Classboxes: A Minimal Module Model Supporting Local Class Extension

Alexandre Bergel, Stéphane Ducasse,
Oscar Nierstrasz, and Roel Wuyts

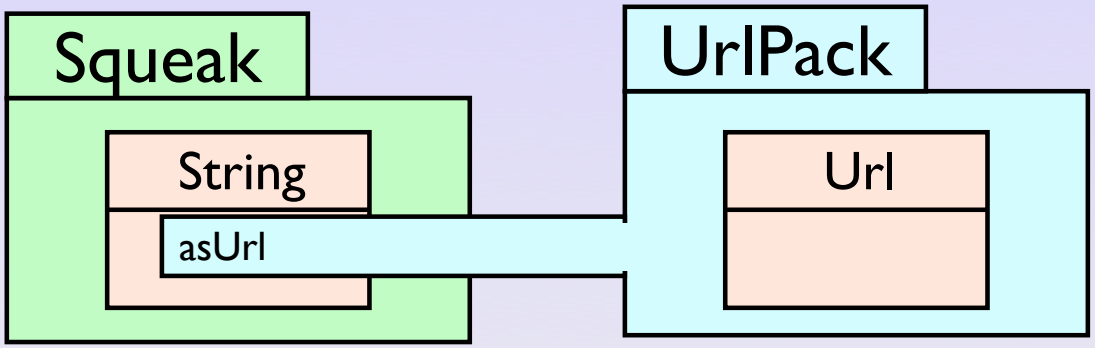
University of Bern (Switzerland)
Université Libre de Bruxelles

bergel@iam.unibe.ch

Outline

1. Class Extension
2. Supporting Unanticipated Changes
3. The Classbox Model
4. Local Rebinding
5. Implementation
6. Case Study: AWT & Swing
7. Conclusion

Class Extension

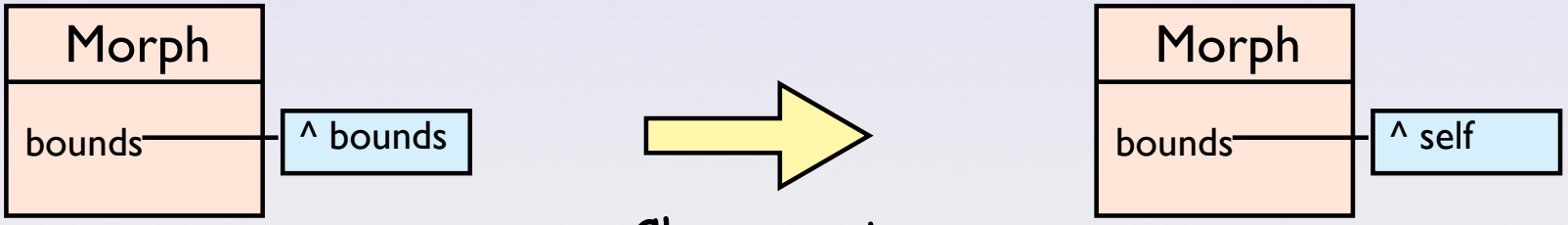


- Adding a new method or redefining one on an already existing class is a **class extension**
- Decoupling a class definition from method definitions
- Relevant: HyperJ, AspectJ, MultiJava, Smalltalk, Clos, ...

In Smalltalk-80

- Class extensions are global
- Any application can modify any class in the system
- Three consequences:
  - Conflicts may arise (e.g., two applications bring the same extensions)
  - Robustness aspect (e.g., an application may redefine a critical method)
  - Implicit dependencies

In Squeak...



