



Towards Unified Aspect-Oriented Programming

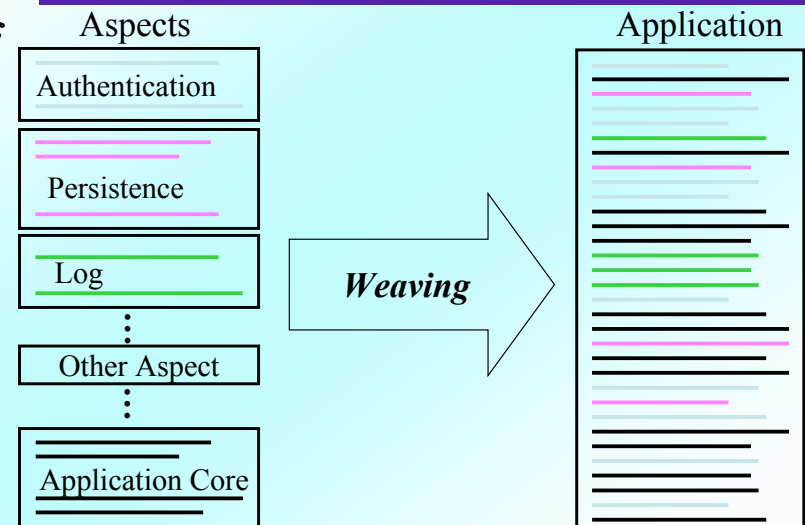
Noury Bouraqadi, Djamel Seriai, Gabriel Leblanc

<http://csl.ensm-douai.fr/research>

ESUG 2005 - Brussels, Belgium



AOP (Aspect-Oriented Programming)



Aspects and Weaving

- Application core = Set of classes
 - Without any crosscutting code
- Aspects = crosscut application core classes
 - Alter application structure
 - Addition/Change of classes, methods, IVs, ...
 - Alter application execution flow
 - Object creation/initialization, Access to IVs, Message dispatch...
- Weaving = Performing changes defined in Aspects



Need for Unification

- Two kinds of crosscutting [Laddad 2003]:
 - Dynamic crosscutting = changes that affect applications execution flow
 - Static crosscutting = changes that affect applications structure
- Limitations of existing AOP platforms:
 - Different constructs for all kinds of crosscutting
 - Code complexity even for simple aspects
 - Aspect reuse not always possible
 - Aspect conflicts not always managed

