# The Power of Abstraction

Barbara Liskov

October 2010

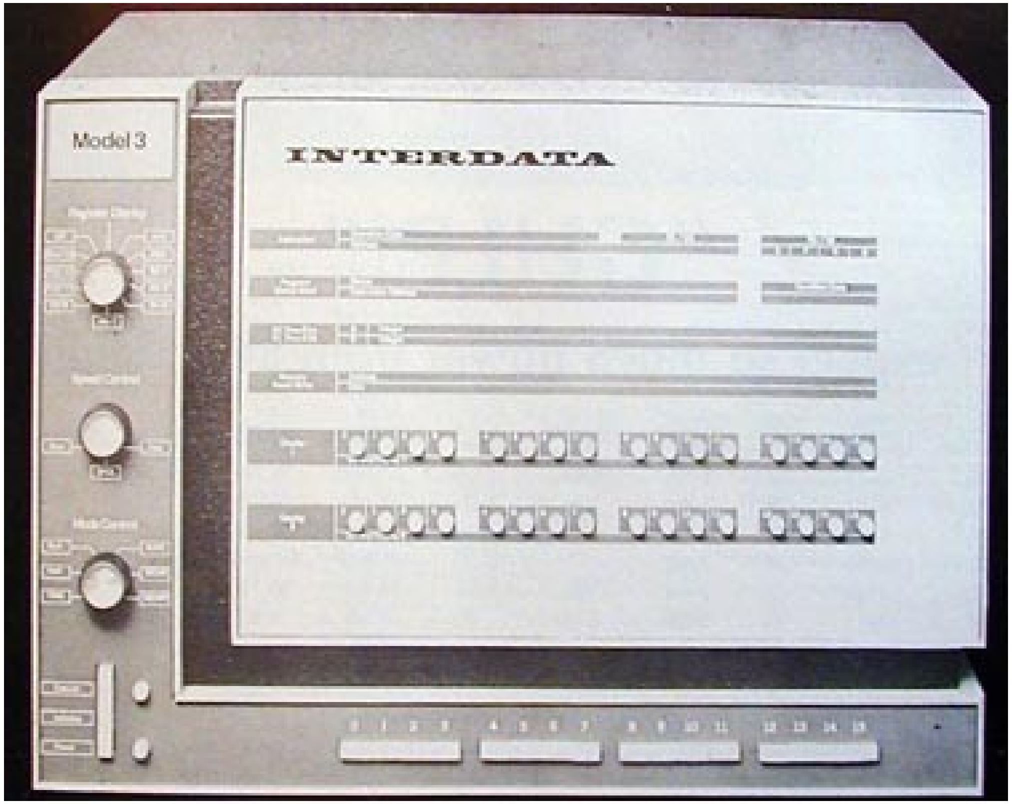
# Outline

- Inventing abstract data types
- CLU
- Type hierarchy
- What next

# Data Abstraction Prehistory

- The Venus machine

# The Interdata 3



# Data Abstraction Prehistory

- The Venus machine
- The Venus operating system

# Data Abstraction Prehistory

- The Venus machine
- The Venus operating system
- Programming methodology

# 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

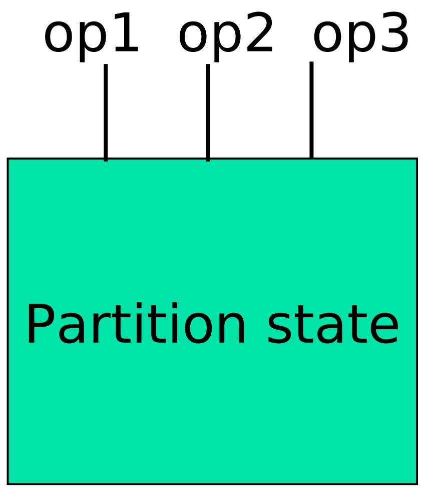
- D. L. Parnas. Information Distribution Aspects of Design Methodology. IFIP Congress, 1971

- “The connections between modules are the assumptions which the modules make about each other.”

# Partitions

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

# Partitions



# From Partitions to ADTs

- How can these ideas be applied to building programs?

