TIM 50 - Business Information Systems

Lecture 12

Instructor: Terry Allen
UC Santa Cruz
11/2/2011

Announcements

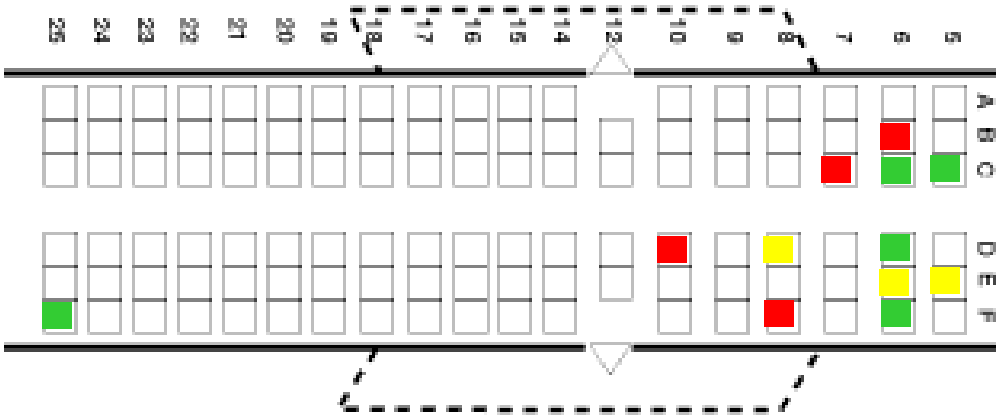
- Database Lab available
  - You will need to attend one session to do the database assignment
  - Tutorial will also be available, run by TAs
- Assignment 3 posted later this week
  - DUE next Monday, 11/14
- For next time read:
  - Messerschmitt Ch. 7

Modularity and Layering

Application Architecture Design

- The most important step to reduce/control complexity
  - Hardest to change
  - Influences everything that follows
- Conceptualization
  - What is it you are trying to do?
- Example Concept:
  - Small HHC for flight attendants.
  - HHC tells flight attendants which passengers are higher priority.
    - Who paid the highest fares
    - Who has been a more valuable customer in past (e.g. frequent flyer points)
  - Flight attendant discriminates based on this
    - Free drinks, meals, and pillows to valued customers

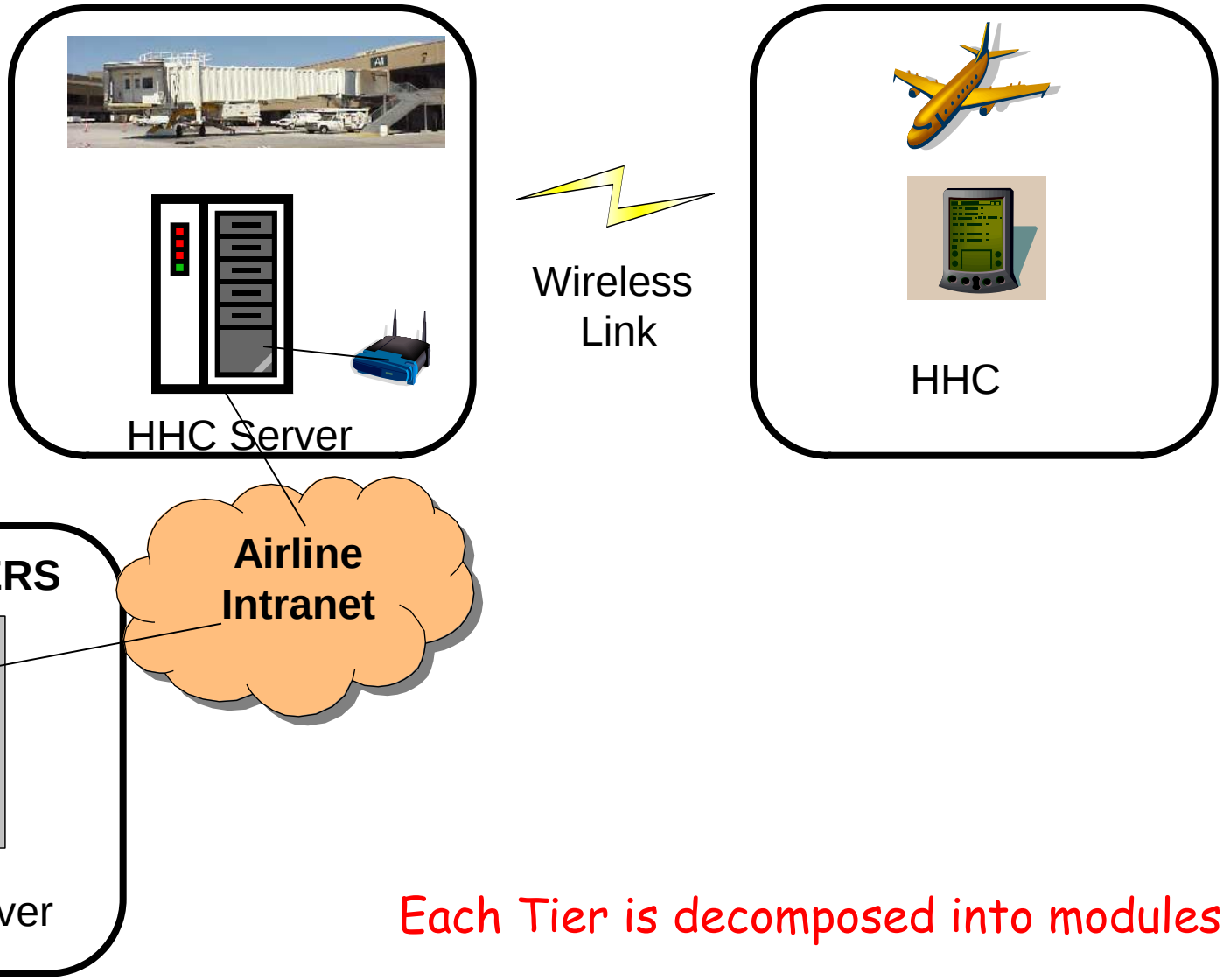
Example Concept:



Architecture

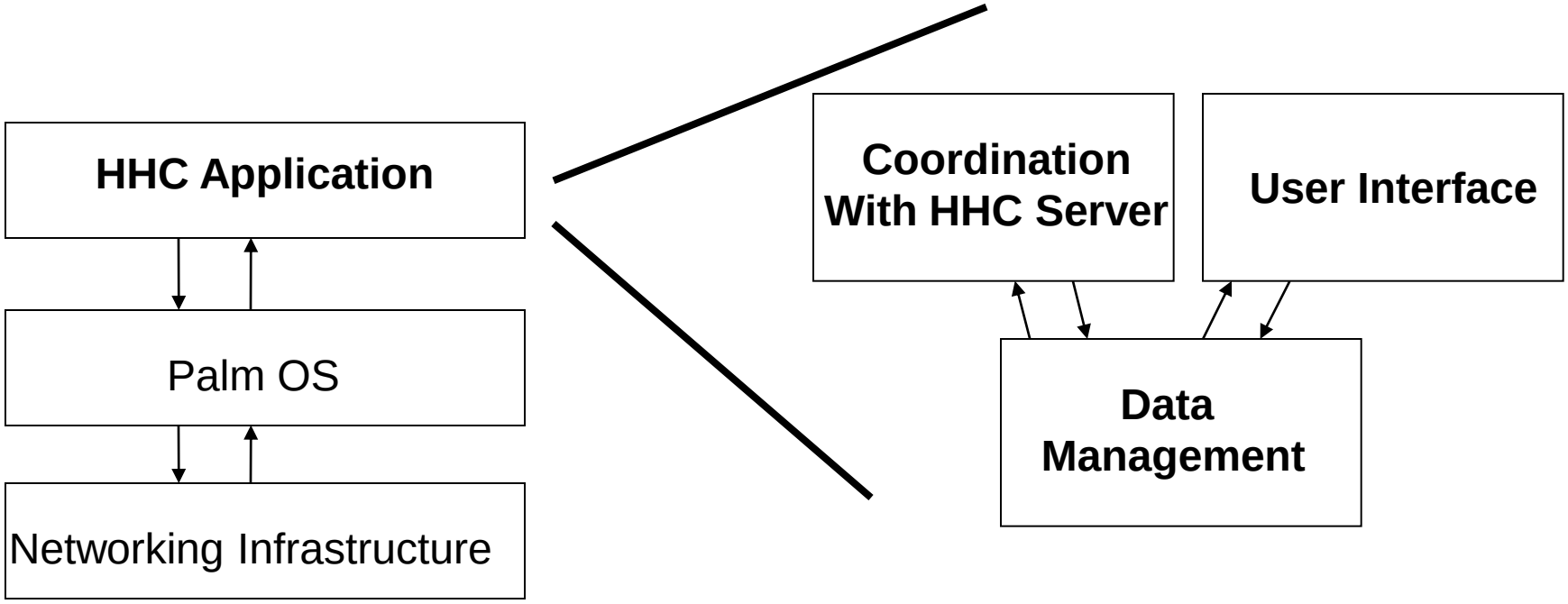
- What is the complexity of such a problem?
- How do you begin to architect a solution for a problem like this?
- Follow the principle that says: Break it into modules!
- What is a "good" architecture?

