Object-Oriented Analysis and Design using JAVA (20B12CS334)

B.Tech (CSE/IT) 5th SEM
2021-2022

Software complexity: benefits and understanding the challenges OOAD can address

Introduction to software complexity

- The first challenge when talking about software complexity is to answer the question: "What is complexity?" There is no consensus on how to define software complexity. It is widely believed that software complexity cannot be described using a single dimension.

- Brooks is referring to an often used classification of complexity: accidental and essential. "The complexity of software is an essential property, not an accidental one. Hence, descriptions of a software entity that abstract away its complexity often abstract away its essence."

- Essential complexity arises from the nature of the problem and how deep a skill set is needed to understand the problem. The reason is that not all complexity is essential. While some complexity is inherent to the problem, we also bring our own complexity while writing the program. This is called accidental complexity.

Brooks,Fredrick P."The Mythical Man Month: Essays on Software Engineering." Addison-Wesley, 1995.
Bill Curtis,"Measurement and experimentation in software engineering." Proceedings of the IEEE, vol.68,no.9,pp.1144–1157, 1980.

- IEEE defines software complexity as “the degree to which a system or component has a design or implementation that is difficult to understand and verify”.

- The concept of software complexity can be categorized into algorithmic and psychological complexity.

- Algorithmic (or computational) complexity characterizes the run-time performance of an algorithm.

- Psychological complexity affects the performance of programmers trying to understand or modify a code module.

IEEE Computer Society: ”IEEE Standard Glossary of Software Engineering Terminology.” IEEE std.,1990.

- Zuse states that the term "software complexity" is poorly defined and that software complexity measurement is a misnomer. He offers a definition of software complexity as: "The true meaning of software complexity is the difficulty to maintain, change and understand software. It deals with the psychological complexity of programs."

- These definitions associate software complexity with the psychological complexity. Three specific types of psychological complexity that affect a programmer's ability to comprehend software have been identified: problem complexity, system design complexity, and procedural complexity.

Zuse H, "Software Complexity Measures and Methods." Walter de Gruyter,Berlin–New York, 1991.

- Problem complexity is a function of the problem domain. Simply stated, it is assumed that complex problems are more difficult for a programmer to comprehend than simple problems.

- System design complexity addresses the mapping of a problem space into a given representation.

- Procedural complexity is associated with the logical structure of a program.

Zuse H, “Software Complexity Measures and Methods.” Walter de Gruyter,Berlin–New York, 1991.

Why Software Is Inherently Complex

- Complexity of problem domain
- Difficulty of managing development process
- Flexibility afforded by software
- Difficulty of characterizing discrete system behaviour

Complexity of Problem Domain

- Many, often contradictory, requirements
- functional (what must be done)
- non-functional (usability, cost, performance, consumption,...)
- Communication gap between customers and developers
- Evolving requirements

Referred sources mentioned in Key References

Difficulty of Managing Development Process

Fundamental task of software development: engineer the illusion of simplicity

However, Modern systems are huge

- Development team is necessary
- More developers =>
- more complex communication
- more difficult coordination
- harder to maintain design unity/integrity

Referred sources mentioned in Key References

Flexibility Afforded by Software

- Software is the ultimate flexible product
- It is technically possible for any developer to create anything with it
- This is both a blessing and a curse
- Other industries have specialization, codes and quality standards
- Software development remains a mostly artisanal labor-intensive business

Referred sources mentioned in Key References

Difficulty of Characterizing Discrete Systems Behavior

- Physical (analog) systems exhibit continuous behavior
- Small external perturbations produce small changes in behavior
- Software (digital) systems exhibit discrete behavior
- Small changes in input can produce large changes in output
- Discrete systems have a combinatorial state explosion
- Describing their behavior precisely and formally is very challenging in general
- Most software professionals are poorly trained for that

Referred sources mentioned in Key References

Handling the Complexity

- Two parts
  - Organized Complexity
    - canonical form of complex system
  - Disorganized Complexity
    - limitation of dealing with human capacity

Hierarchies of complex system

- There are many types of hierarchies possible but two are significantly impact the design process
  - Decomposition - **part-of** and **Has-a**
  - Abstraction - **IS-A**

Hierarchy

- Example of decomposition hierarchy
  - A PC is composed of
    - CPU
    - memory
    - keyboard
    - HDD and many more

- Example of Abstraction hierarchy
  - A CPU can be one of pentium, pentium II, pentium III, celeron, …
  - Processors may be different but they share the same functionality like Pentium **IS-A** CPU and celeron **IS-A** pentium II

