Object-Oriented Programming

Yi-ting Chiang

Department of Applied Mathematics
Chung Yuan Christian University

September 13, 2023

Complex Systems

Many systems in the real world are complex

- Personal Computers
- Space Shuttle
- Cars
- Business Organizations

And most of these complex systems are hierarchic, which means they are composed of some major elements, and these elements are also composed of other elements.

Complex Systems

The main elements of personal computers

- central processing unit (CPU)
- motherboard (adapters, main memory)
- storage devices (hard drive, DVD/BD)
- input devices (keyboard, mouse, microphone, etc)
- output devices (monitor, printer, speaker, etc)

The main elements of CPU

- arithmetic logic unit (ALU)
- control unit
- cache memory (L1, L2, L3 cache)
- registers (general-purpose registers, program counter, program status registers, address registers)
- bus (data bus, address bus, control bus)

The main elements of ALU

- arithmetic components (Adder/Subtractor, Multiplier)
- logic components
- multiplexer

And most of the elements in ALU are composed of logic gates (for example, the NAND gate)

Complex Systems

Main components in CYCU:

- Academics (學術單位)
- Administration (行政單位)
- People

In Academics, there are

- College of Science
- College of Engineering
- College of Business
- College of Design
- College of Himanities and Education
- School of Law

Departments in CYCU College of Science includes

- Applied Mathematics
- Physics
- Chemistry
- Psychology
- Bioscience Technology
- Master Program in Nanotechnology

