

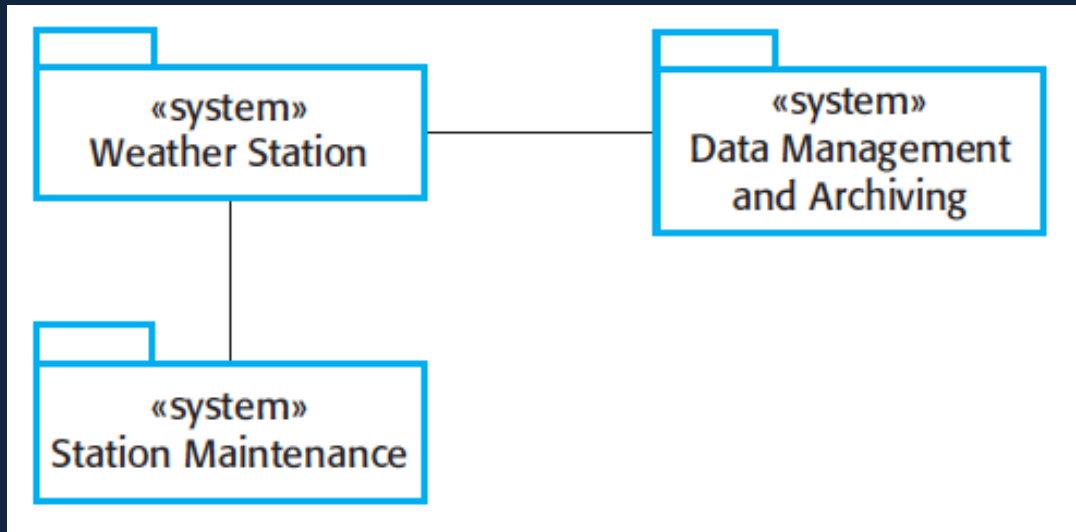
CS 123

Introduction to Software Engineering

06-C: Object-Oriented Design

DISCS
2013 – 2014

Wilderness Weather Station

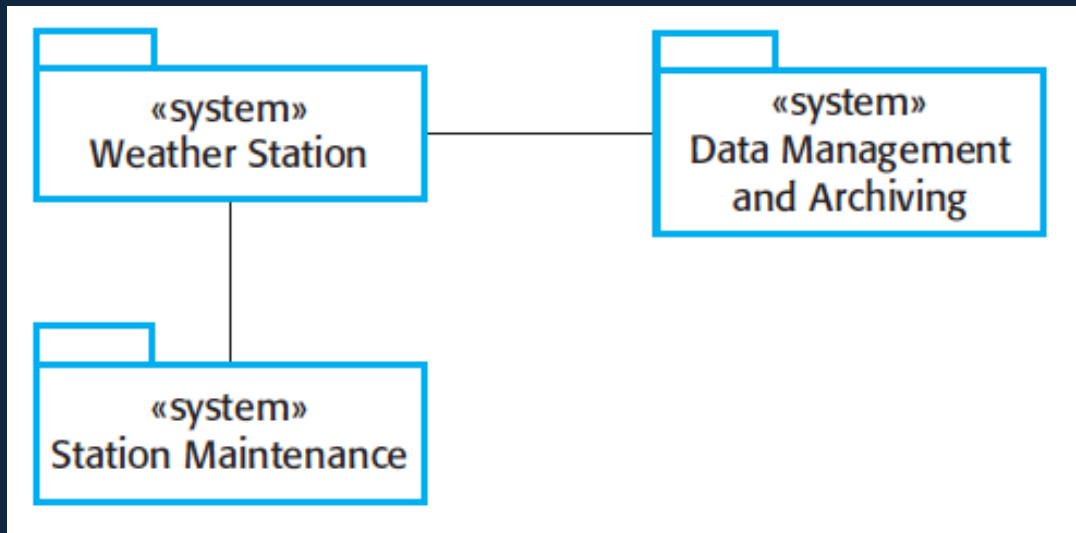


To help monitor climate change and to improve the accuracy of weather forecasts in remote areas, the government of a country with large areas of wilderness decides to deploy several hundred weather stations in remote areas.

The weather station collects data from a set of instruments and measure temperature and pressure, sunshine, rainfall, wind speed, and wind direction.

