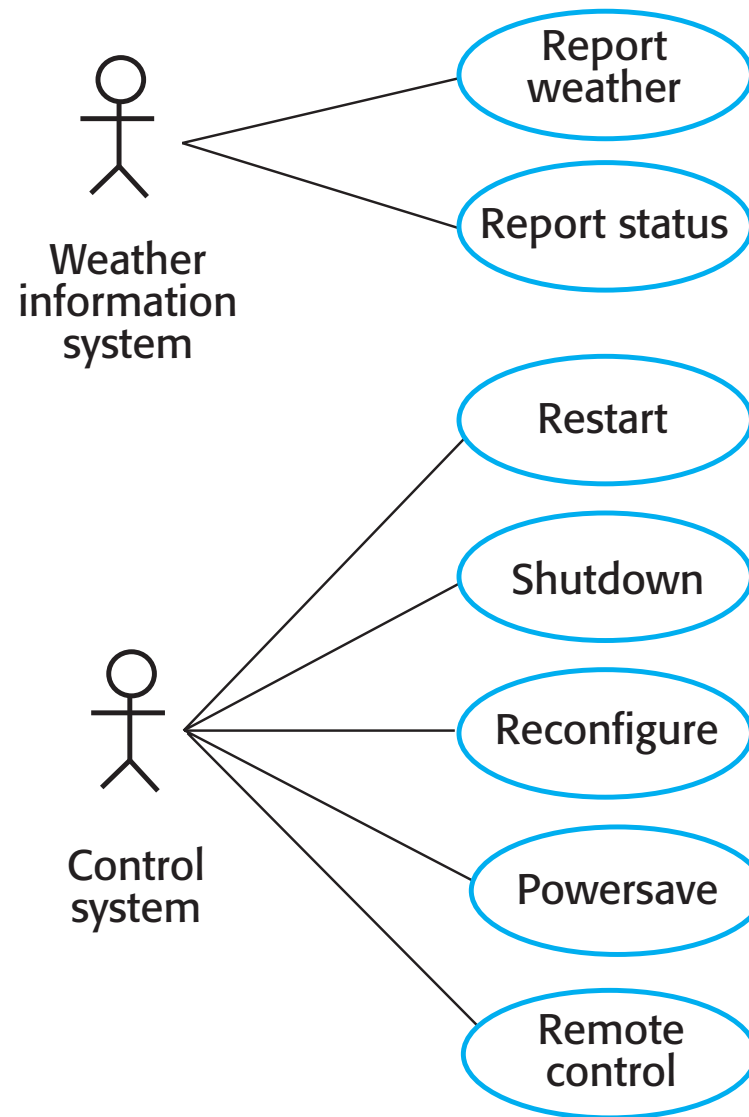
Mô hình ngữ cảnh và mô hình tương tác

- Mô hình ngữ cảnh hệ thống
 - ▣ Mô hình cấu trúc chỉ ra các hệ thống khác trong môi trường của hệ thống đang được phát triển.
- Mô hình tương tác hệ thống
 - ▣ Mô hình động mô tả cách hệ thống tương tác với môi trường của nó.
 - ▣ Sử dụng các use case để chỉ ra các tương tác.

Ngữ cảnh hệ thống cho trạm thời tiết



Use case cho trạm thời tiết





Các giai đoạn của quy trình thiết kế

- Định nghĩa ngữ cảnh và các tương tác bên ngoài với hệ thống
- Thiết kế kiến trúc hệ thống
- Nhận diện các đối tượng chính
- Phát triển các mô hình thiết kế
- Đặc tả giao diện đối tượng

