

Q R S T U V W X Y Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

V W X Y Z

A JOURNEY FROM SMALLTALK TO NATIVE CODE

CLEMENT BERA  
CAMILLO BRUNI

**1 THE STANDARD:  
FROM SMALLTALK TO BYTECODE**

**2 THE ONCE HANDSOME:  
THE COG**

**3 THE GOOD:  
THE NATIVEBOOST & THE MATE**

**1.1 BYTECODE COMPILER**

**1.2 BYTECODE OPTIMIZATION**

**1.3 CONNECTING THE VM**

**SOURCE CODE**

**AST**

**IR**

**BYTE CODE**

**D E M O**

**1.1 BYTECODE COMPILER**

**1.2 BYTECODE OPTIMIZATION**

**1.3 CONNECTING THE VM**

AST

OPTIMIZE

IR

OPTIMIZE

BYTECODE

**D E M O**



**1.1 BYTECODE COMPILER**

**1.2 BYTECODE OPTIMIZATION**

**1.3 CONNECTING THE VM**

**SMALLTALK**

**BYTECODE**

**VM**

**SMALLTALK**

**P R I M I T I V E**

**VM**

**D E M O**

**SMALLTALK**

**P L U G I N**

**VM**

**D E M O**

**SMALLTALK**

**SPECIAL OBJECTS**

**VM**

**D E M O**



# **SMALLTALK**

**BYTECODE**

**PRIMITIVES**

**SPECIAL OBJECTS**

**VM**

**1 THE STANDARD:  
FROM SMALLTALK TO BYTECODE**

**2 THE ONCE HANDSOME:  
THE COG**

**3 THE GOOD:  
THE NATIVEBOOST & THE MATE**

# **SMALLTALK ENVIRONMENT**

# SMALLTALK ENVIRONMENT

VM ENVIRONMENT

↓  
BYTECODES

↓  
PRIMITIVES

↑  
SPECIAL OBJECTS

# SMALLTALK ENVIRONMENT

## VM ENVIRONMENT

BYTECODES

PRIMITIVES

SPECIAL OBJECTS

JIT

BYTECODE  
INTERPRETER

# HARDWARE ENVIRONMENT

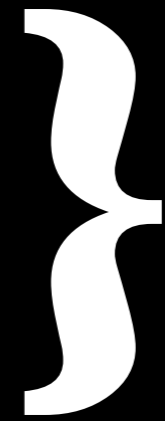
1 + 2

**BYTECODE**

push 1

push 2

send +



**BYTECODE  
INTERPRETER**

1. 



**HARDWARE ENVIRONMENT**

1 + 2

**BYTECODE**

push 1

push 2

send +

2. ○

JIT → ASM

mov RegA 1

mov RegB 2

add RegA RegB



**HARDWARE ENVIRONMENT**

# SMALLTALK ENVIRONMENT

## VM ENVIRONMENT

BYTECODES

PRIMITIVES

SPECIAL OBJECTS

JIT

BYTECODE  
INTERPRETER

## HARDWARE ENVIRONMENT



# Object basicNew

```
<primitive: 70>
```

```
objectSize := Object instanceSize.
```

```
(heap hasEnoughSpace: objectSize)  
  ifFalse: [ heap grow ].
```

```
raw := heap allocate: objectSize.
```

```
raw initializeObjectStructure.
```

```
^ raw asObject
```

# MEMORY MANAGEMENT

**JIT**

**HEAP**

**OLD SPACE**

**YOUNG SPACE**

**STACK**

# GARBAGE COLLECTION



Array new: 1'000'000 →



heap grow → heap garbageCollectYoungSpace



→ heap garbageCollectAll  
⌚.....



→ heap basicGrow



# Object basicNew

```
<primitive: 70>
```

```
objectSize := Object instanceSize.
```

```
(heap hasEnoughSpace: objectSize)  
  ifFalse: [ heap grow ].
```

```
raw := heap allocate: objectSize.
```

```
raw initializeObjectStructure.
```

```
^ raw asObject
```

**VM MAKER**

**SLANG**



**VM MAKER**

**SLANG**



**VM MAKER**

**SLANG**



**1 THE STANDARD:  
FROM SMALLTALK TO BYTECODE**

**2 THE ONCE HANDSOME:  
THE COG**

**3 THE GOOD:  
THE NATIVEBOOST & THE MATE**



# VM MAKER

SLANG SLANG SLANG SLANG SLANG SLANG SLANG SLANG  
SLANG SLANG SLANG SLANG SLANG SLANG SLANG SLANG  
SLANG SLANG SLANG SLANG SLANG SLANG SLANG SLANG  
SLANG SLANG SLANG SLANG SLANG SLANG SLANG SLANG



