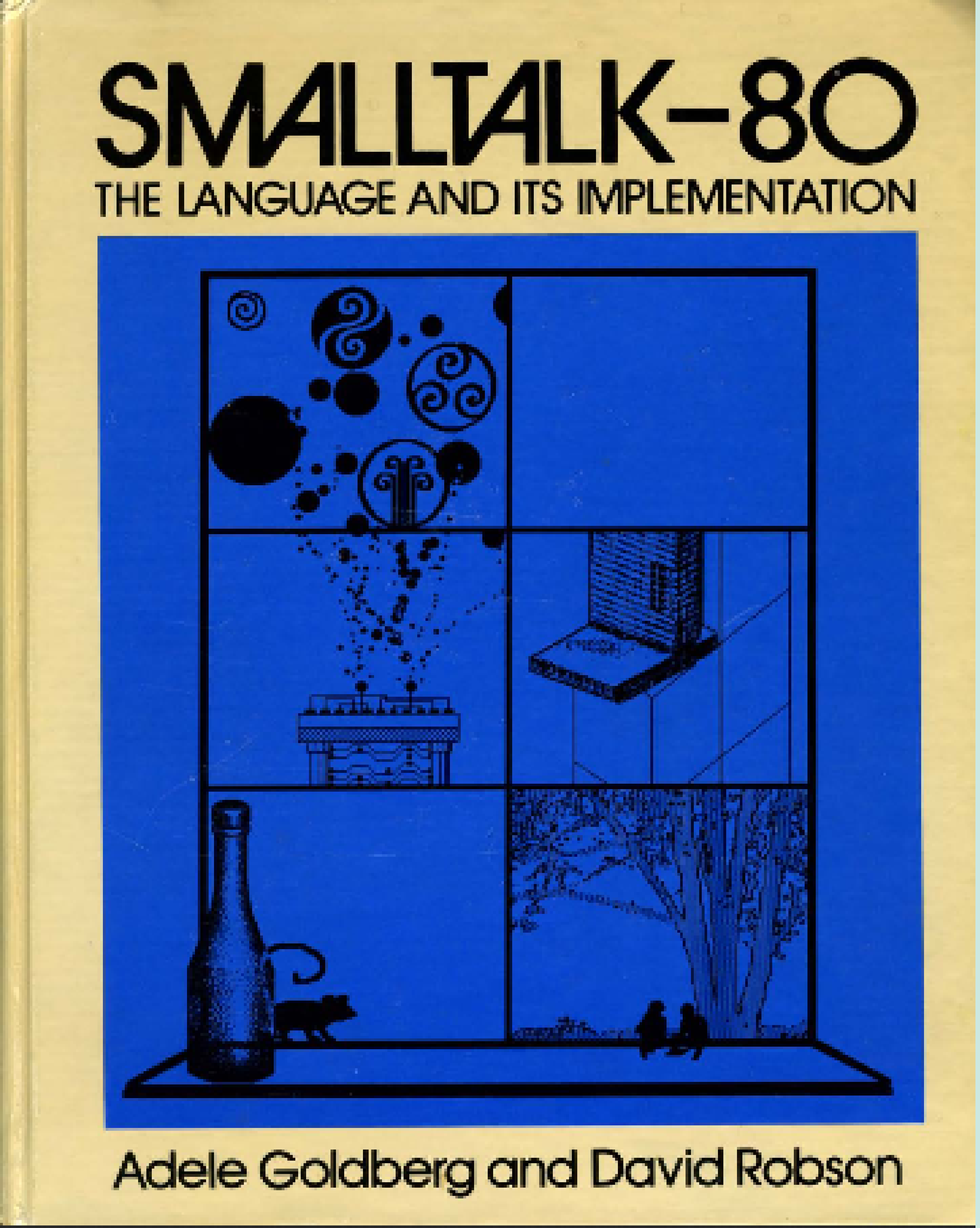

SMALLTALK-80
THE LANGUAGE AND ITS IMPLEMENTATION
Adele Goldberg and David Robson

SMALLTALK

Presented by
Team Smol
-
Simbajon,Sison,Solis,Tabanao

Presentation Outline

Smalltalk

SMALLTALK →

HISTORY OF SMALLTALK

Smalltalk ?

Smalltalk is a general purpose object oriented programming language

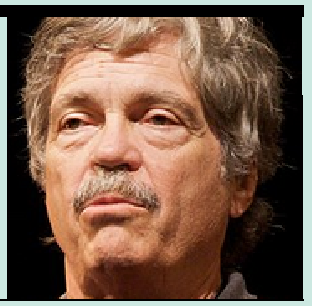
- **Smalltalk was created as the language underpinning the "new world" of computing exemplified by "human–computer symbiosis".**
- **first embodied and articulated the fundamental concepts of OOP**

NEXT

History of Smalltalk

People Behind the Creation of Smalltalk

ALAN KAY



- Designed most of the early Smalltalk versions

DAN INGALLS

- Designer and implementer of five generations of Smalltalk environments.

ADELE GOLDBERG

- Wrote most of the documentation of Smalltalk.

NEXT

History of Smalltalk

EARLY 1970S

Smalltalk was born at Xerox Palo Alto Research Center (PARC). Alan Kay developed the very first version of the language.

- Smalltalk-71
- Smalltalk-72
- Smalltalk-76

1980S

Smalltalk first went public with the release of Smalltalk-80 version 1 which was released on a limited basis to a few selected organizations including Apple, Hewlett-Packard, and UC Berkley.

LATE 1980S TO MID-1990S

Slow Commercial Growth and Open-Source Proliferation. Smalltalk environments were sold by two competing organizations (Parkplace and Digitalk).

1998

ANSI Smalltalk was ratified and represents the official version of Smalltalk on which modern implementations are based.