Changeset
that modify
Morph>>bounds

Result: unrecoverable crash

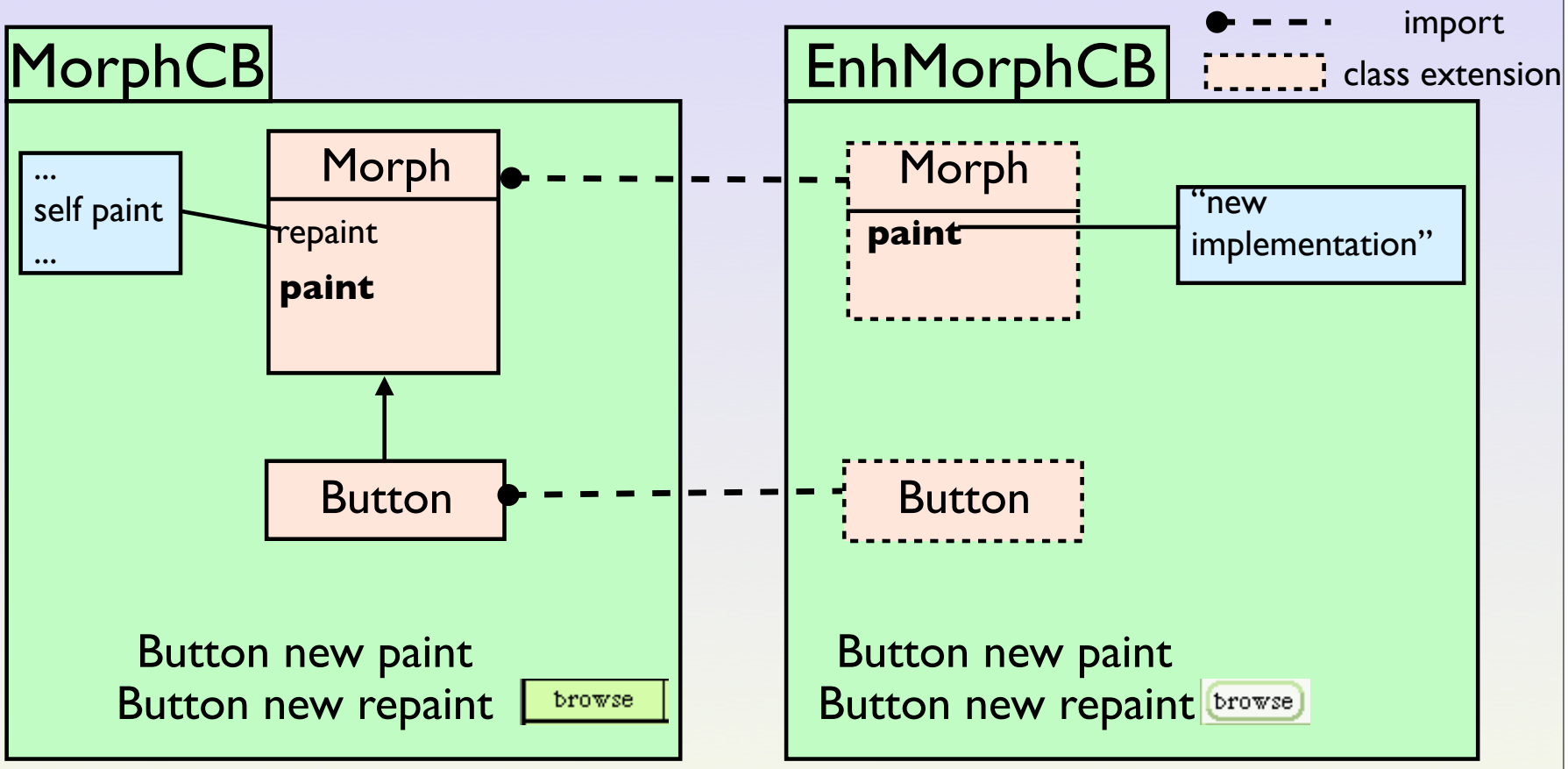
Supporting Unanticipated Changes

- Java, Modula3 provide package mechanism and no class extension
- AspectJ, Smalltalk, CLOS provide class extension and weak packages
- **How to combine module + class extension?**
- Scoped changes: avoiding **globality** and reconciling with **security**

The Classbox Model

- A **classbox** is a unit of scoping (it behaves as a namespace).
- Within a classbox:
  - Classes can be defined
  - Classes can be imported from other classboxes
  - Methods can be defined **on any visible class**
  - Code can be evaluated
- Local redefinitions take precedence over previous definitions