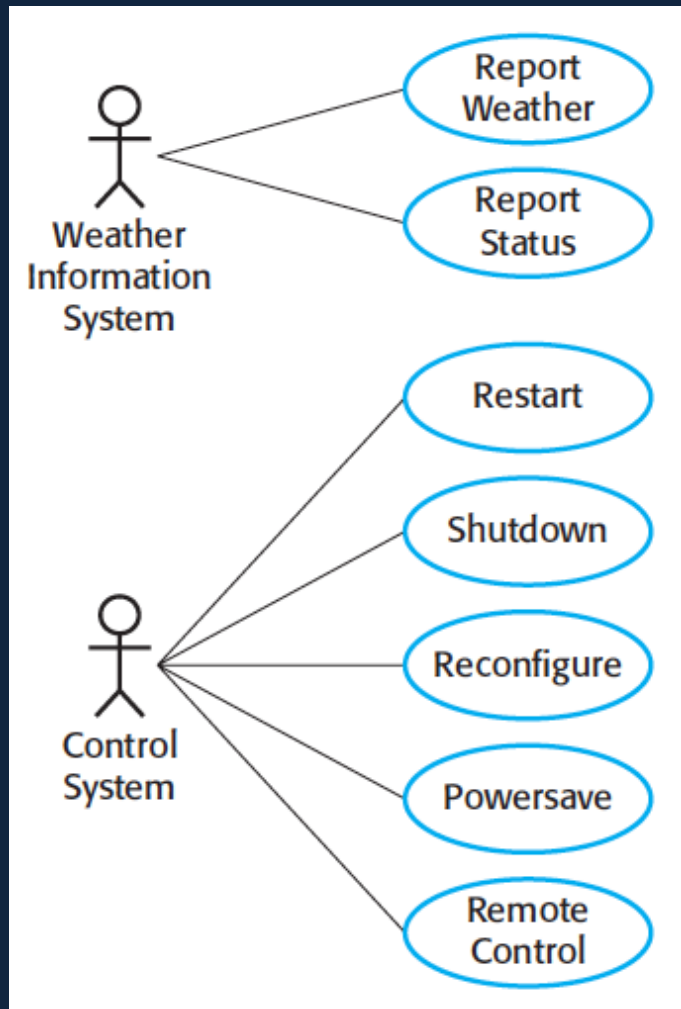
Wilderness weather stations are part of a larger system (see image above)

Wilderness Weather Station



1. **Weather Station System:** responsible for collecting data, carrying out some initial data processing, and transmitting data to the data management system
2. **Data Management System and Archiving System:** collects the data from all of the wilderness weather stations, carries out data processing and analysis, and archives the data in a form that can be retrieved by other systems, such as weather forecasting systems.
3. **Station Maintenance System:** communicates by satellite with all wilderness stations and provides reports of problems. It can update the embedded software in these systems. In the event of system problems, it can remotely control a wilderness weather system.

Use Cases



1. **Report Weather** – send weather data to the WIS
2. **Report Status** – send status info to WIS
3. **Restart** – if the weather station is shutdown, restart the system
4. **Shutdown** – shut down the weather station
5. **Reconfigure** – reconfigure the weather station software
6. **Powersave** – put the weather station into power-saving mode
7. **Remote Control** – send control commands to any weather station subsystem

Use Case: Report Weather

Actor	Weather Information System (WIS)
Description	<p>WS sends to the WIS a summary of the weather data that has been collected from the instruments in the collection period.</p> <ul style="list-style-type: none">• max, min, ave ground & air temp• max, min, ave air pressure• max, min, ave wind speeds• total rainfall• wind direction <p>** Sampling is done at 5-min intervals</p>
Stimulus	WIS establishes a satellite communication link to the WS and requests transmission of data.
Response	The summarized data is sent to the WIS
Comments	WSs are usually asked to report once per hour, but this frequency may differ from one station to another and may be modified in the future

Use Case: Report Status

Actor	Weather Information System (WIS)
Description	WS sends a summary of its status to the WIS. This includes details of the functioning of its instruments as tested by the self-test system and any other information about potential problems that have been detected.
Stimulus	WIS establishes a satellite communication link to the WS and requests transmission of the status data.
Response	The WS status data is sent to the WIS.
Comments	WSs are usually asked to report once per hour, but this frequency may differ from one station to another and may be modified in the future

Use Case: Restart

Actor	Control System (CS)
Description	WS should be in a shutdown state. The restart command causes all instruments to be initialized and the collection of data should begin.
Stimulus	CS establishes a satellite communication link to the WS and issues a restart command.
Response	The WS should start up.
Comments	This should be rare. Normally, it is only required after the system software has been updated by a reconfigure command.

