Understanding Networked Applications:
A First Course

Chapter 6

by

David G. Messerschmitt

Goal

- Appreciate the importance of complexity management in networked computing
- Understand better the role of architecture in complexity management
- Examine infrastructure layering in more depth

Complexity

- A system that cannot be understood in all its detail by a single person or small group of people is complex
- The intricacy of the logic embodied in software
  - suffers no physical limitations
  - complexity is a primary limitation
  - advances allow us to extend that complexity

Some sources of complexity

- Problem domain is complex
- Top-down design (as opposed to independent actors in the economy)
- Software is not adaptable like people
- Large team efforts required
- Integration of heterogeneous suppliers

Caution

- The applications considered in this course are relatively simple
- We have addressed
  - only the top of the hierarchy
  - ignored details
  - but this is the essence of hierarchical design: make that which is complex appear simple

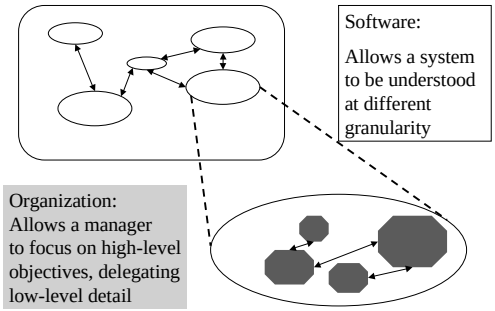
Some solutions to complexity

- Modularity properties
  - separation of concerns
  - reuse
- Interoperability through interfaces
  - abstraction
  - encapsulation

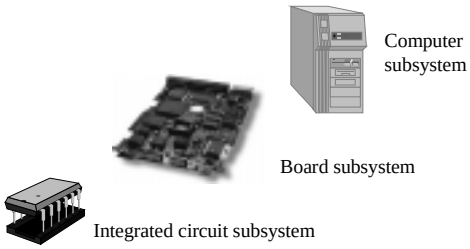
Modularity

- A system is modular when it is divided into subsystems (called modules) with good properties
  - Modules have distinct functional groupings
  - Hierarchy supports views at different granularity and scale
  - Separation of concerns among modules
  - Reusability of some modules

Hierarchy



Hierarchy in hardware architecture



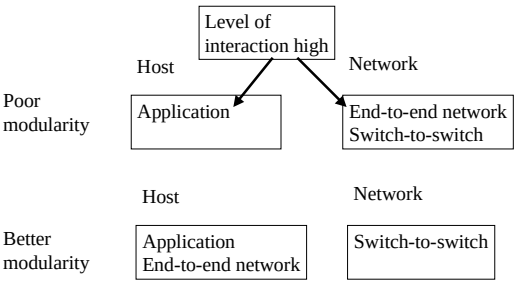
Separation of concerns

- The assignment of functionality to different modules should allow them to be designed and implemented as independently as possible
- The level of interaction
  - may be internally high
  - should be externally low
- They can then be assigned to different groups or companies for design
  - minimum coordination costs

Physical-world example

	Customer service	Credit checking
Poor modularity	Customer service Janitorial	Credit checking Floor polishing
	Loan department	Physical plant
Better modularity	Customer service Credit checking	Janitorial Floor polishing

Infrastructure example



Parts of a module

Module

Interface

What other modules see

Implementation

What only the implementer sees

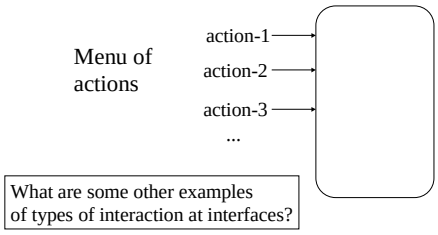
Interfaces

- Focus of module interaction and interoperability
- Two purposes:
  - Informs other modules how to interact
  - Informs implementer about what has been promised to other modules

Hardware interface

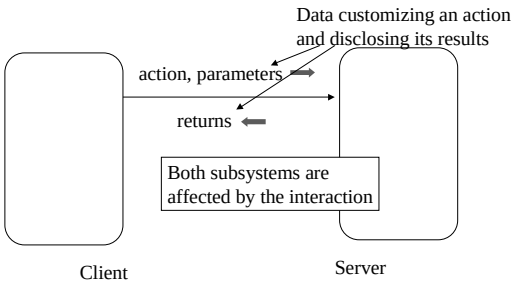
- Physical connection
- Electrical properties
- Formats of data passing through the interface (structure and interpretation)

Possible software interface



What are some other examples of types of interaction at interfaces?