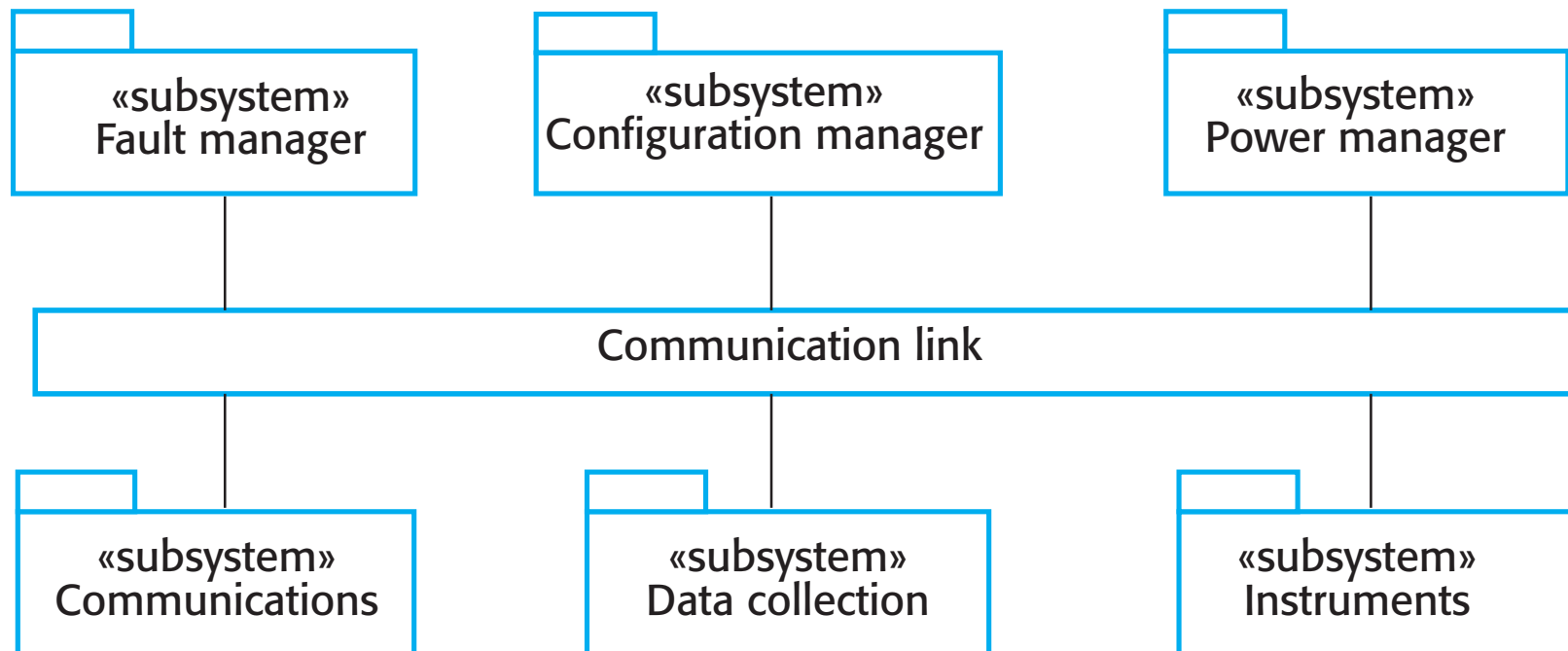


Thiết kế kiến trúc

- Sử dụng thông tin về các tương tác giữa hệ thống và môi trường để thiết kế kiến trúc hệ thống.
 - ▣ Nhận diện các component chính hình thành nên hệ thống và mối quan hệ giữa các component này,
 - ▣ Tổ chức các component này sử dụng một kiến trúc mẫu có sẵn: mô hình phân tầng, mô hình client-server,...