Use Case: Shutdown

Actor	Control System (CS)
Description	WS should be in an operational state. The shutdown command causes all collected data to be stored in permanent memory and the WS instruments should then be shutdown so that no further data is collected.
Stimulus	CS establishes a satellite communication link with the WS and requests the system to shut down.
Response	The WS should shut down.
Comments	This should be rare. Normally, it is only required when the system software is to be updated. It should normally be preceded by a Report weather command.

Use Case: Reconfigure

Actor	Control System (CS)
Description	WS should be in shutdown state. The reconfigure command allows the existing software to be replaced by a software update which is remotely transmitted to the system. This is transmitted as a software download which is installed to replace the previous system.
Stimulus	CS establishes a satellite communication link with the WS and requests the system to accept a software download.
Response	After the download has been completed and installed, the WS should send an acknowledgement that it has been correctly downloaded and installed.
Comments	The command is used when software is to be updated. It should be preceded by a shutdown command and followed by a restart command.

Use Case: Powersave

Actor	Control System (CS)
Description	The effect of this command is to run the WS in a minimal energy state. The frequency of data collection is reduced, self-checking is only run once per day and instruments are turned off between data collection.
Stimulus	CS establishes a satellite communication link with the WS and requests the system to move to power saving mode.
Response	The WS should acknowledge that it is in a power saving state.
Comments	This should be rare. Normally, it is only required when there has been some failure of the hardware that charges the WS batteries.