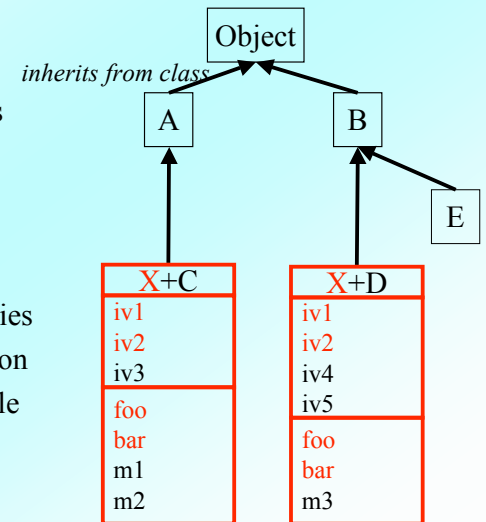
Mixins

Unified Aspects = Mixins + Reflection

Need for Mixin-Based Inheritance

Context

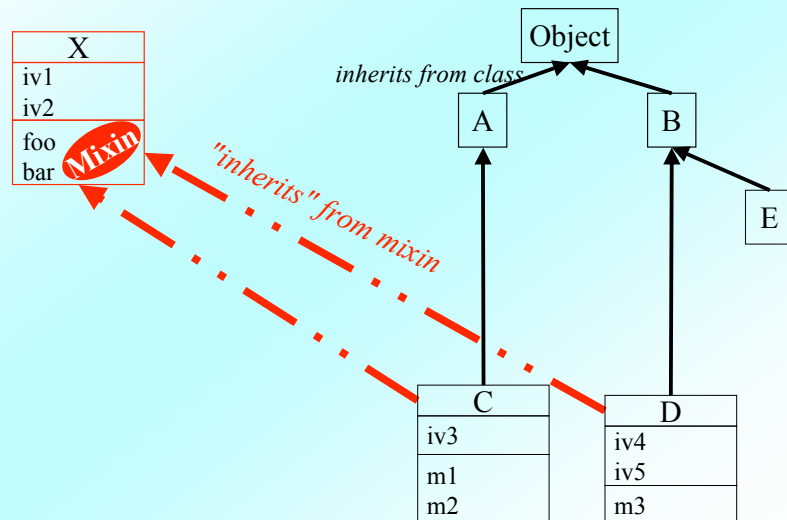
- Single inheritance
- Unrelated hierarchies
- Same Properties



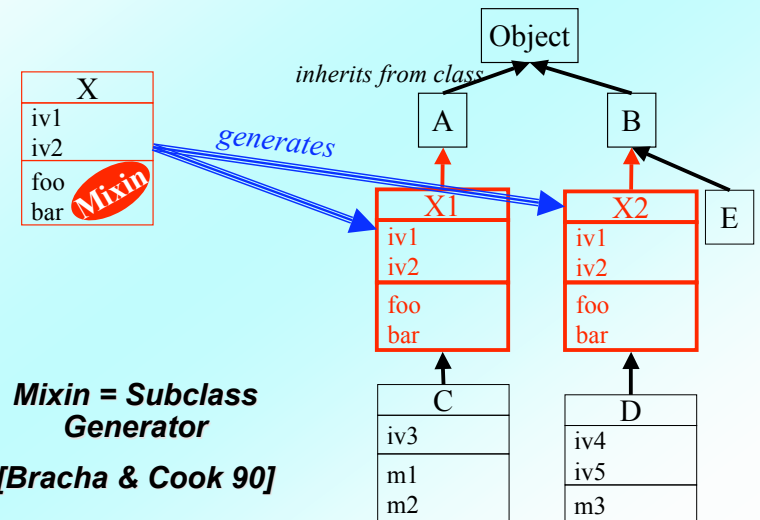
Goal

- Reuse shared properties
- Avoid code duplication
- Alternative to multiple inheritance

Mixin-Based Inheritance

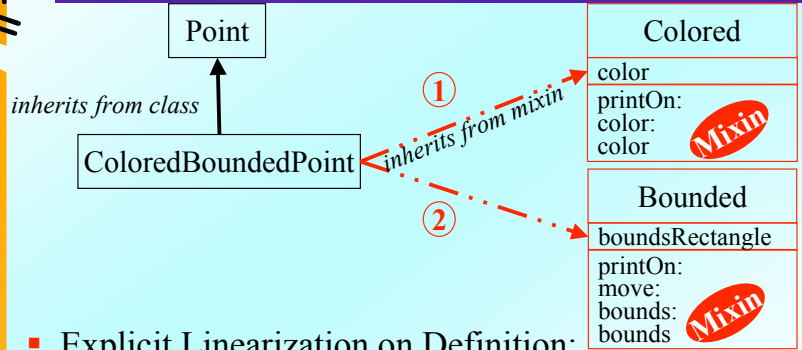


Single Inheritance Behind the Scene



Mixin = Subclass Generator
[Bracha & Cook 90]

