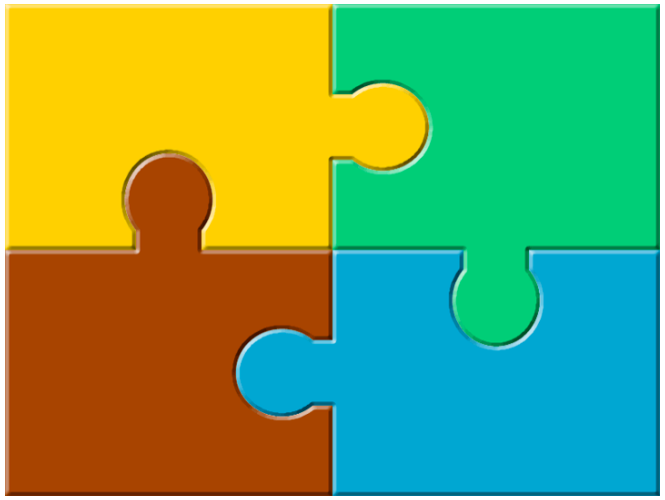


Composition Mechanisms in OO Languages: Traits and ClassBoxes

Roel Wuyts
ULB Séminaire

25/03/04

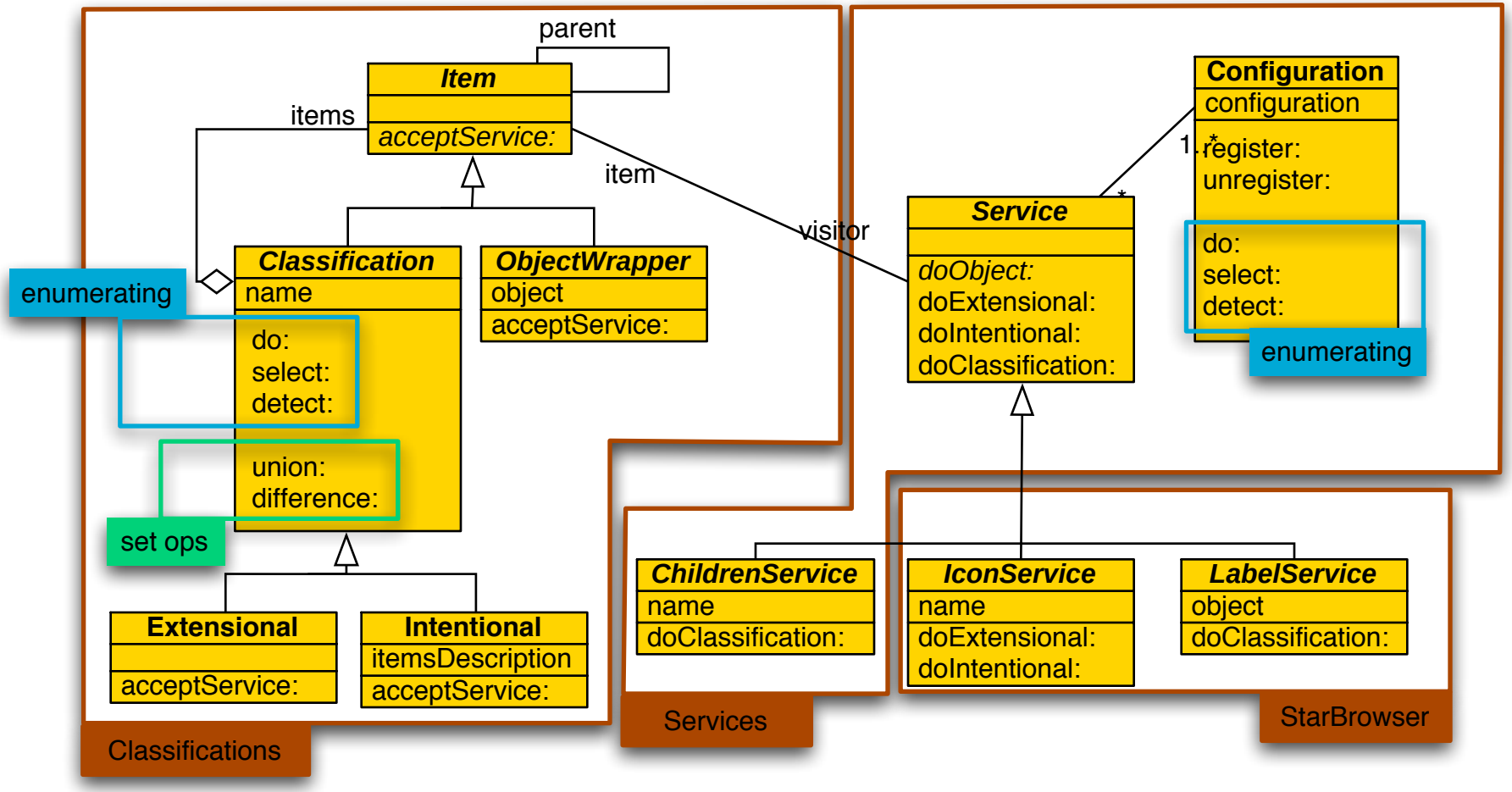
4 talks

- "Logic Meta Programming and Language Symbiosis"
- "A Data-centric Approach to Composing Embedded, Real-time Software Components"
- "Unanticipated Integration of Development Tools and the StarBrowser"
- "Composition Mechanisms in OO Languages: Traits and ClassBoxes"

Roadmap

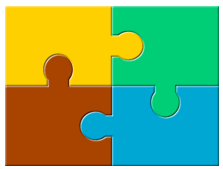
- Context
- Traits
  - Problem, model, validation
- Classboxes
  - Problem, model, validation
- Validation
- Conclusion