Local Rebinding

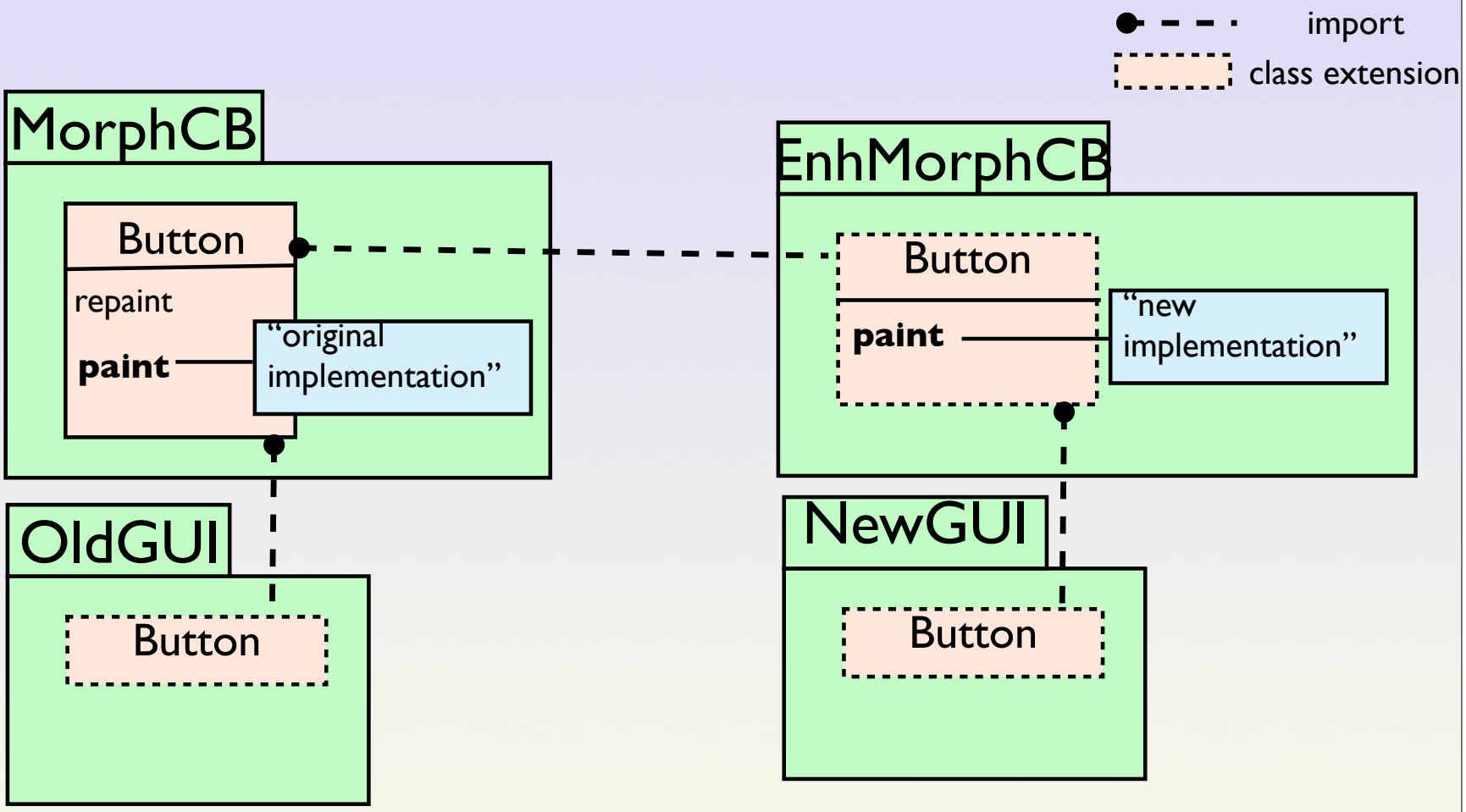


Flatten view within a classbox

Properties of the Model

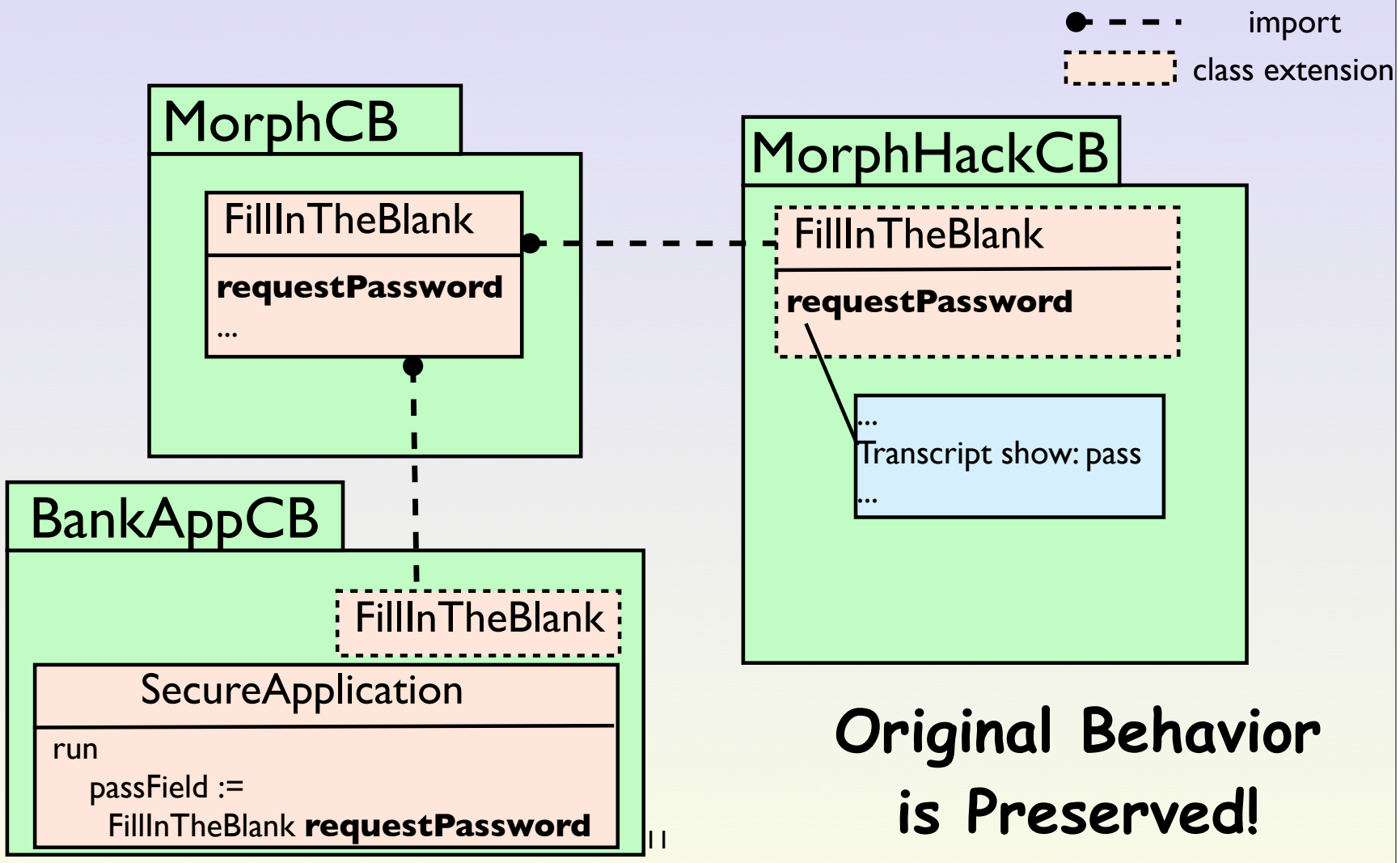
- From within a classbox: flattened view of the world.
- Local changes are as if they would have been global.