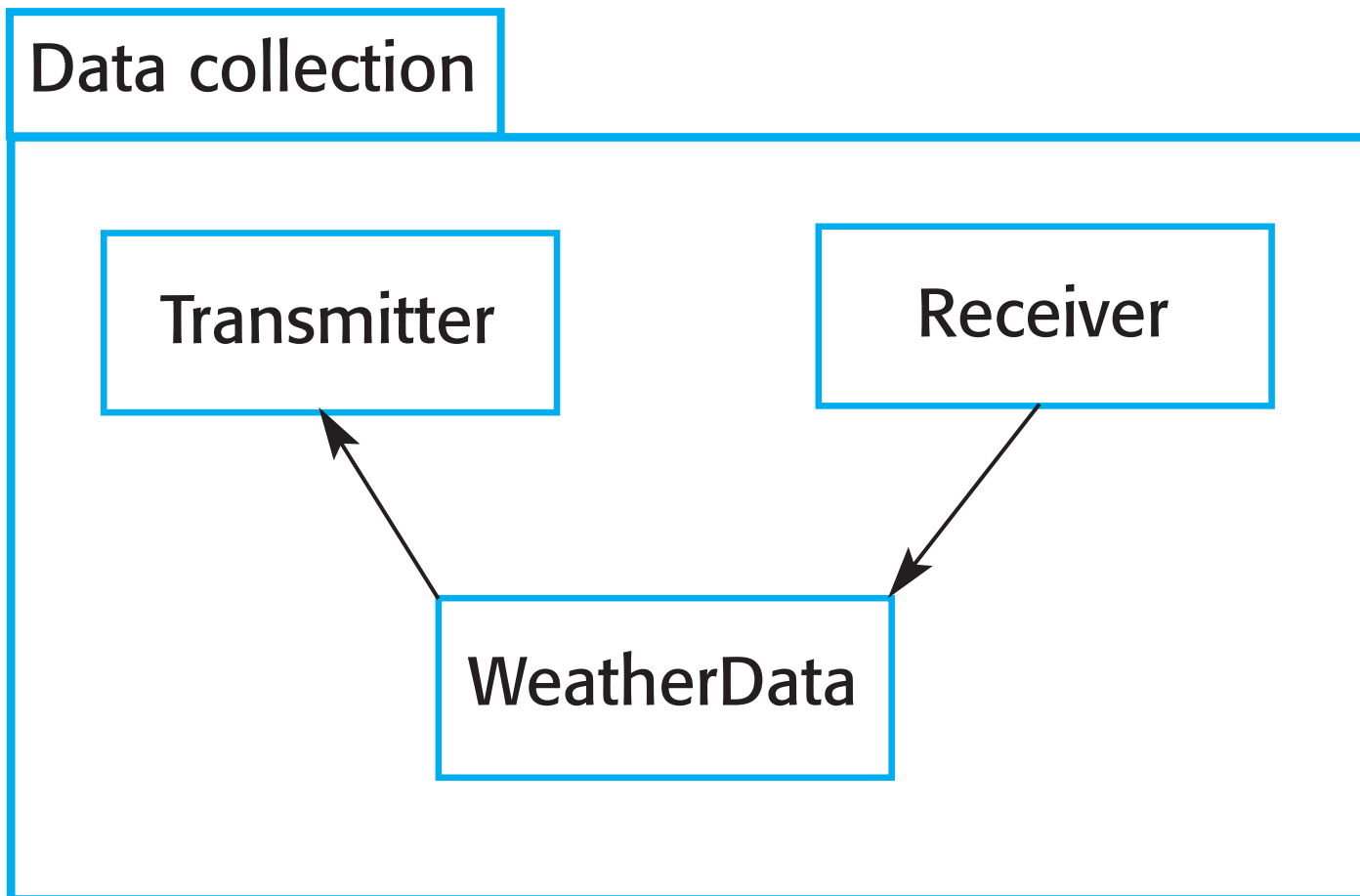


Kiến trúc ở mức cao của weather station





Kiến trúc của hệ thống thu thập dữ liệu





Các giai đoạn của quy trình thiết kế

- Định nghĩa ngữ cảnh và các tương tác bên ngoài với hệ thống
- Thiết kế kiến trúc hệ thống
- Nhận diện các đối tượng chính
- Phát triển các mô hình thiết kế
- Đặc tả giao diện đối tượng



Nhận diện lớp đối tượng

- ☐ Là phần khó của thiết kế hướng đối tượng.
- ☐ Không có một công thức tổng quát nào
- ☐ Đây là quy trình lặp lại.



Các phương pháp để nhận diện

1. Phân tích ngữ pháp dựa vào mô tả hệ thống.
2. Dựa vào việc nhận diện những đối tượng hữu hình có trong miền ứng dụng.
3. Phân tích dựa vào kịch bản.



Ví dụ: mô tả Weather station

A weather station is a package of software controlled instruments which collects data, performs some data processing and transmits this data for further processing. The instruments include air and ground thermometers, an anemometer, a wind vane, a barometer and a rain gauge. Data is collected periodically.

When a command is issued to transmit the weather data, the weather station processes and summarises the collected data. The summarised data is transmitted to the mapping computer when a request is received.



Ví dụ: mô tả Weather station

A **weather station** is a package of software controlled instruments which collects data, performs some data processing and transmits this data for further processing. The instruments include **air and ground thermometers**, an **anemometer**, a **wind vane**, a **barometer** and a **rain gauge**. Data is collected periodically.

When a command is issued to transmit the **weather data**, the weather station processes and summarises the collected data. The summarised data is transmitted to the mapping computer when a request is received.



Các lớp đối tượng trong Weather station

- Nhận diện đối tượng dựa vào những dữ liệu và phần cứng hữu hình trong hệ thống:
 - Ground thermometer, Anemometer, Barometer,...
 - Các đối tượng của miền ứng dụng, là các đối tượng phần cứng liên quan đến thiết bị trong hệ thống.
 - Weather station
 - Giao diện cơ bản của weather station với môi trường của nó. Do đó, các thao tác của nó phản ánh các tương tác được nhận diện trong mô hình use case.
 - Weather data
 - Chịu trách nhiệm xử lý các yêu cầu về báo cáo thời tiết. Đối tượng này gửi một bản tóm tắt về dữ liệu từ thiết bị đến hệ thống thông tin thời tiết.