Context: OO Systems

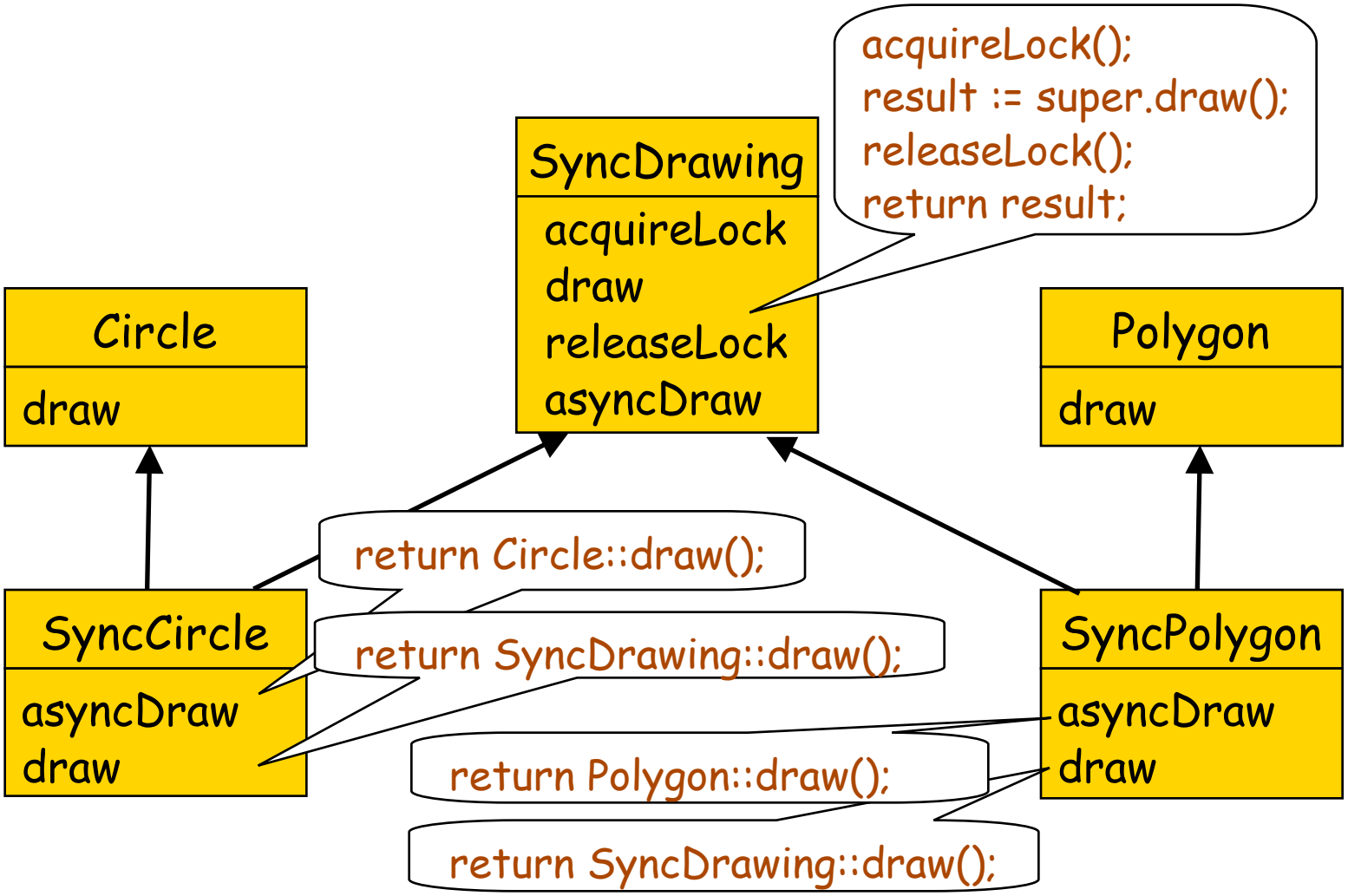


Traits

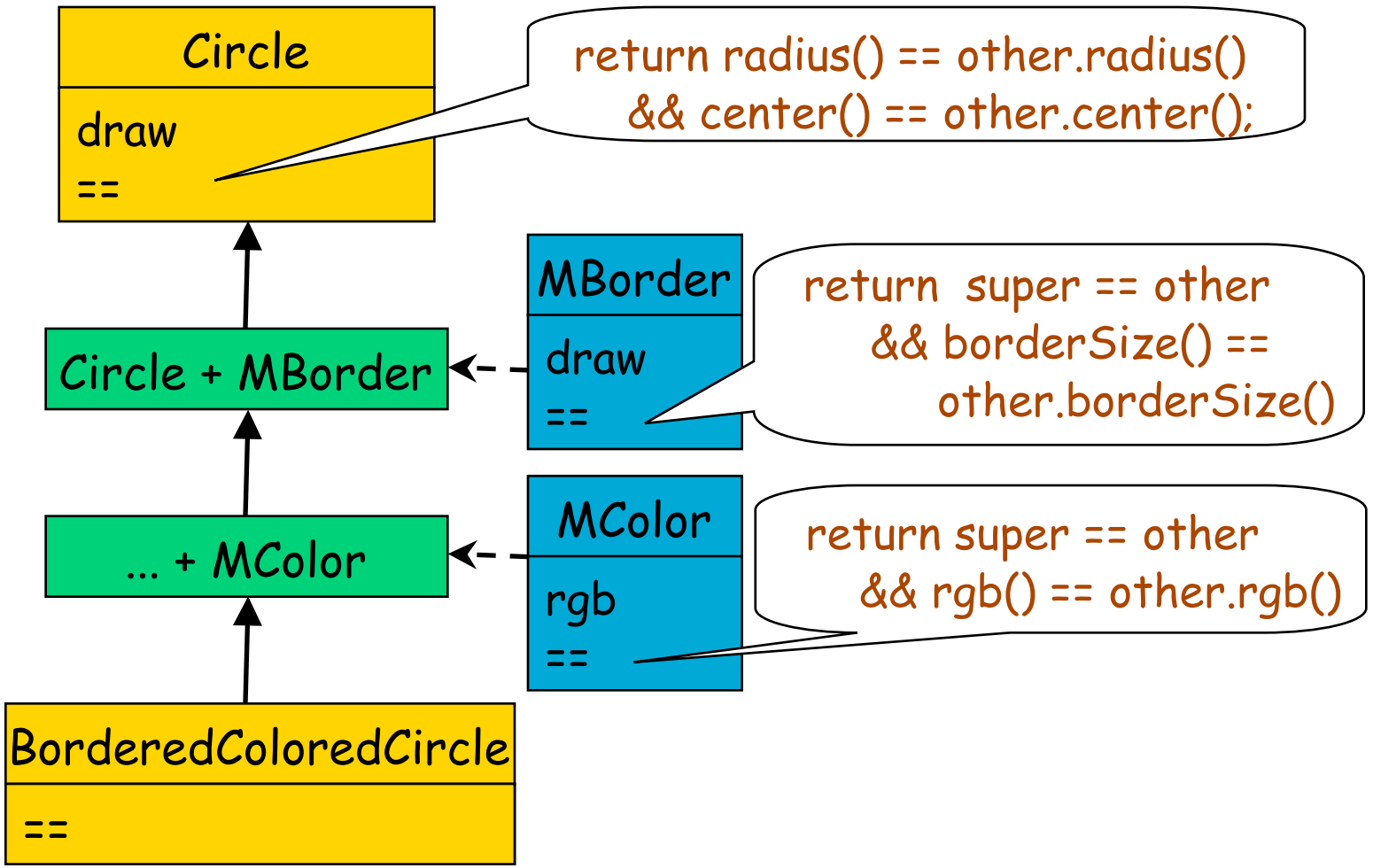
- Modularity of methods within classes
  - Solve shortcomings with inheritance
- Compose methods in groups
- Compose groups to form classes
- Advantage: level between single methods and complete classes