- Extending some classes does not impact their clients.

Visibility Bounded to a Classbox



Multiple versions of a method at the same time

Limited Class Extension Impact

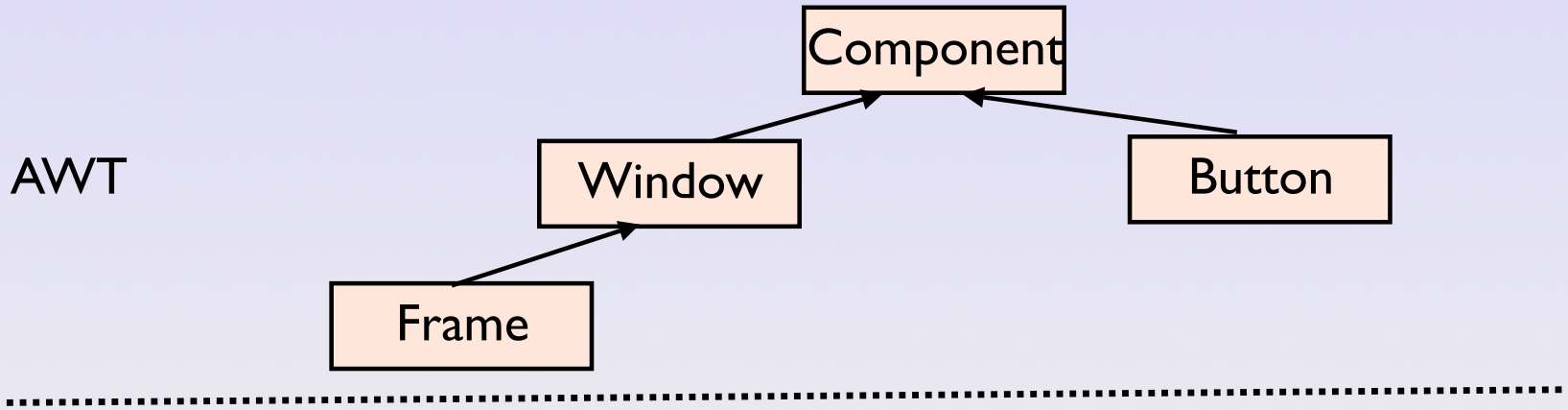


Original Behavior is Preserved!