Architecture



Each Tier is decomposed into modules

HHC Architecture

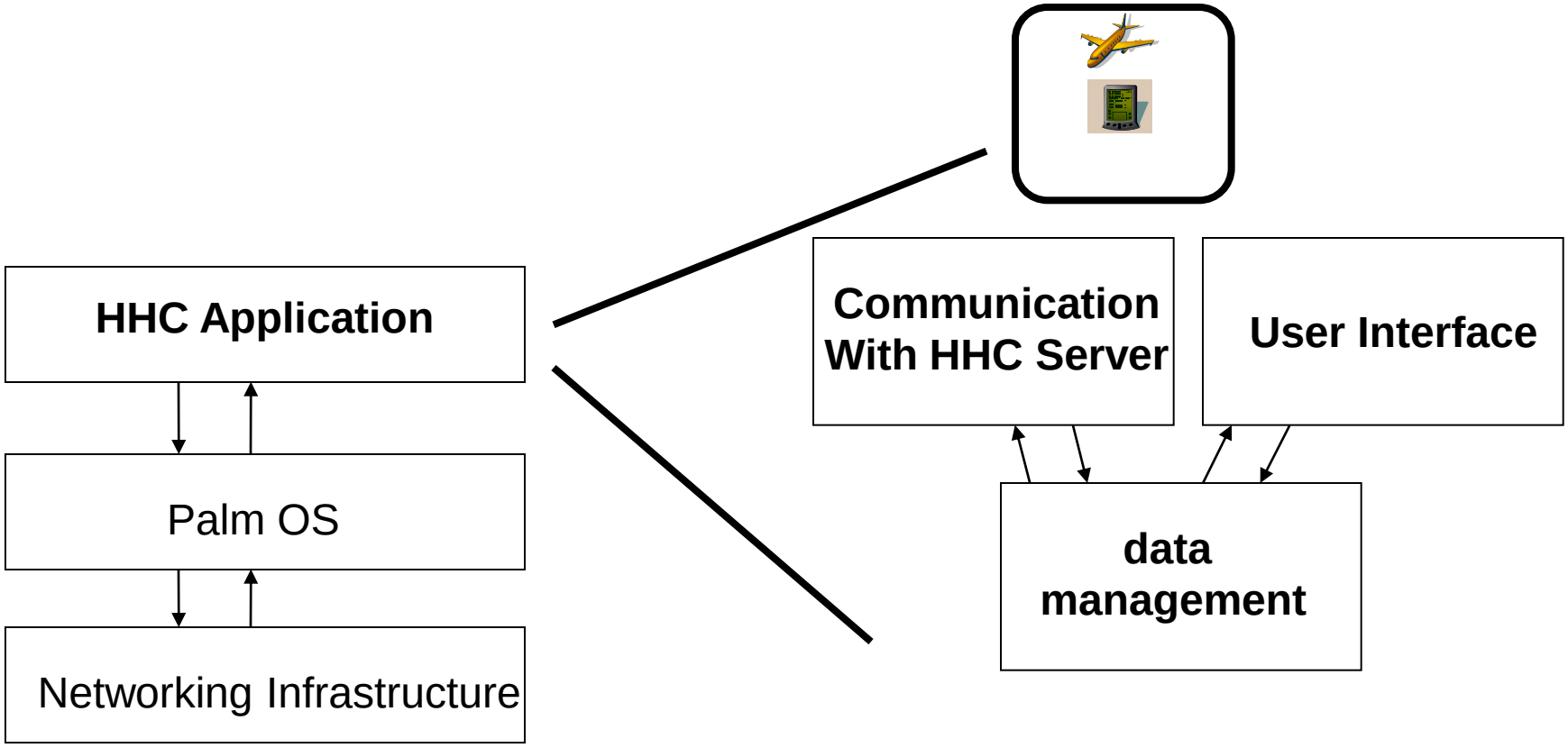


When a module is composed of sub-modules, the architecture is **hierarchical**.

HHC Architecture

We are using a ***layered architecture*** as well.

Allows reuse of previously built infrastructure.



Some aspects of software complexity

1) The number of elements (or participants) increases → system's complexity increases
2) The problem domain is complex
3) A lot of constraints
4) Every case must be foreseen
5) Continuous vs discrete, cannot exhaustively check every case
6) Team effort
7) Integration of different parts

Properties of Modularity

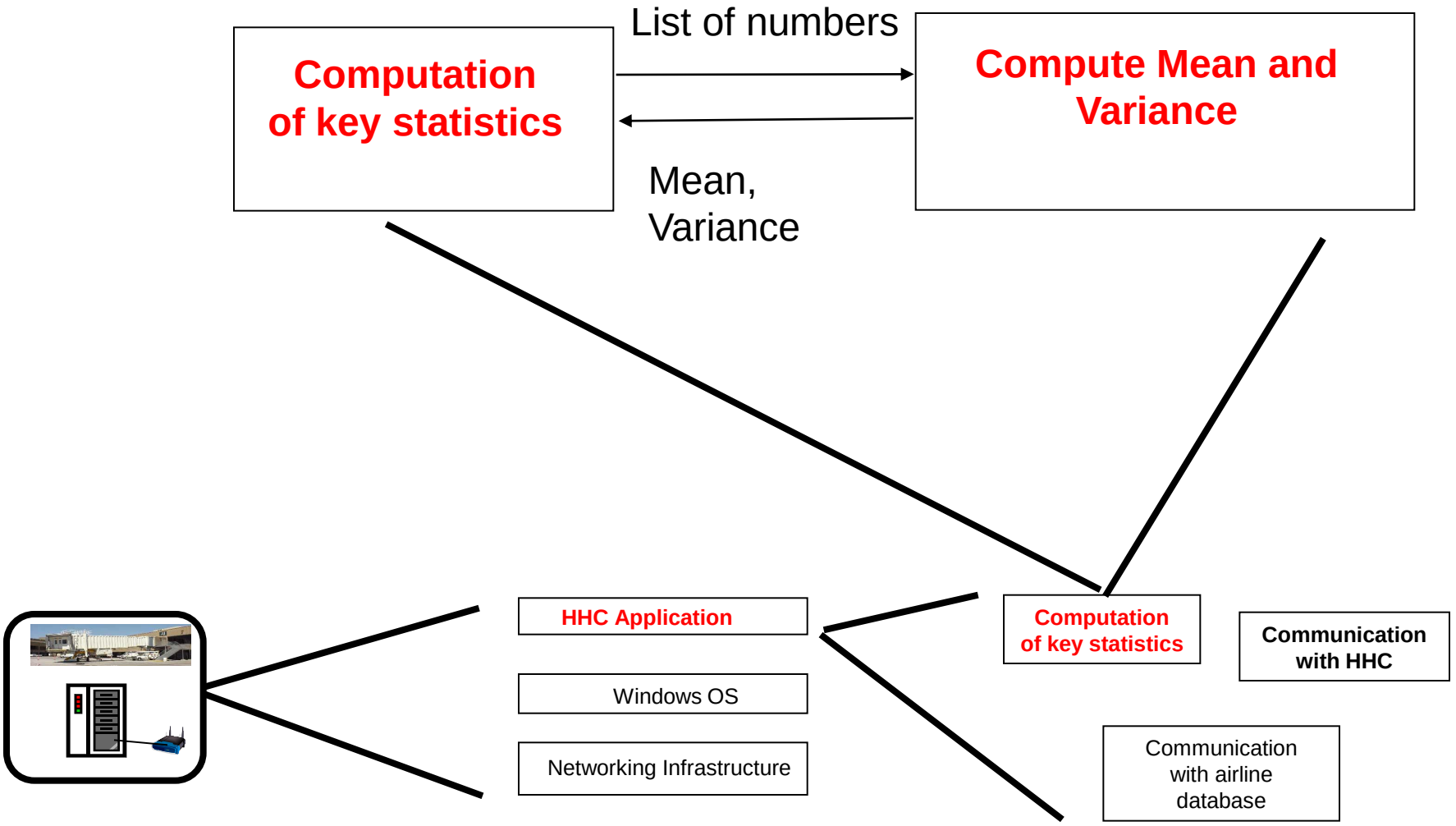
- *(idea: divide into smaller parts and deal with each part separately)*