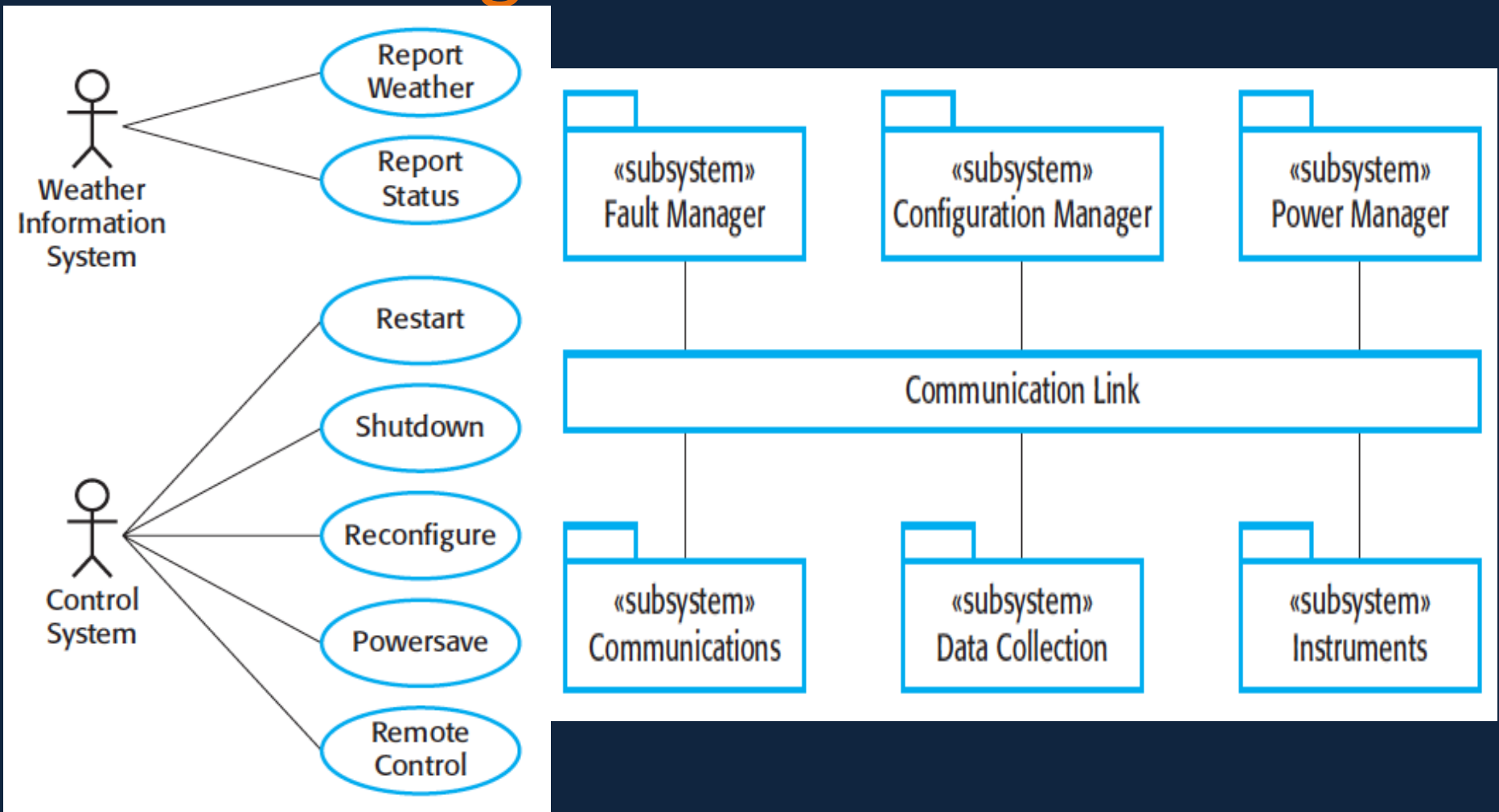
Use Case: Remote Control

Actor	Control System (CS)
Description	The feature allows the WS instruments to be controlled directly from the CS, rather than from the WS control software. The system establishes a direct channel of communication to the instrument interface. Commands are received as a string, decoded and sent to the instrument being controlled.
Stimulus	CS establishes a satellite communication link with the WS and sends a string of instrument commands.
Response	The WS should collect data as instructed by the commands.
Comments	This should be rare. Normally, it is only required when the system software has irretrievably failed

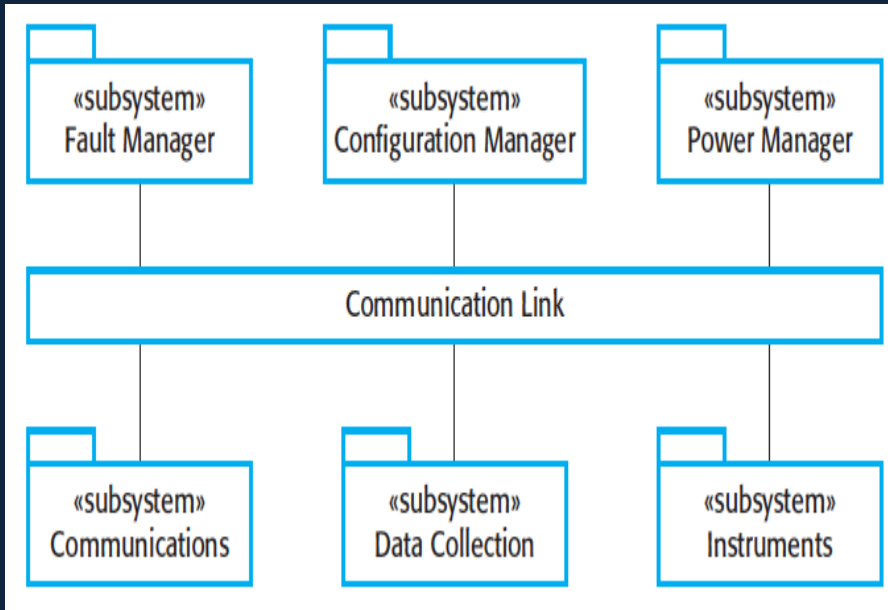
Architectural design

- Once interactions between the system and its environment have been understood, you use this information for designing the system architecture.
 - Identify the major components that make up the system and their interactions
 - Organize the components using an architectural pattern (such as a layered or client-server model).
- The weather station is composed of independent subsystems that communicate by broadcasting messages on a common infrastructure.

