Self Introduction

Ellen.Van.Paesschen@vub.ac.be

Programming Technology Lab - Vrije Universiteit Brussel
Schedule

- Introduction to prototypes
- Important features of PBLs
- History of Self
- Self’s basics
- The power of Self
- Efficiency
- References
Introduction To Prototypes

Prototype theory is founded in the middle of the twentieth century as a response to the ancient class theory.

In 1986 prototypes are introduced into the OO community by Lieberman in "Using Prototypical Objects to Implement Shared Behavior in OO Systems".

“A prototype represents the default behavior for a concept”

“New objects can reuse (part of) the prototype’s knowledge by saying how the new object differs from the prototype”

PBLs are class-less OO languages.

Most PBLs are dynamically typed (only Omega isn’t).

Self, Kevo, Agora, NewtonScript, …
Schedule

- Introduction to prototypes
- Important features of PBLs
  - History of Self
  - Self’s basics
  - The power of Self
- Efficiency
- References
Important Features of PBLs (1/3)

Three ways to create new objects

Cloning a prototype

Creation ex-nihilo

Extension
Important Features of PBLs (2/3)

Delegation or object-centered inheritance

**LOGO example**

- Executed in context of original receiver
- Delegates to a prototypical Pen at (50,200)
- Delegates to a Pen at (100,200)
- Delegates to a prototypical Turtle at (25,200) with heading 90

*draw* searching...

*forward* searching...

*heading* 25
**Important Features of PBLs**

Parent sharing

- A Pen at (50, 200)
- A Pen at (100, 200)
- A Turtle at (50, 200) with heading 90

```
*draw
self
50 200
x y

*forward
*draw
90
heading

y: 100
```
Introduction to prototypes

Important features of PBLs

**History of Self**

Self’s basics

The power of Self

Efficiency

References
History of Self

1986
Initial implementation @Stanford

1987
Research vehicle

1990
1991
1992
1993

1st Public release Stanford

1995
Self 4.0

today
Self 4.2.1

Designed by Ungar&Smith @ XEROX
“Self: The Power of Simplicity” @OOPSLA’87

Moved to Sun Microsystems
More public releases

Researched@Sun

Dynamic Languages Day
Schedule

- Introduction to prototypes
- Important features of PBLs
- History of Self
- **Self’s basics**
- The power of Self
- Efficiency
- References
Self's Basics

Visual exploratory programming environment
Self's Basics

Language features (1/2)
- Syntax and semantics resemble those of Smalltalk
  - Blocks
- Everything is an object
  - Manipulated via message sending
- Objects consist of slots
- Slots contain data or methods
  - Assignable and constant data slots
Language features (2/2)

- Default creation is cloning prototype via `copy`
- Creation ex-nihilo via slot definition, e.g. `(| x . y <- 5 |)`
- Delegation for behavioral inheritance
  - Through parent slots, e.g. `parent1*`
- Extension (concatenation) for state inheritance
  - `copy-down`
Self’s Basics

Traits objects gather shared reusable behavior

Traits Pen

Delegation via parent* pointer

Con

vention

prototypical Pen

x

y

50

200

a Pen

x

y

50

200

a Pen

x

y

50

200

Dynamic Languages Day
Self’s Basics

Maps *transparently* group objects

**Map for Pen**

<table>
<thead>
<tr>
<th>x</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>offset</td>
</tr>
</tbody>
</table>

prototypical Pen

50 200

a Pen

100 200

a Pen

20 30

**Clone family**
Dynamic delegation

- Traits Turtle
  - draw
  - forward
  - backward

- becomeDashed

- Traits DashedTurtle
  - draw

- a Turtle at (50,200) with heading 90

- x
- y
- 90
- heading
Multiple inheritance

Traits MovingTurtle
forward backward

Traits DrawingTurtle
draw

Dynamic Languages Day

Self's Basics

a Turtle at (50,200) with heading 90
Schedule

- Introduction to prototypes
- Important features of PBLs
- History of Self
- Self’s basics
- The power of Self
- Efficiency
- References
The Power of Self

Reflective meta-programming with mirrors

Annotations
Consulting size, parents, ...
Changing structure
Testing
...

addSlots: (z <= 0)
The Power of Self

Example 1: Method overwriting

```
Traits Pen

draw = (body body)

asMirror

a mirror on Traits Pen

at: 'draw' PutContents: (reflect: (newBody))
```

```
draw = (newBody)
```

```
a mirror on
```

```
Traits Pen
```

```
draw = (body body)
```

```
50  200
x   y
```

```
a Pen
```

```
50  10
x   y
```

```
a Pen
```

```
1   3
x   y
```

```
a Pen
```
Example 2: Assignment shadowing (1/3)

Setters are automatically provided per assignable slot via a hidden data slot containing the assignment primitive.
The Power of Self

Example 2: Assignment shadowing (2/3)

How to override this x: ?

x: = (←)
y: = (←)

Create a new object
Add a parent link to original Pen
Change all references from original Pen to new object
Add new method possibly with super call

x:5

x: = (    )
y: = (    )

➔

x: = (    )
y: = (    )

➔

x: = (    )
y: = (    )

➔

How to override this x:
The Power of Self

Example 2: Assignment shadowing (3/3)

Six statements

```
shadow: selector = (  
  |new . me|  
  me: (reflect: self).

Create a new object
new: (reflect: ( | |)).

Add new method possibly with super call
new at:(selector,'::') PutContents:  
(('|:arg| resend.',selector, ': arg. \'new code\'.')) parseObjectBody).

Change all references from original Pen to new wrapper a new object
(browse referencesOf: self)do:[  
  |:ref| (ref isFake) ifFalse:[(ref mirror)at:(ref storedName PutContents:new]]

Add a parent link to original Pen
new at:(selector,'assignmentParent') PutContents: (reflect:self).  
(new at:(selector,'assignmentParent')) isParent: true.)
```
Schedule

- Introduction to prototypes
- Important features of PBLs
- History of Self
- Self’s character
- The power of Self
- Efficiency
- References
Efficiency

Is Self slow?

Significant optimisations of the Self compiler by Chambers and Hölzle @ Sun

- Branch prediction
  - Optimisation of the fetch-decode-execute processor cycle for branch instructions

- Inline caching
  - Traits containing the intended behavior are “guessed” during method lookup

≈ 2 * speed of optimised C
Schedule

- Introduction to prototypes
- Important features of PBLs
- History of Self
- Self’s character
- The power of Self
- Efficiency
- References
References

- Self online tutorial - Wolzcko, De Corte: [http://research.sun.com/research/self/release_4.0/Self-4.0/Tutorial/index.html](http://research.sun.com/research/self/release_4.0/Self-4.0/Tutorial/index.html)
- Self folder /manuals
References

- Programming as an Experience: The Inspiration for Self (1995) - Smith, Ungar
- Parents are Shared Parts: Inheritance and Encapsulation in Self (1991) - Chambers, Ungar, Chang, Hölzle
- Organizing Programs Without Classes (1991) - Ungar, Chambers, Chang, Hölzle
Questions
Figure 2 How SELF programs are compiled