- Functionality
- Hierarchy
- **Separation of concerns**
  - **"Easier to code for one goal than ten goals"**
- Interoperability
- Reusability

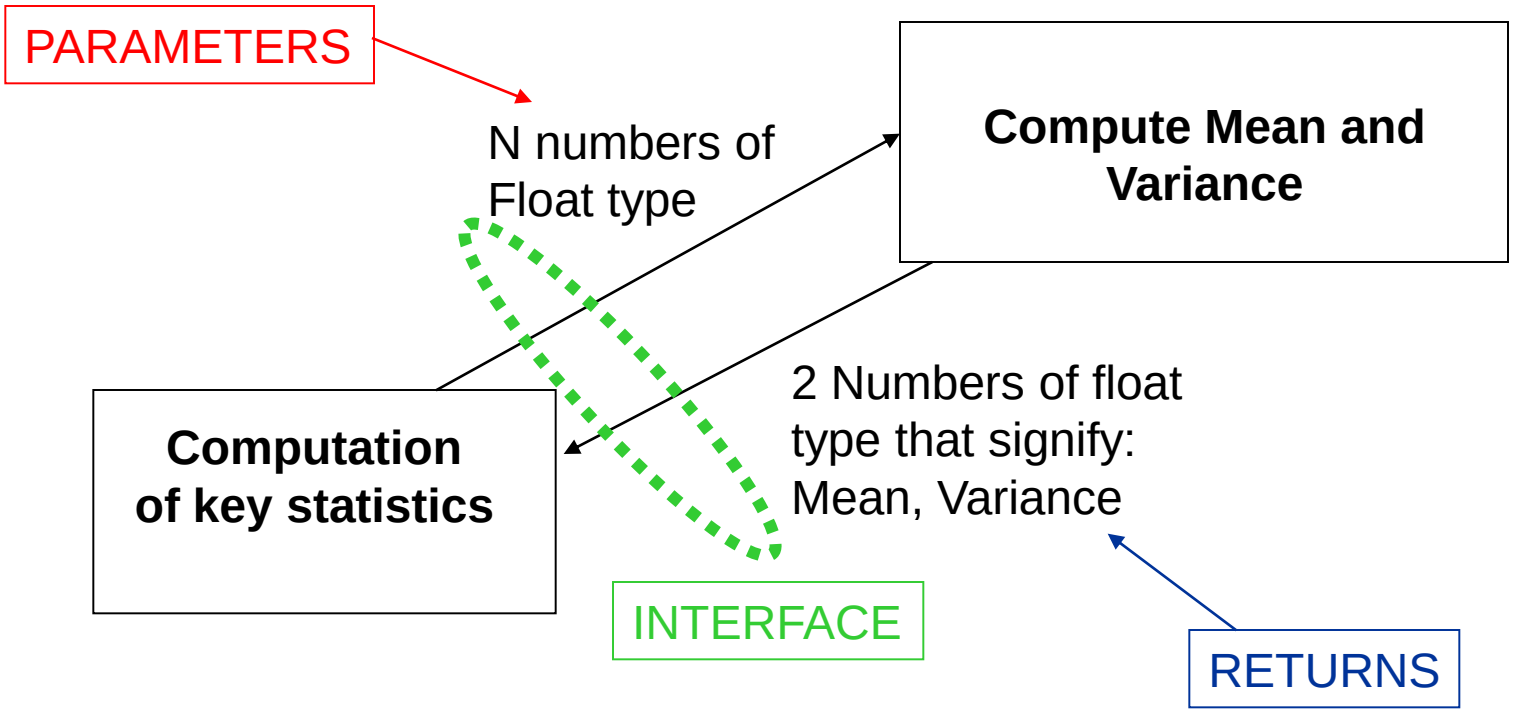
Student Talks

- Wai-Son Wong

A simple interface: from within our HHC Server Architecure



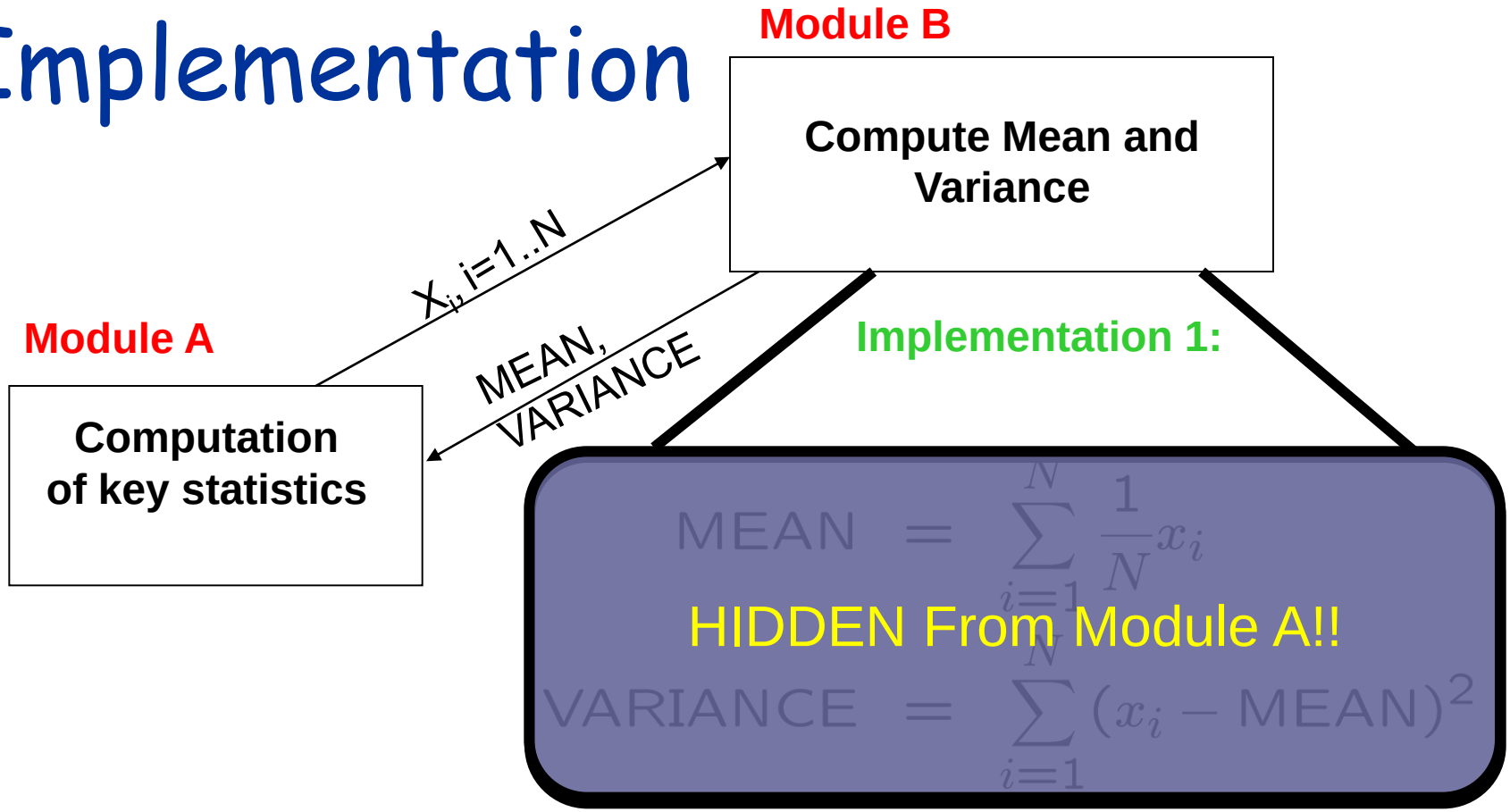
Interfaces



The data passed through an interface have 3 properties:

