# История и перспективы языка Оберон

# A History and Perspective of the Programming Language Oberon

Проф. Никлаус Вирт
Prof. Niklaus Wirth

Москва. Политехнический музей

21 сентября 2005 г.

# Modula-2 and Oberon

Niklaus Wirth

Structured Programming: Pascal (1970)

Modular Programming: Modula-2 (1979)

- The Language
- Implementations

Object-oriented Programming: Oberon (1988)

- The Language
- Implementations

# Programming in 1975

- Main Frame Computers
- Use over terminals and low-speed lines
- Time-sharing
- Fortran for scientific applications
- Cobol for business applications
- IBM tried to join the two fields through
  - the 360 family of computers
  - the language PL/1
- My 1st sabbatical at Xerox 1976/77

# System Programming and HL-Languages

- Available: Pascal, C, PL/1, Lisp, …
- Introducing structures and data types
- Large systems require programming in teams
- Decomposition of system into modules
  - Precise definition of interfaces by team
  - Implementation of components by members
- Independent compilation → Separate comp.
  - Guarantees type checking over modules
  - Mesa (Xerox PARC)

# Modula-2 (1979)

- Modules
- Procedure types
- Low-level facilities
- The type CARDINAL
- What was left out
  - Concurrency
  - Exception handling
  - Storage management

# Modules

## Information hiding, encapsulation

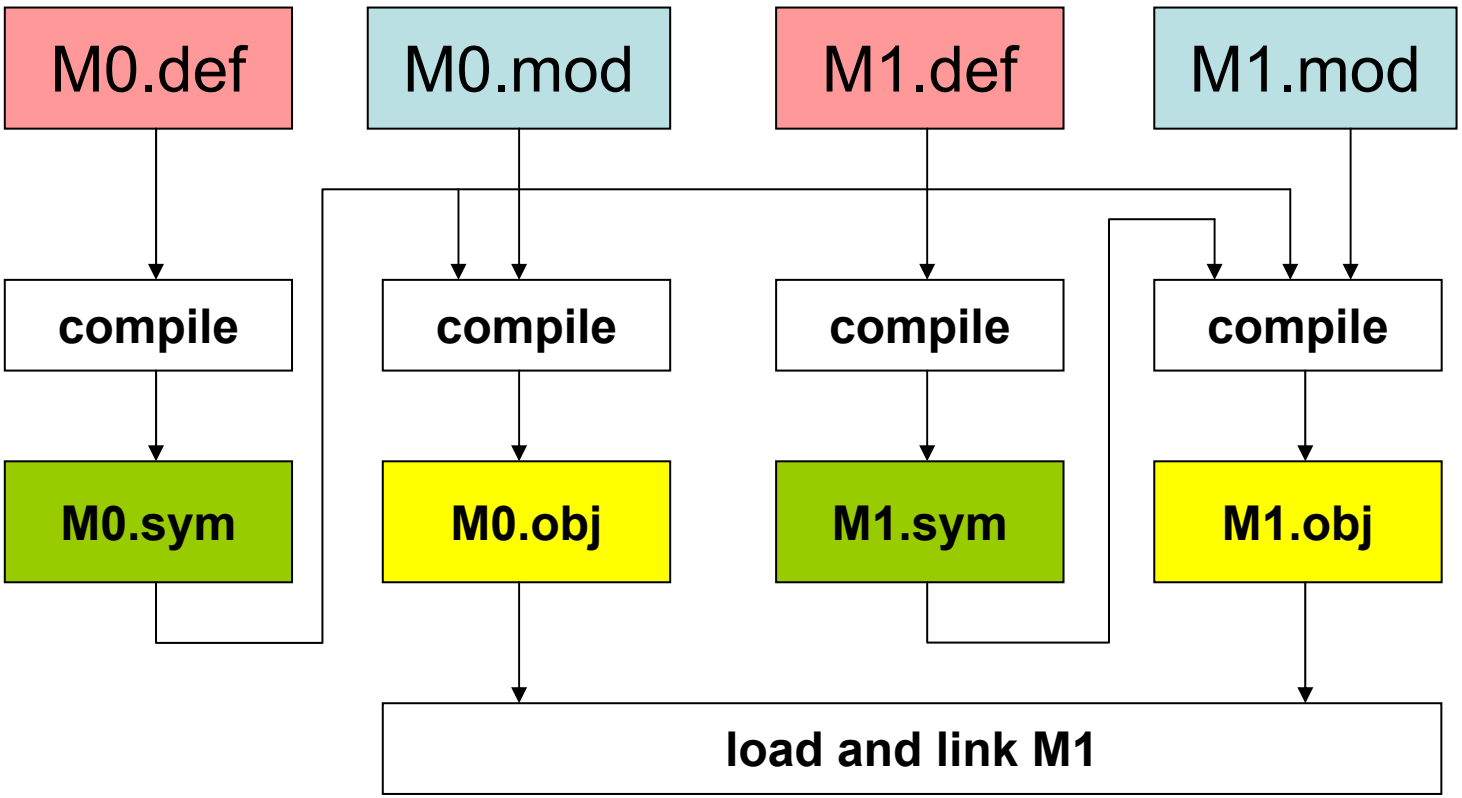
```
module M0;
   export random;
   var x: integer;
   procedure random(): integer;
   begin x := (x*a + b) mod c; return x
   end random;
begin x := 0
end M0;
```