Complex Systems

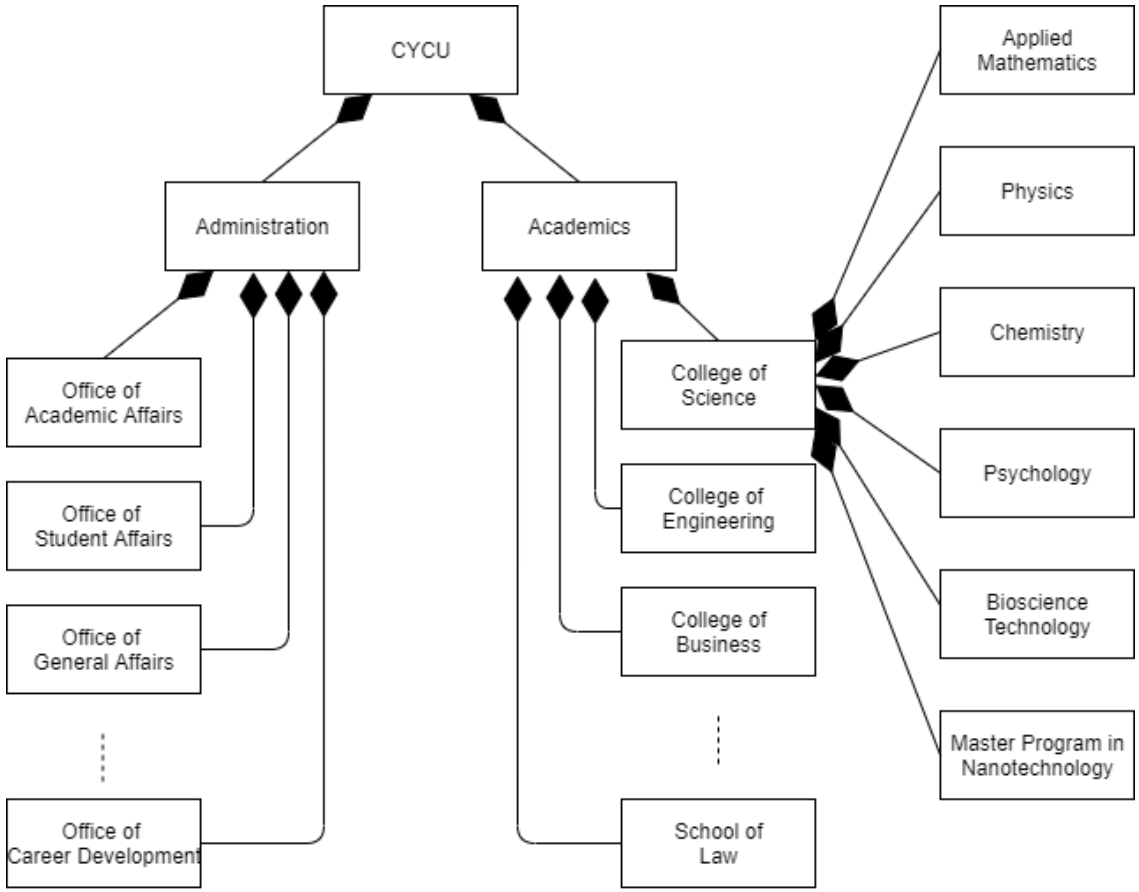


Figure: The Hierarchy of CYCU

Complex Systems

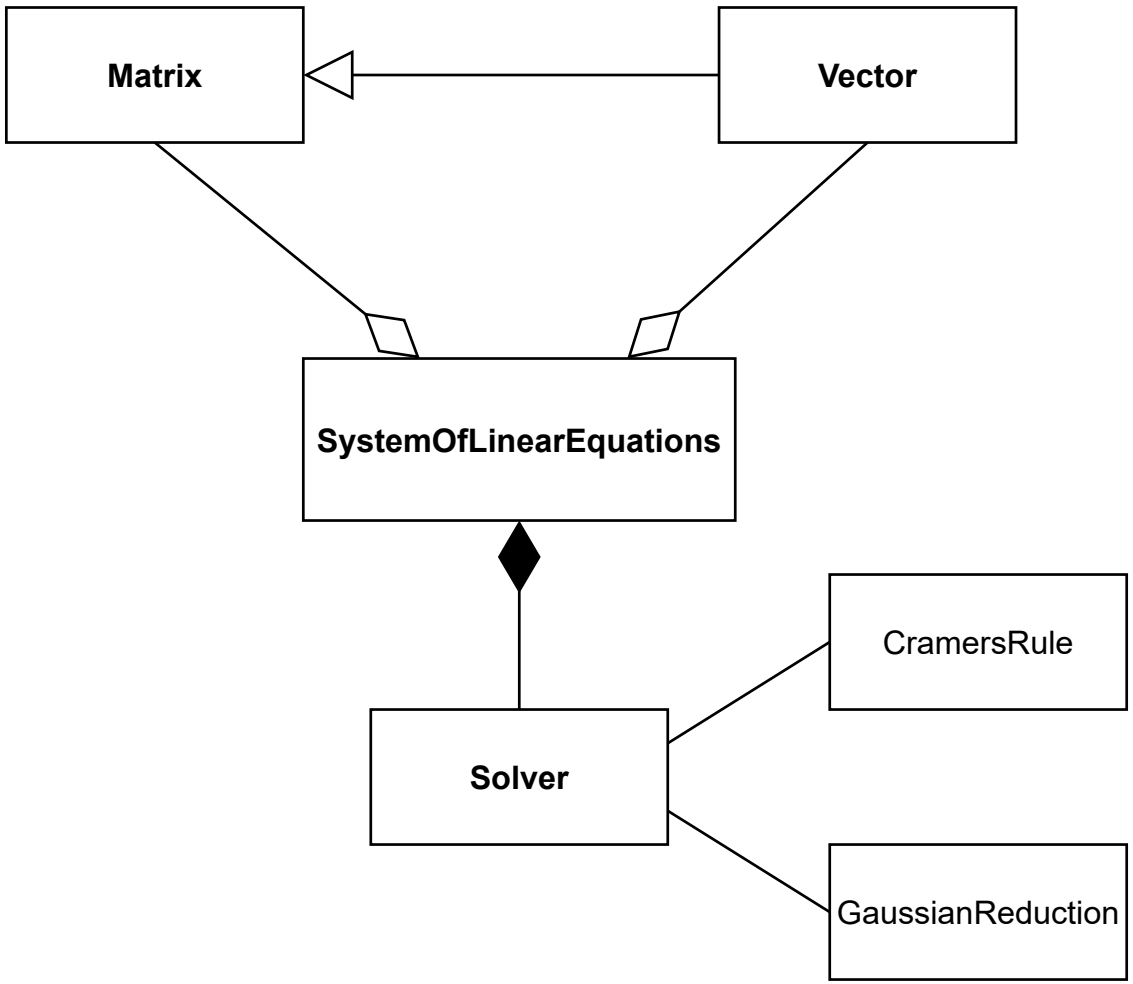


Figure: A solver system of the system of linear equations

Complex Systems

Decomposing a complex system

- Components of a complex system can be studied separately
  - One can independently study the operation of one element (for example, CPU) without knowing others (Hard drive)
- The hierarchy of complex system's components represents different levels of abstraction
  - Lower-level components provide services to higher ones
  - One can focus on a specific level for a particular purpose
    - To find out a memory failure problem, you don't need to check if there is any software bug
- Complex systems may be composed of similar building blocks
  - NAND gates compose many computer elements
  - Different kinds of cells are the building blocks of both plants and animals