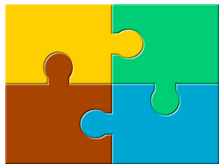


Multiple inheritance problem



Mixins

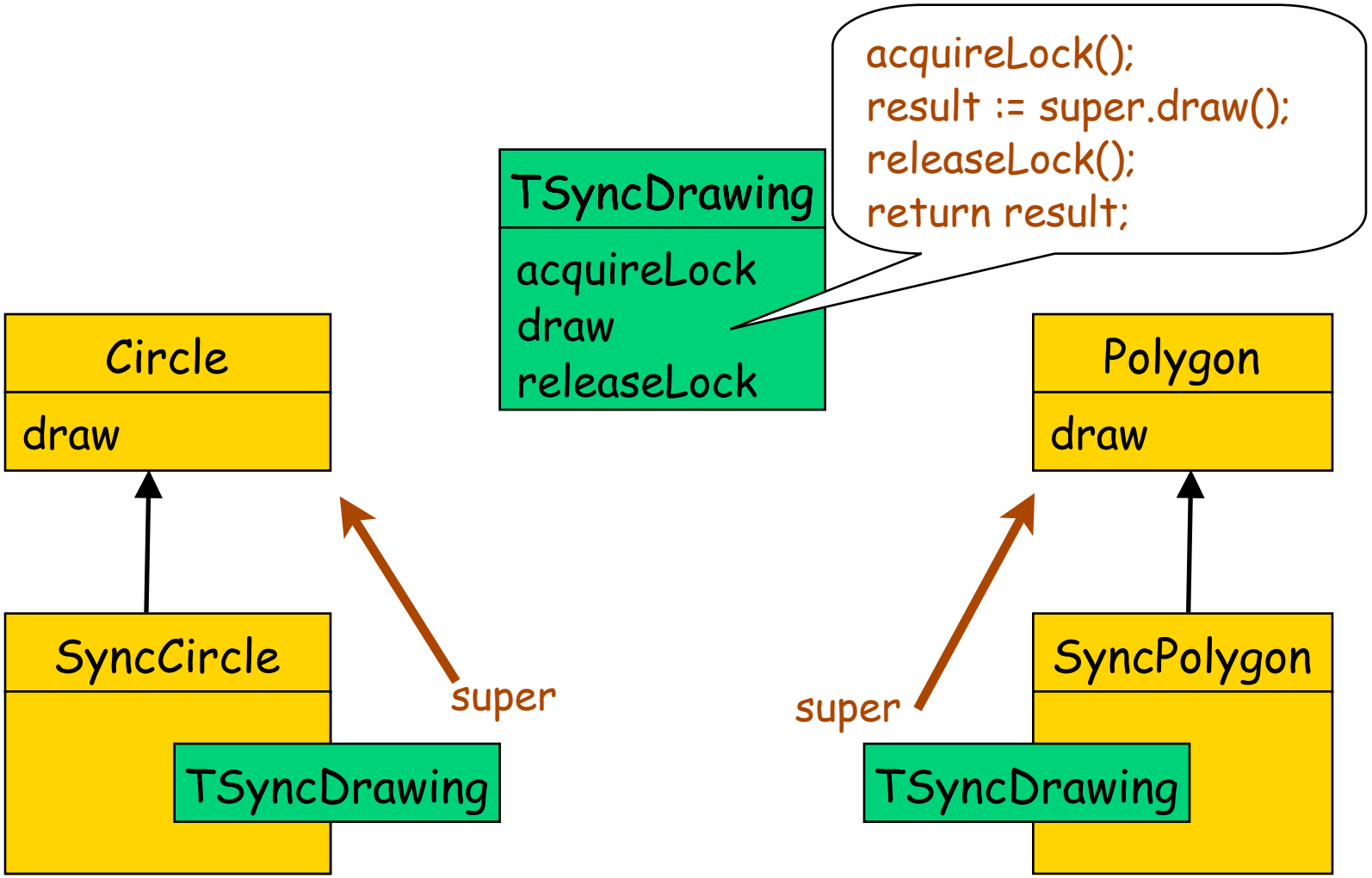

Circle
draw
==
return radius() == other.radius()
&& center() == other.center();
Circle + MBorder
MBorder
draw
==
return super == other
&& borderSize() ==
other.borderSize()
... + MColor
MColor
rgb
==
return super == other
&& rgb() == other.rgb()
BorderedColoredCircle
==