# Modules, export and import

- Local (possibly nested) modules
  - with export and import lists in heading
- Global modules
  - Separate texts for definition of interface and implementation
  - Definition part replaces export list
  - Separate compilation. Symbol files
  - Module import: **import** M0, M1
  - Qualified import: **from** M0 **import** x, y, z

# Separate Compilation

M1 imports M0



# Procedure types

Algol, Pascal: u := integral(fct, a, b)

Modula: f := fct u := integral(f, a, b)

**var** f: **procedure** (x: real): real

- Type safety
- Makes up-calls possible
- Important for, e.g., operating systems

# Low-level Facilities

- Type transfer functions

  i := integer(x)  x := real(i)

  interpretation of real as integer depends on representation
- Type Address  (address arithmetic)
- Misuse of variant record structure

| i: integer | x: real |
|---|---|
| z: complex | w: bitset |

# The type Cardinal

- integer: $-2^{15} \leq k < 2^{15}$
- cardinal: $0 \leq k < 2^{16}$
- Larger range of addresses
- Assume x: cardinal

  while $x \geq 0$ do S; x := x-1 end

  while x > 0 do x := x-1; S end
- Different interpretation of division

  (-x)/y = -(x/y)  $q \times y + r = x,\ 0 \leq r < y$

  (-5)/2 = -2  (-5)/2 = -3  (r = +1)

# What was left out?

- Concurrency
  - Coroutine, transfer
  - Together with synchronization primitives packaged inside a utility module
- Exception handling
  - Controlled exit from procedure and/or module, deallocation of stack frames
- Storage management (garbage collection)
  - Too many “unsafe features”

# Implementation of Modula

1977 first notes about the language

1978 7-pass compiler for PDP-11 (K. van Le)
64K byte memory, 2M byte disc

1979 5-pass compiler (U. Ammann)
ported to Workstation Lilith (128K byte)

1981 single-pass compiler on Lilith (N. Wirth)

entier operating system, compiler, editor, Ethernet, servers etc. written in a single language without using any assembler

# Oberon, Language and System

- Object-oriented programming
- My 2nd sabbatical at Xerox, 1984/85
- The Cedar System
  - Mesa → Modula
  - Cedar → Oberon
- Return to the Essentials!