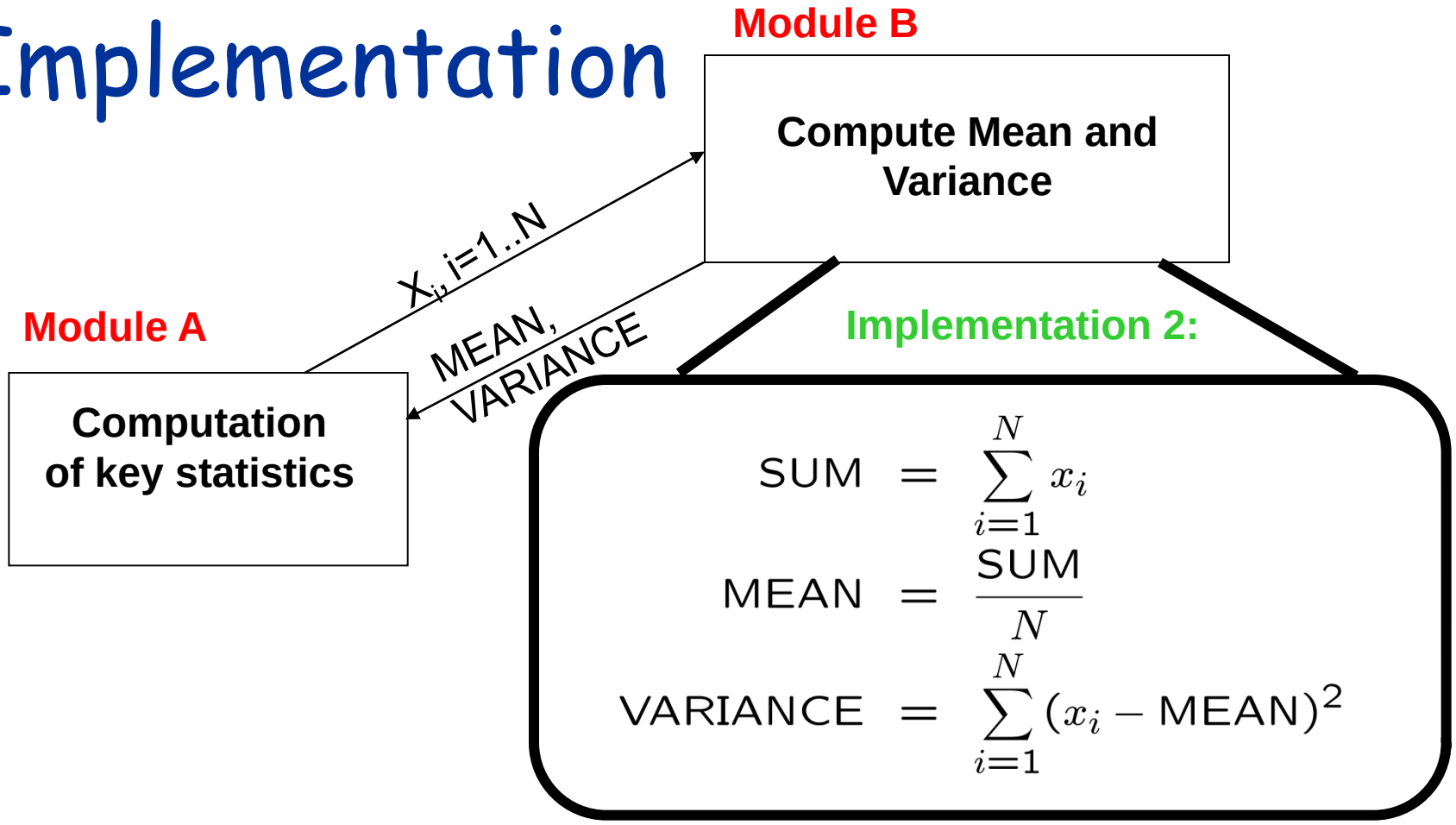
1. Name (e.g. employee_name)
2. Type (e.g. string)
3. Value (e.g. "John Smith")

Implementation



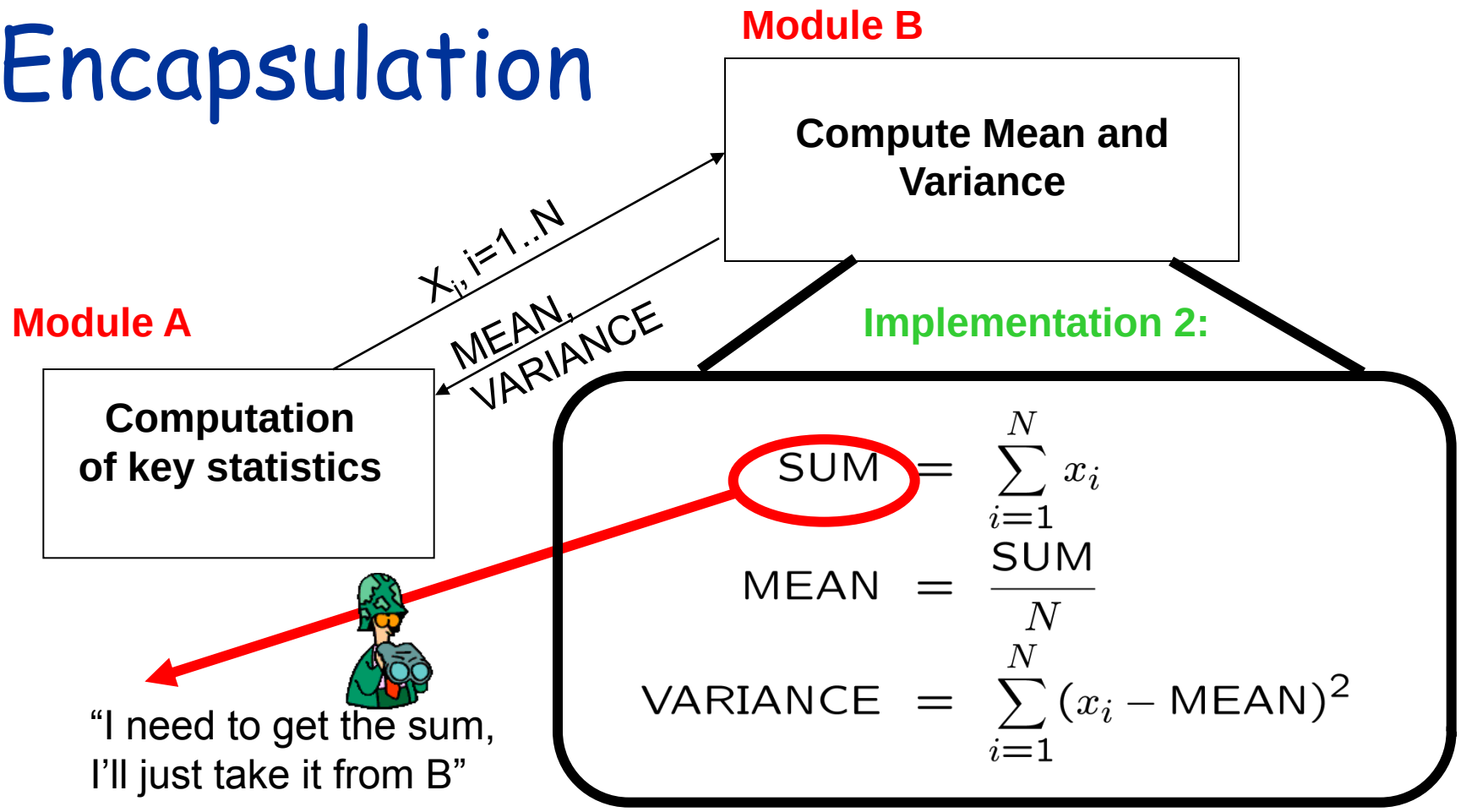
- One module should not be concerned with other module's implementation
  - → "Separation of concerns."
- One module should see the other only through its interface – implementation details hidden.
  - → Abstraction

Implementation



- Though different, this implementation is ok too.
- We can choose the implementation details however we want, as long as we comply with the agreed interface.