The Inherent Complexity of Software

Large software systems

- need to be reused
- need to be repaired
- need to extend their functionality
- are difficult (even impossible) for an individual developer to understand or maintain them

The Inherent Complexity of Software

Software is inherently complex due to

- the complexity of the problem domain (external complexity)
  - requirements are difficult to comprehend (for large systems like automatic driving systems)
  - requirements may be competing or even contradictory (performance or reliability v.s. cost)
  - requirements often changes during its development
  - communication gap between users and developers
    - users are usually hard to express their needs in the form that developers can understand
    - users may only have vague ideas of what they want
    - user requirements may changes after they use the system and better understand their needs
- the difficulty of managing the development process
- the flexibility possible through software
- the problems of characterizing the behaviour of discrete systems

The Inherent Complexity of Software

Software is inherently complex due to (cont.)

- the complexity of the problem domain (external complexity)
- the difficulty of managing the development process
  - today, software systems are usually composed of a large number of lines of codes
  - software systems are usually developed by not one person but a team of developers
  - more developers means more complex communication
    - they may even in different countries!
- the flexibility possible through software
- the problems of characterizing the behaviour of discrete systems

The Inherent Complexity of Software

Software is inherently complex due to (cont.)

- the complexity of the problem domain (external complexity)
- the difficulty of managing the development process
- the flexibility possible through software
  - software is often customized case by case
  - there is usually no uniform building codes and standards
  - software developers have to express almost any kind of abstraction
  - software developers have to craft virtually all the primitive building blocks on which these high-level abstractions stand
- the problems of characterizing the behaviour of discrete systems

The Inherent Complexity of Software

Software is inherently complex due to (cont.)

- the complexity of the problem domain (external complexity)
- the difficulty of managing the development process
- the flexibility possible through software
- the problems of characterizing the behaviour of discrete systems
  - a large application may contains hundreds or even thousands (but a finite numbers) of variables
    - it is discrete, and can have a very large number of possible states
  - its behaviour is not continuous
    - small changes in inputs can cause large changes in outputs
    - an unexpected input can corrupt the system
    - vigorous testing is necessary, but exhaustive testing is usually impossible

Five Attributes of a Complex System

The following five attributes are common to all complex systems

- Hierarchic Structure
  - the architecture of a complex system is a function of its components as well as the hierarchic structure relationships among these components
- Relative primitives
  - an component may be considered primitive by one person while another observer says it is at a high level of abstraction
- Separation of concerns
  - intracomponent linkages are generally stronger (interacts more frequently) than intercomponent linkages
  - this helps to separate the systems into relatively isolated components
- Common patterns
  - complex systems have/reuse common patterns
    - For example, NAND gates are used in many computer components
- Stable intermediate forms
  - a complex system designed from scratch never works
  - you have to start from a system which is working and simple (in a stable intermediate form)

Canonical Form of a Complex System

The two hierarchies of systems:

- "part of" hierarchy
  - A wheel is one part of a car
  - Also called "object structure"
- "is a" hierarchy
  - A sports car is a specialized kind of car
  - Also called the "class structure"

Both hierarchies are layered. More abstract classes/objects are built on more primitive ones.