Inheritance Problems

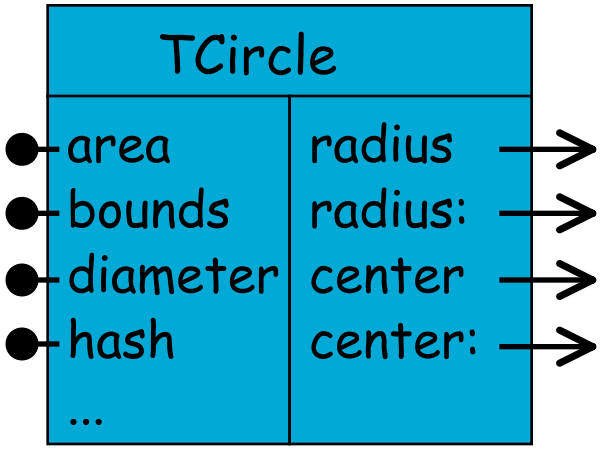
- Single: not expressive enough
- Multiple:
  - Complex solution → Hard to understand
  - Explicit class references → Fragile
  - Sometimes requires code duplication
- Mixins:
  - Implicit conflict resolution → Surprises!
  - Composite entity is not in full control
    - → Dispersal of glue code
    - → Fragile hierarchies

What we want..


acquireLock();
result := super.draw();
releaseLock();
return result;
TSyncDrawing
acquireLock
draw
releaseLock
Circle
draw
Polygon
draw
SyncCircle
TSyncDrawing
super
SyncPolygon
TSyncDrawing
super

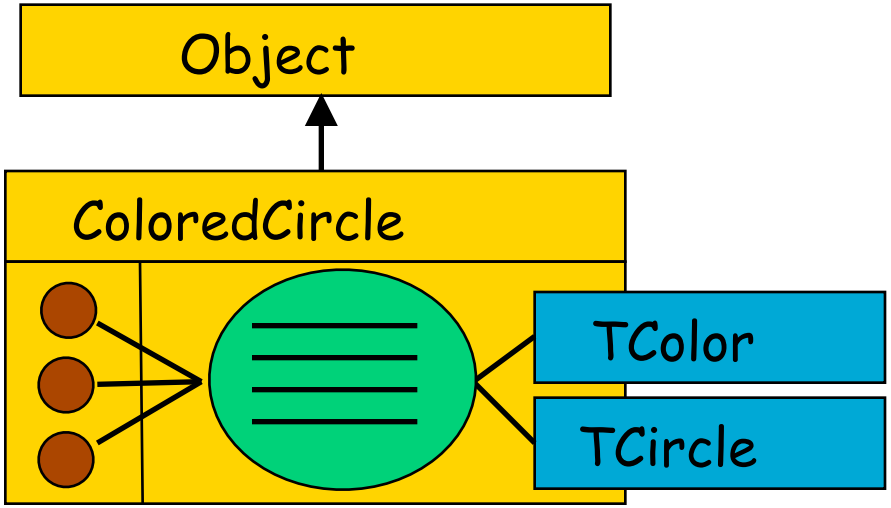
What are Traits?

- Traits are parameterized behaviors
  - Traits provide a set of methods (●—)
  - Traits require a set of methods (→)
  - Traits are purely behavioral (no state)



How are Traits Used?

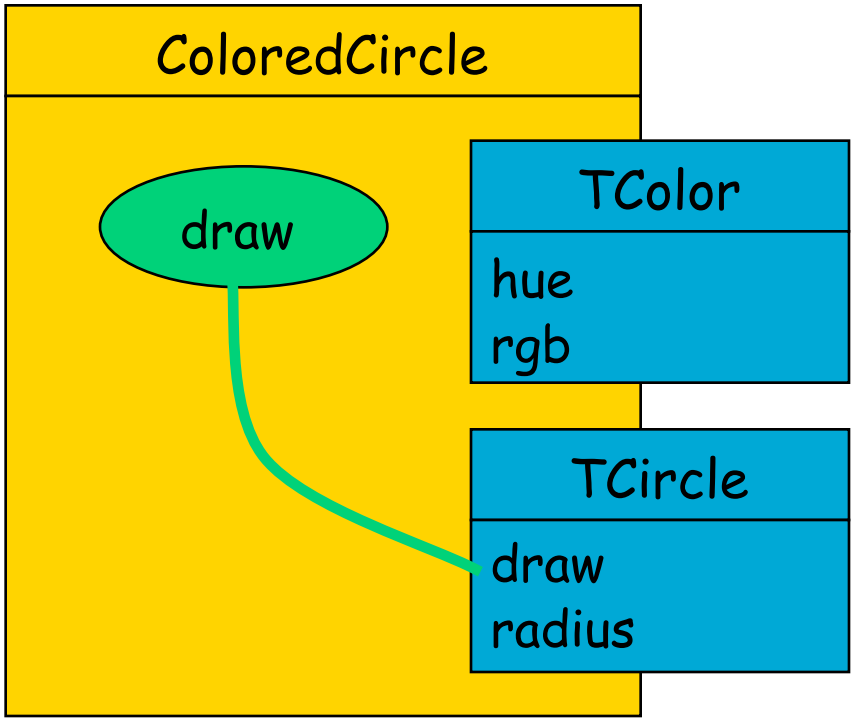
- Traits are the building blocks of classes
  - Class = superclass + state + traits + glue



- Traits do not replace single inheritance
  - They provide modularity *within* classes