Các lớp đối tượng Weather station

WeatherStation
identifier
reportWeather () reportStatus () powerSave (instruments) remoteControl (commands) reconfigure (commands) restart (instruments) shutdown (instruments)

WeatherData
airTemperatures groundTemperatures windSpeeds windDirections pressures rainfall
collect () summarize ()

Ground thermometer
gt_Ident temperature
get () test ()

Anemometer
an_Ident windSpeed windDirection
get () test ()

Barometer
bar_Ident pressure height
get () test ()



Các giai đoạn của quy trình thiết kế

- Định nghĩa ngữ cảnh và các tương tác bên ngoài với hệ thống
- Thiết kế kiến trúc hệ thống
- Nhận diện các đối tượng chính
- Phát triển các mô hình thiết kế
- Đặc tả giao diện đối tượng



Các mô hình thiết kế

- ☐ Chỉ ra các đối tượng, lớp đối tượng và mối quan hệ giữa các thực thể này.
- ☐ Mô hình tĩnh
 - ☐ mô tả cấu trúc tĩnh của hệ thống về các lớp đối tượng và quan hệ.
- ☐ Mô hình động
 - ☐ mô tả tương tác động giữa các đối tượng.



Các ví dụ về mô hình thiết kế

Mô hình hệ thống con

- Chỉ ra việc gom nhóm các đối tượng vào trong hệ thống con tương ứng

Mô hình tuần tự

- Chỉ ra chuỗi tuần tự các tương tác của các đối tượng.

Mô hình trạng thái

- Chỉ ra cách các đối tượng riêng lẻ thay đổi trạng thái để trả lời các sự kiện.

Các mô hình khác: use case, cộng gộp, tổng quát hóa, ...

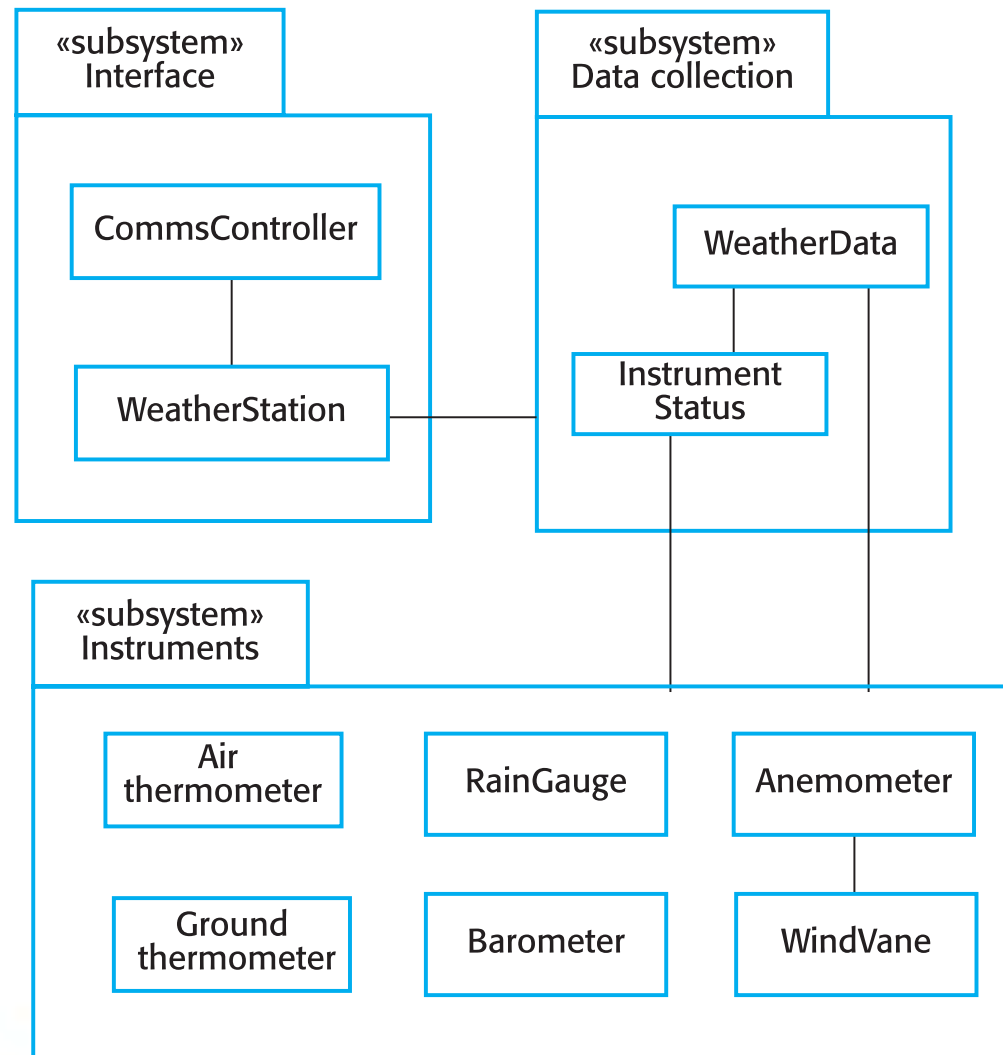


Mô hình hệ thống con

- ☐ Là mô hình tĩnh
- ☐ Chỉ ra cách gom nhóm các đối tượng liên quan đến nhau về mặt logic như thế nào.