Module interaction through interfaces



Data types

- Data passing an interface is often specified in terms of a limited number of standard data types
- Data type = range of values and allowable manipulation
- Data type does *not* presume a specific representation, to allow heterogeneous platforms
  - Representation must be known when data passes a sepecific module interface

Example data types

- Integer
  - "natural number between -32,767 and +32,768"
  - Could be represented (in many ways) by 16 bits
    - since $2^n = 65{,}536$
- Float
  - "number of the form $m*10^n/32768$, where m is in the range -32,767 to +32,768 and n is in the range -255 to +256"
  - Could be represented by 16+8 = 24 bits

More data types

- Character
  - "values assuming a-z and A-Z plus space and punctuation marks"
    - could be represented by 7 or 8 bits
- Character string
  - "collection of n characters, where n is customizable"
    - could be represented by $7*n$ bits

Compound data types

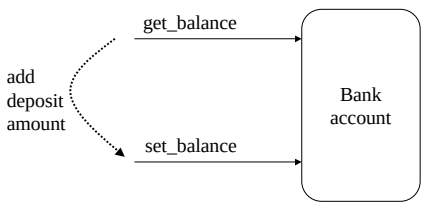
- Programmer-defined composition of basic data types
- Example:

```
Employee {
    String name;
    String address;
    Integer year_of birth;
    etc.
    }
```

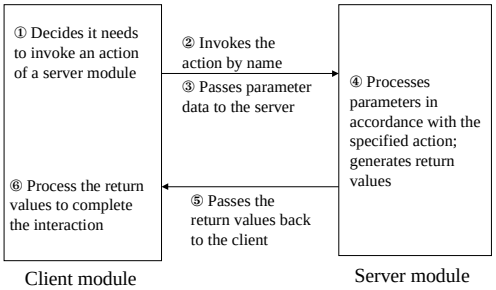

Protocols

- A defined sequence of actions between/among two or more subsystems required to achieve some higher-level functionality
- Interface specification focuses on actions (including formats of parameters and returns) *and* protocols

Example protocol: deposit



Anatomy of an action invocation



Understanding Networked Applications:
A First Course

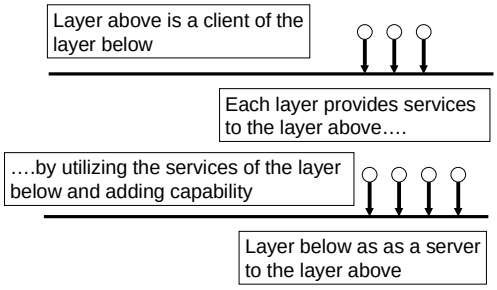
More on layering

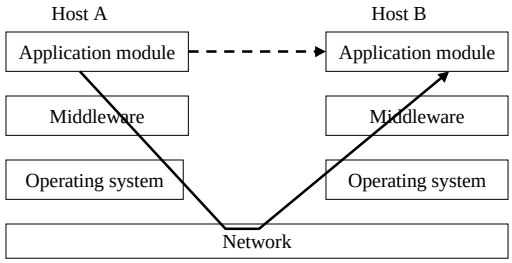
by
David G. Messerschmitt

Goals

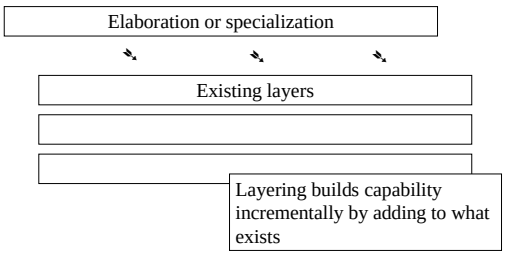
- Understand better
  - how layering is used in the infrastructure
  - how it contains complexity
  - how it coordinates suppliers
  - how it allows new capabilities to be added incrementally