# SMALLTALK ENVIRONMENT



The diagram illustrates the Smalltalk environment architecture. It features a large, irregular black shape representing the environment, with several white circles of varying sizes scattered within it. At the top, the text 'SMALLTALK ENVIRONMENT' is written in a bold, serif font. Below this, on the left side, the text 'BYTECODE' is written in a bold, serif font, with a small white arrow pointing downwards towards the black shape. In the center-right of the black shape, the text 'BYTECODE INTERPRETER' is written in a bold, serif font. Below the black shape, a solid black horizontal bar spans the width of the diagram, with the text 'HARDWARE ENVIRONMENT' written in a bold, serif font below it. Two vertical white double-headed arrows connect the bottom edge of the black shape to the top edge of the hardware environment bar, one on the left and one on the right.

BYTECODE

BYTECODE  
INTERPRETER

HARDWARE ENVIRONMENT

**SMALLTALK**

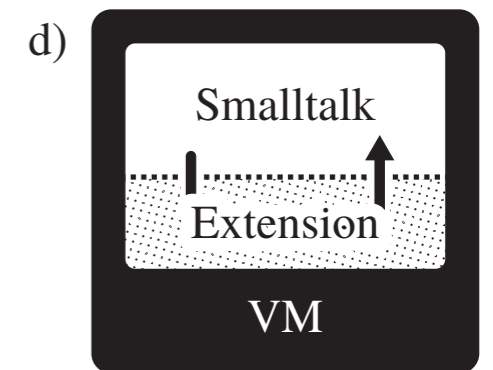
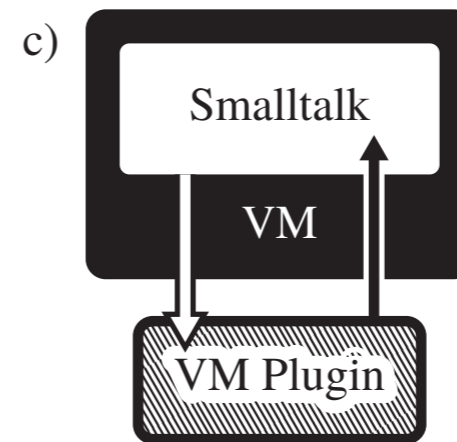
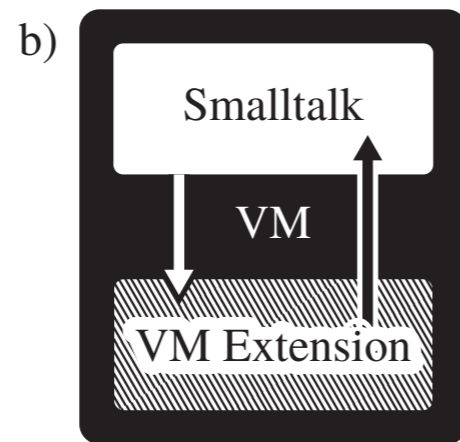
**NATIVE BOOST**

**HARDWARE**

# NATIVE BOOST

- CALL NATIVE CODE FROM THE LANGUAGE SIDE
- REPLACE PLUGINS
- REPLACE PRIMITIVES
- REPLACE JIT COMPILER

# VM EVOLUTION



# DRAWBACKS

-   **COMPLEX ARCHITECTURE**
-  **NEEDS CLEANING**
-  **NEEDS DOCUMENTATION**

# THE MATE

*Bringing finest*  *Reflection to VMs*

# THE GOAL

- &• **ST-LIKE HIGH-LEVEL VM**
- &• **METACIRCULAR**
- &• **STRONG REFLECTION**
- &• **CONTROL DOWN TO THE METAL**
- &• **MAINTAINABILITY OVER PERFORMANCE**



# THE BENEFITS

- ✧ **SMALLTALKISH DEVELOPMENT PROCESS**
  - ✧ **IMMEDIATE FEEDBACK**
  - ✧ **LIVE DEBUGGER**
  - ✧ **LIVE INSPECTION**
- ✧ **C-INDEPENDENT**
- ✧ **EASY TO UNDERSTAND**
- ✧ **EASY TO MAINTAIN**

# THE STATUS

- &• **MEMORY MANAGEMENT SIMULATOR**
- &• **HAZELNUT BOOTSTRAP IN MATE**
- &• **AST INTERPRETER**
- &• **NATIVEBOOST**
- &• **NATIVE COMPILATION INFRASTRUCTURE**
- &• **SIMPLE GC IMPLEMENTATION**

**D E M O**

**BREAK WITH OLD HABITS TO  
EXPLORE NEW POSSIBILITIES**