Hệ thống con Weather station





Các giai đoạn của quy trình thiết kế

- 
- Định nghĩa ngữ cảnh và các tương tác bên ngoài với hệ thống
- 
- Thiết kế kiến trúc hệ thống
- 
- Nhận diện các đối tượng chính
- 
- Phát triển các mô hình thiết kế
- 
- Đặc tả giao diện đối tượng



Đặc tả giao diện

- ☐ Giao diện đối tượng phải được đặc tả sao cho đối tượng và các hệ thống con có thể được thiết kế song song với nhau.
- ☐ Thiết kế giao diện liên quan đến việc đặc tả chi tiết giao diện của một đối tượng hoặc một nhóm đối tượng.
- ☐ Giao diện có thể được đặc tả trong UML sử dụng cùng ký hiệu với biểu đồ lớp.



Đặc tả giao diện

- ☐ Không nên chứa chi tiết việc biểu diễn dữ liệu trong đặc tả giao diện.
- ☐ Cùng một đối tượng có thể có vài giao diện
 - ☐ mỗi giao diện là một góc nhìn khác nhau về các phương thức mà đối tượng cung cấp.
- ☐ Một nhóm các đối tượng có thể được truy cập thông qua một giao diện duy nhất.



Weather station interfaces

«interface» Reporting
weatherReport (WS-Ident): Wreport statusReport (WS-Ident): Sreport

«interface» Remote Control
startInstrument(instrument): iStatus stopInstrument (instrument): iStatus collectData (instrument): iStatus provideData (instrument): string



Giao diện Weather station

```
interface WeatherStation {  
  
    public void WeatherStation () ;  
  
    public void startup () ;  
    public void startup (Instrument i) ;  
  
    public void shutdown () ;  
    public void shutdown (Instrument i) ;  
  
    public void reportWeather ( ) ;  
  
    public void test () ;  
    public void test ( Instrument i ) ;  
  
    public void calibrate ( Instrument i) ;  
  
    public int getID () ;  
  
} //WeatherStation
```



Nội dung

1. Thiết kế hướng đối tượng sử dụng UML
2. Thiết kế mẫu
3. Các vấn đề về cài đặt



Các mẫu thiết kế

- ☐ Là một mô tả của vấn đề và điểm chính của giải pháp.
- ☐ Không phải là một đặc tả chi tiết
 - ☐ Nên biểu diễn đủ trườ tượng để có thể tái sử dụng ở các thiết lập khác.



Các vấn đề về thiết kế

- Khi gặp phải một vấn đề về thiết kế, ta có thể tìm được một mẫu thiết kế phù hợp có thể áp dụng được.
- Ví dụ:
 - Khi cần báo cho một vài đối tượng rằng trạng thái của đối tượng nào đó bị thay đổi (Observer pattern).
 - Cung cấp một giao diện đơn giản cho cho một tập các giao diện trong hệ thống con, làm cho hệ thống con dễ sử dụng hơn (Façade pattern).
 - Để truy cập vào các phần tử của một tập hợp, bỏ qua việc tập hợp đó được cài đặt thế nào (Iterator pattern).
 - Cho phép khả năng mở rộng tính năng của một lớp đã có sẵn tại thời gian thực thi (Decorator pattern).
 - ...



Các thành phần của mẫu thiết kế

- ☐ Tên
 - ☐ Một tên có nghĩa để nhận diện.
- ☐ Mô tả vấn đề.
- ☐ Mô tả giải pháp.
 - ☐ Là một template cho một giải pháp thiết kế trong đó giải pháp này có thể được sử dụng theo cách khác.
- ☐ Hệ quả
 - ☐ Kết quả sau khi áp dụng mẫu này.



Mẫu Observer

- ☐ Tên
 - ☐ Observer.
- ☐ Mô tả
 - ☐ Tách rời việc biểu diễn trạng thái ra khỏi đối tượng.
- ☐ Mô tả vấn đề
 - ☐ Được sử dụng khi có nhiều cách hiển thị trạng thái.
- ☐ Mô tả giải pháp
 - ☐ Xem slide tiếp theo.
- ☐ Hệ quả
 - ☐ Sẽ không thực tế nếu muốn tối ưu hóa để làm tăng hiệu suất của việc hiển thị.



