Atomic Install in Newspeak
Gilad Bracha & Ryan Macnak

Joint work with the Newspeak team at Cadence
Newspeak Overview

- Purely object-oriented, class-based, dynamically typed in the tradition of Smalltalk
- Like Smalltalk, strong emphasis on live programming (reflectivity)
- Unlike Smalltalk, strong support for modularity, security, interoperability
Reflectivity

- Programs can be modified live
- Methods can be added, removed, modified
- Fields can be added or removed
- Class hierarchy can change
- All instances immediately adapt to the new definition
Why Atomic Install?

- In traditional Smalltalk
  - Each change applies instantly
  - Changes are applied sequentially
  - Intermediate states may be inconsistent!
  - Ordering of changes matters
  - Each step needs error-handling
  - Performance
Why Atomic Install?

- We want a set of changes to be applied atomically, as a transaction.
- Old fashioned edit-compile cycle does this, but lacks liveness.
- Atomic install allows consistent modifications to running programs.
Use Cases

- Live IDE - modifiable within itself
- Live IDE w/integrated Source Control
- Objects as Software Service
Objects as Software Services

- Maintain software and data on server
- Provide backup, audit trail, sharing, software distribution & maintenance
- Software distributed and updated when client syncs with server
- Client can run offline using locally cached software and data
- Sync does not imply application restart
As Opposed To ...
Tell me what code is changing and how, and I will make it so.
Inputs to Atomic Install

In a class-based system: tell me what classes are changed, added or removed and how and I will make it so
Inputs to Atomic Install

In Newspeak, we have mixins, so

Tell me what mixins are changed, added or removed and how and I will make it so
Mixins

class Point3D extends Point2D {

    ...The class body is the “mixin”

}

Imagine this body can be re-used attaching it to different superclasses. A mixin is such a reusable class body.
Mixins

mixinPoint3D(N, Superclass) {
    class N extends Superclass {
        ...The class body
    }
}

One view of mixins is as functions from superclasses to subclasses
Mixins in Newspeak

class Point3D x: a y: b z: c = Point2D x: a y: b {

...The class body is the “mixin”

}

In Newspeak, all names are late bound - even superclass names. So we cannot statically tie class bodies to superclasses. Hence classes are compiled as mixins.
A Simple World of Mixins and Classes
Atomic Install Process

Concrete input is a list of new or revised mixin descriptions

Existing ones will be replaced, new ones added

For each altered mixin, all its applications, their subclasses and instances will be modified as needed
Modify M1 & M2, Add M5, No Schema Changes
M1, M2, C0, C1, C2 are Impacted
Modify M1 & M2, Add M5, Change Schema of M2
Impacted: M1, C0, C1, C2

Instances of C0, C2
Modify M1 & M2, Add M5, Change Schema of M1
Impacted: M1, C0, C1, C2
Instances of C0, C1
How to do Atomic Install?

- New mixins are presumed top level and added, including their nested mixins.
- Existing mixins are added to a list of objects to be changed, as are their applications and all subclasses.
- If a class’ shape has changed, all its instances are added to the list of objects to be changed.
How do we convert old objects to new objects?

We use become:
A key primitive in Smalltalk like systems

Swaps the identity of two objects

Allows us to replace old objects with new ones throughout the running process, supporting shape change
become:

- Easy if you use an object table
- Nowadays, sweeps through memory
- A lazy approach would be better
We use bulk one-way become:

Given two object arrays $a$ and $b$, both of size $n$, all references to $a[i]$ are changed to refer to $b[i]$, for $i = 1 \ldots n$
Impacted: M1, C0, C1, C2

Instances of C0, C2

M3 → C3

C1' → M1' → C0'

M2' → C2' → C4

M4
What’s missing

- UI for transactional updates in the IDE
- Setting values for new slots, possibly depending on old object’s contents
- Sync’ing remotely, updating existing objects
Links

http://newspeaklanguage.org


http://gbracha.blogspot.com/2009/10/atomic-install.html

http://bracha.org/objectsAsSoftwareServices.pdf


https://www.youtube.com/watch?v=_cBGtvjaLM0
Credits

Peter Ahe
Vassili Bykov
Felix Geller
Yaron Kashai
Matthias Kleine
Bill Maddox
Eliot Miranda
Bob Westergaard
Newspeak
It's double lusgood
This file is licensed under the Creative Commons Attribution ShareAlike 3.0 License. In short: you are free to share and make derivative works of the file under the conditions that you appropriately attribute it, and that you distribute it only under a license identical to this one. Official license.

The Newspeak eye used in the bullets, slide background etc. was designed by Victoria Bracha and is used by permission. Newspeak doubleplusgood logo designed by Hila Rachmian and used by permission.