WS High-Level Architecture



WS High-Level Architecture

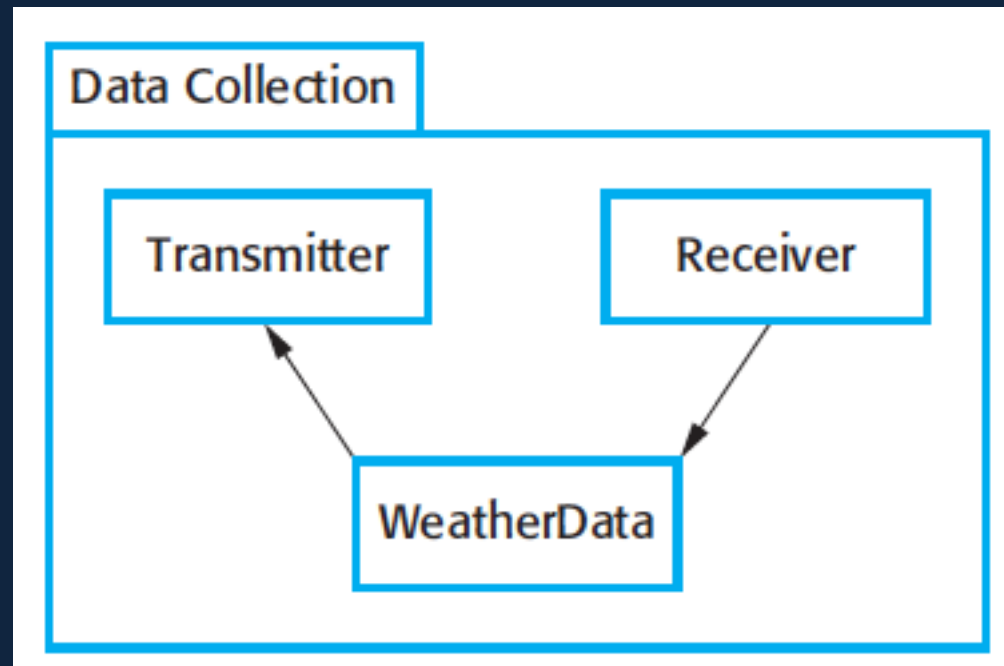
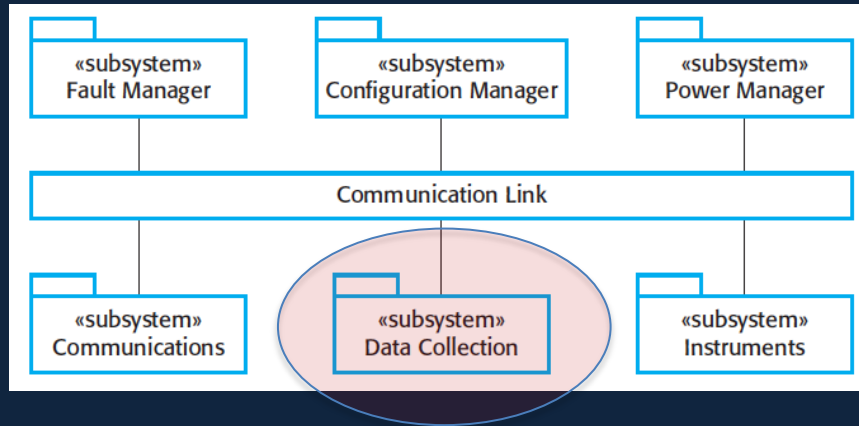


Another commonly used architecture:

- independent subsystems
- communication by broadcasting messages on a common infrastructure
- each subsystem listens for messages (and picks up messages that are intended for it).

A shutdown command is easily propagated to concerned subsystem.

Data Collection Subsystem



WeatherData Object

WeatherData

airTemperatures
groundTemperatures
windSpeeds
windDirections
pressures
rainfall

collect ()
summarize ()

WeatherData is responsible for processing the Report Weather command.

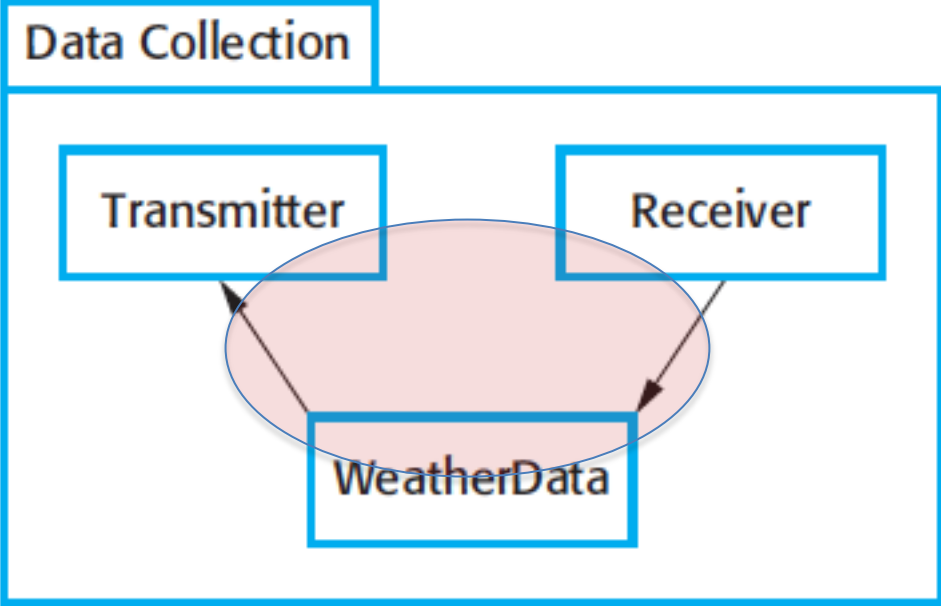
It collects data from the Receiver and sends the summarized data to the Transmitter