Mẫu Observer

Tên mẫu	Observer
Mô tả	<p>Tách rời việc hiển thị trạng thái ra khỏi đối tượng và cung cấp các hiển thị thay thế. Khi trạng thái của đối tượng thay đổi, tất cả các hiển thị được thông báo và tự động cập nhật sự thay đổi đó.</p> <p>Trong nhiều tình huống, ta phải cung cấp nhiều hiển thị khác nhau về thông tin trạng thái, ví dụ như một hiển thị đồ họa và một hiển thị bảng. Các biểu diễn thay thế nên hỗ trợ tương tác, và khi trạng thái bị thay đổi, tất cả các hiển thị phải được cập nhật.</p>
Mô tả vấn đề	<p>Mẫu này cũng có thể được sử dụng trong tất cả các tình huống ở đó nhiều hơn một định dạng hiển thị về thông tin trạng thái được yêu cầu và ở đó không cần thiết phải duy trì thông tin trạng thái để biết về định dạng hiển thị cụ thể được sử dụng.</p>

Tên mẫu

Observer

Mô tả giải pháp

Gồm hai đối tượng trừu tượng: Subject và Observer, và hai đối tượng cụ thể: ConcreteSubject và ConcreteObject thừa kế thuộc tính của các đối tượng trừu tượng liên quan.

Các đối tượng trừu tượng chứa các thao tác chung có thể áp dụng được trong mọi tình huống. Trạng thái được hiển thị được duy trì trong ConcreteSubject, cho phép thêm hoặc loại bỏ các Observer (mỗi observer tương ứng với một hiển thị) và đưa ra một thông báo khi trạng thái bị thay đổi.

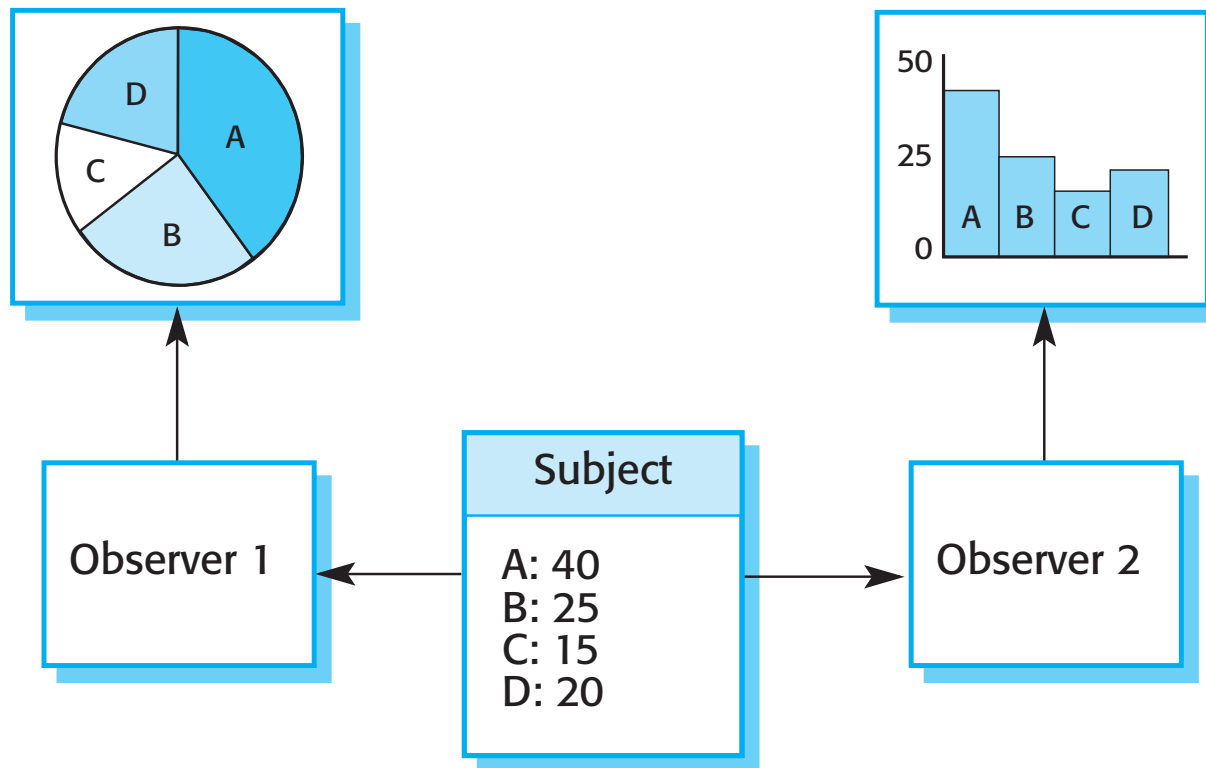
ConcreteObserver duy trì một bản copy trạng thái của ConcreteSubject và cài đặt giao diện Update() của Observer. ConcreteObserver tự động hiển thị trạng thái và phản ánh sự thay đổi khi trạng thái được cập nhật.