# Idea

- Connect partitions to data types

# Meeting in Savanah

- ACM Sigplan-Sigops interface meeting. April 1973. (Sigplan Notices, Sept. 1973)
- Started to work with Steve Zilles

# The Landscape

- Extensible Languages
  - S. Schuman and P. Jourrand. Definition Mechanisms in Extensible Programming Languages. AFIPS. 1967
  - R. Balzer. Dataless Programming. FJCC 1967

# 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

- W. Wulf and M. Shaw. Global Variable Considered Harmful. Sigplan Notices. Feb. 1973.

# 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 operations
  - And a set of objects
  - The operations provide the only way to use the objects

# What that paper proposed

- Abstract data types
  - Clusters with encapsulation
- Polymorphism
- Static type checking (we hoped)
- Exception handling

# From ADTs to CLU

- Participants
  - Russ Atkinson
  - Craig Schaffert
  - Alan Snyder

# Why a Programming Language?

- Communicating to programmers
- Do ADTs work in practice?
- Getting a precise definition
- Achieving reasonable performance

# Language Design

- Goals
  - Expressive power, simplicity, performance, ease of use

  - Minimality
  - Uniformity
  - Safety

# Language Design

- Restrictions
  - No concurrency
  - No go tos
  - No inheritance

# Some Assumptions/Decisions

- Heap-based with garbage collection!
- No block structure!
- Separate compilation
- Static type checking

# CLU Mechanisms

- Clusters
- Polymorphism
- Exception handling
- Iterators

# Clusters

IntSet = cluster is create, insert, delete, isIn, ...

end IntSet

# Clusters

```
IntSet = cluster is create, insert, delete, ...
end IntSet

IntSet s := IntSet$create( )
IntSet$insert(s, 3)
```

# Clusters

```
IntSet = cluster is create, insert, delete,
    ...

    rep = array[int]
```

# Clusters

```
IntSet = cluster is create, insert, delete, …

    rep = array[int]

    create = proc ( ) returns (cvt)
        return (rep$create( ))
        end create
```

# Polymorphism

```
Set = cluster[T: type] is create, insert,
    ...
end Set



Set[int] s := Set[int]$create( )
Set[int]$insert(s, 3)
```

# Polymorphism

Set = cluster[T: type] is create, insert, ...

where T has equal: proctype(T, T)

returns (bool)

# Polymorphism

```
Set = cluster[T: type] is create, insert, ...
  where T has equal: proctype(T, T)
     returns (bool)

    rep = array[T]

    insert = proc (x: cvt, e: T)
       ... if e = x[i] then ...
```

# Exception Handling

- J. Goodenough. Exception Handling: Issues and a Proposed Notation. Cacm, Dec. 1975
  - Termination vs. resumption
  - How to specify handlers

# Exception Handling

```
choose = proc (x: cvt) returns (T)
    signals (empty)
        if rep$size() = 0 then signal empty
        ...
```

# Exception Handling

```
choose = proc (x: cvt) returns (T)
  signals (empty)
    if rep$size() = 0 then signal empty
    ...

set[T]$ choose(s)
   except when empty: ...
```

# Exception Handling

- Handling
- Propagating
- Shouldn’t happen
  - The failure exception
- Principles
  - Accurate interfaces
  - Avoid useless code

# Iterators

- For all x in C do S

# Iterators

- For all x in C do S
  - Destroy the collection?
  - Complicate the abstraction?

# Visit to CMU

- Bill Wulf and Mary Shaw, Alphard
- Generators

# Iterators

```
sum: int := 0
for e: int in Set[int]$members(s) do
    sum := sum + e
    end
```

# Iterators

```
Set = cluster[T] is create, ..., members, ...

    rep = array[T]

    members = iter (x: cvt) yields (T)
      for z: T in rep$elements(x) do
        yield (z) end
```

# After CLU

- Argus and distributed computing
- Type Hierarchy

# The Landscape

- Inheritance was used for:
  - Implementation
  - Type hierarchy

# Type hierarchy

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

# The Liskov Substitution Principle (LSP)

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

- B. Liskov. Data abstraction and hierarchy. Sigplan notices, May 1988

# What Next?

- Modularity based on abstraction is the way things are done

# Challenges

- New abstraction mechanisms?
- Massively Parallel Computers
- Internet Computer
  - Storage and computation
  - Semantics, reliability, availability, security

# The Power of Abstraction

Barbara Liskov

October 2010