Interaction of layers





Layering



Three types of software

Application

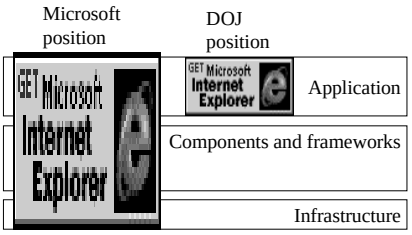
•Components and frameworks:

What is in common among applications

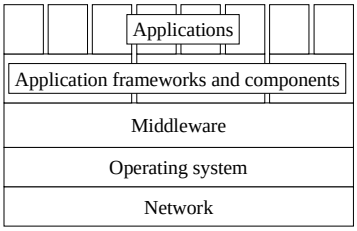
•Infrastructure:

Basic services (communication, storage, concurrency, presentation, etc.)

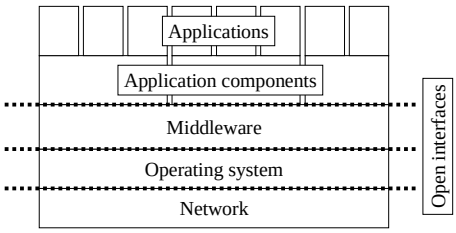
Part of Microsoft vs. DOJ dispute



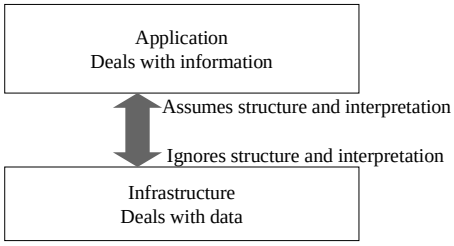
Major layers



Open layer interfaces



Data and information



Data and information in layers

- The infrastructure should deal with data, or at most minimal structure and interpretation of data suitable for a wide range of applications
- The application adds additional structure and interpretation
- This yields a separation of concerns

Package = file, message

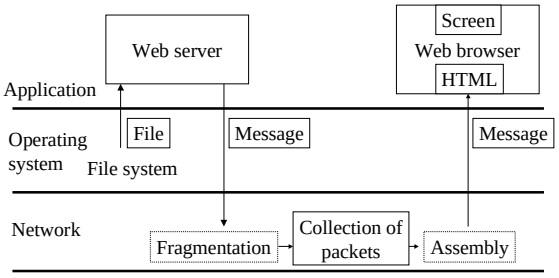
- In the simplest case, the infrastructure deals with a package of data (non-standard terminology)
  - collection of bits
  - specified number and ordering
- The objective of the infrastructure is to store and communicate packages while maintaining data integrity
- File for storage, message for communication

Data integrity

- Retain the
  - values
  - order
  - number

of bits in a package

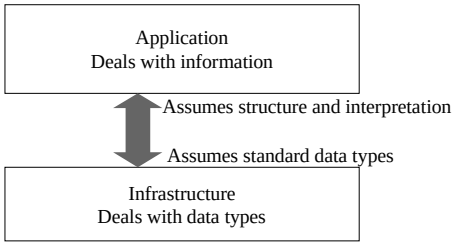
Example



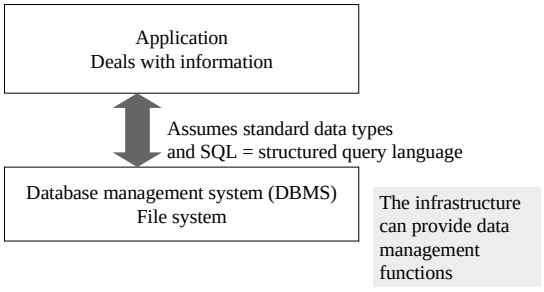
Information in the infrastructure

- 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
- At most, data types

Data and information

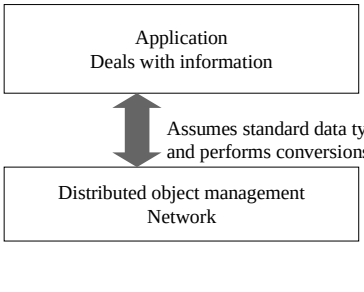


Storage



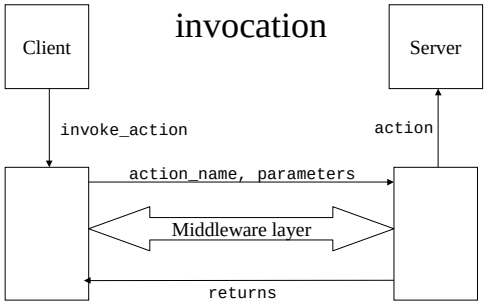
The infrastructure can provide data management functions