Hệ quả

Subject chỉ biết Observer và không biết về chi tiết của lớp cụ thể. Vì vậy có ít mối liên hệ giữa các đối tượng này. Vì thiếu thông tin, việc tối ưu để nâng cao hiệu năng hiển thị là không thực tế. Thay đổi Subject có thể gây nên một loạt các cập nhật đối với các Observer được phát sinh một cách không cần thiết.

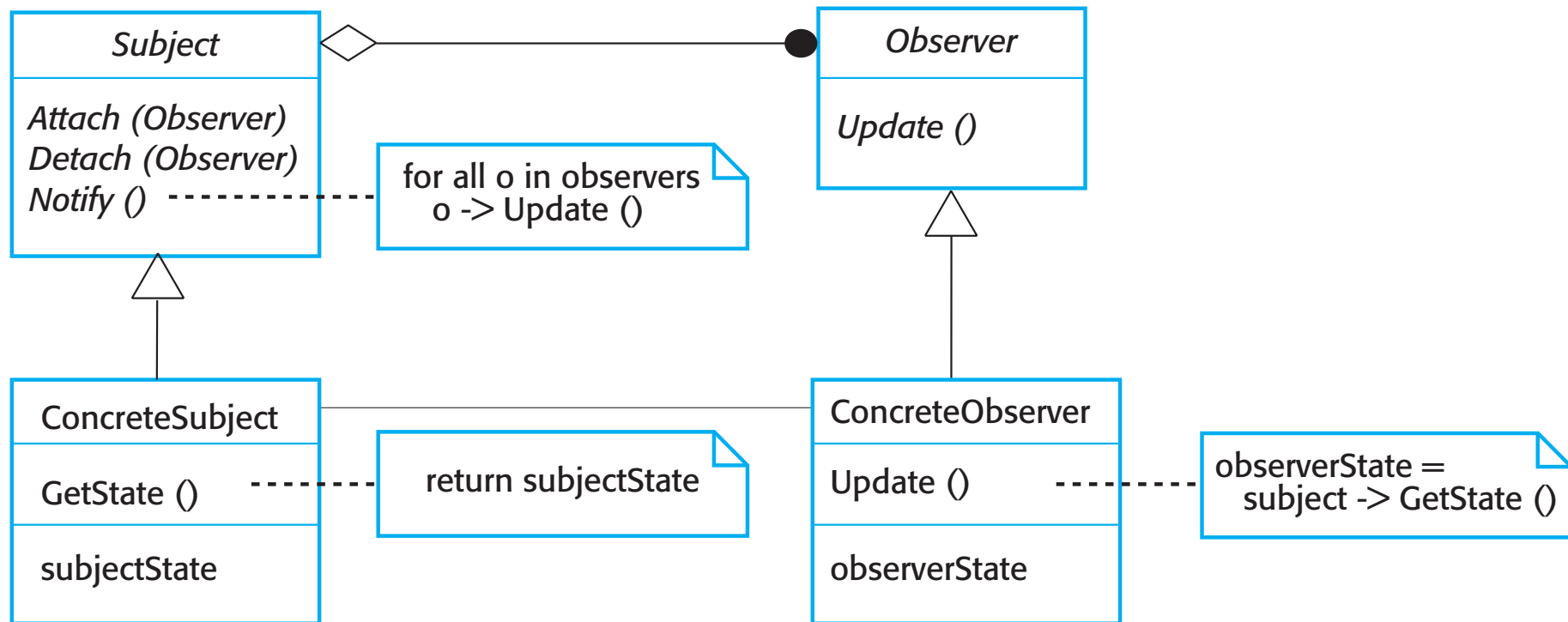


Đa hiển thị sử dụng mẫu Observer





Mô hình UML của mẫu Observer





Nội dung

1. Thiết kế hướng đối tượng sử dụng UML
2. Thiết kế mẫu
3. Các vấn đề về cài đặt



Các vấn đề về cài đặt

- Phần này không tập trung về lập trình, mặc dù phần lập trình khá quan trọng, chỉ tập trung vào các vấn đề liên quan đến cài đặt:
 - Tái sử dụng
 - Quản lý cấu hình
 - Phát triển host-target



Tái sử dụng

- Trong khoảng 1960 – 1990, đa phần các phần mềm mới đều được phát triển từ đầu, bằng cách viết tất cả các mã nguồn trong ngôn ngữ lập trình bậc cao.
 - ▣ Việc tái sử dụng chủ yếu là sử dụng hàm và đối tượng trong thư viện của ngôn ngữ lập trình.
- Áp lực về chi phí và tiến độ dự án → phương pháp này không thực tế, đặc biệt là cho các hệ thống dựa vào web và thương mại.
- Tái sử dụng là phương pháp phổ biến trong việc phát triển các phần mềm ngày nay.



