Pragmatic Smalltalk

David Chisnall

February 7, 2009

http://etoileos.com

Outline

The Smalltalk Family

- Smalltalk - first dynamic, object-oriented, language.
- Self - Smalltalk without classes.
- JavaScript - Self with Java syntax.

A Quick Introduction to Smalltalk

Assignment:

```
aVariable := anExpression.
```

Message Sending:

```
'Unary message'
receiver run.
'Message with one parameter'
receiver doSomethingTo: anObject.
'Message with two parameters'
receiver doSomethingWith: a and:b.
```

Smalltalk Flow Control

- No flow control other than message sending in the language.
- If statements implemented with *blocks* (closures).
- Block literals are defined with square brackets.

```
1 aBoolean ifTrue: [ obj message. ]
2          ifFalse: [ obj2 message. ].
```

`True` and `False` are subclasses of `Boolean` with different implementations.

```
1 whileTrue: body
2     ^(self value) ifTrue:[
3         body value.
4         self whileTrue: body
5     ].
```

Objective-C...

- Smalltalk-like extensions to C.
- Very easy to learn for C programmers.

...is not smalltalk...

- Has non-object types.
- Has primitive flow control.
- Is fast enough for device drivers on a 25MHz m68K.

Why do we want Smalltalk?

- Very simple language to learn.
- Very clean language to read.
- Close match to Objective-C.

The Pragmatic Smalltalk Compiler

- Initial release as part of Étoilé 0.4.
- ABI-compatible with Objective-C.
- JIT compiler based on LLVM.
  - Also does static compilation.
  - Produces .o files that can be linked with C/ObjC code.

A modular approach

Composed of several components:

LanguageKit contains a dynamic language AST and, in separate, loadable libraries:

- A set of runtime code needed for LanguageKit programs.
- A code generator based on LLVM.
- Language front-ends are loaded dynamically.

Smalltalk is a bundle loaded by LanguageKit.

edlc is the compiler driver for running / compiling Smalltalk from the terminal.

How big is it?

Framework	Component	Lines of Code
LanguageKit	AST	3608
LanguageKit	Runtime	774
LanguageKit	Code Generation	4505
LanguageKit		8887
Smalltalk	Parser	661
Smalltalk	Runtime Support	151
Smalltalk	Test Suite	304
Smalltalk		812
Everything		**9699**

Under 1000 lines of code are specific to Smalltalk.

What does this mean?

- Objective-C is a pure superset of C.
  - Calling C from Objective-C has no overhead.
  - C or Objective-C can have inline assembly (if you really need it).
- Objective-C objects and Smalltalk objects use the same structure.
- An object can have methods in Objective-C and Smalltalk.
- No virtual machine, just a small runtime library.

How does it work?

- Gets type information from selectors.
- Boxes and unboxes primitive types automatically.
- Implements SmallInt methods as C functions, compiled to bitcode and inlined by LLVM.

Selector

A selector is an abstract method name. With the GNU runtime this also includes a string representing the return type and argument types.

The Difficult Bits

Smalltalk blocks: Closures are first-class objects and must be fast.

Non-local returns: Returning from inside a block returns from the method where the block was returned.

Blocks

```
1 array map: [ :x | x log. x permuteInSomeWay ].
```

- Two objects, the block and the context.
- Both allocated on the stack (fast).
- The block object is stateless.
- The context object contains all of the locals for a given stack frame.

Retaining blocks

When a block is retained (i.e. the receiver keeps a reference to it after it returns):

1. A copy on the heap is created, pointing to the same context.
2. The class of the context is changed from `StackBlockContext` to `RetainedBlockContext`.
3. A pointer to the new block is stored on the stack, just in front of the stack context.
4. Execution proceeds until...
5. ...the scope containing the retained block exits.
6. The retained block is copied off onto the heap, just before the stack frame is popped.
7. All of the pointers to it are updated.

Non-Local Returns

```
1 - aMethod [
2 "This return is in a block, but causes this
      method to return true"
3 someCondition ifTrue: [ ^true ].
4 ^false.
5 ]
```

- New exception handling personality function written (part of LanguageKitRuntime).
- Non-local returns use the platform's unwind library, just like exceptions.
- Cleanup code for intervening stack frames run by their EH personality functions.
- Very expensive, but doesn't slow anything down when not used.

What can we do with it?

- Write applications. Mélodie uses lots of Smalltalk, first pure-Smalltalk app committed to svn in January.
- Write scripts. Corner activation and gesture app uses Smalltalk for scripting.
- Modify existing apps...

Smalltalk categories...

- Objective-C allows methods to be replaced at run time.
- We can now do this with Smalltalk.
- Smalltalk bundles are loaded from the user's home directory when the app starts...
- ...for all GNUstep applications.