  Algol → Algol W → Pascal → Modula → Oberon
- entier operating system, compiler, editor, Ethernet, servers etc. written in a single language without using any assembler

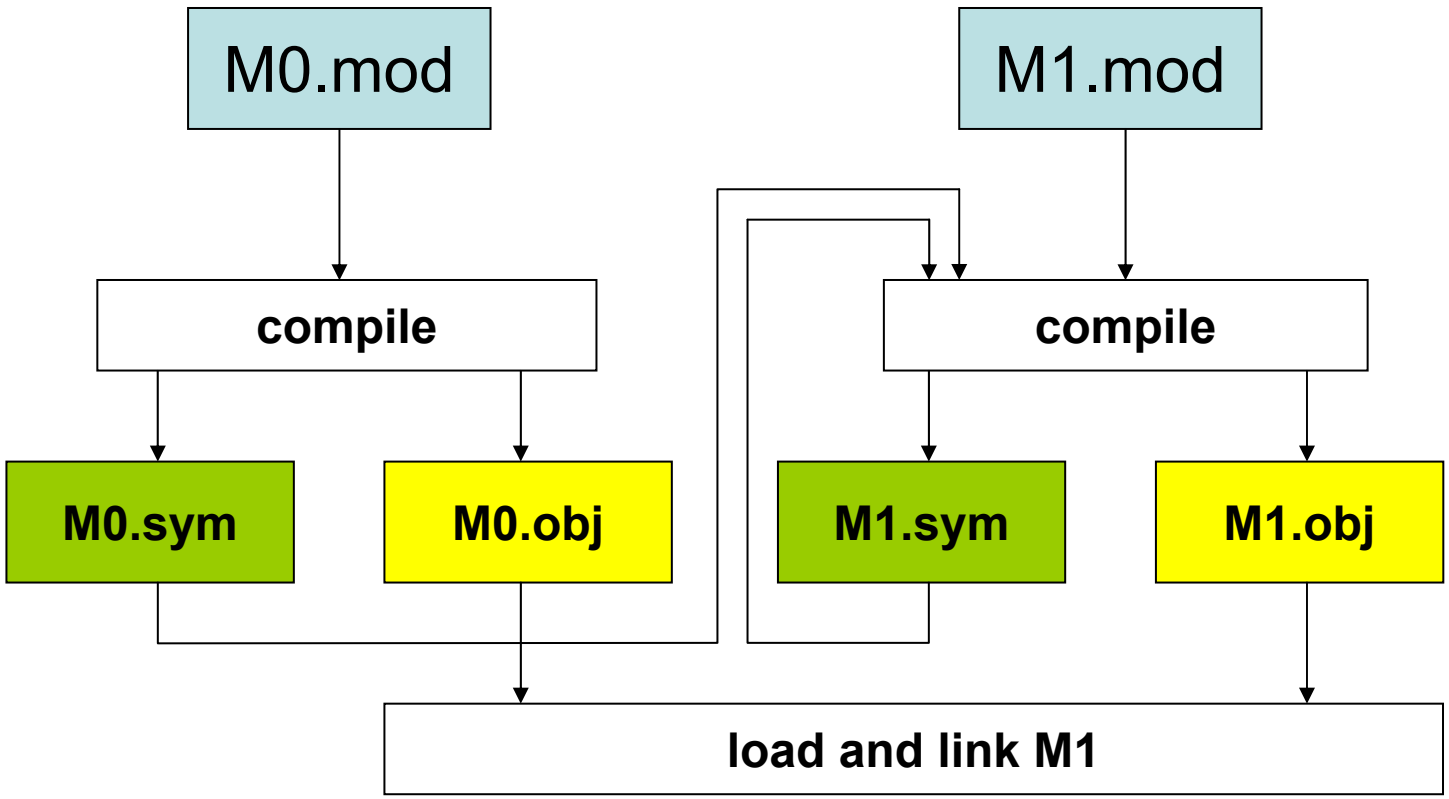
- Better system programming through use of a simpler, structured language
- More efficient system development through use of a flexible, interactive environment
- Oberon System on workstation Ceres (1989)
  - Non-hierarchical file system
  - Non-overlapping viewer system (windows)
  - Single process, single language, single user system
  - Fast compiler, flexible document editor
  - 200K bytes, compilation in 40s
  - Allows systematic teaching of the fundamentals