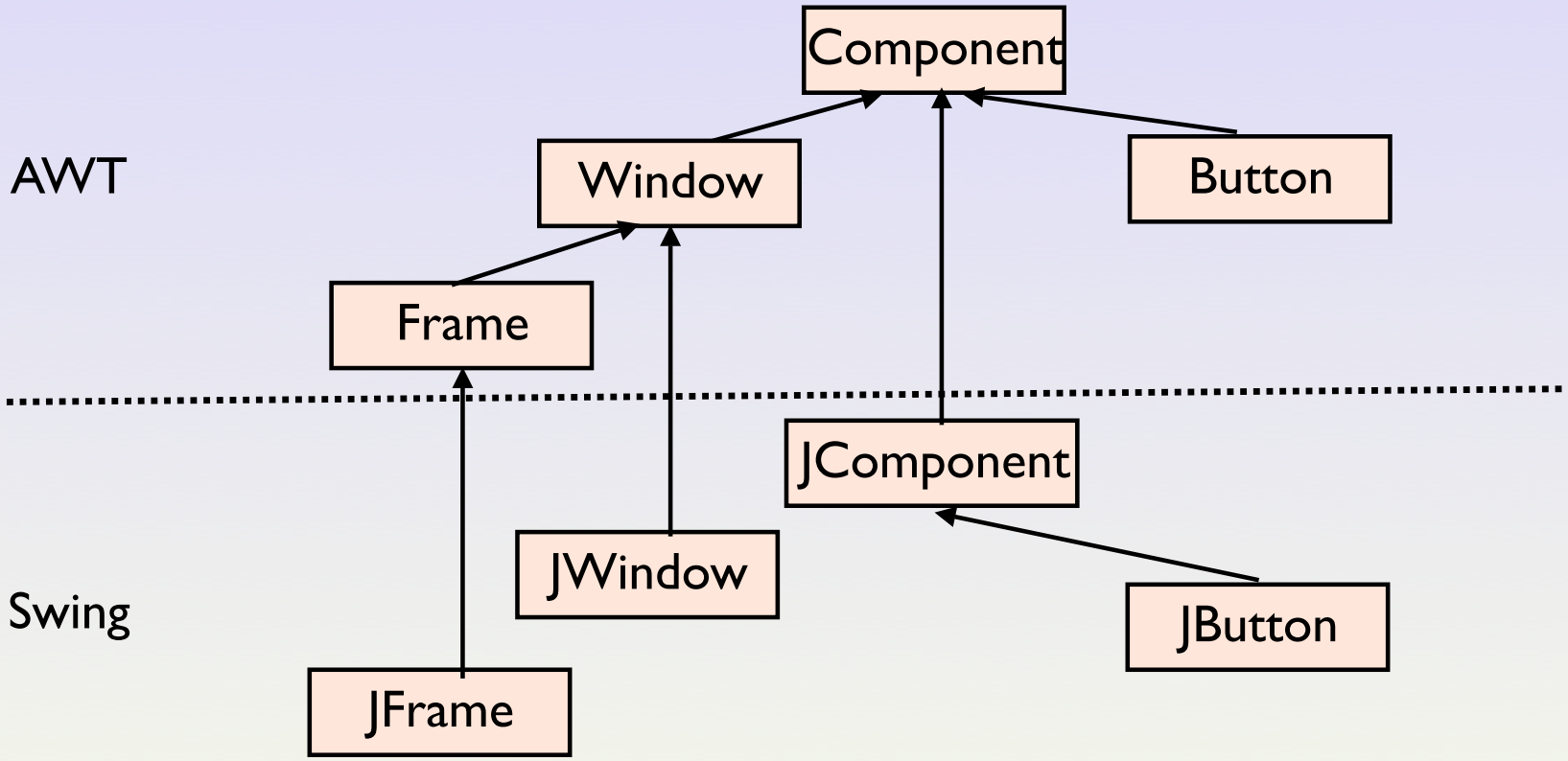
Implementation

- In Squeak, an open-source Smalltalk.
- New method lookup semantics.
- No need to modify the VM.
- Uses a cache mechanism.
- Cache is checked in redefined methods by adding 5 extra byte-codes.
- No overhead for added method invocation
- Redefined method invocation (worst case): 2.5 times
- Java implementation based on source code transformation