Encapsulation

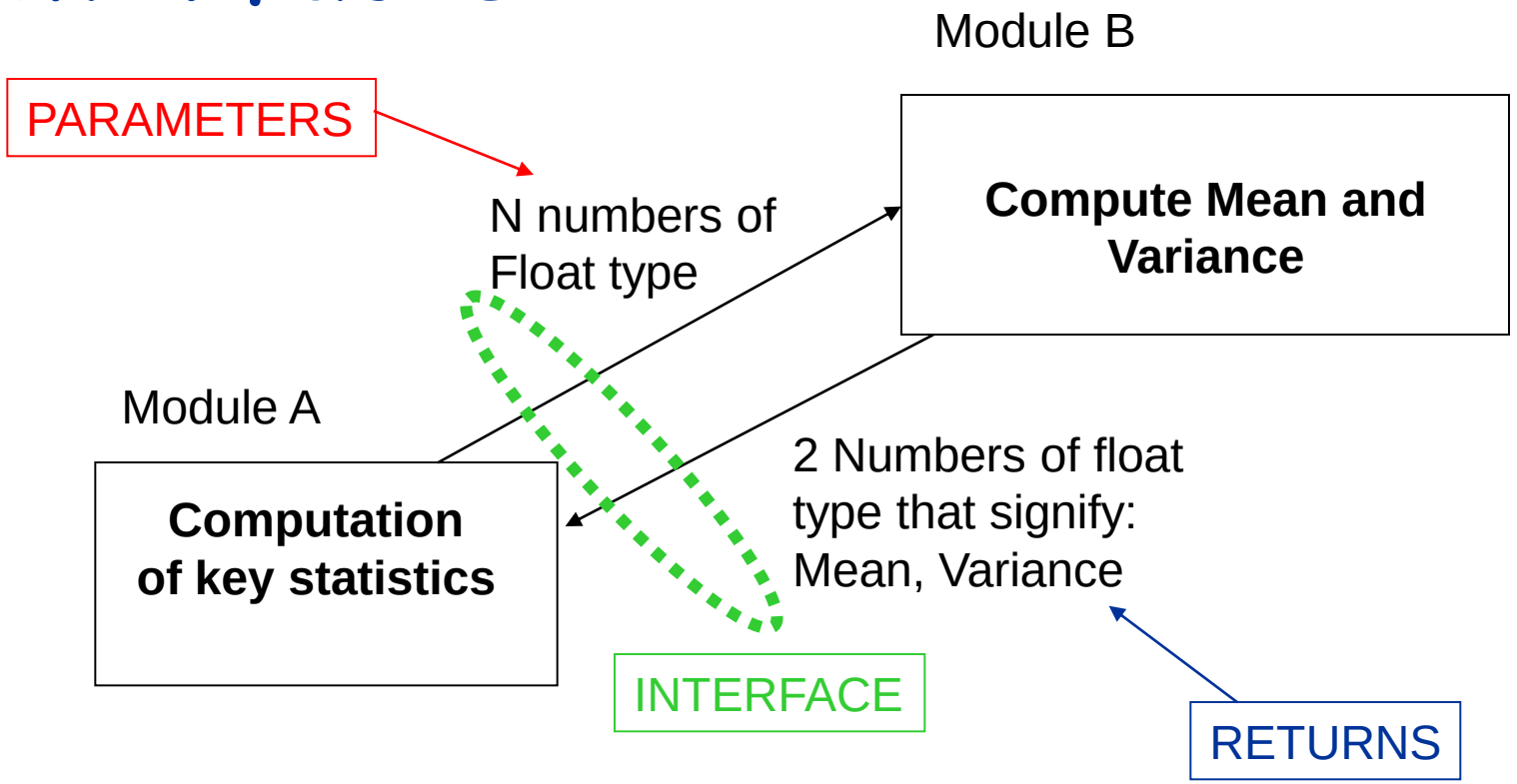


- Should he use it?
  - NO!!!! Why??
- Either A should compute "SUM" itself, or the interface of B should be redesigned

Encapsulation

- The designer of B might take measures to hide "SUM" from A so that A is not able to violate the agreed interface.
  - Example: B does not declare "SUM" as a global variable.
  - Modern languages allow developers to make this explicit (i.e., developer must affirmatively declare all publicly available items)
- Making a modules implementation details inaccessible to other modules is called ***encapsulation***

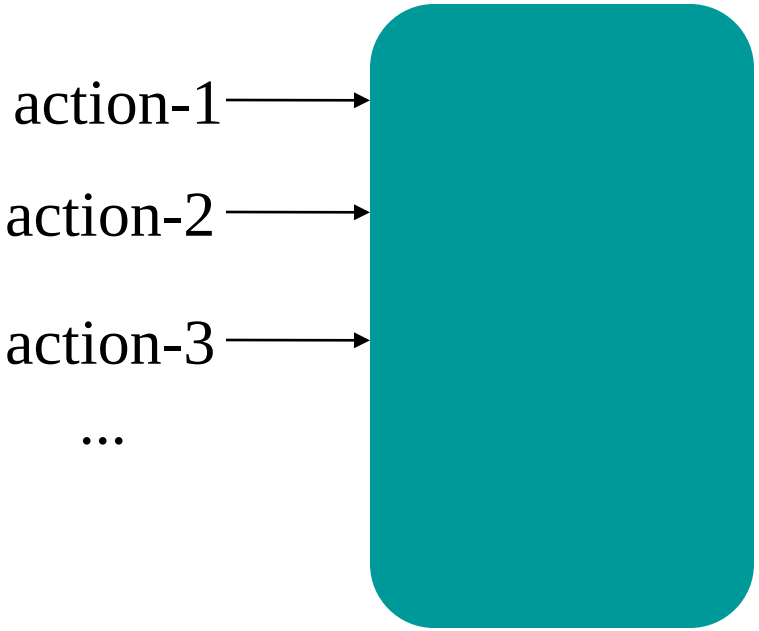
Interfaces



- This simple interface example allows for only one action of module B.
  - Action is "Compute mean and variance."
- Other examples are possible.

Possible software interface

Menu of actions



Example:

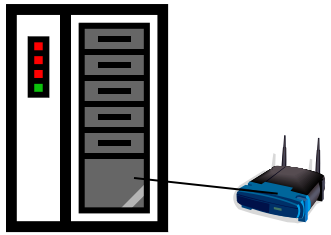
Action 1: Compute mean
Action 2: Compute variance
Action 3: Compute sum
Etc..

Protocol

In addition to atomic actions, an interface may define protocols

- Protocol == finite *sequence* of actions required to achieve a higher level function
- One action can be shared by multiple protocols
- Multiple modules may participate in a protocol

Protocol Example



HHC Server

Hello: I'm the HHC of Airplane#1234

Hello: I'm the gate 32 server

Tell me about the passengers of my next flight

Return Passenger Data

Tell me about the weather at my next destination.

Return Weather Data

HHC

(Might be passed as an array of a compound data type "passenger," which in turn is composed of standard types like integer, and string)

Another Interface Example:
Automatic teller machine (ATM)

What is the interface between this machine and the customer?

Steps

1. Identify interface building blocks

2. Define available actions

3. Define, for each higher level function, a **protocol**
   - Single action or a finite sequence of actions

1.Interface building blocks

Message on screen or printed

- Menu of actions or returns from an action
- Touch selection of action

Keypad

- Input parameters to an action

Card reader

- Authentication, input parameters

Money output slot

- Returns money

2.ATM actions

- Authentication
- Account specification
- Amount specification
- Options (e.g., transaction record)

Action: authentication

Parameters

- Identity (card in slot)
- Institution (card in slot)
- PIN (typed on keypad)

Internally, it contacts institution and matches against its database, institution noted for all subsequent actions

Returns

- Screen message
  - "Invalid PIN", or
  - Menu of available actions

Action: specify_account

Parameters

- Account (touch screen from menu of choices)

Internally, choice noted for all subsequent actions

Returns

- None

Action: amount

Parameters

- Dollars_and_cents (typed on keypad)