Example of Mixin Inheritance

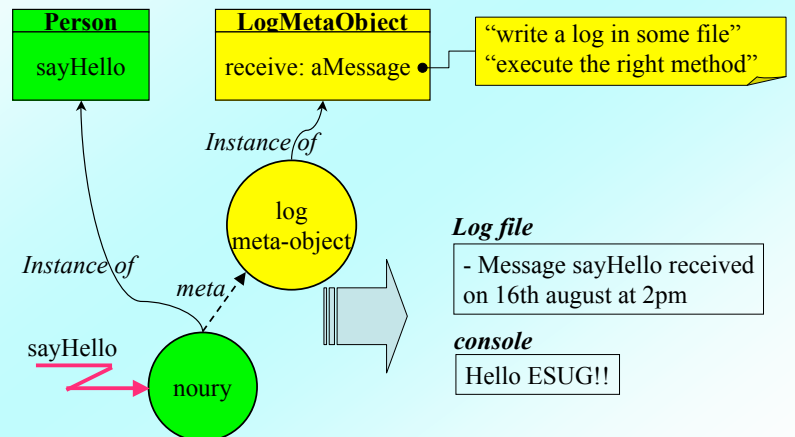


- Explicit Linearization on Definition:
 - *ColoredBoundedPoint* mixins: {*Colored*, *Bounded*}
- Lookup list
 - *ColoredBoundedPoint*, *Colored*, *Bounded*, *Point*

OO Reflective Languages

- A Reflective language gives access to its own semantics
- Two programming levels
 - evaluator → meta-level → meta-objects
 - program → base-level → base-objects
- Meta-object = an object that **controls** one or more base-objects
 - i.e. Evaluates messages sends, field accesses, ...

