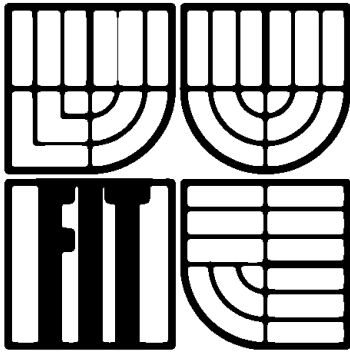
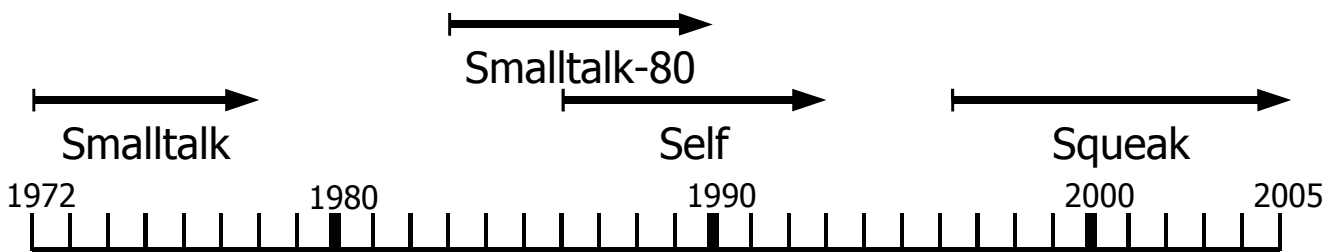


Prototype-based programming tools for Squeak Smalltalk



Author: Pavel Křivánek

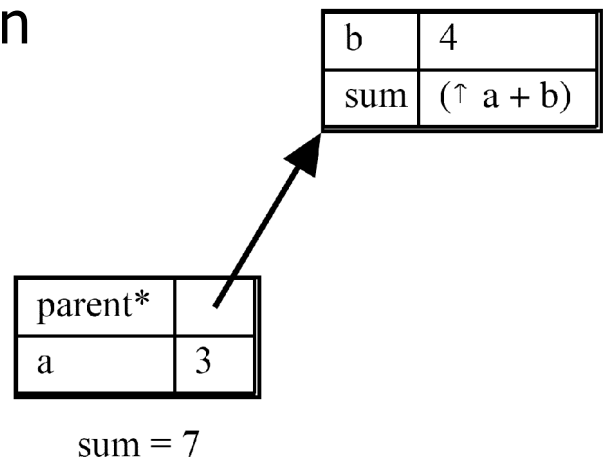
Supervisor: Janoušek Vladimír, Ing., Ph.D.



Self

- classless object orientation
- object = set of slots
- delegation
- dynamic inheritance

- class ~ trait + prototype



Self & Squeak

- MVC - Morphic (GUI)
- eToys
 - visual programming
 - class for every new scripted morph!

„Simplicity can be marvellously powerful“ - Rahul Jindal

Self benefits against Smalltalk

- simpler, well-advised, more powerful
- dynamic and multiple inheritance, mixins
- name spaces, image modularization

Negatives

- Sun neglects it
- slow Linux VM, none for Windows
- small community, multimedia, Seaside absence...

Squeak and prototypes

- several implementations
- low effective, problematic syntax...
- unusable for bigger projects

Why not Smalltalk

- class-oriented language
- pseudo-variables, literals...