Các mức tái sử dụng

- ☐ Mức trừu tượng
 - ☐ Tái sử dụng kiến thức của những thiết kế ở mức trừu tượng.
- ☐ Mức đối tượng
 - ☐ Sử dụng trực tiếp đối tượng từ thư viện có sẵn.
- ☐ Mức component
 - ☐ Component là tập hợp các đối tượng và lớp đối tượng mà ta tái sử dụng trong hệ thống ứng dụng.
- ☐ Mức hệ thống
 - ☐ Tái sử dụng toàn bộ hệ thống ứng dụng.



Chi phí của việc tái sử dụng

- ☐ Chi phí cho thời gian tìm kiếm phần mềm để tái sử dụng và đánh giá liệu nó có đáp ứng được yêu cầu đặt ra hay không.
- ☐ Chi phí để chỉnh sửa và cấu hình lại các component/hệ thống phần mềm được tái sử dụng để đáp ứng yêu cầu của hệ thống đang phát triển.
- ☐ Chi phí tích hợp các thành phần tái sử dụng và mã nguồn mới mà ta phát triển.



Quản lý cấu hình

- Quản lý cấu hình là tên gọi cho quy trình quản lý hệ thống phần mềm đang thay đổi.
- Mục tiêu: Hỗ trợ quy trình tích hợp hệ thống sao cho
 - ▣ tất cả người phát triển có thể truy cập vào tài liệu và mã nguồn của dự án theo cách được kiểm soát,
 - ▣ tìm ra sự thay đổi đã được thực hiện và
 - ▣ biên dịch và liên kết các component để tạo ra hệ thống.



Các hoạt động của quản lý cấu hình

☐ Quản lý phiên bản

- ☐ hỗ trợ việc theo dõi các phiên bản khác nhau của các component

☐ Tích hợp hệ thống

- ☐ giúp người phát triển định nghĩa các phiên bản nào của component được sử dụng để tạo ra mỗi phiên bản của hệ thống.

☐ Theo dõi vấn đề

- ☐ cho phép người dùng report bugs và các vấn đề khác; và cho phép tất cả người phát triển thấy ai đang giải quyết vấn đề này và khi nào chúng được sửa lỗi.



Phát triển host-target

- Hầu hết phần mềm được phát triển trên một máy tính (host) và chạy trên một máy tính khác (target).
- → Ta đang đề cập đến nền tảng (platform) phát triển và nền tảng thực thi.
 - Một nền tảng không chỉ là phần cứng.
 - Còn bao gồm hệ điều hành + các phần mềm hỗ trợ (hệ quản trị CSDL) hay nền tảng phát triển (môi trường phát triển tương tác).
 - Nền tảng phát triển thường có các phần mềm khác nhau được cài đặt hơn là nền tảng thực thi; những nền tảng này có thể có các kiến trúc khác nhau.



Công cụ nền tảng phát triển

- ☐ Bộ biên dịch tích hợp và hệ thống chỉnh sửa cú pháp.
- ☐ Hệ thống debug
- ☐ Công cụ chỉnh sửa đồ họa, ví dụ như công cụ chỉnh sửa các mô hình UML .
- ☐ Các công cụ kiểm thử (ví dụ: JUnit)
- ☐ Các công cụ hỗ trợ dự án.



Integrated development environments (IDEs)

- Các công cụ phát triển phần mềm thường được gom nhóm lại để tạo thành môi trường phát triển tích hợp (IDE).
- Một IDE là tập các công cụ hỗ trợ nhiều khía cạnh khác nhau của việc phát triển phần mềm.



Các nhân tố triển khai component/hệ thống

- Nếu một component được thiết kế cho một kiến trúc phần cứng cụ thể, hoặc dựa vào một số hệ thống khác, nó phải được triển khai trên nền tảng cung cấp phần cứng và phần mềm được yêu cầu.
- Các hệ thống hỗ trợ đa nền tảng có thể yêu cầu các component được triển khai nhiều hơn một nền tảng. Nghĩa là, trong trường hợp triển khai trên nền tảng đó thất bại, phải có cài đặt thay thế của component đó.
- Nếu lượng truy cập thông tin giữa các component lớn, nên triển khai chúng trên cùng một nền tảng hoặc các nền tảng gần nhau về mặt vật lý → giảm độ trễ giữa thời gian gửi và nhận thông điệp.