Example of Meta-Object Usage

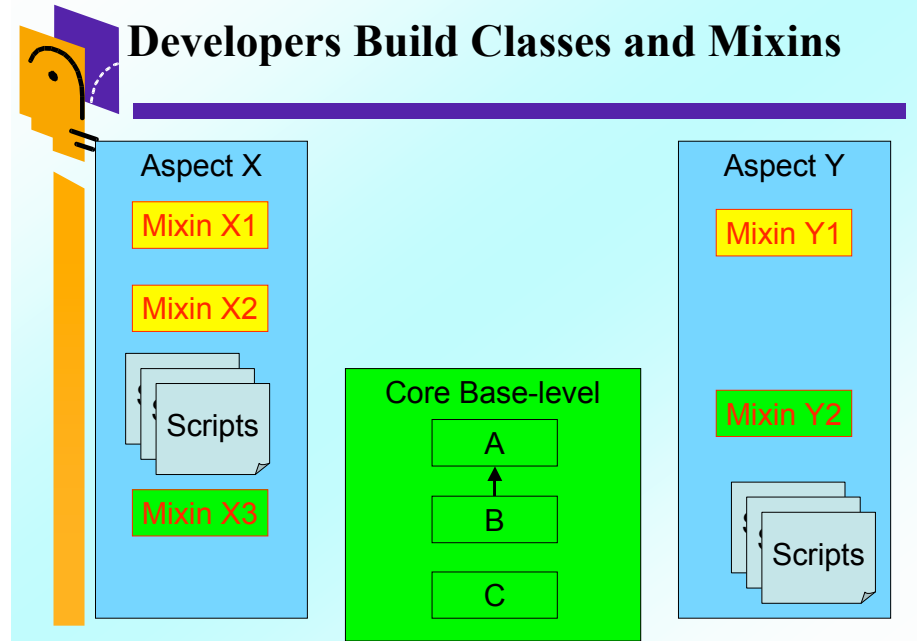


Log file
 - Message sayHello received on 16th august at 2pm

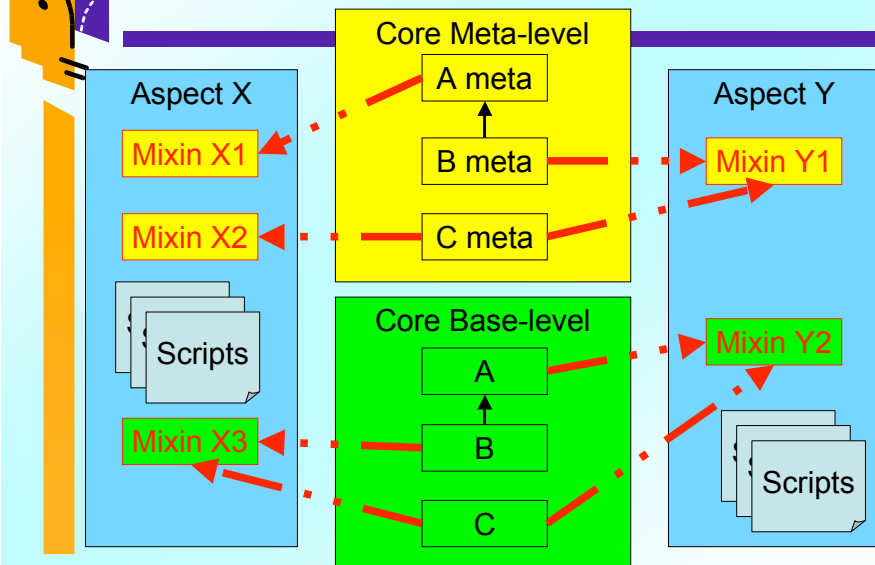
console
 Hello ESUG!!

Unified Aspects

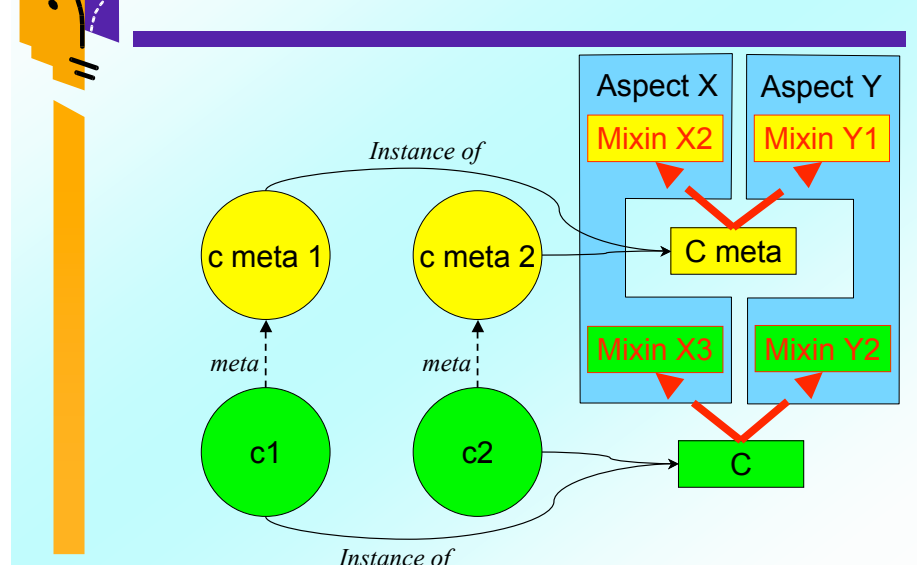
Unified Aspects = Mixins + Reflection



Static View After Weaving



Dynamic (partial) view after weaving





Weaving and Aspect Reuse

- Unified Aspects are reusable
 - Application independent mixins
- Weaving aspects into specific applications
 - Mapping mixins to application core classes
 - Pre/Post weaving scripts
- Weaving =
 1. Evaluate aspects pre-weaving scripts
 2. Link classes to mixins
 3. Evaluate aspects post-weaving scripts



Summary

- Crosscutting can be static or dynamic
 - Static : Alters applications structure
 - Dynamic : Alters applications behavior
- A **unified** representation of crosscutting
 - Mixins at base-level = static crosscutting
 - Mixins at meta-level = dynamic crosscutting
- **Reuse** is encouraged
- Simple conflict management = Mixins ordering



Future Work

- Weaving
 - Pre/Post weaving scripts reuse
 - High-level language for expressing pointcuts
- Advanced conflict management
 - Persistence: support application rebuilds
 - Order of pre/post weaving scripts evaluation
 - Finer grain: Method/Instance variable level

Conclusion