Internally, amount noted

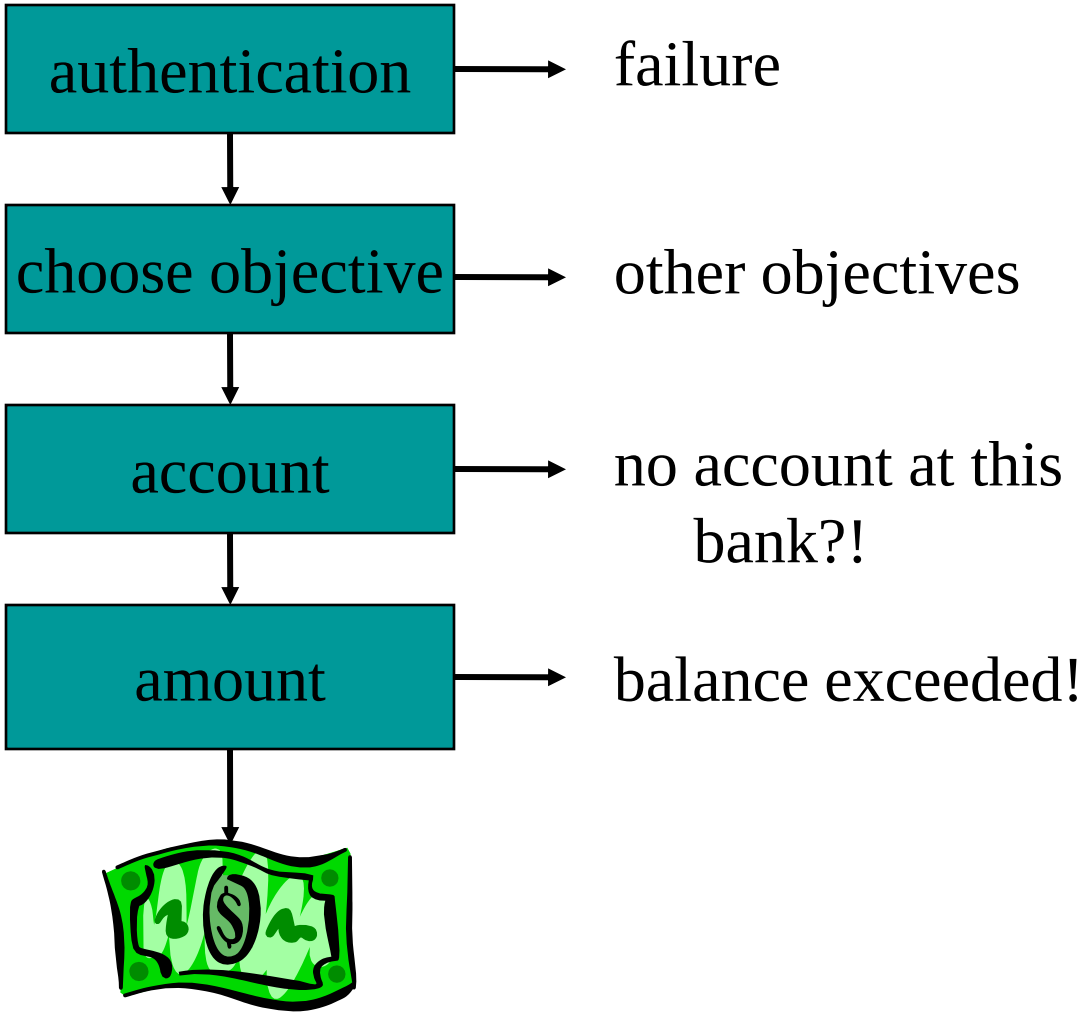
Returns

- Success or failure (state dependent, for example for a withdraw failure when dollars_and_cents exceeds balance)

Protocol: cash_withdrawal

What is the sequence of actions?

Protocol: cash_withdrawal



More on layering

by
David G. Messerschmitt

Interaction of layers

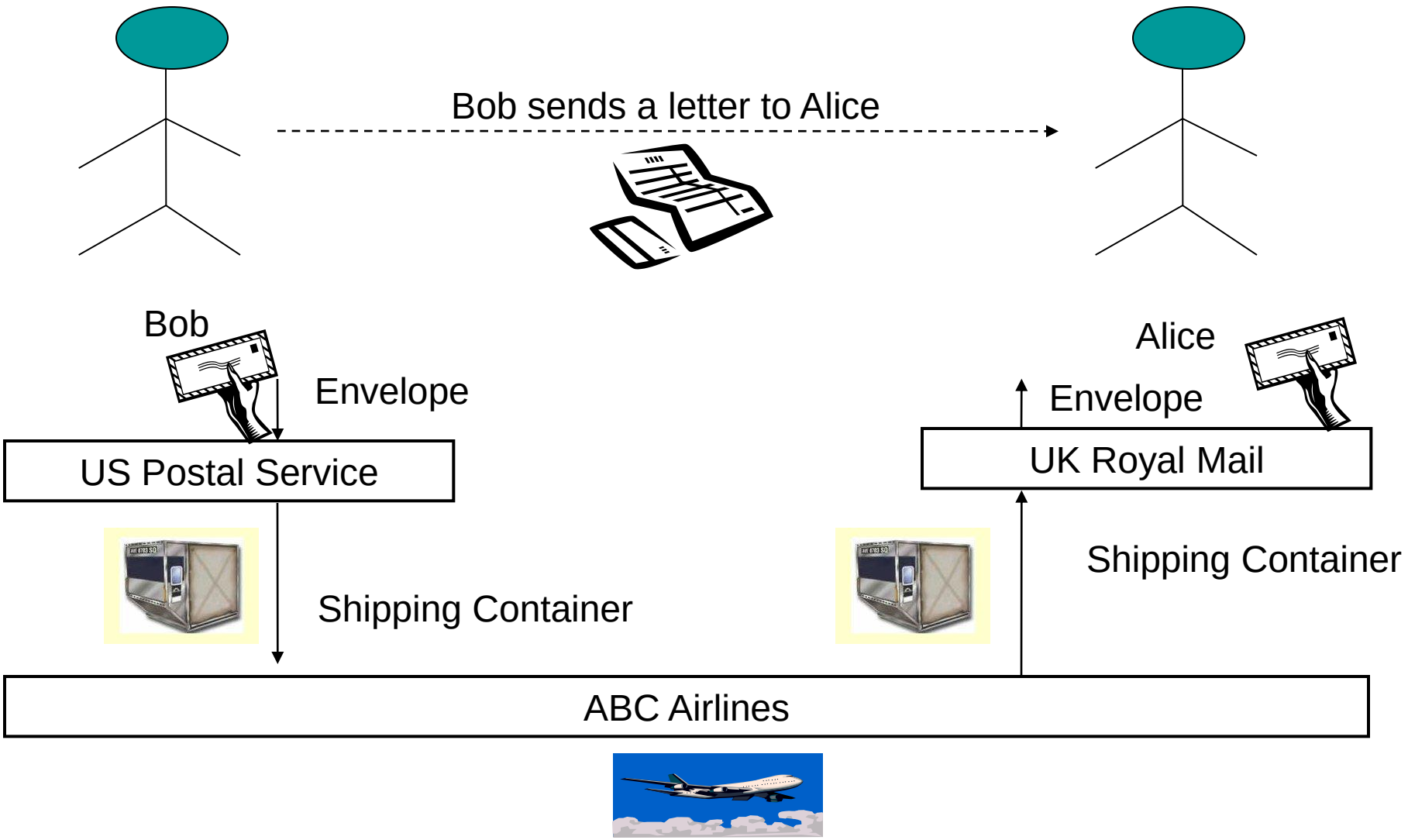
Layer above is a client of the layer below

Each layer provides services to the layer above….

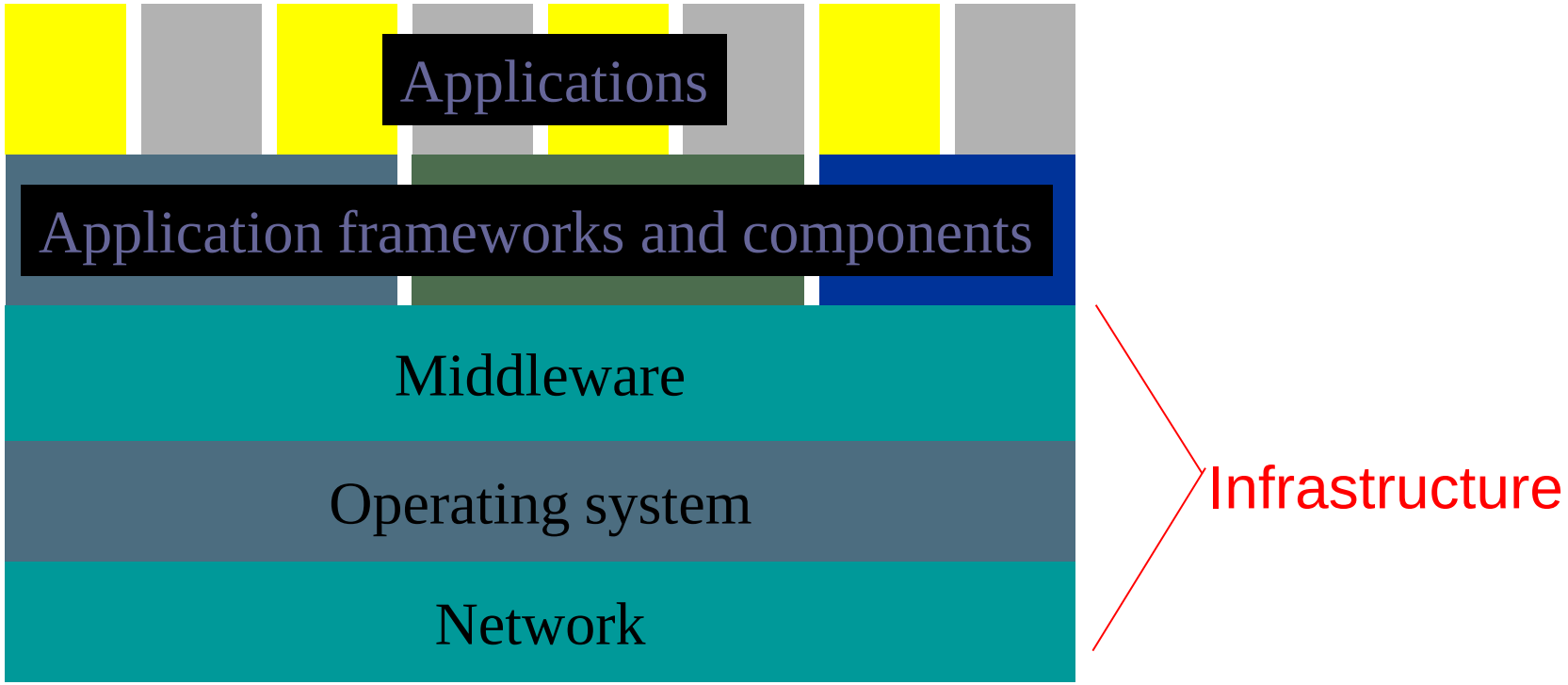
….by utilizing the services of the layer below and adding capability

