RTALK -
A Smalltalk 'Live' Environment
Built on the JVM
Roos Instruments, Inc.

HARDWARE AND SOFTWARE
FOR IC TEST

[Diagram of Roos Instruments equipment]
Since 1989

- Efficiency - 3 to 9x Java
- Low errors - 1/3 Java
- 500K lines of code vs 2.5M

But we now have Obsolete Platforms
- OS/2
- Digitalk STV
- Heisenbug
Rtalk is a Fork of Smalltalk

- Not sure the JVM would support a full implementation
- Commercial distribution focus
  - Code repo based not image based
- Freedom to change
- Not in the spirit ST > Language
Why look at Smalltalk

- A Language should make you think (Perlis)
- It was an interesting time for computing
- Smalltalk was the start of a paradigm
- Maybe the current approaches are just fads

The real romance is out ahead and yet to come... Don't be misled by the enormous flow of money into... poor adaptations of incomplete ideas

Alan Kay
Languages at the time of Smalltalk

- ALGOL
- SIMULA
- LISP
- SKETCHPAD
- LOGO
- Fortran etc
Dynabook

Goal to give users full flexibility for the creation and manipulation of knowledge especially for children
Smalltalk Philosophy

- Simpler but no simpler
- Support complex problems
- All about communication
Smalltalk Basics

- Everything is an Object
- Message Based not procedures
  - each Object is responsible to handle
- Class holds common behavior
  - shared message support
- Its all about communication
Impact of message sending

receiver perform:#foo with:args

anArray detect:[ :a | a name = 'foo']

2 + 3 * 5 = ?
Existing Coding Flow

- Editor
- APPLICATION CODE TEXT
- COMPILER
- println()
- EXECUTABLE
- DEBUGGER
Smalltalk Coding Flow

OBJECTS

Editor

APPLICATION
OBJECTS

DEBUGGER

INSPECTOR
'Live'? 

Live means an instance based environment

Fully Reflective
Fully Manipulative
What do we need to go 'LIVE'?

- Full Access to all Objects
- Run time code replacement
  - Code editing
- Access to all objects on heap
  - instance manipulation
- Access to stack slots
  - Debugger
- Thread Control (stop/step/run)
Smalltalk Workspace

1 + 2 3

| tmp |
tmp := 'test'.
tmp copyFrom:1 to:3 'tes'

Object inspect

1 halt

Process allInstances inspect
WORKSPACE

- Text Pane
- All text panes are REPLs
Smalltalk Inspector

 zelf
  -- Class --
classPool
sharedPools
  -- Behavior --
comment
dictionaryArray
instances
name
structure
subclasses
superClass

(MethodDictionary
rilLogMsgLocal: ==> Object>>rilLogMsgLocal:
  = ==> Object>>=
  == ==> Object>>==
yourself ==> Object>>yourself
instVarAt: ==> Object>>instVarAt:
isBoolean ==> Object>>isBoolean
riMessage:to:beep: ==> Object>>riMessage:to:beep:
printOn: ==> Object>>printOn:
isCollection ==> Object>>isCollection
isCompiledMethod ==> Object>>isCompiledMethod
withQuotes ==> Object>>withQuotes
releasePrim ==> Object>>releasePrim
jaAdminLog ==> Object>>jaAdminLog
printInspectString ==> Object>>printInspectString
isLocalVarValue ==> Object>>isLocalVarValue
asString ==> Object>>asString
fullCopy ==> Object>>fullCopy
class ==> Object>>class
class: ==> Object>>class:
isInstrument ==> Object>>isInstrument
isString ==> Object>>isString
storeOn: ==> Object>>storeOn:
hash ==> Object>>hash
isBitmap ==> Object>>isBitmap
Goal: Analyze all Objects

Heap
  - instances
  - References

Used JNI wrapping JVMTI
Smalltalk Inspector

```
self
1
2
```

User IF=false
Priority=1
ProcessScheduler>>suspendActive
ProcessScheduler>>yield
ProcessScheduler>>resume:
Semaphore>>signal
Process class>>pmInterrupt
Process class(Object)>>perform:
True(Object)>>vmInterrupt:
ProcessScheduler>>schedule
Smalltalk Editor

readInfoFrom:aFile

"<modified:sys=G9TQQ5YA,time=10/25/11 at 07:41:07 pm>  "
"<modified:sys=G9TQQ5YA,time=10/25/11 at 06:59:23 pm>  "
    "parse the info section"
    | tmp tmp1|
    [ aFile atEnd or:[aFile peek =$:] whileFalse:
        tmp := (aFile nextLine) riPair.
        tmp isEmpty ifFalse:
            attributes at:(tmp at:1) put:(tmp at:2).
        ].
    ]
• Open (inspect) a root class
• All displays are reflective
• Text Pane compile context is list selection
Private - Compile and execute the selected text. If no error, log it on the change log.

Cursor reset.

CursorManager execute changeFor:
  [self dolt: |
    self clearMouseCapture.
    CursorManager normal change.
    ^self]].
Debugger

- Stack var inspection
- Hop step jump (thread control)
- Done with MethodHandles
- JVM stacks as well
- Convert all jvm errors to rtalk halts
MH Chain for debugger

callsite

MH Test suspend

target

callsite

Match

GWT

target

GWT

Fallback
Profiling

- CallSites on JVM side collect
- represented as Rtalk Objects
- Inspector just opens objects
- Done with MethodHandles
Performance for Hanoi 25

- java prims 151 ms
- smalltalk 350 ms
- java boxed 310 ms
- RtObject 425 ms
- RtTalk 960 ms Indy
Performance for RI software

- smalltalk  95 s
- Rtalk      35.881 s
FUTURE WORK

- Use and Share
- Async Messages (Linda)
- Actor base large scale concurrency
- Performance
- UI on Browser
- Objects in cloud
- coroutines
What I want from the JVM

- Visibility and access from the app
  - Hotspot
  - Heap
  - stack (coroutines?)
- Objects everywhere
- And at xmas a PIC methodHandle
References

- Viewpoints: http://vpri.org/
- dynabook: http://tkbr.ccsp.sfu.ca/dynabook/
- L. Peter Deutsch:
  www.ifs.uni-linz.ac.at/~ecoop/cd/papers/ec89/ec890073.pdf
- Free Smalltalk books: stephane.ducasse.free.fr/FreeBooks.html
- sample code (DropBox link on jvm summit wiki)
- mlvm mailing list
  http://mail.openjdk.java.net/mailman/listinfo/mlvm-dev
- me mroos@roos.com