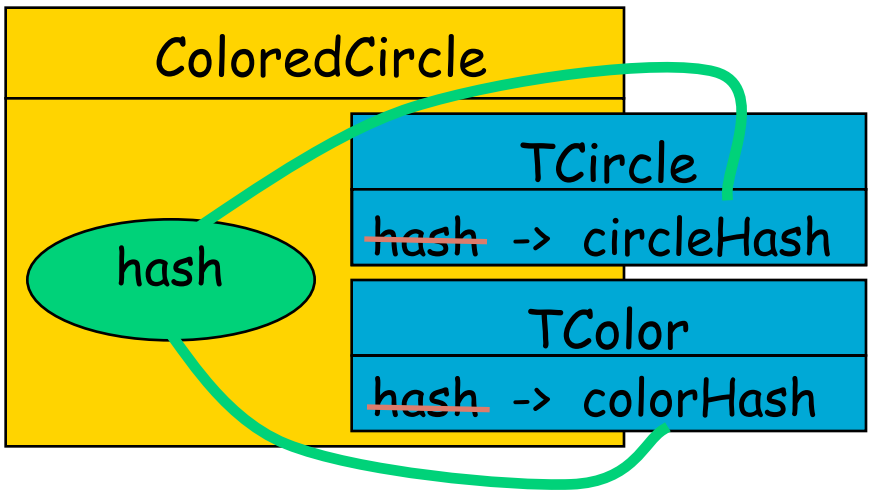
Composition Rules

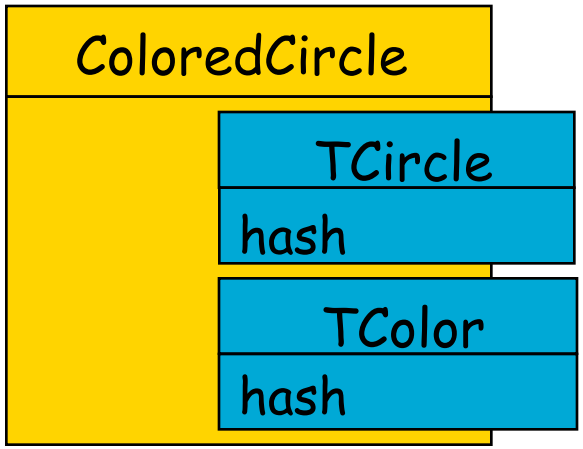
- Class methods take precedence over trait methods



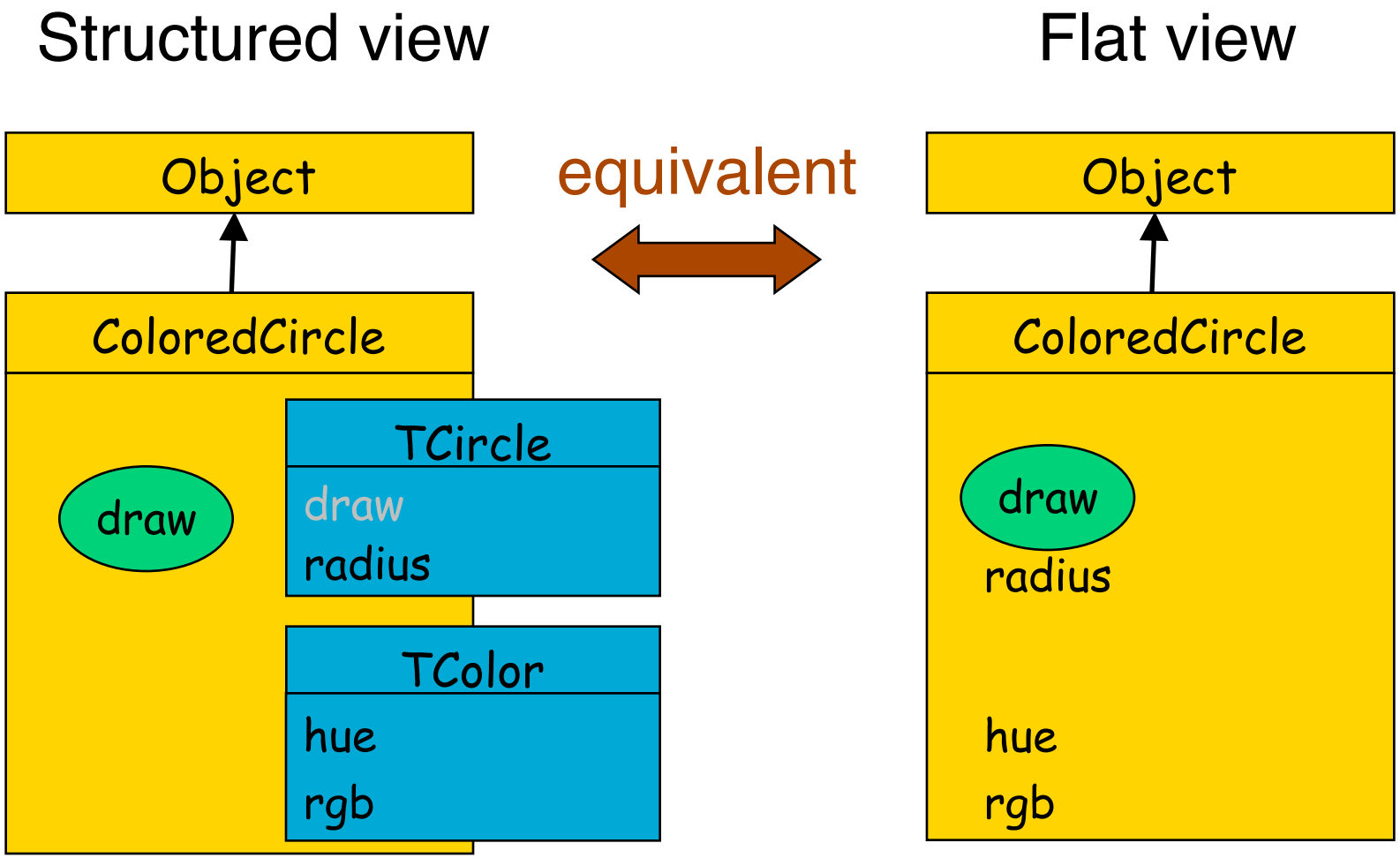
Explicit Conflict Resolution

- Override the conflict with a glue method
  - Aliases to access to the conflicting methods
- Avoid the conflict
  - Exclude conflicting method from one trait





