# Paradigmen der Programmierung

Hans-Jürgen Hoffmann

Herbst 1983

Einige Anmerkungen zur Beschäftigung mit Objektorientierung, mit Interaktion und besonders mit Smalltalk.

Werbung für Smalltalk, Aufmerksamkeit wecken für das Paradigma der Objektorientierung.

(leider etwas eingestaubte Folien aus 1983, eingescannt 2015)

## Leitgedanken ("Paradigmen") der Programmierung

- prozedural — PASCAL
- funktional — LISP
- objekt-orientiert — SMALLTALK (-Sprache)

### Was heißt "objekt-orientiert"?

So etwa das einzige, was sich der Programmierer vorzustellen hat, ist, daß
- es "Aktoren" gibt und daß
- Aktoren miteinander durch Nachrichtenaustausch kommunizieren.

Wir könnten also ein SMALLTALK-Programm durch Rollenspiel vollziehen.

### Konstanz eines Programms versus Dynamik der Rechnernutzung

Σ Iteration / Rekursion

Σ Nutzer spielt als Bediener einer Datenstation die Rolle eines Aktors und tauscht Nachrichten mit den maschinell realisierten Aktoren aus.

und

Im aus Nutzer und Rechner gebildeten System können ganz nach Bedarf weitere Akteure (von der maschinell realisierten Seite selbstverständlich nur) eingerichtet und durch Zusenden einer Nachricht von irgendeinem anderen Akteur aus für die Lösung irgendeiner Aufgabe, die sie bearbeiten können, herangezogen werden.

ihren "Fähigkeiten"

## Leitgedanken der Rechnernutzung

- konventionell
- objekt-orientiert

**SMALLTALK**
- Sprache und -System

Jeder Akteur ist ein Objekt / Jedes Objekt ist ein Akteur.

Objekte kann man in Betrachtung ihrer Fähigkeiten in Objektklassen zusammenfassen.

Es gibt eine Objektklasse, die alle Objekte in Betrachtung der primitivsten Fähigkeiten, die allen gemeinsam sind, zusammenfaßt.

Zu den primitivsten Fähigkeiten gehört die Fähigkeit, ein anderes Objekt einzurichten.

Beim Einrichten eines Objekts können Fähigkeiten vererbt und weitere Fähigkeiten durch Programmierung mitgegeben werden.

Jedes Objekt weiß, entsprechend seinen Fähigkeiten, wie es auf ihm zugesandte Nachrichten zu reagieren hat (Erkennung aufgrund syntaktischer Merkmale).
Jede seiner Fähigkeiten ist durch eine Methode beschrieben / Jede Methode seiner Beschreibung realisiert eine Fähigkeit

(in der Beschreibung können nur die primitivsten Fähigkeiten angesprochen werden, wozu das Zusenden einer Nachricht an ein anderes Objekt*) gehört).

*) Auch sich selbst!

Die Beschreibung (einer Objektklasse) erfolgt in einem festen Schema. Das Schema wird zu diesem Zweck dem Bediener in geeigneter Form am Bildschirm der Datenstation visualisiert; er kann die Beschreibung entsprechend den Bedürfnissen abfassen.

| | |
|---|---|
| CLASS NAME | identifier |
| SUPERCLASS | identifier |
| INSTANCE VARIABLE NAMES | identifier * |
| CLASS VARIABLE NAMES | identifier * |
| CLASS MESSAGES & METHODS | |
| method * | |
| INSTANCE MESSAGES & METHODS | |
| method * | |

# Beispiele für das Aussehen von Nachrichten

Receiver Selector Argument

```
3 + 4                                   Tischrechner

someList size                           } Datenstruktur
  ↑ Variablenname
someList add: newComponent              } Datenstruktur
                ↑ Variablenname

[index <= limit]                        } Ablaufstruktur
        whileTrue:
                [index := index + 1.
                 collection at: index put: value]
```

Was steckt hinter dem selektierten Methodennamen whileTrue?