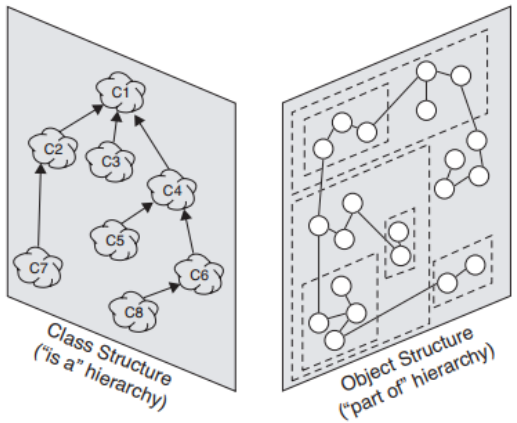
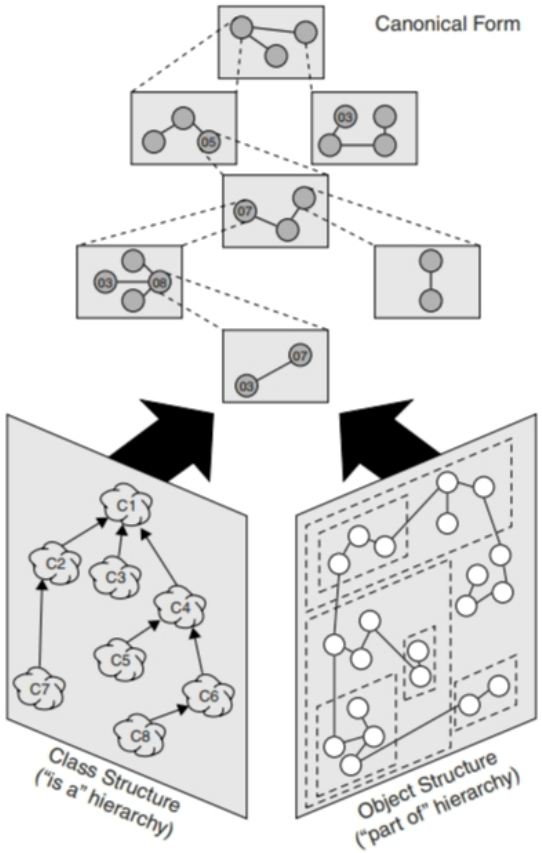


Figure: The two hierarchies of systems.[1]

[1] From: Object-Oriented Analysis and Design with Applications, 3rd

Canonical Form of a Complex System

The class and object structures of a system are system's architecture[2]



- Each object in the object structure represents a specific instance of some class
  - Notice that there are several object 03, which are the instance of class C3, in the canonical form
- Showing the system in both hierarchies helps to find the redundancy of the system

Figure: The canonical from of a complex systems.

[2]Figure from: Object-Oriented Analysis and Design with Applications, 3rd

To Deal with Complexity

A single person cannot keep track of all of the details at once

- Human can simultaneously understand a limited number of chunks of information
  - The channel capacity of the short-term memory
  - The information processing speed of human mind

Three ways help to deal with the complexity of software systems:

- Decomposition
- Abstraction
- Hierarchy

The Role of Decomposition

Why decomposition?

- The capacity of human for dealing with complexity is limited
- Decompose the system so the developers need only to comprehend a few parts at once

The ways to do decomposition

- Algorithmic decomposition: highlights the ordering of events
- Object-Oriented decomposition: emphasizes the agents that either cause action or are the subjects on which these operations act

The Role of Decomposition

Algorithmic decomposition

- Decompose the system into modules which denote major steps
- Highlight the ordering of events

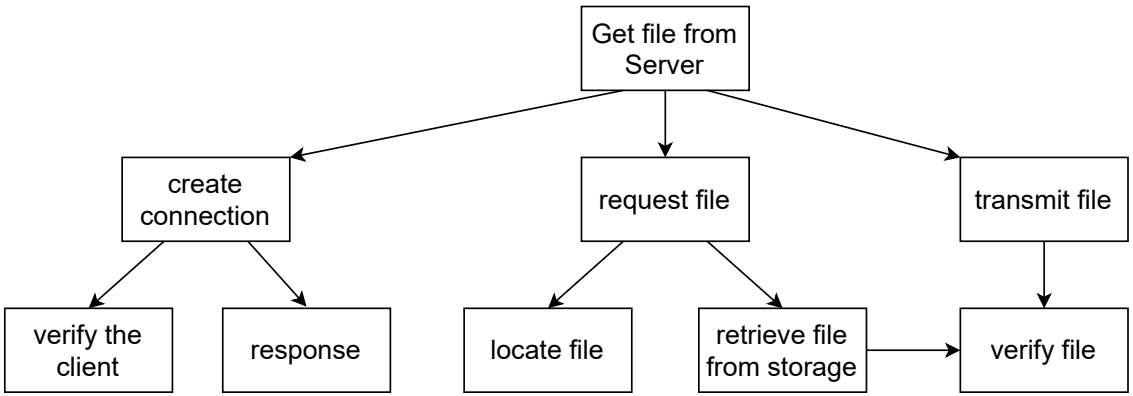


Figure: Algorithmic decomposition of a part of a program that requests a file from the server.