Flattening Property (2 views on code)



Validation

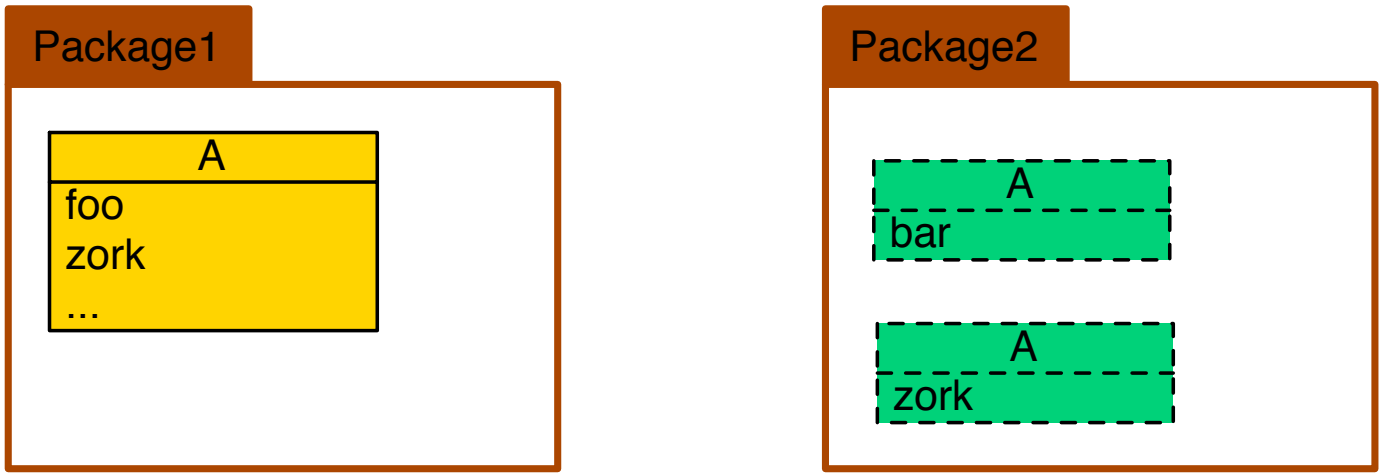
- Implemented in Smalltalk
  - Smart method dictionary manipulation
  - Development environment extended
- Set-theoretic formalization to prove the flattening property
- Applications
  - Refactored the collection hierarchy
  - applied on metaclass composition
- Broad impact

Classboxes

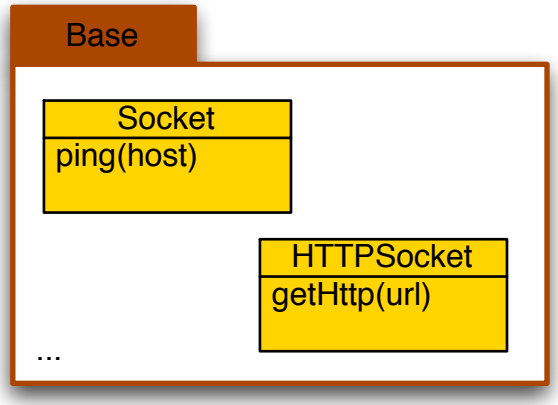
- Modularity of (groups of) classes
  - address packaging problems
- Compose classes and methods
- Import classes
- Extend existing classes locally
- Advantage: Module system for OO language handling class extensions

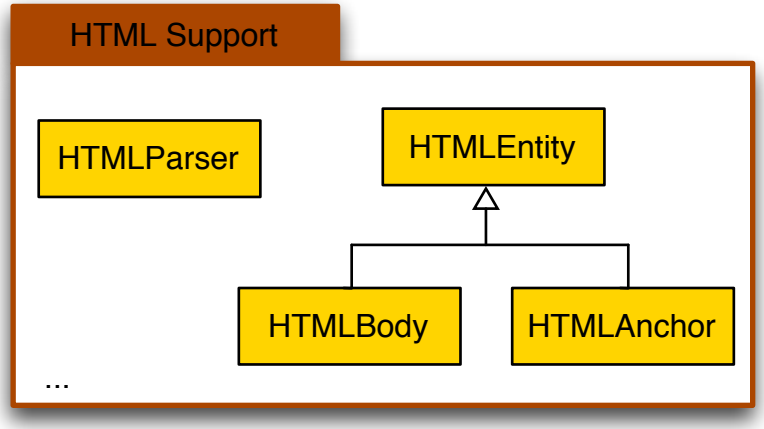
Class extensions

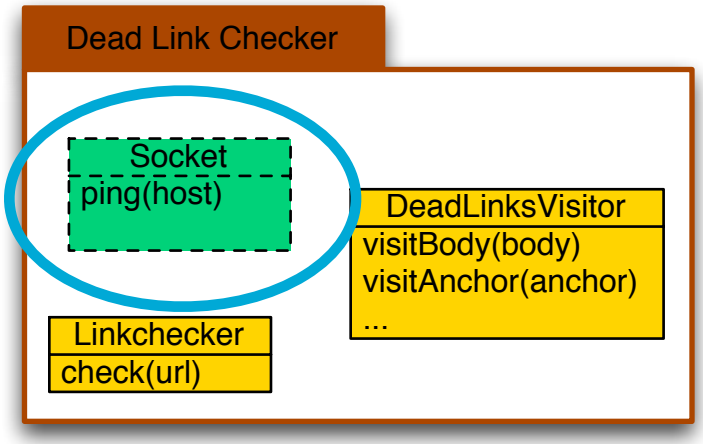
- method defined in a package
  whose class is defined elsewhere
- can be an *extension* or a *redefinition*
- exists in CLOS, Smalltalk, Ruby, ...