1. Der Empfänger [...] soll sich selbst auswerten.
2. Solange die Auswertung das Objekt true liefert, soll er das Argument [...] auswerten

# Beispiele für das Aussehen von Nachrichten

**Receiver** **Selector** **Argument**

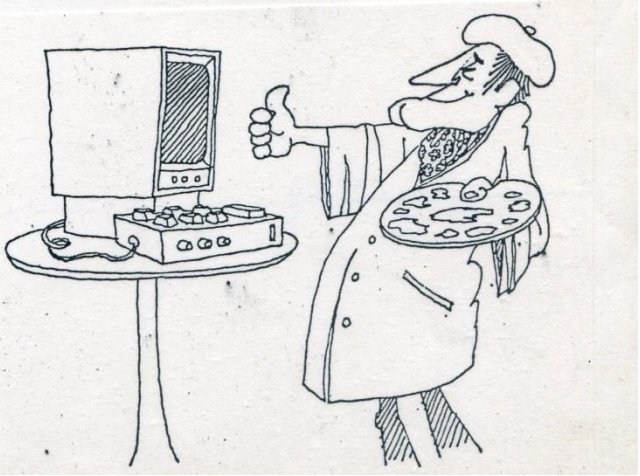
```
3 + 4                                   Tischrechner

someList size
  ↑ Variablenname                       } Datenstruktur

someList add: newComponent              }
                ↑ Variablenname

[index <= limit                         } Ablaufstruktur
  whileTrue:
    [index ← index + 1.
     collection at: index put: value]
```

SMALLTALK-Interpreter

Was steckt hinter dem selektierten Methodennamen **whileTrue**?

1. Der Empfänger **[ ... ]** soll sich selbst auswerten.
2. Solange die Auswertung das Objekt **true** liefert, soll er das Argument **[ ... ]** auswerten

SMALLTALK-Editor



konventionell

DIE SMALLTALK-Vision

1. Schritt

2. Schritt

Der Nutzer (ganz groß!)

Hexenmeister's (Systemprogrammierer's) Wolke

DIE Programmierumgebung

Baukasten!

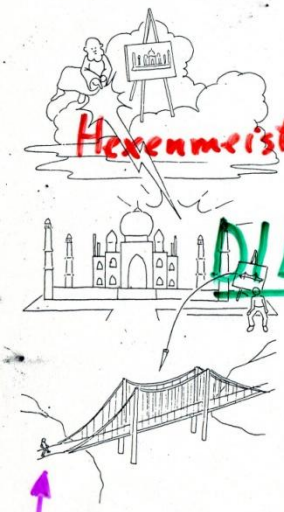


Figure 1

Der Nutzer (ganz klein!).

Figure 2

Der Nutzer verwendet den Baukasten.

Figure 3

Der Nutzer kann sich seinen eigenen Baukasten machen.

