ID Masters Module 3

Introduction to Software Engineering

prof .loe feijs

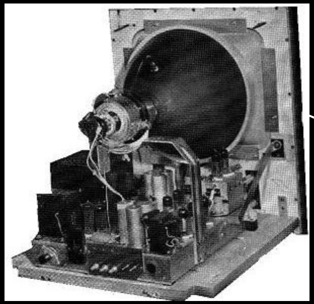
Intelligent products, systems and services

empowering people:

intelligent spaces:

THE INTELLIGENCE IS EMBODIED IN COMPLEX COMPUTER PROGRAMS

How computer programs are made



1950-1960: machine code

1960-1970: high-level language

1970-1980: structured programs

1980-1990: object orientation

1990-2000: component software

2000-2010: software agents

Machine language

```
$TEST:    PUSHF
          SETST     4
          SETST     3
          SKBIT     15
          CLRST     4
          PUSH      0
          LD        0,MANT
          SKBIT     15
          CLRST     3
          LI        0,0
          SKSTF     4
          SETBIT    0
          SKSTF     3
          SETBIT    1
          ST        0,$STAT
          PULL      0
          PULLF
          RTS
NORM:     SKNE      0,$NUL
          JMP       $H
          JMP       $I
$H:       SKNE      1,$NUL
          JMP       $J
          JMP       $I
$J:       LI        2,0
          RTS
$I:       PUSHF     ETC
```

Concepts:

- memory
- arithmetic
- logic
- stack
- jumps

average productivity:2.5 lines per hour

-- Anonymous.

Like the old joke, "He has experience, he wrote over 350 kloc personally... Then he discovered loops."

High-level language

```
     SUBROUTINE TOASC (N,M,NADE)
C ***************************
C TOASC CONVERTS INTEGER N   *
C OF MAX M DIGITS TO ASCII   *
C RESULT IN ARRAY NADE       *
C****************************
     DIMENSION NADE(M)
     L=N
     I=0
     DO 10 J=1,M
     K=M-J
     I=N/(!)**K)
     NADE(J)=I+48
     N=N-(I*(10**K))
10   CONTINUE
     N=L
     RETURN
     END
```

Concepts:

- types
- variables
- If-then-else
- for, while, repeat, etc.
- procedures, parameters

average productivity: 2.5 lines per hour

Structured programs

```
procedure straightselection;
      var i,j,k: index;
          x : item;
begin for i := 1 to n-1 do
          begin k := i;
                x := a[i];
                for j := i+1 to n do
                    if a[j].key < x.key then
                       begin  k := j;
                              x := a[j]
                       end;
                a[k] := a[i];
                a[i] := x;
     end
end
```

Concepts:

- indentation
- data records
- nested scopes
- elimation of goto
- recursive procedures
- axiomatic theory

average productivity: 2.5 lines per hour

Prof. Edsger Dijkstra
1930-2002

Object-orientation

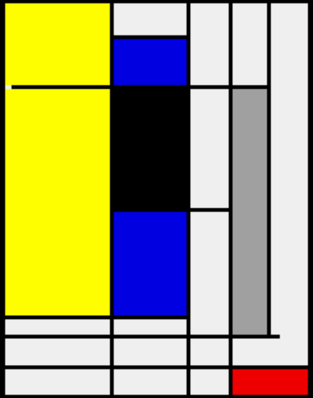
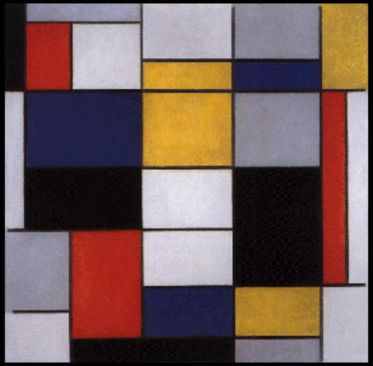
Concepts:

- encapsulation
- inheritance
- polymorphism

average productivity: 2.5 lines per hour

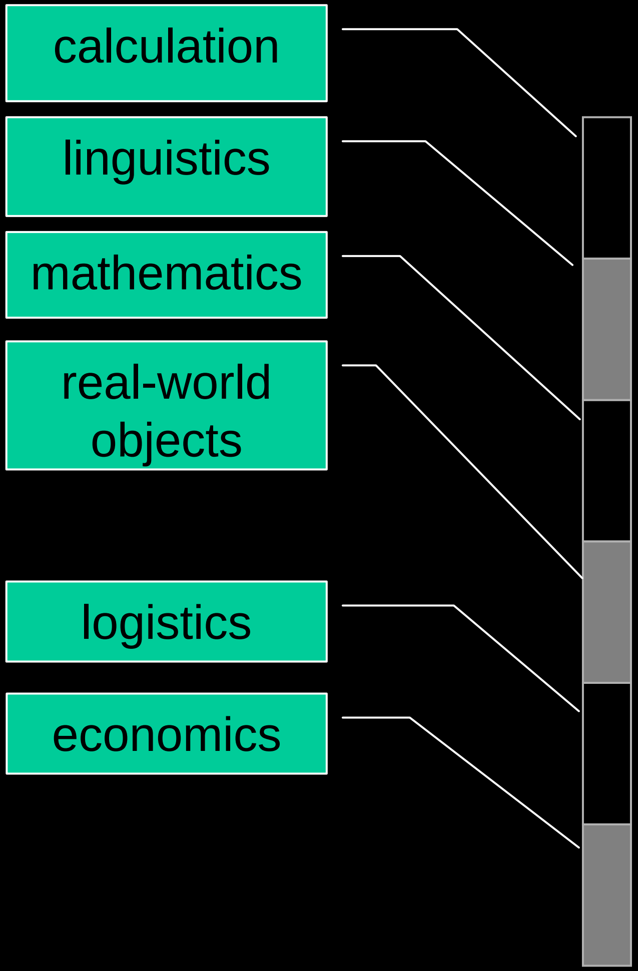
```
procedure Painting.pntA;
var
  i : integer;
begin
  self.color := LightGrey;
  self.MaxCell := 1 + random(25);
  for i := 0 to self.MaxCell do begin
      self.Cells[i] := mkCell(self.mkKernelA([]));
  end; {for}
  Delay(100);
end;

procedure Painting.pntB;
var
  i : integer;
begin
  self.color := LightGrey;
  self.MaxCell := 1 + random(50);
  for i := 0 to self.MaxCell do begin
      self.Cells[i] := mkCell(self.mkKernelB([]));
  end; {for}
  Delay(100);
end;
```

Feijs, Matematica e cultura, Venezia, aanstaande vrijdag!

Nature of the innovations



1950-1960: machine code

1960-1970: high-level language

1970-1980: structured programs

1980-1990: object orientation

developments based on language

1990-2000: component software

2000-2010: software agents

developments based on cooperation and competition between programs

Component software

Concepts:

- registration
- interface specification
- downward compatibility
- language independence

Software agents

Concepts:

- security
- authentication
- emergent behaviour
- economic and game theory

Software size

High-end TV: ± l.5 M lines

Windows OS: ± 30M lines

Drawing application: ± 4K lines

Telephone exchange: ± 6M lines

Ehhh … ?????

TESTING

COMMUNICATION

SPECIFICATION

Software specification:

Flow charts, Nassi-Sneidermann diagrams, SDL, Yourdon diagrams, Message sequence charts, Entity relationship diagrams, Class diagrams, ITU, OSI ...

UML: the Unified Modeling Language

- describing user behaviour
- describing software behaviour

Thank you for your attention,