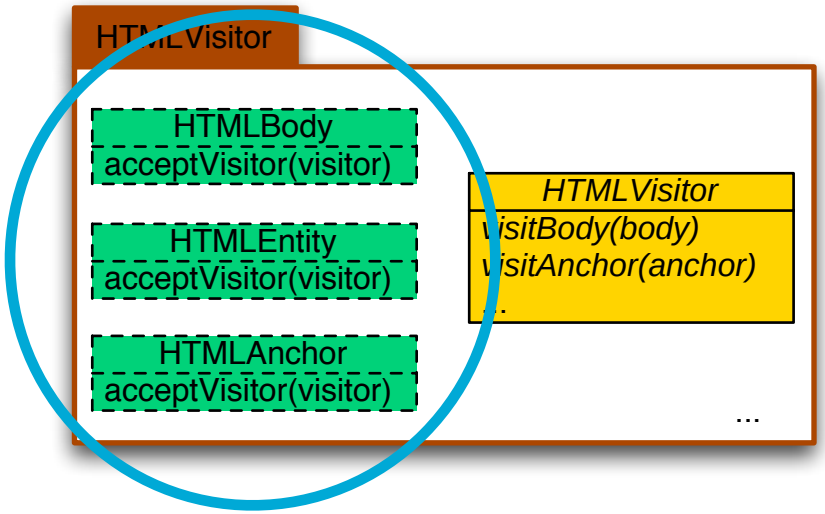


What we want...


Base
Socket
ping(host)
HTTPSocket
getHttp(url)
...


HTML Support
HTMLParser
HTMLEntity
HTMLBody
HTMLAnchor
...


Dead Link Checker
Socket
ping(host)
DeadLinksVisitor
visitBody(body)
visitAnchor(anchor)
...
Linkchecker
check(url)

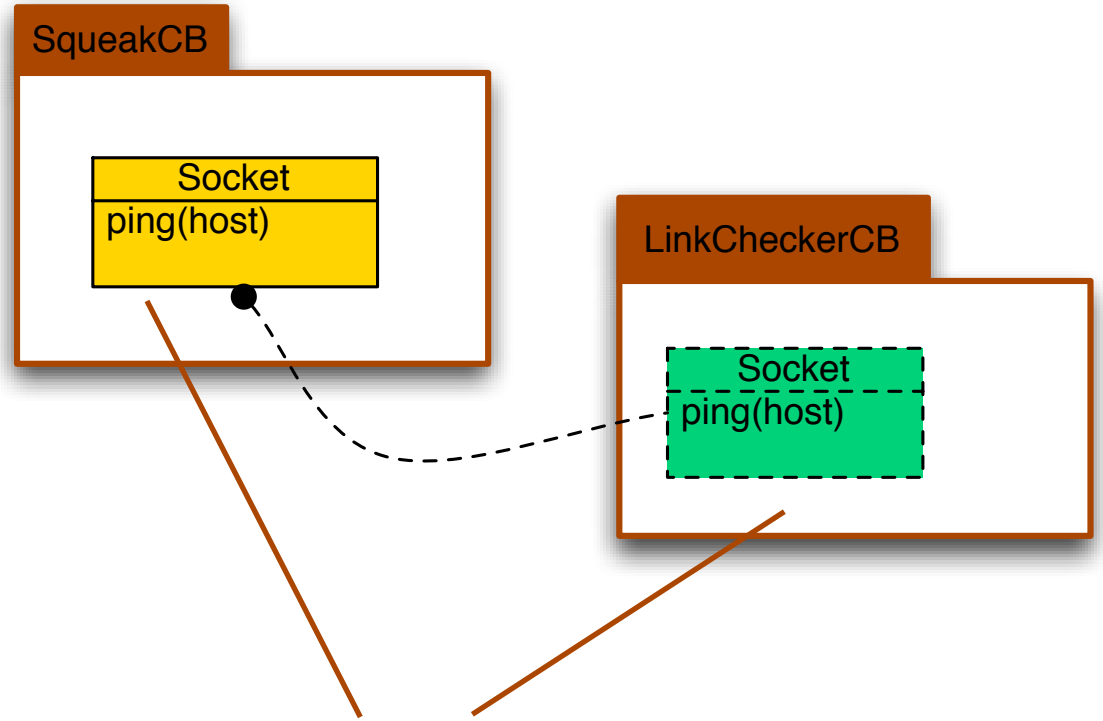

HTMLVisitor
HTMLBody
acceptVisitor(visitor)
HTMLEntity
acceptVisitor(visitor)
HTMLAnchor
acceptVisitor(visitor)
HTMLVisitor
visitBody(body)
visitAnchor(anchor)
...
...

Classbox Model Definition

- The Classbox Model is a module system supporting local rebinding
- A Classbox
  - Is a unit of scoping (acts as a namespace)
  - Can define classes, methods, variables
  - Can import class definitions
- Classes and methods always belong to exactly one classbox