Data Collection

Transmitter

Receiver

WeatherData



```
graph TD; Receiver --> WeatherData; WeatherData --> Transmitter;
```

The diagram illustrates the data collection process within a system. It features three main components: a Transmitter, a Receiver, and a WeatherData object. The Receiver is shown sending data to the WeatherData object, which then sends the summarized data back to the Transmitter. The entire process is enclosed in a box labeled 'Data Collection'.

Object class identification

- Identifying object classes is often a difficult part of object oriented design.
- There is no 'magic formula' for object identification. It relies on the skill, experience and domain knowledge of system designers.
- Object identification is an iterative process. You are unlikely to get it right first time.

Approaches to identification

- Use a grammatical approach based on a natural language description of the system (Abbott, 1983).
- Base the identification on tangible things in the application domain (Wirfs-Brock et al., 1990).
- Use a behavioural approach and identify objects based on what participates in what behaviour.
- Use a scenario-based analysis. The objects, attributes and methods in each scenario are identified.

Weather station object classes

- Object class identification in the weather station system may be based on the tangible hardware and data in the system:
 - Ground thermometer, Anemometer, Barometer
 - Application domain objects that are 'hardware' objects related to the instruments in the system.
 - Weather station
 - The basic interface of the weather station to its environment. It therefore reflects the interactions identified in the use-case model.
 - Weather data
 - Encapsulates the summarized data from the instruments.

Weather station object classes

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 ()

Weather station object classes

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

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

Focus first on identifying the objects/classes.

Later on you can refine the object design

- Create Instrument superclass
- Add attributes (e.g., frequency of data collection)

Ground thermometer
gt_Ident temperature
get () test ()

Anemometer
an_Ident windSpeed windDirection
get () test ()

Barometer
bar_Ident pressure height
get () test ()

Design Models

- Design Models
 - aka System Models
 - Bridge between system requirements (analysis) and implementation (development)
 - Show the objects or classes in a system
 - Show the relationships and association between the entities
- Conflicting goals for design models:
 - Must be abstract enough to show relationships between the components and the system requirements
 - Must be detailed enough for programmers to make implementation decisions
- Solution: Develop models at various level of details

Design Models

- Unified Modeling Language (UML) – standard general-purpose modeling language in the field of OO software engineering
 - Includes a set of graphic notation techniques
 - Used to specify, visualize, modify, construct and document the artifacts of an OO system under development
- UML 2.2 has 14 types of diagrams divided into 2 categories
 - **Static** (or structural) models describe the static structure of the system in terms of object classes and relationships.
 - **Dynamic** (or behavioral) models describe the dynamic interactions between objects.