Communication

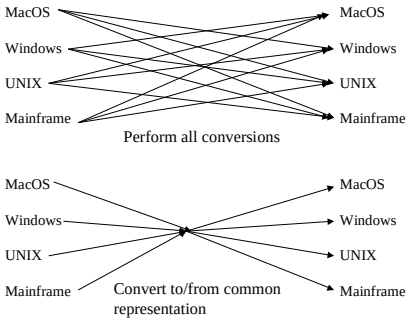


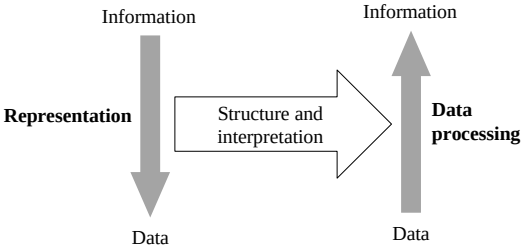
The infrastructure can transparently convert representations across platforms

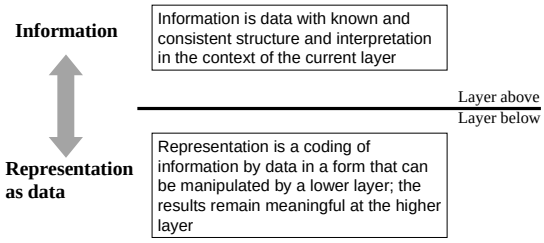
Idea behind remote action invocation



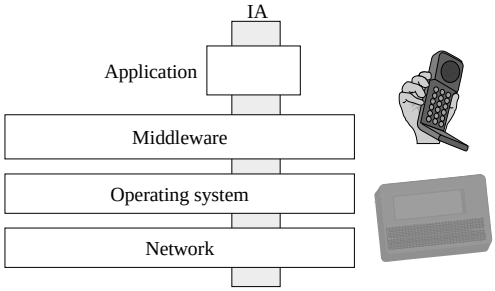
Using a common intermediate form







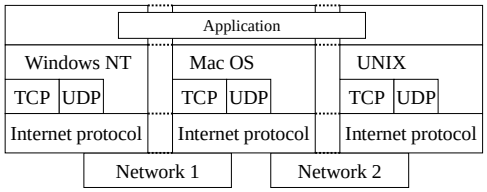
Information appliances



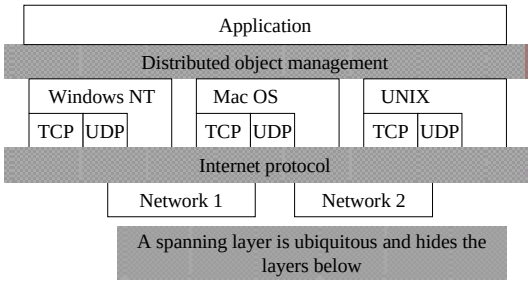
Question

- What advantages and disadvantages do you see for the information appliance?

Horizontal structure in layers



Spanning layer



Abstraction

- A property of well-designed interfaces to modules
- Hide detail, displaying only what is necessary
- Simplify, displaying only what is meaningful to the outside
- Important for complexity management

Encapsulation

- Module implementation details (anything not explicit at interface) should be inaccessible from the outside
  - So other modules cannot become inadvertently dependent on implementation
  - In the case of components, for proprietary or security reasons

Summary of modularity

- Divide and conquer: decomposition of the system into modules with well-defined functional groupings
- Separation of concerns: great dependency internally, little dependency externally
- Abstraction: hide detail and simplify
- Encapsulation: make internal implementation inaccessible
- Reusability: meet generalized needs, configurable