Layer below as as a server to the layer above

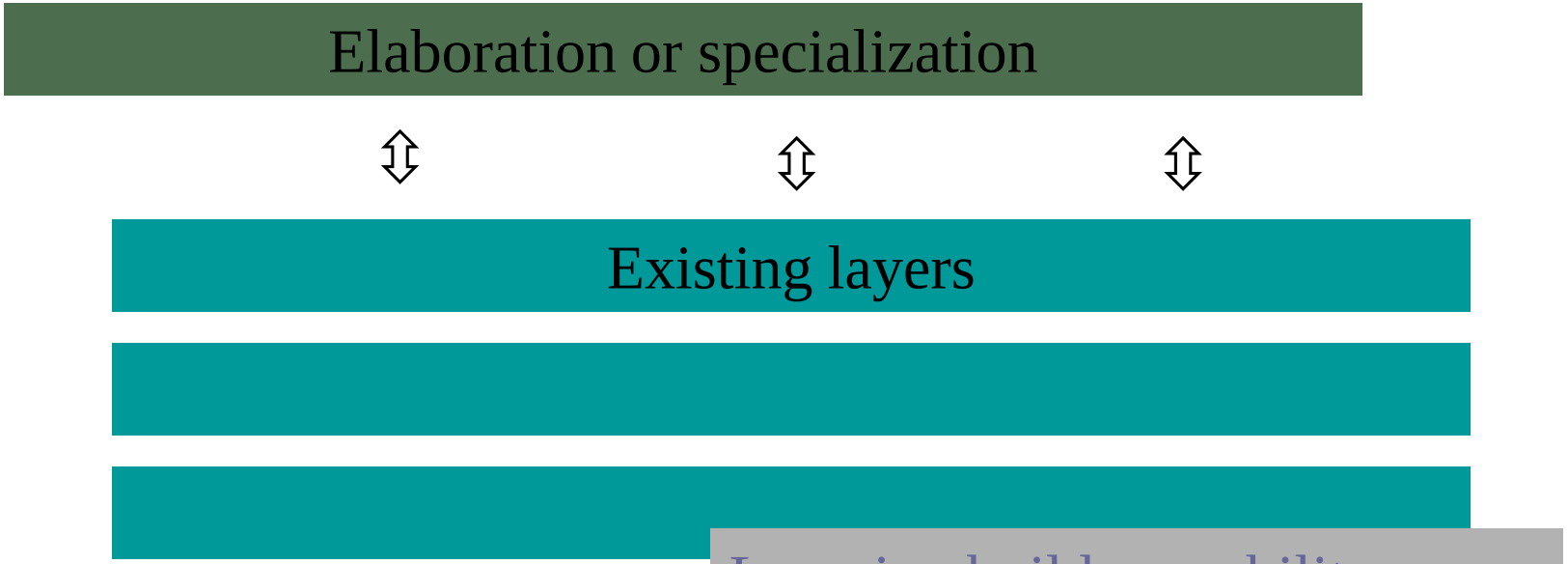
Example 1


Bob sends a letter to Alice
Bob
Envelope
US Postal Service
Shipping Container
Alice
Envelope
UK Royal Mail
Shipping Container
ABC Airlines