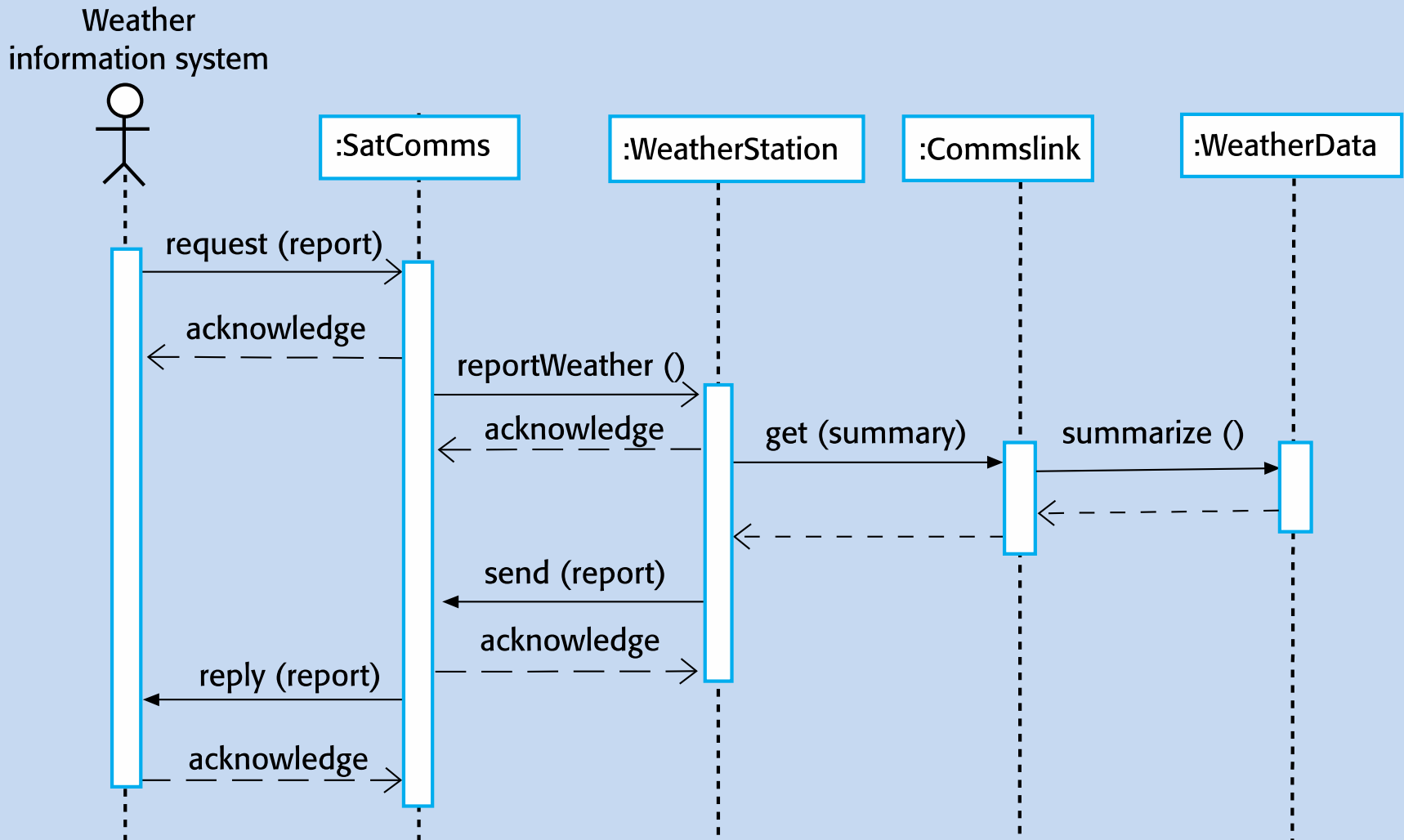
Examples of design models

- Subsystem models that show logical groupings of objects into coherent subsystems.
- Sequence models that show the sequence of object interactions.
- State machine models that show how individual objects change their state in response to events.
- Other models include use-case models, aggregation models, generalisation models, etc.

Sequence models

- Sequence models show the sequence of object interactions that take place
 - Objects are arranged horizontally across the top;
 - Time is represented vertically so models are read top to bottom;
 - Interactions are represented by labelled arrows, Different styles of arrow represent different types of interaction;
 - A thin rectangle in an object lifeline represents the time when the object is the controlling object in the system.