This figure shows various functional elements (or "steps") and the relationships between these elements

The Role of Decomposition

Object-oriented decomposition

- Decompose into key abstractions (objects), not steps
- There are a set of autonomous agents that collaborate to perform the jobs
- Modules (steps) in algorithm decomposition are operations of the objects
  - "transmit file" becomes an operation of the server

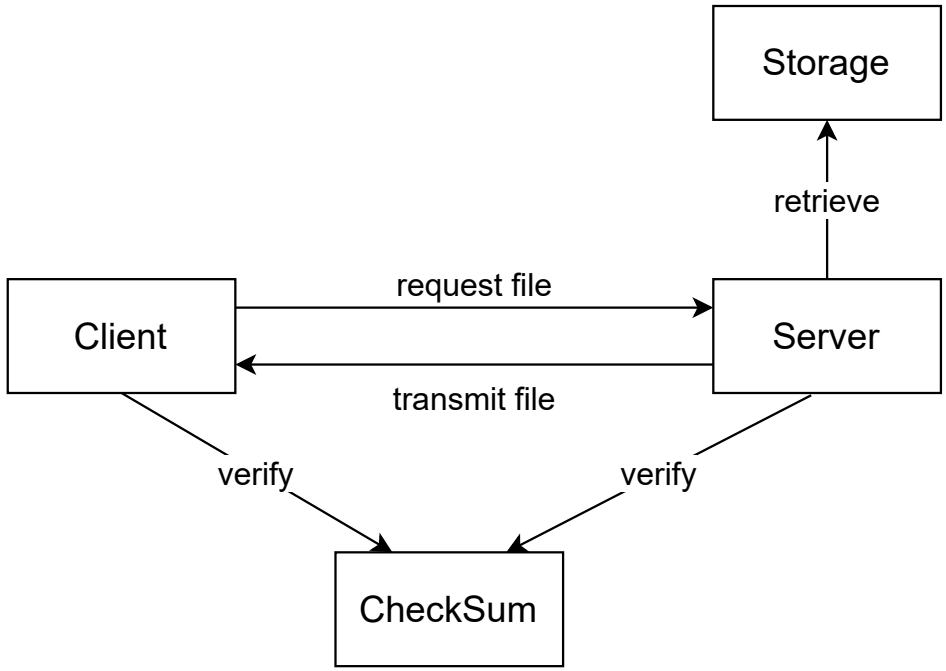


Figure: Object-Oriented decomposition view the world as a set of agents.

The Role of Decomposition

Algorithmic decomposition and object-oriented decomposition provide completely different views of the system. We can only choose one way and then use the resulting structure as the framework for expressing the other perspective.

It's recommended to apply object-oriented view first because

- it helps to organize the inherent complexity
- it yields smaller systems through the reuse of common mechanisms
- based on stable intermediate forms, it is more resilient to change
- designed to evolve incrementally from smaller systems, it greatly reduces the risk of building complex software systems

The Role of Abstraction

- An individual can comprehend only about seven, plus or minus two, chunks of information at one time[3]
- Therefore, we need to ignore the inessential details and deal with the generalized, idealized model of the object
- By abstraction, we use chunks of information with increasingly greater semantic content

[3] Miller, G. March 1956. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. The Psychological Review vol. 63(2), p. 86.

The Role of Hierarchy

The roles of hierarchy are

- to recognize the class and object hierarchies within a complex software system
  - Object (part of) hierarchy helps to understand how different objects collaborate with one other
  - Class (is a) structure highlights common structure and behaviour within a system
- to classify objects into groups of related abstractions so we can distinguish the common and distinct properties of different objects

Design Complexity Systems

The purpose of design is to construct a system on the environment that satisfies:

- Functionality:
  - Satisfies a given (perhaps informal) functional specification
- Constraints of developing the system:
  - satisfies implicit or explicit design criteria
  - satisfies restrictions on the design process itself, such as its length or cost, or the tools available for doing the design
- Constraints of running the system:
  - meets implicit or explicit requirements on performance and resource usage
  - agree with the limitations of the target medium

Design Complexity Systems

There is no "best" design method. Object-oriented analysis and design is just the method for object-oriented decomposition.

Applying object-oriented design helps

- create the software that is easy to and adaptive to change, and written with economy of expression
- achieve a greater level of confidence in the correctness of the software
- reduce the risk inherent in developing complex software systems