Major layers

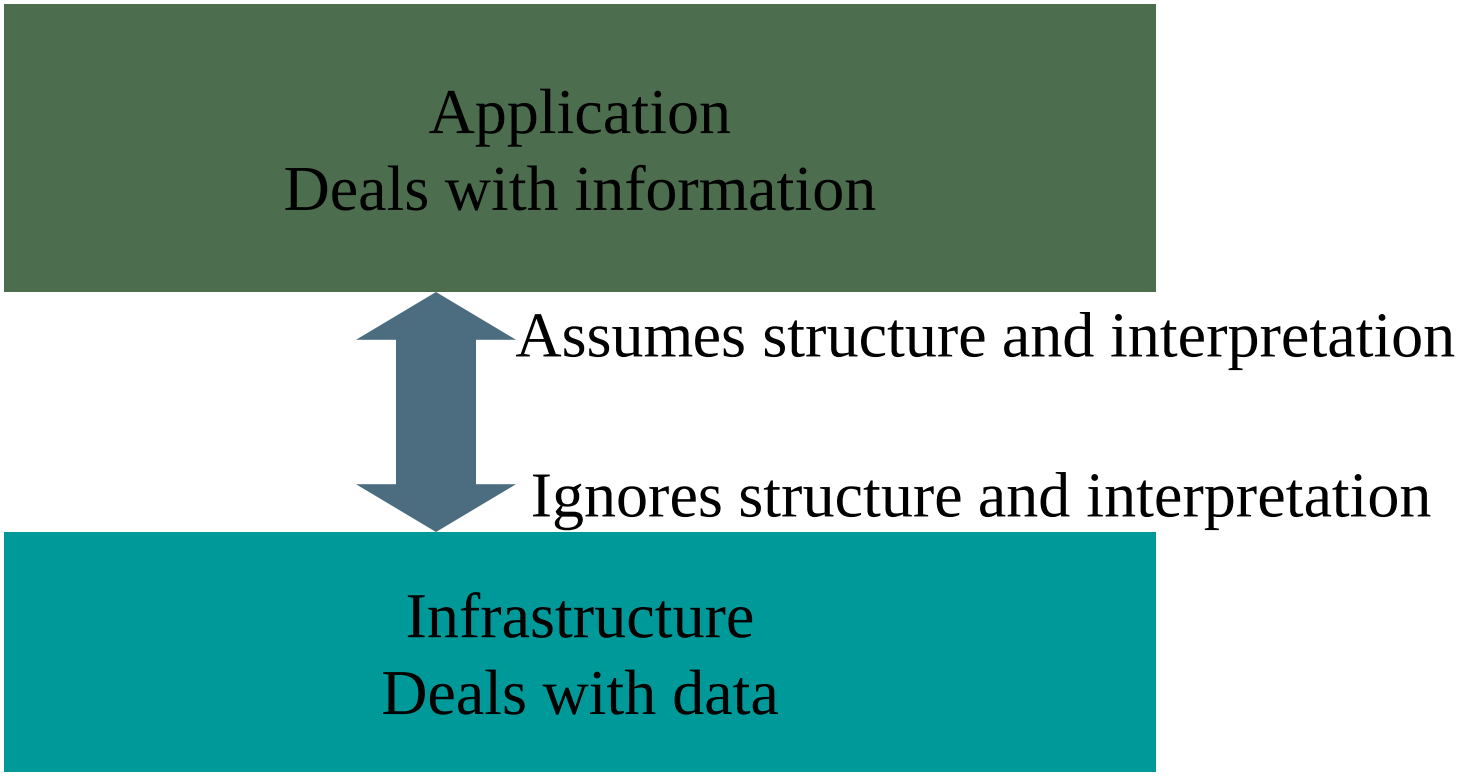


Layering

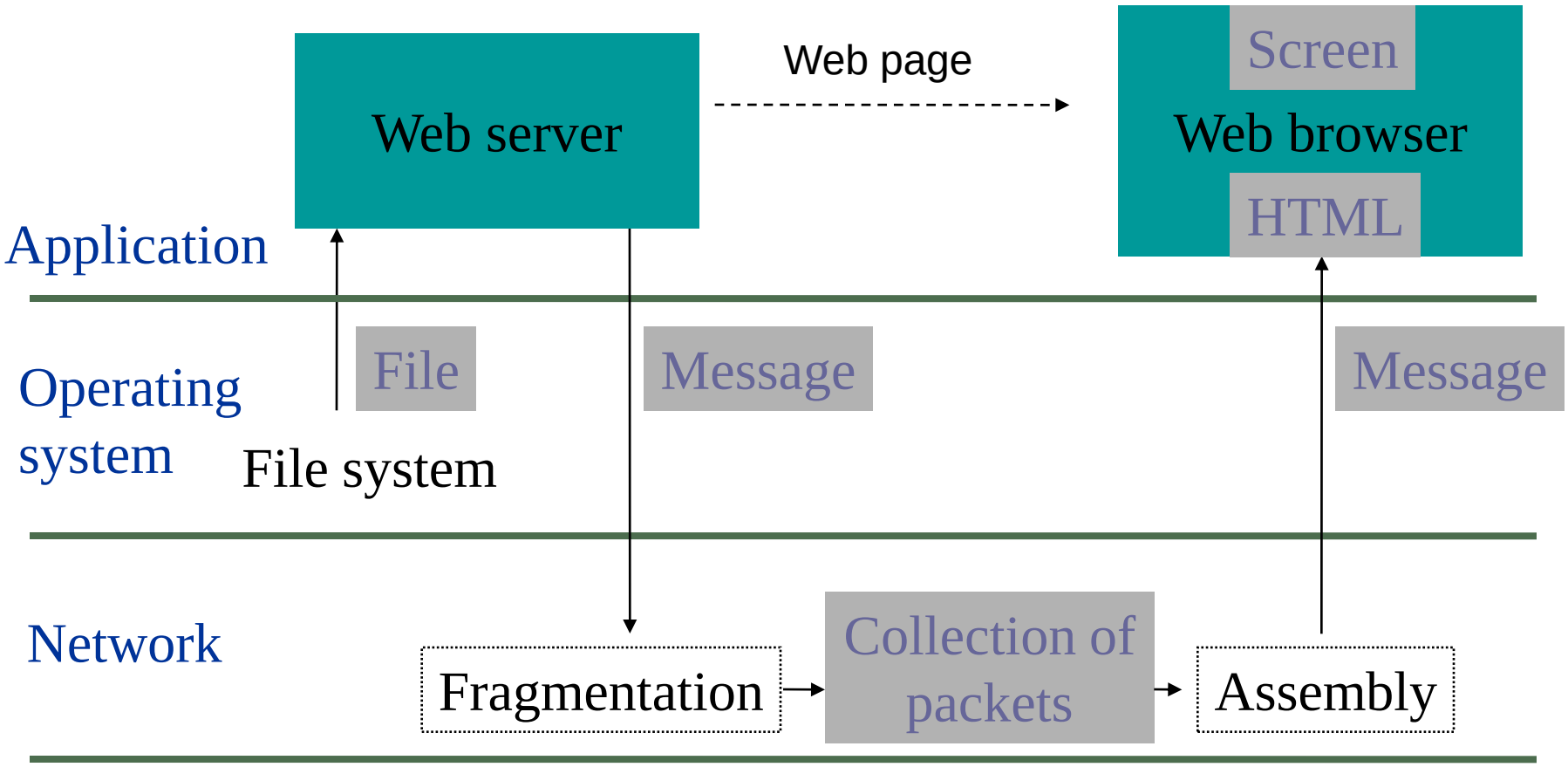


Layering builds capability incrementally by adding to what exists

Data and information



Example 2


Web server
Web page
Screen
Web browser
HTML
Application
Operating system
File
Message
Message
File system
Network
Fragmentation
Collection of packets
Assembly

Package = file or message

Infrastructure deals with a package of data (non-standard terminology)

- collection of bits
- specified number and ordering

Infrastructure stores and communicates packages while maintaining **data integrity**

→ File for storage

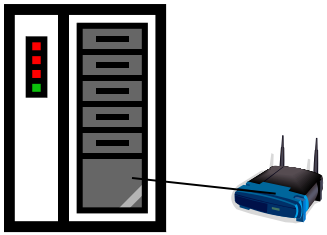
→ Message for communication

Data integrity

- Nothing is lost/changed in the representation/recovery of information
- Retain the
  - values
  - order
  - number

  of bits in a package
- Also applies to more complicated forms of representation and data processing
  - E.g. Data Integrity in Databases

Example 3



HHC Server

HHC

HHC Server Application

Passenger Information

HHC Client Application

message

message

Windows OS

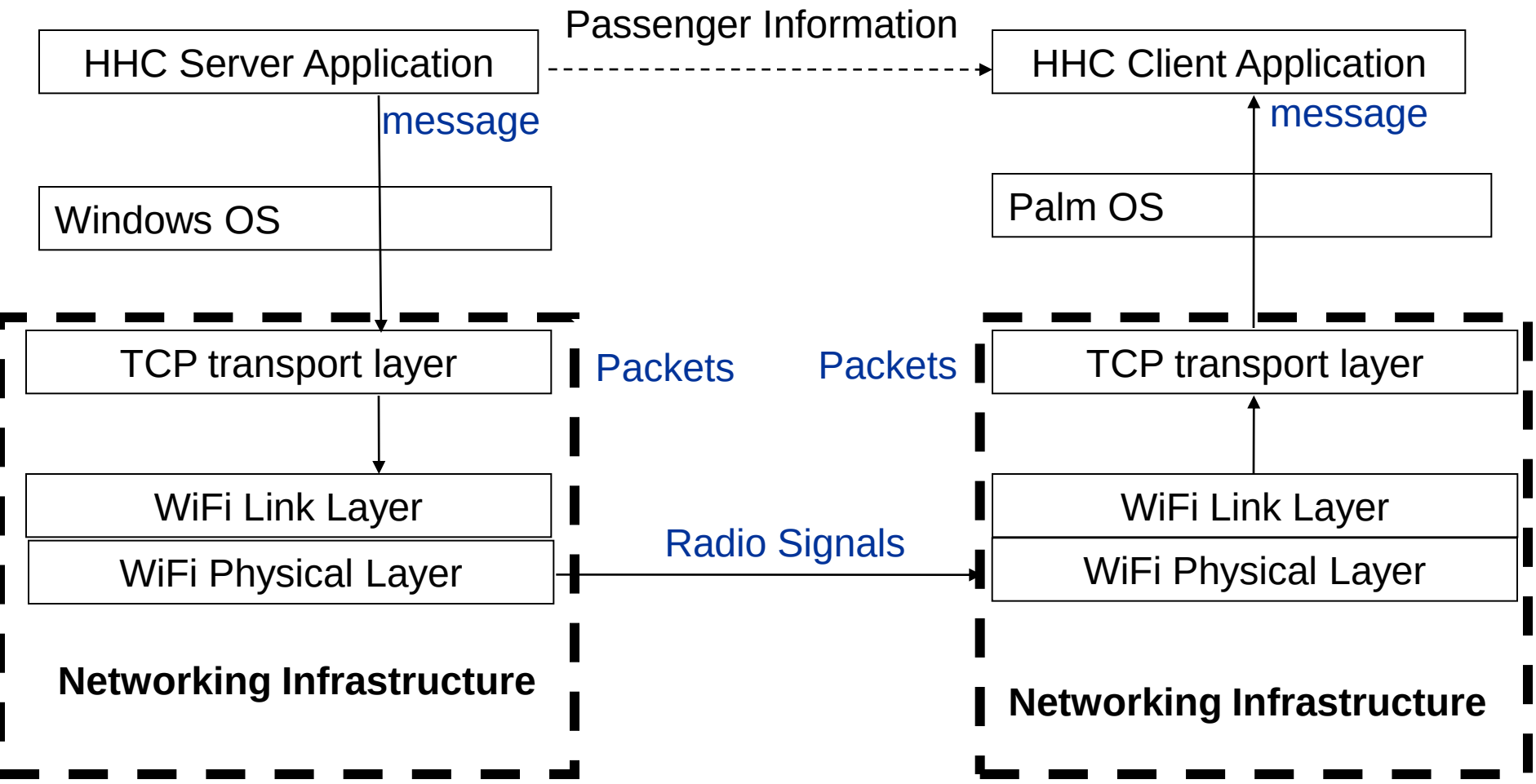
Palm OS

Networking Infrastructure
(Contains: TCP/IP, WiFi)

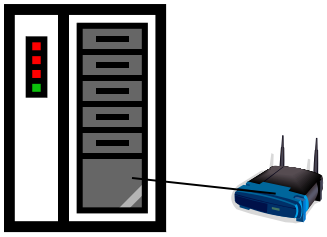
Collection of Packets

Networking Infrastructure
(Contains: TCP/IP, WiFi)

Example 3: Network Infrastructure Expanded

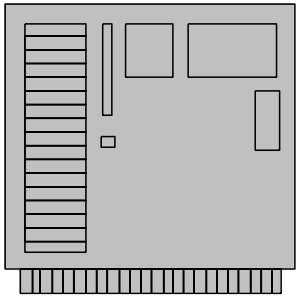


Example 4



HHC Server

HEADQUARTERS



Airline Dataserver

HHC Server Application

"Send me today's flight information"

DBMS

message

message

Windows OS

Unix OS

Networking Infrastructure Layers within TCP/IP, WiFi

Collection of Packets

Networking Infrastructure Layers within: TCP/IP, WiFi

Data and information in layers

- The infrastructure should deal with data,
  - or at most minimal structure and interpretation

- The application adds additional structure and interpretation

- This yields a separation of concerns

Information in the infrastructure

- *Sometimes* it is appropriate for the infrastructure to assume structure and interpretation for data
  - to add capabilities widely useful to applications
  - to help applications deal with heterogeneous platforms, where representations differ
- Data types