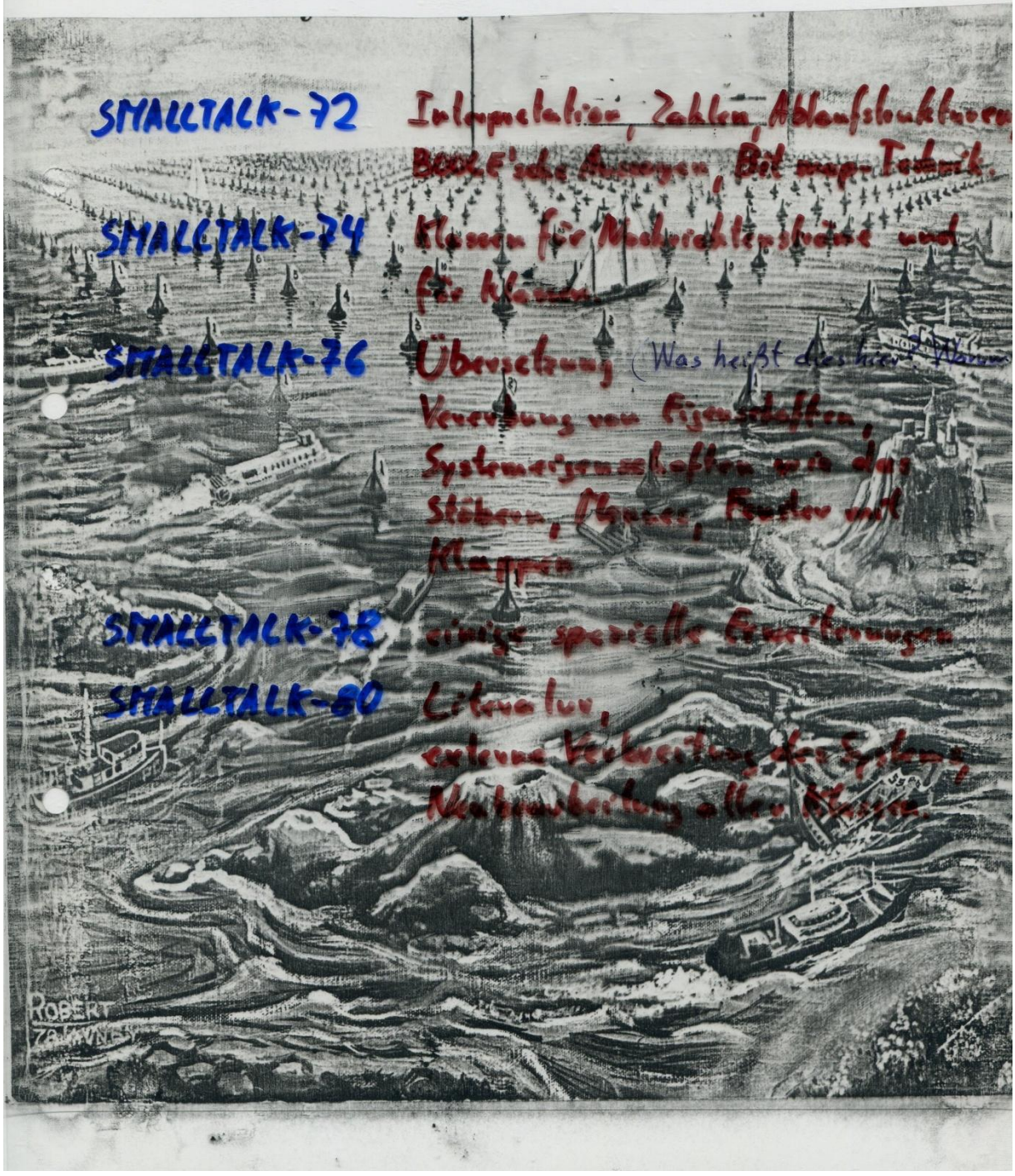
Figure 4

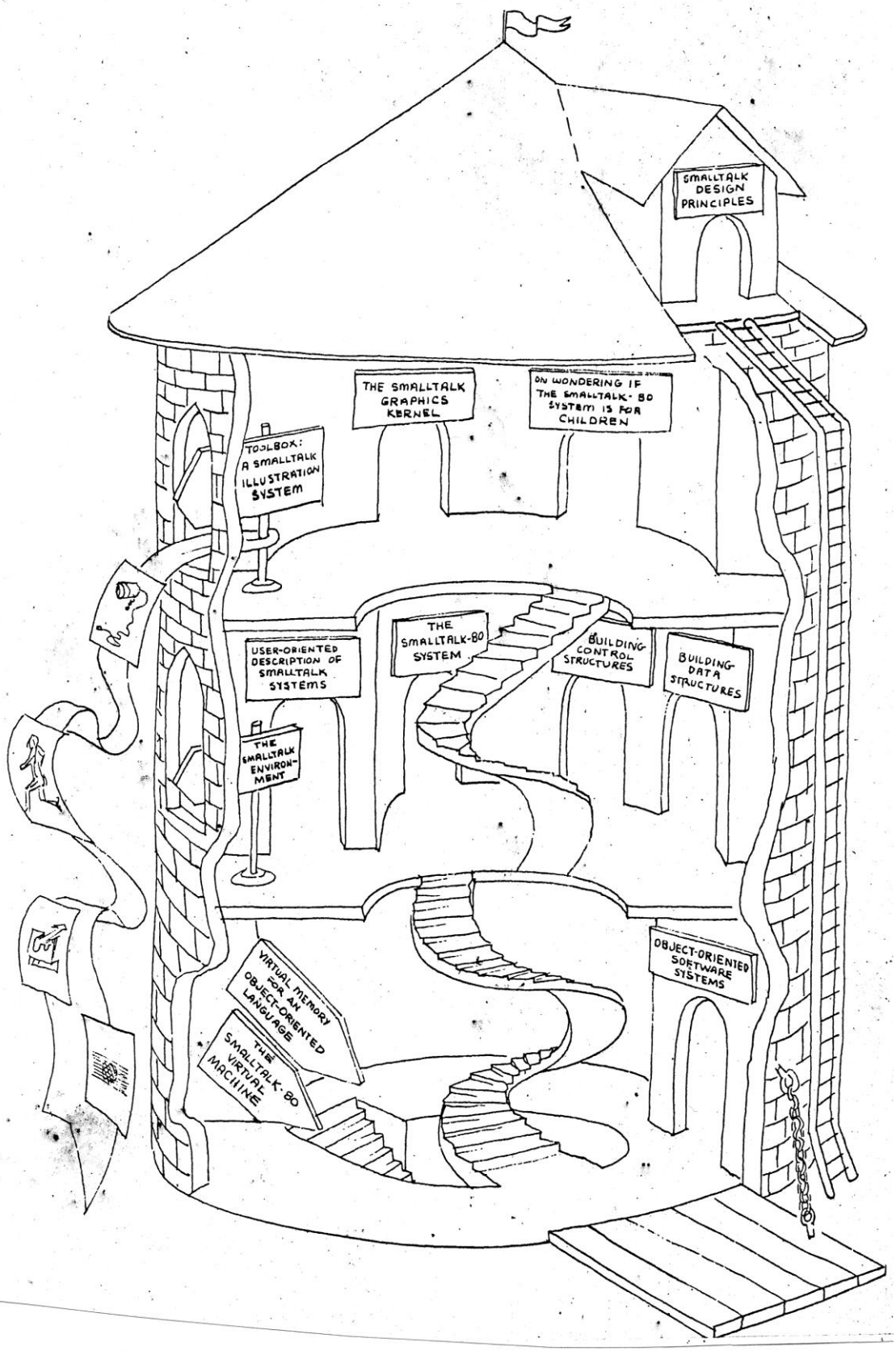
DYNAMIC PERSONAL MEDIA

Die individuelle, persönliche Wolke jedes Nutzers.

# SMALLTALK

Learning Research Group
of the XEROX PALO ALTO RESEARCH CENTE




SMALLTALK DESIGN PRINCIPLES
THE SMALLTALK GRAPHICS KERNEL
ON WONDERING IF THE SMALLTALK-80 SYSTEM IS FOR CHILDREN
TOOLBOX: A SMALLTALK ILLUSTRATION SYSTEM
USER-ORIENTED DESCRIPTION OF SMALLTALK SYSTEMS
THE SMALLTALK-80 SYSTEM
BUILDING CONTROL STRUCTURES
BUILDING DATA STRUCTURES
THE SMALLTALK ENVIRONMENT
VIRTUAL MEMORY FOR AN OBJECT-ORIENTED LANGUAGE
THE SMALLTALK-80 VIRTUAL MACHINE
OBJECT-ORIENTED SOFTWARE SYSTEMS