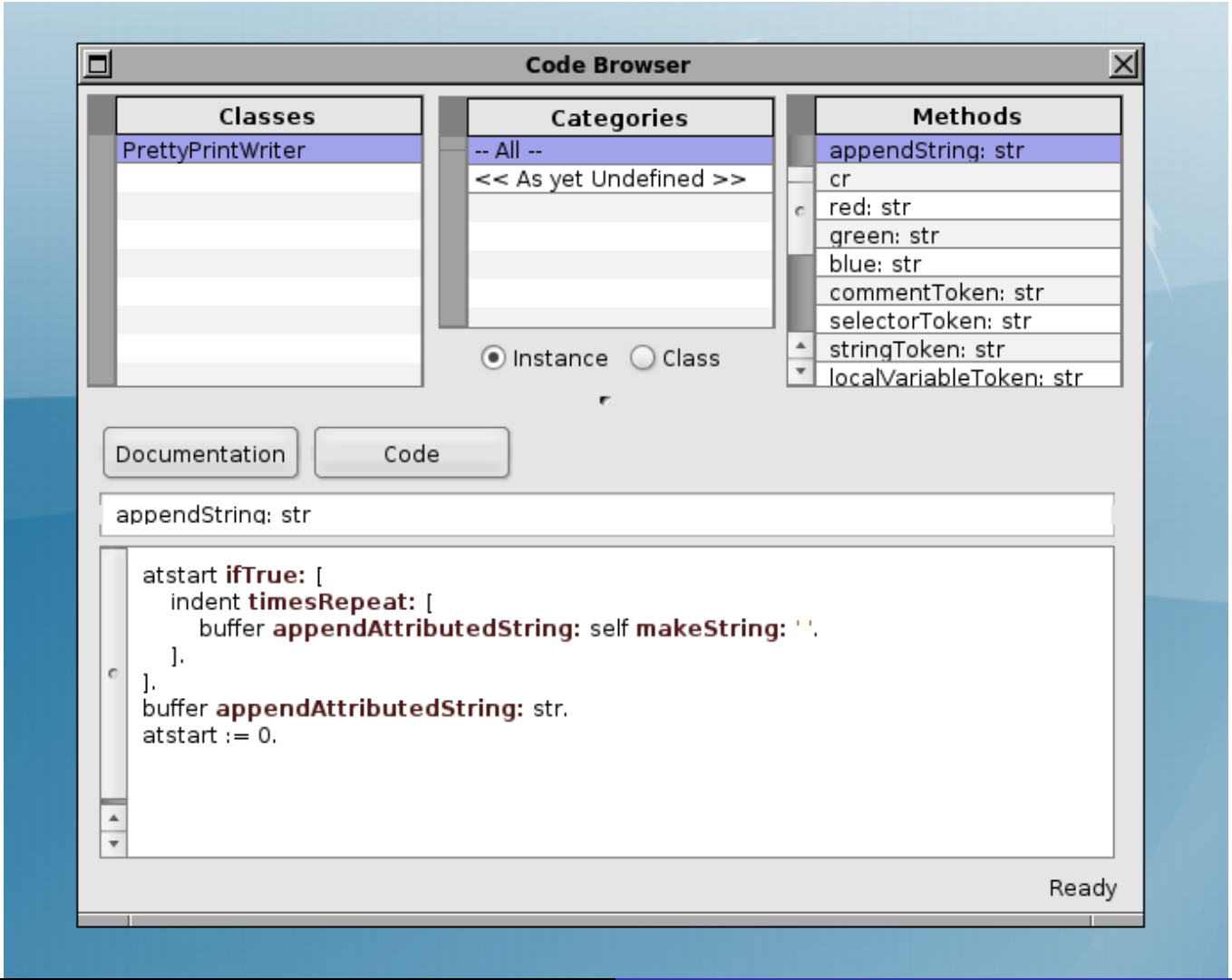
The Free Software Foundation's Third Freedom

This is no longer true:

> *The freedom to improve the program, and release your improvements (and modified versions in general) to the public, so that the whole community benefits (freedom 3).* **Access to the source code is a precondition for this.**

We don't need access to the source code to do this, in a high-level language. We can inspect classes in a code browser, see method names, and write replacements in any running application.
In a perfect Free Software system, any user can make any changes.

Code Monkey


Code Browser
Classes
PrettyPrintWriter
Categories
-- All --
<< As yet Undefined >>
Methods
appendString: str
cr
red: str
green: str
blue: str
commentToken: str
selectorToken: str
stringToken: str
localVariableToken: str
Instance
Class
Documentation
Code
appendString: str
atstart ifTrue: [
indent timesRepeat: [
buffer appendAttributedString: self makeString: ' '.
].
].
buffer appendAttributedString: str.
atstart := 0.
Ready

Code Monkey

- Syntax highlighting.
- Class browser using runtime introspection.
- Allows run time modification of loaded classes.
- Will be embedded in existing apps.

The Future

- More languages
  - EScript - new front end based on JavaScript being worked on by Truls Becken.
  - Io, maybe Self.
  - Others?
- Optimization. Some as AST transforms, some in LLVM. We're fast already, but we could be a lot faster.
- Étoilé Objective-C runtime. Written by David Chisnall, will eventually replace the GNU runtime in Étoilé.

Acknowledgements

Truls Becken: Smalltalk parser that actually worked.

David Chisnall: LanguageKit AST and code generation, first (buggy and horrible) Smalltalk parser.

Günther Noack: Test suite and bug hunting.

Nicolas Roard: Code Monkey IDE.

Eric Wasylishen: First real Étoilé apps written in Smalltalk (finding lots of bugs along the way).