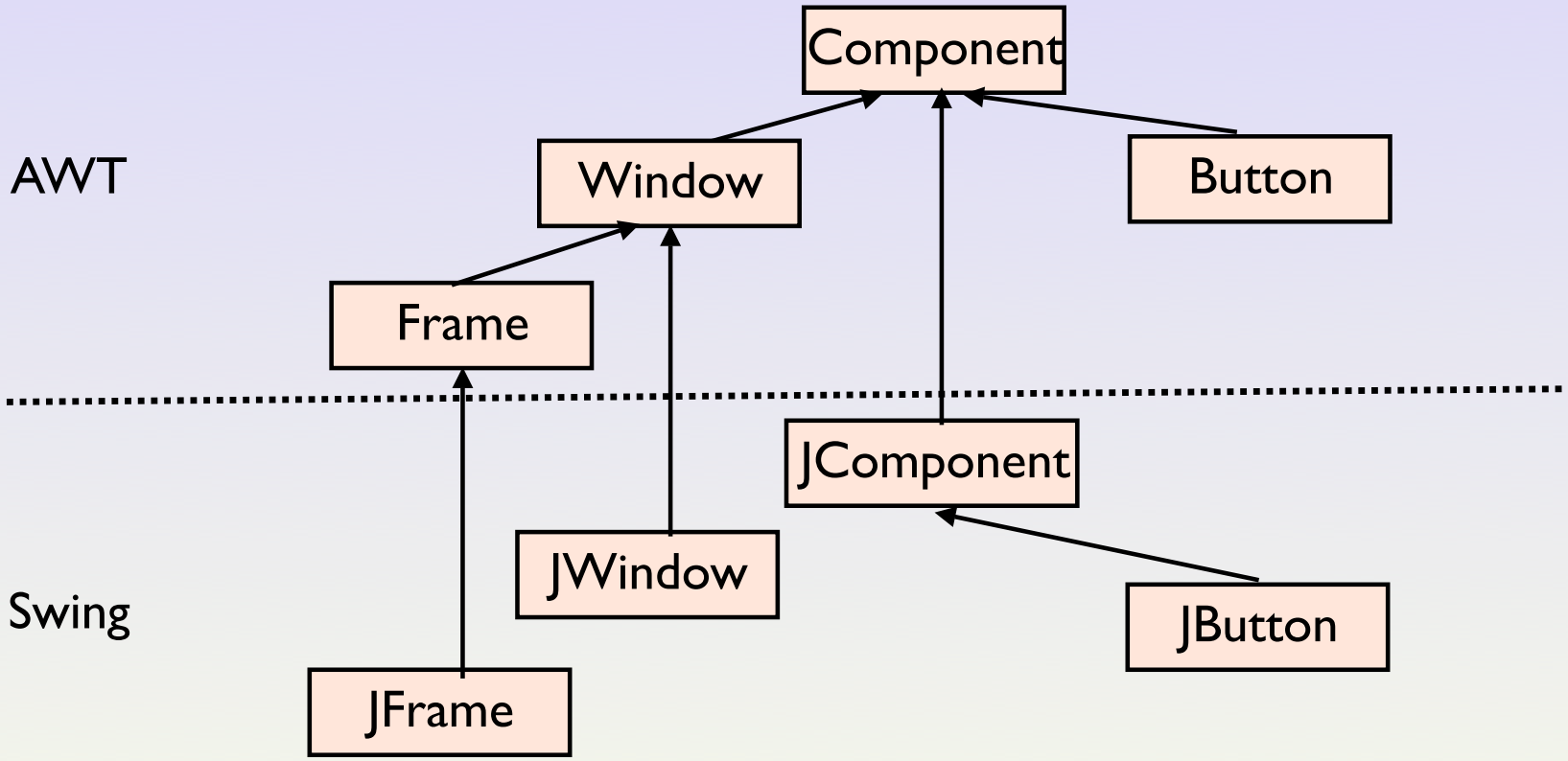
Case Study: AWT & Swing



Case Study: AWT & Swing

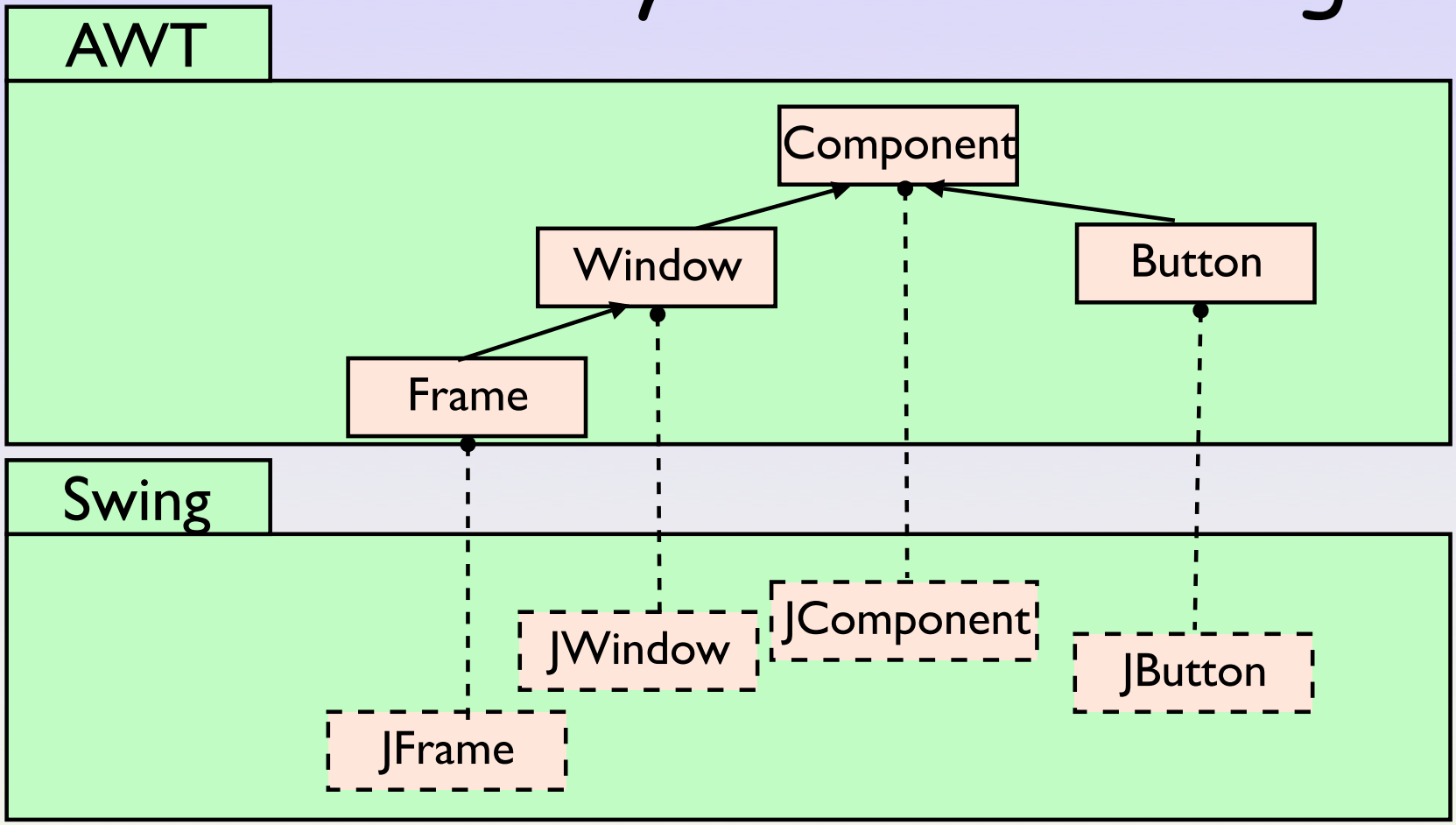


Case Study: AWT & Swing



- Code duplication (i.e., JFrame and JWindow, Button and JButton)
- a Frame is a Window, but a JFrame is not a JWindow
- a JWindow is not a JComponent

Case Study: AWT & Swing



- Less Code duplication
- a Frame is a Window in Swing Classbox
- a Window is Component in Swing