# The Language Oberon (1988)

- Features omitted from Modula:
  - Variant records
  - Enumeration types
  - Subrange types
  - Set types  (single type **set**)
  - Qualified import
  - Low-level facilities  (type transfer functions)
  - Merging definition and implementation texts

# Separate compilation

M1 imports M0



# Features added to Modula

- Type inclusion (5 arithmetic types)

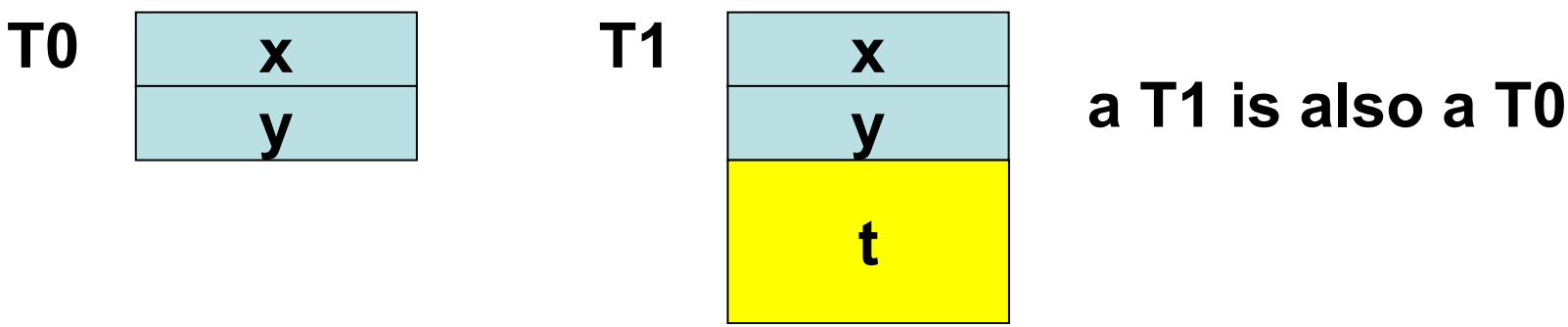
  longreal ⊇ real ⊇ longint ⊇ integer ⊇ shortint

  k: integer; x: real; x := k; k := x; k := entier(x)

- Type extension (inheritance)