class and object structures

The discussed hierarchy can be referred to as

- **Class** Structure: Abstraction
- **Object** Structure: Decompositions

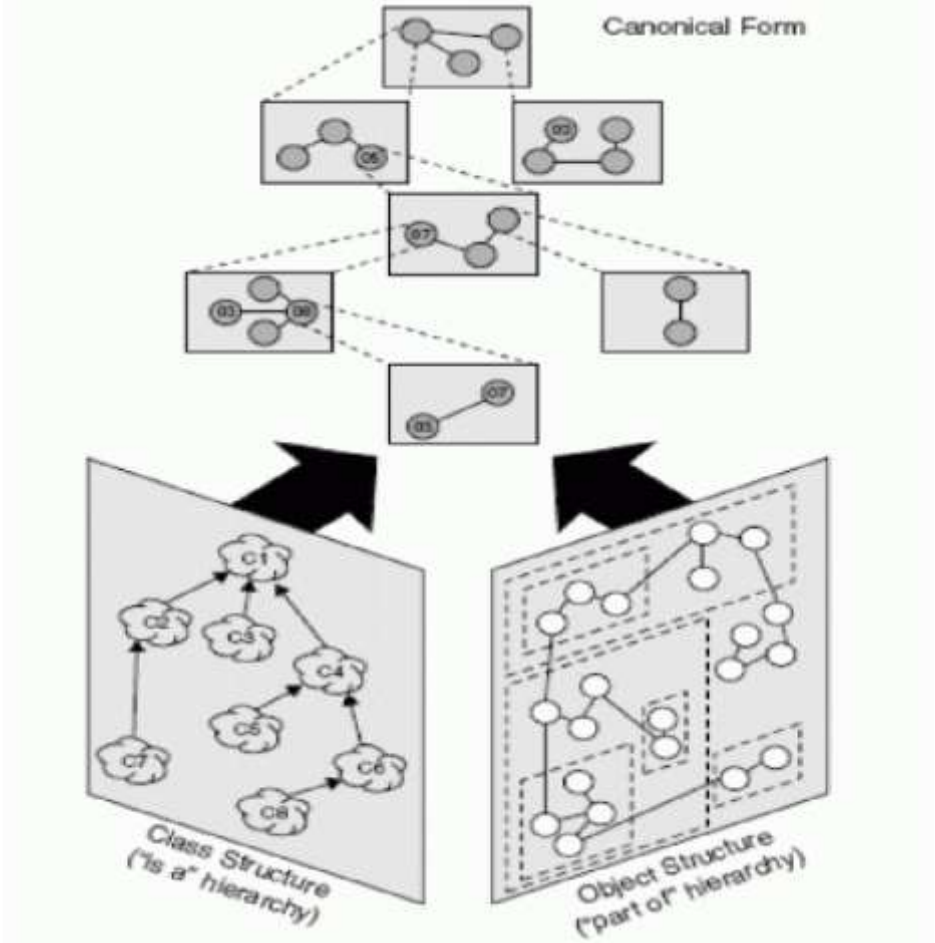
Note:

1. These are class and object structures at higher level of abstraction and not same as coding in software
2. these are makeup of complex systems

Canonical Form of a Complex System

- System Architecture
  - class structure
  - object structure
- Five attributes
  - Hierarchic structure
  - Relative primitives (multiple level of abstraction)
  - Separation of concepts
  - Common patterns
  - Stable Intermediate Forms
- **Organized complexity**

Canonical Form of a Complex System



Referred sources mentioned in Key References

Disorganized complexity

- Human beings
- maximum chunks of information that an individual can simultaneously comprehend is on the order of seven, plus or minus two
- short-term memory
- processing speed is limited - mind take 5 sec to accept new chunk of information

Successful Complex Software Systems

Exhibit the following attributes characterizing good complex systems.

Have well designed and built class and object structures

- captures common features and behaviour within a system
- Illustrates how different objects collaborate with one another

Benefits and challenges OOAD can address

- Among the many claimed benefits of OO systems are faster development, higher quality, easier maintenance, reduced costs, increased scalability, better information structures, and increased adaptability.

- One of the primary reasons for these claims is that OO approaches control complexity of a system by supporting hierarchical decomposition through both data and procedural abstraction.

- The complexity of OO systems can be represented by a set of measures defined at different levels.

- For OO systems, the software complexity of OO systems are described at four levels:
    - variable,
    - method,
    - object, and
    - system.

E. Da-wei, "The Software Complexity Model and Metrics for Object-Oriented," *2007 International Workshop on Anti-Counterfeiting, Security and Identification (ASID)*, Xiamen, Fujian, 2007, pp. 464-469,

- Variable level complexity is associated with the definition and use of variables throughout the system.
- Method level complexity is associated with the definition and use of methods throughout the system.
- Object level complexity combines variable and method complexity with measures of the inheritance structure.
- System level complexity provides high level representations of OO system size and organization.
- At each level, measures are identified to account for the cohesion (intra) and coupling (inter) aspects of the system at that level.
- The measures account for both procedural and data characteristics of an OO system, and are applicable throughout the lifetime of an OO system.

E. Da-wei, "The Software Complexity Model and Metrics for Object-Oriented," *2007 International Workshop on Anti-Counterfeiting, Security and Identification (ASID)*, Xiamen, Fujian, 2007, pp. 464-469,

Key references

1. Object-Oriented Analysis and Design with Applications-Third Edition-Addition Wesley Authors-Grady Booch Robert A. Maksimchuk Michael W. Engle Bobbi J. Young, Ph.D. Jim Conallen Kelli A. Houston

Thank You