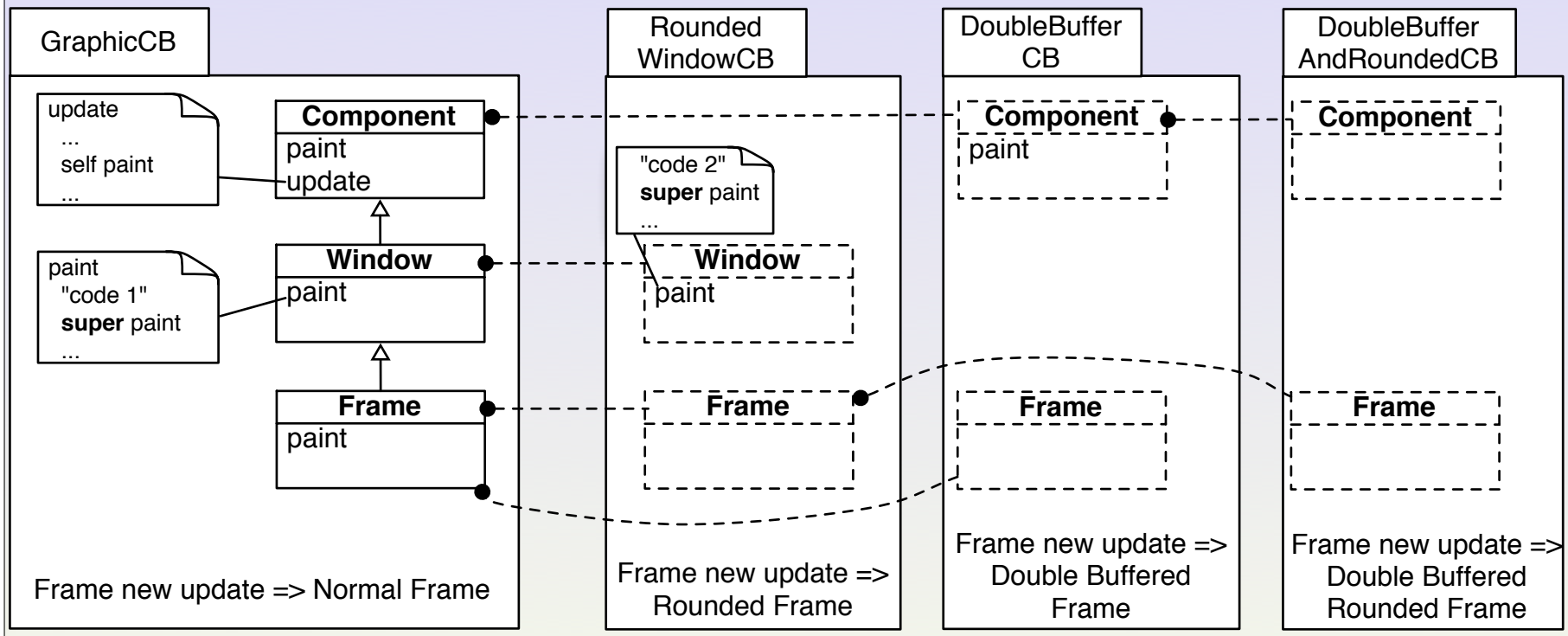
Conclusion

- Secure module system for controlling class extensions:
  - Control the visibility of method addition and replacement
  - Support for unanticipated evolution
- New method lookup semantics
  - Use of a cache mechanism

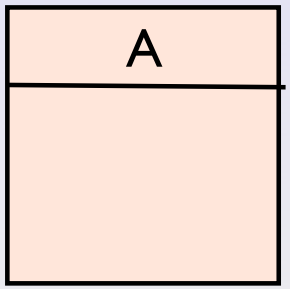
The Classbox Model

-
- http://www.iam.unibe.ch/~scg
- or Ask google.com about **classbox**
- Information: bergel@iam.unibe.ch

Import Before Inheritance

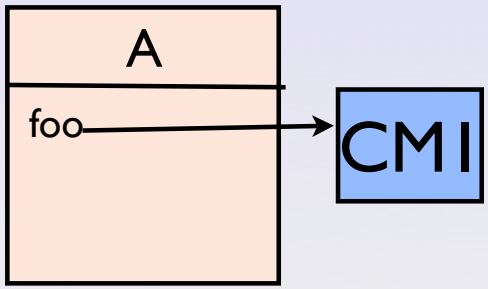


Cache Mechanism (1/4)



Class creation

Cache Mechanism (2/4)

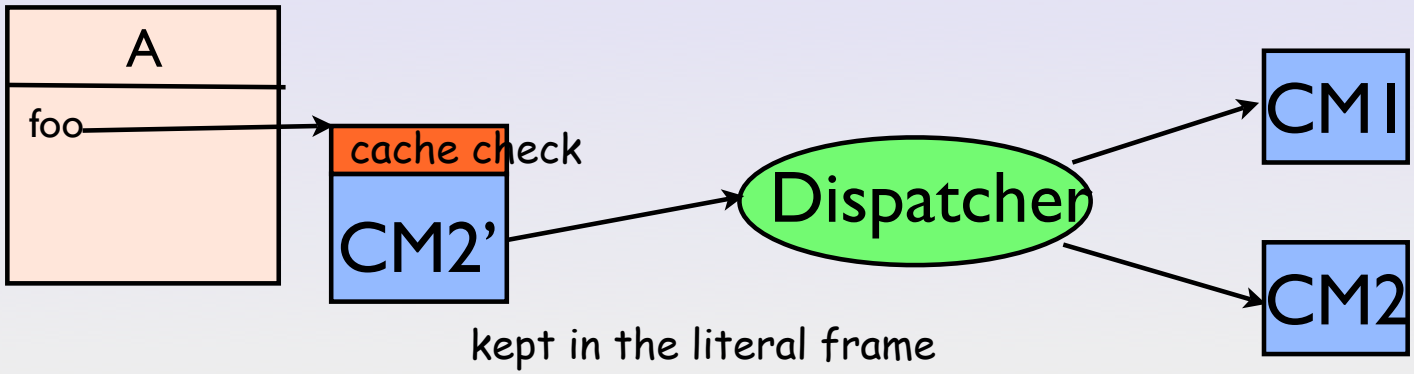


Method addition

Cache Mechanism (3/4)

Method redefinition

Cache Mechanism (4/4)



Method execution: *A new foo*

- Model composed of 3 classes: Classbox, ClassboxSystem and Dispatcher

Scope of a Method