2000S

Smalltalk growth stalled out. However, it is enjoying a resurgence today due in no small part to the success of Smalltalk web application frameworks like Seaside and AIDA/web.

Ray Anthony Solis

Domain & Paradigm

Domain

01

It was designed and created in part for educational use, more so for Constructivist teaching, at Xerox PARC by Alan Kay during the 1970s, influenced by Sketchpad and Simula.

02

Smalltalk was so good for business use that in the 1990s, IBM chose Smalltalk as the centrepiece of their VisualAge enterprise initiative to replace COBOL

Paradigm

Everything in Smalltalk is done by sending messages to objects..

Everything in Smalltalk is an object

This makes Smalltalk as flexible and agile as other dynamically typed languages

Smalltalk is dynamically typed

Smalltalk program is able to inspect its own structure and computation at runtime.

Smalltalk is reflective

Smalltalk is a language virtual machine

that supports image persistence

The Smalltalk image allows you to save the execution state of your program at any time and resume execution later on from exactly where you left off!

Live coding and debugging is a powerful way to program and is the principal reason for Smalltalk's tremendous productivity.

Smalltalk has a live coding IDE

This makes Smalltalk a functional language, as well, except that it doesn't have immutability.

Smalltalk has lambdas

virtual computers
universally connected
through virtual networks.
There are only objects

Smalltalk as "Software Internet"

What did Smalltalk give us?

VM

Smalltalk introduced the world to the language virtual machine (or VM), which allows software to be platform-independent.

JIT

Smalltalk also pioneered JIT (just-in-time) compilation, a technique for dramatically improving the performance of bytecode software

Modern IDE

From Smalltalk came the first modern IDE (integrated development environment), which included a text editor, a system or class browser, an object or property inspector, and a debugger.

Live programming

Smalltalk was the first language tool to support "live" programming and advanced debugging techniques such as on-the-fly inspection and code changes during execution.

MVC

Smalltalk introduced MVC (Model-View-Controller) to the world. MVC is a software architectural pattern for implementing user interfaces.

TDD & XP

To a large extent, Smalltalk is responsible for giving us test-driven development (or TDD) and extreme programming (or XP), which are both very influential in today's standard agile practices.

GUI/WYSIWYG

·Smalltalk was instrumental in developing the graphical user interface (or GUI) and the "what you see is what you get" (WYSIWYG) user interface.

Refactoring browsers

Smalltalk gave us the first refactoring browser. Of course, refactoring support can be found in most IDEs today.

Duck typing

Smalltalk made "duck typing" a household word (well, if your house has a programmer in it). Duck typing is where "type checking" is deferred until runtime—when reflection capabilities are used to ensure correct behavior.

Object Database

Smalltalk pioneered the development of object databases. While they didn't make it into the mainstream, object databases have their niche markets.

Mark Kenith Simbajon

Features

- Data types
- Operators
- Data Structures
- Control Structures

Data Types

Data types available in smalltalk are the following:

- **SmallInteger**
- **Float**
- **String**
- **Boolean**

A dynamically typed reflective programming language.

```
|x y z a|
x := 4
y := 1.5
z := 'apple'
a := true
```

Data Types

In Smalltalk, there is a unary message for doing each kind of conversion. The tradition is to give these methods names beginning with "as..." such as "asFloat" or "asString".

```
X := 4   asInteger
Y := 1.5   asFloat
Z := 'apple'   asString
```

Operators

Arithmetic:

+ - / *

Logical:

and or not

Comparison:

< > <= >=

Arithmetic operators in smalltalk has equal priority

```
x :=  3 + 2 * 4
```

Ans: 20

Using parenthesis will give different result

```
x :=  3 + (2 * 4)
```

Ans: 11

Operators

Gets: :=

Gets operator is used in assigning a value

```
x := y
b := 3
y :=  'orange'
```

Returns: ^

Returns operator is used to return a value

```
name: aSymbol
      name := aSymbol.
      ^name.
```

Data Structures

- **Dictionary**

A dictionary is a special kind of collection. With a regular array, you must index it with integers. With dictionaries, you can index it with any object at all

```
y := Dictionary new
y at: 'One' put: 1
y at: 'Two' put: 2
y at: 1 put: 'One'
y at: 2 put: 'Two'
```

- **Array**

An array in Smalltalk is similar to an array in any other language, although the syntax may seem peculiar at first. Here is an example of an array with rooms for 20 elements.

```
x:= Array new: 20
```

Assigning a value to an array

```
x at: 1 put: 99
```

Data Structures

- **Sets**

A collection of unordered values

```
x:= Set new
x add: 5
x add: 7
x add: 'foo'
x remove : 5
```

- **String**

is just a Collection of Characters, where each Character is stored in the position it occupies inside the String

```
x:= 'this is a string'
```

Control Structures

Conditional Execution

In Smalltalk, there are two messages, one for when the condition is true and one for when it's false. Both messages are sent to an instance of Boolean (ie, to true or false).

```
booleanValue ifTrue: [some code].
booleanValue ifFalse: [some code].

3 < 4 ifTrue: [Transcript cr; show: 'True']
```

Control Structures

Looping

Smalltalk has no looping constructs in the language. Instead, it provides looping functionality by sending messages to BlockClosures. The most basic type of loop is one that continues to loop while some condition is true.

```
[some code] whileTrue.

count := 0.
[Transcript cr; show: count printString.
count := count + 1.
count < 10] whileTrue.
```

Control Structures

Repetition

If you want to loop from one number to another, incrementing by one each time, send to:do: passing the code block as a parameter. The block expects to receive the index number as a parameter.

```
1 to: 5 do: [ :index | Transcript cr; show: index printString]
```