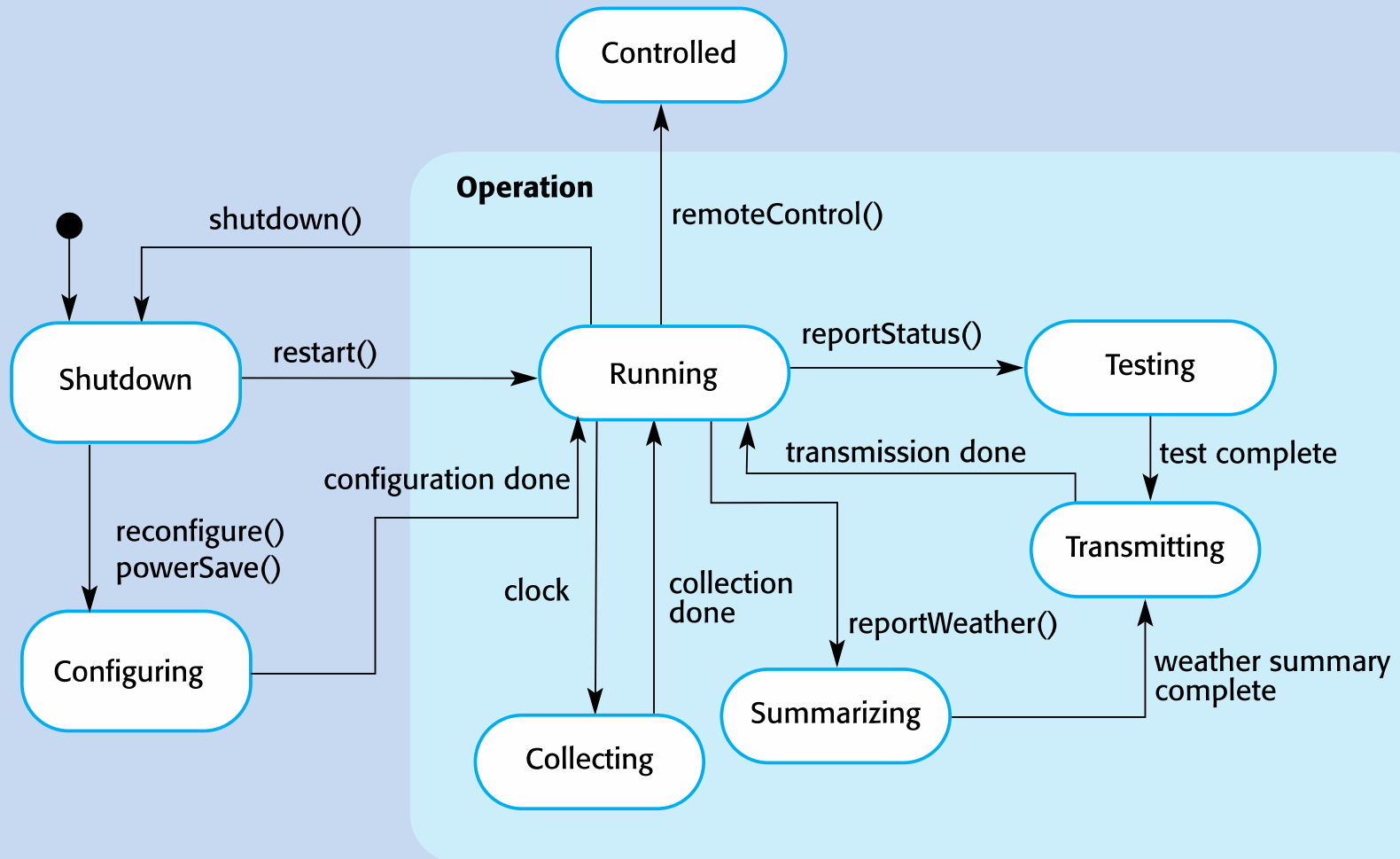
Sequence diagram describing data collection



State diagrams

- State diagrams are used to show how objects respond to different service requests and the state transitions triggered by these requests.
- State diagrams are useful high-level models of a system or an object's run-time behavior.
- You don't usually need a state diagram for all of the objects in the system. Many of the objects in a system are relatively simple and a state model adds unnecessary detail to the design.

Weather station state diagram



Interface specification

- Object interfaces have to be specified so that the objects and other components can be designed in parallel.
- Designers should avoid designing the interface representation but should hide this in the object itself.
- Objects may have several interfaces which are viewpoints on the methods provided.
- The UML uses class diagrams for interface specification but Java may also be used.

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

Key points

- Software design and implementation are inter-leaved activities. The level of detail in the design depends on the type of system and whether you are using a plan-driven or agile approach.
- The process of object-oriented design includes activities to design the system architecture, identify objects in the system, describe the design using different object models and document the component interfaces.
- A range of different models may be produced during an object-oriented design process. These include static models (class models, generalization models, association models) and dynamic models (sequence models, state machine models).
- Component interfaces must be defined precisely so that other objects can use them. A UML interface stereotype may be used to define interfaces.

References

- I. Sommerville, *Introduction to Software Engineering 9th ed.* Addison-Wesley, 2011
- http://en.wikipedia.org/wiki/Unified_Modeling_Language