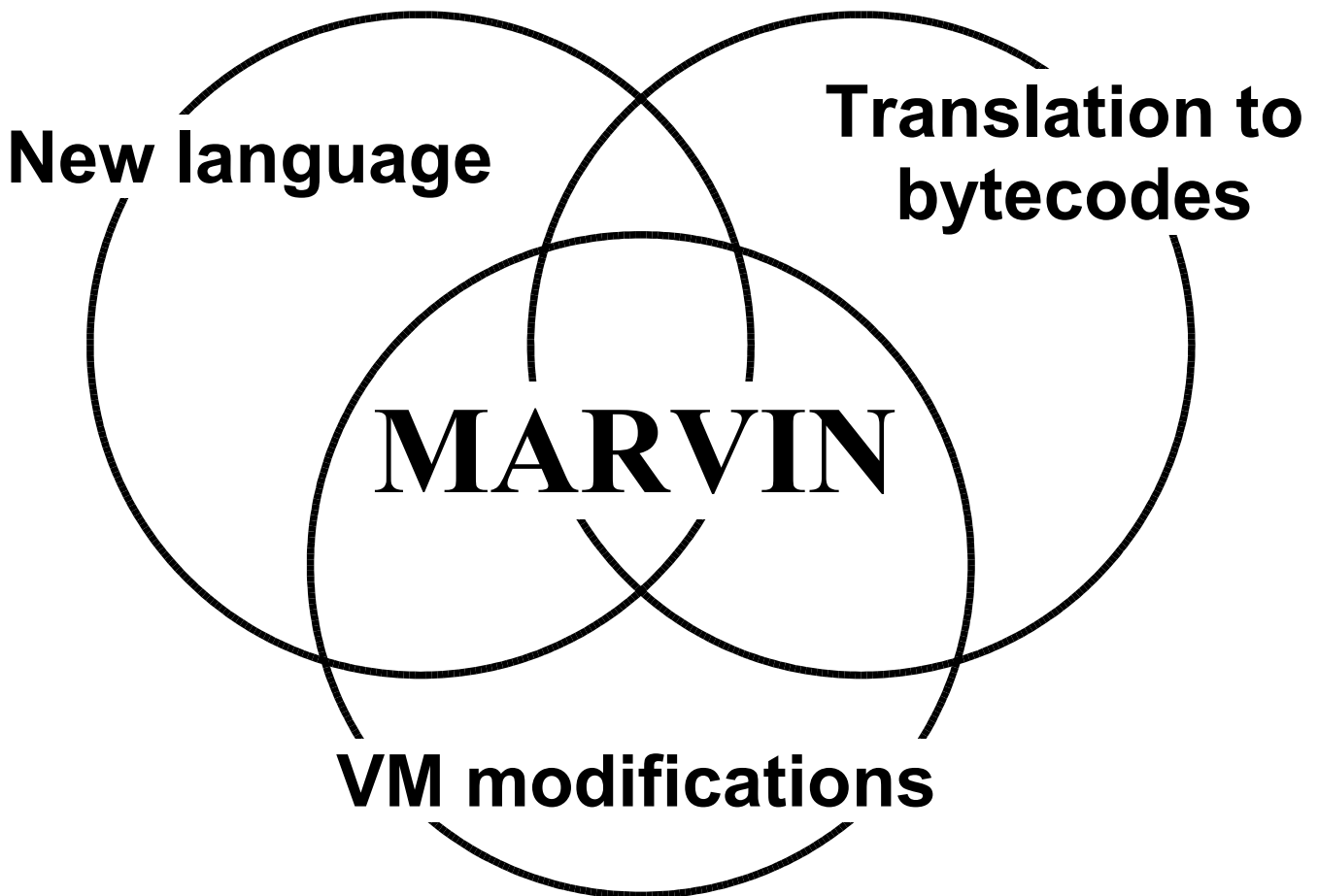
Why not Self, Slate etc.

- problematic Squeak integration
- slow interpretation

New complex approach

Smalltalk and Self combination
literals for objects and slots
Smalltalk conventions for messages
Smalltalk literals

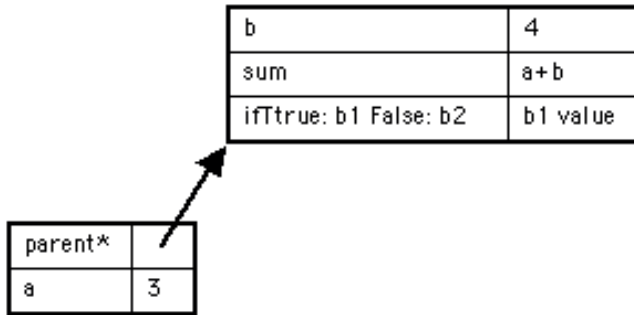
directly to the bytecodes of Squeak
Smalltalk equivalent
native Squeak blocks
local slots as temp. variables
full integration with Squeak



prototypes support
delegation support
resend support
independent part

Marvin vs. Self

- same expression capabilities
- better blocks, numbers, unicode and namespaces support
- full integration with Squeak



Self

```
( |
  parent* = ( |
    b = 4.
    sum = (a+b).
    ifTrue: b1 False: b2 = ( b1 value ) | ).
  a = 3.
| )
```

Marvin

```
( |
  parent* = [( |
    b = [4].
    sum = (^ a+b).
    ifTrue: b1 iffFalse: b2 = ( ^ b1 value ) | )].
  a = [3].
| )
```

Smalltalk

```
Object subclass: #Parent1
  instanceVariableNames: 'b'
  classVariableNames: ''
  poolDictionaries: ''
  category: 'Demo'
```

```
!Parent1 methodsFor: 'as yet unclassified' stamp: 'pk 6/16/2005 16:16!'
b
```

```
^ b! !
```

```
!Parent1 methodsFor: 'as yet unclassified' stamp: 'pk 6/16/2005 16:17!'
ifTrue: b1 iffFalse: b2
```

```
^ b1 value! !
```

```
!Parent1 methodsFor: 'as yet unclassified' stamp: 'pk 6/16/2005 16:19!'
initialize
```

```
b := 4! !
```

```
!Parent1 methodsFor: 'as yet unclassified' stamp: 'pk 6/16/2005 16:17!'
sum
```

```
^ self a + self b! !
```

```
Parent1 subclass: #Object1
  instanceVariableNames: 'a'
  classVariableNames: ''
  poolDictionaries: ''
  category: 'Demo'
```

```
!Object1 commentStamp: 'pk 6/16/2005 16:20' prior: 0!
Object1 new!
```

```
!Object1 methodsFor: 'as yet unclassified' stamp: 'pk 6/16/2005 16:18!'
a
```

```
^ a! !
```

```
!Object1 methodsFor: 'as yet unclassified' stamp: 'pk 6/16/2005 16:19!'
initialize
```

```
a := 3! !
```

**Squeak objects and classes can be used
and vice versa**

Next progress

- outliner, debugger and next development tools
- optimizations
- image prototypization
- 3D multi-user development environment based on Croquet