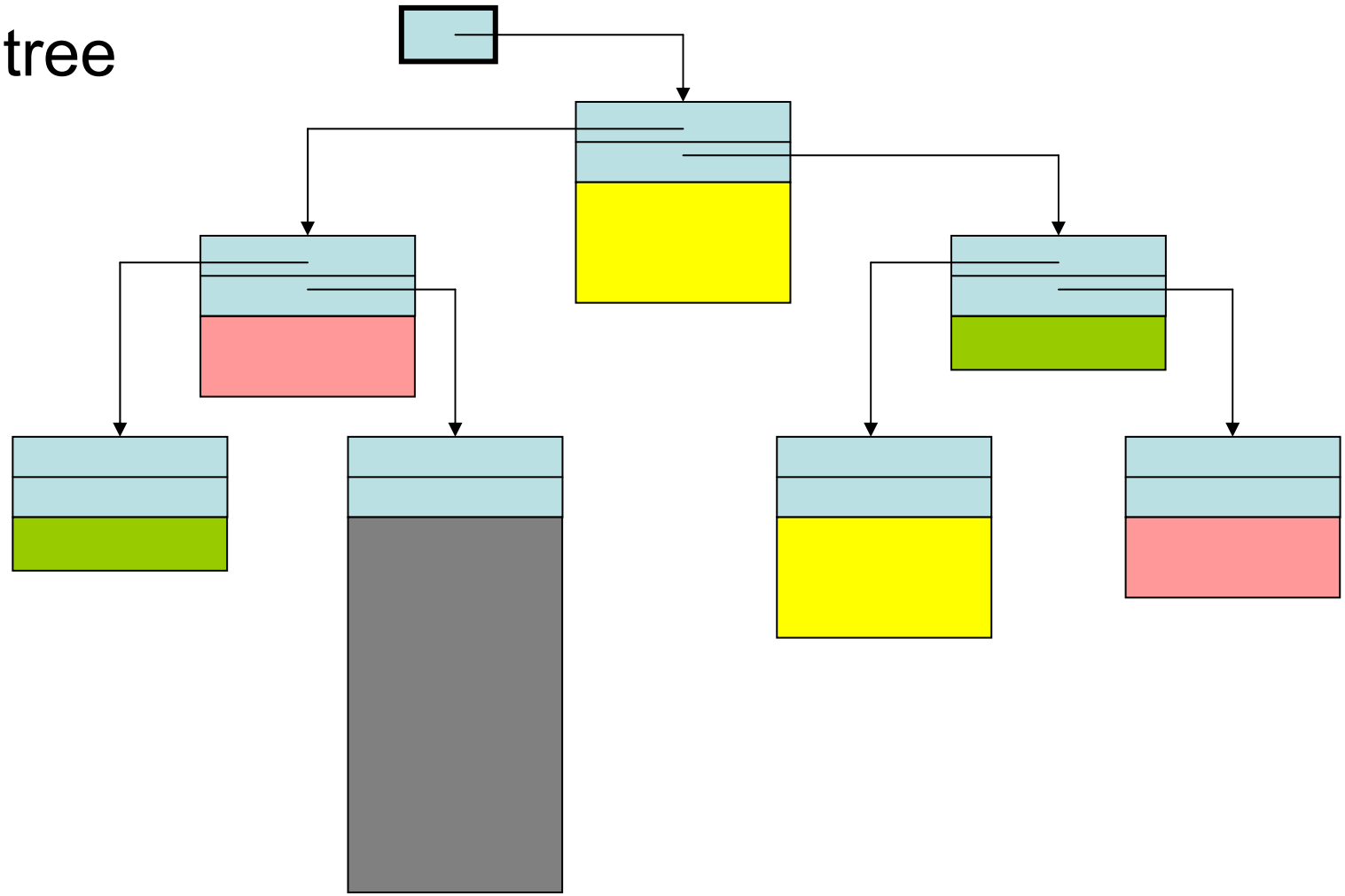
  T0 = **record** x, y: integer **end**

  T1 = **record** (T0) t: Text **end**



# Object-orientation and inhomogeneous data structures

a tree



# Methods = bound procedures

```
type figure = pointer to record
        x, y, w, h: integer;
        move: procedure (f: figure; dx, dy: integer);
        draw: procedure (f: figure; mode: integer)
    end ;
var f: figure;
… f.x := 48; …  f.move(f, 10, 10) …
```

f qualifies method move

f designates object moved

# Implementations

- First ideas in early 1985
- Development of Workstation Ceres-1
- First compiler devel. on Lilith in Modula
- Compiler completed and ported to Ceres, and Language Report published in 1988
- Implementation of Oberon System by Wirth and Gutknecht 1986-88
- Entire system programmed exclusively in Oberon, by 2 persons in less than 2 years

# Porting Oberon

- Concentrated effort to port Oberon to various platforms:
- Intel 80x86 (IBM PC)
- Motorola 680x0 (Apple Mac) M. Franz
- Sparc (Sun) J. Templ
- MIPS (Silicon Gr.) R. Crelier
- PowerPC (IBM) M. Brandis
- NS32x32 (National) N. Wirth

# Why Oberon for Teaching?

- Oberon is the “natural” descendant of Pascal
- Adopts Pascal’s syntax, power, and style
- Clearly exhibits structure of program + data
- Implies 30 years of experience in structured programming with a structured language
- Incorporates object-orientation
- Sound basis for effective implementation and clean abstractions
- We have 15 years experience with Oberon

# References

- www.Oberon.ethz.ch
- www.Oberon.ethz.ch/WirthPubl/
- www.Oberon.ethz.ch/books.html