



# Reflections on Programming Methodology

---

Barbara Liskov  
MIT CSAIL  
December 2020



# My Background

---

- UC Berkeley Math 1961
- Mitre
- Harvard
- Stanford PhD (1963-1968)
- Mitre



# The Situation in 1970

---

- The software crisis!



# The Situation in 1970

---

- The software crisis!
- We did not understand how to build programs that worked
- Software development efforts failed



# Programming Methodology

---

- How should programs be designed?
- How should programs be structured?



# The Landscape

---

- E. W. Dijkstra. Go To Statement Considered Harmful. Cacm, Mar. 1968



# The Landscape

---

- N. Wirth. Program Development by Stepwise Refinement. Cacm, April 1971



# The Landscape

---

- D. L. Parnas. Information Distribution Aspects of Design Methodology. IFIP Congress, 1971
- “The connections between modules are the assumptions which the modules make about each other.”





# Modularity Today

---

- A program is a collection of modules
  - Each module has an **interface**, described by a **specification**
  - E.g., a sort routine



# Modularity Today

---

- A program is a collection of modules
  - Each has an **interface** described by a **specification**
  - A module's **implementation** is correct if it meets the specification



# Modularity Today

---

- A program is a collection of modules
  - Each has an **interface** described by a **specification**
  - A module's **implementation** is correct if it meets the specification
- **Local reasoning** provided using modules **depend only** on the specification



# Modularity in 1970

---

- We knew we wanted it
- We understood its benefits
  - Local reasoning
  - Independent development
  - Modifiability



# Modularity in 1970

---

- Procedures were the only type of module
  - e.g. a sort routine
- Not powerful enough
  - e.g., a file system
- Complicated connections



# Partitions

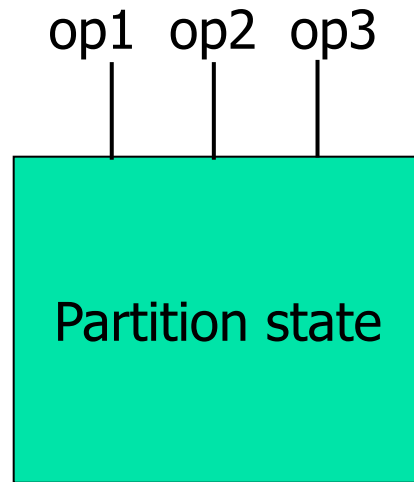
---

- B. Liskov. A Design Methodology for Reliable Software Systems. FJCC, Dec. 1972



# Partitions

---





# Move to MIT 1972

---





# From Partitions to ADTs

---

- How can these ideas be applied to building programs?



# Idea

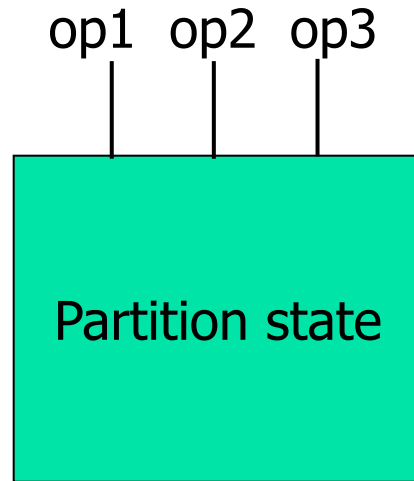
---

- Connect partitions to data types



# Partitions as Data Types

---





# Why This Idea Mattered

---

- Links modularity to design
  - Design by inventing abstractions
- Programmers understood data types
- They would be able to invent new ones
  
- But requires programming language support!



# Exploring Abstract Data Types

---

- Joint work with Steve Zilles



# The Landscape

---

- O-J. Dahl and C.A.R. Hoare. Hierarchical Program Structures. Structured Programming, Academic Press, 1972



# The Landscape

---

- J. H. Morris. Protection in Programming Languages. Cacm. Jan. 1973



# The Landscape

---

- J. H. Morris. Protection in Programming Languages. Cacm. Jan. 1973
  - Code outside the module must **not modify** the data managed by the module
  - Nor even **observe** it





# Abstract Data Types

---

- B. Liskov and S. Zilles. Programming with Abstract Data Types. ACM Sigplan Conference on Very High Level Languages. April 1974



# What That Paper Proposed

---

- Abstract data types
  - A set of objects
  - A set of operations
  - The operations provide the **only** way to create and use the objects
- A sketch of a programming language



# From ADTs to CLU

---

- Participants

- Russ Atkinson

- Craig Schaffert

- Alan Snyder

- Abstraction Mechanisms in CLU, B. Liskov et al, CACM August 1977





# Rationale

---

- Precise rules
- A programming language is a tool
  - Convenience
  - Expressive power
  - Performance



# Some Facts about CLU

---

- Static type checking
- Heap-based
- Separate compilation
- No concurrency, no gotos, no inheritance



# CLU Mechanisms

---

- Clusters
- Polymorphism (generics)
- Iterators
- Exception handling



# After CLU

---

- Distributed computing
  - Viewstamped replication
  - Practical BFT (Byzantine fault tolerance)
  - DIFC (Decentralized information flow control)





# After CLU

---

- Programming methodology
  - Modular program design
  - Reasoning about correctness
  - 6.170
    - With John Guttag



# After CLU

---

- Programming methodology
  - Modular program design
  - Reasoning about correctness
  - 6.170
  - Type hierarchy



# From CLU to Object-Oriented Programming

---

- SmallTalk provided **inheritance**



# From CLU to Object-Oriented Programming

---

- SmallTalk provided **inheritance**
- Inheritance was used for
  - Implementation
  - **Type hierarchy**



# Type Hierarchy

---

- Wasn't well understood
  - E.g., stacks vs. queues



# Behavioral Subtyping

---

- Objects of subtypes should **behave** like those of supertypes if used via supertype methods



# Behavioral Subtyping

---

- Objects of subtypes should behave like those of supertypes if used via supertype methods
- The “Liskov Substitution Principle”



# Behavioral Subtyping

---

- Objects of subtypes should behave like those of supertypes if used via supertype methods
  - B. Liskov. Data abstraction and Hierarchy. *Sigplan Notices*, May 1988
  - B. Liskov and J. Wing. A Behavioral Notion of Subtyping. *ACM Toplas*, Nov. 1994





# Programming Today

---

- Modularity based on abstraction is the way things are done



# Reflections on Programming Methodology

---

Barbara Liskov  
MIT CSAIL  
December 2020