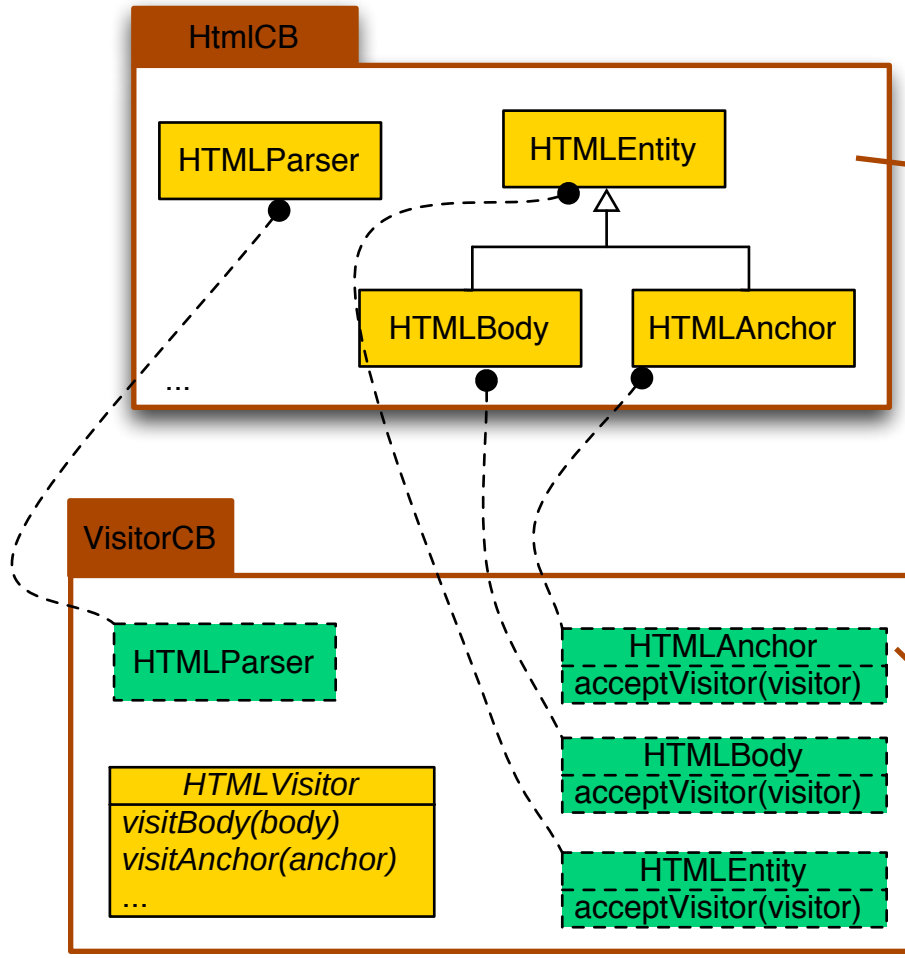
Local rebinding

- Extensions act as if they were global, but they are local to the classbox



Two different implementations!

Local rebinding (ctd)



Parser creates HTML nodes

Parser creates "visitable" instances

Validation

- Implemented in Smalltalk
  - new method lookup algorithm
  - extended development environment
- Classboxes
  - Set-theoretic formalization to prove the local rebinding property
  - applied on web application framework

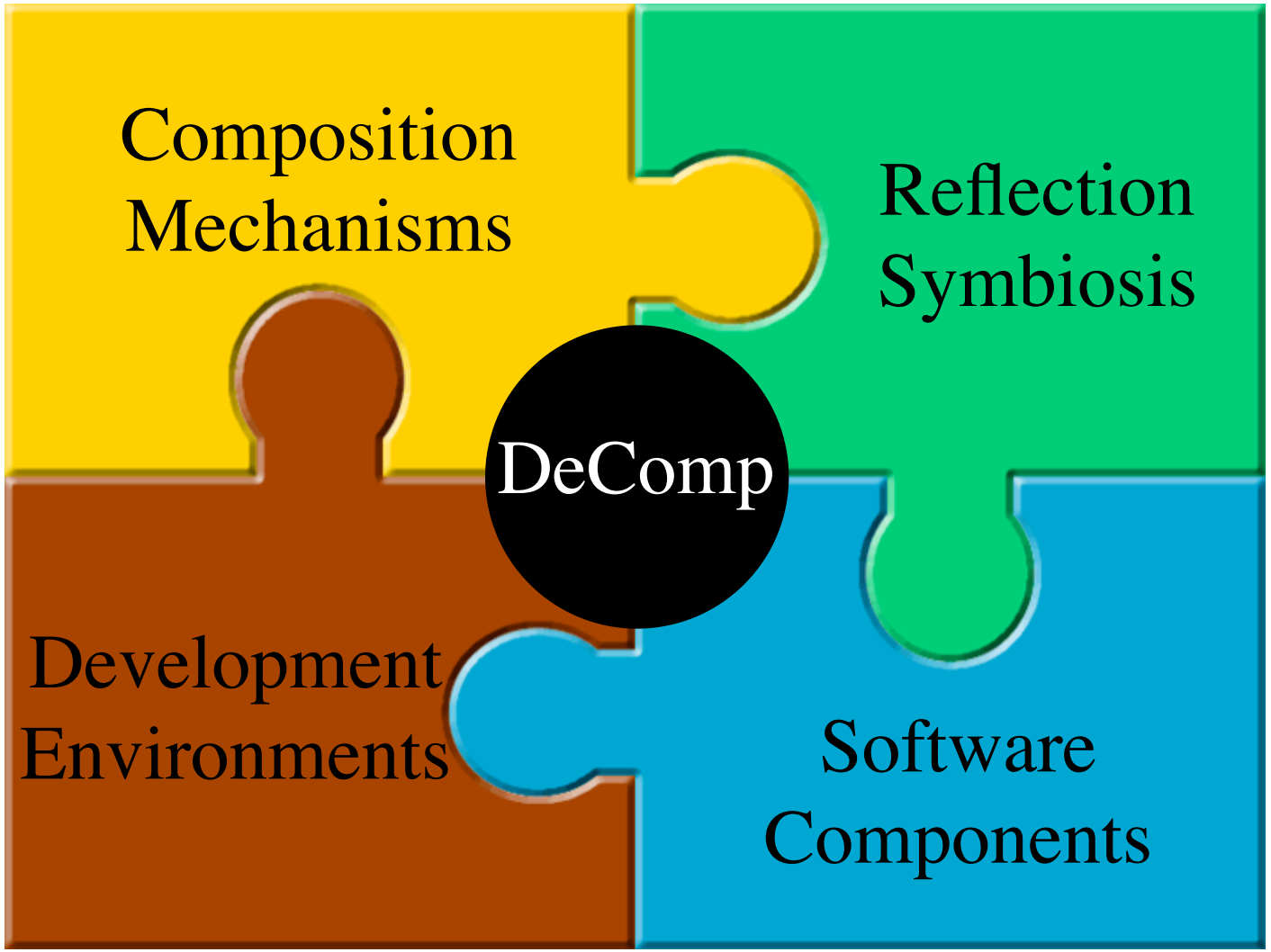
Conclusion

- Composition at different levels
  - Traits: methods in classes
  - Classboxes: classes in modules
- Automatic conflict detection
- Explicit (manual) conflict resolution
  - overriding, cancellation, aliasing

More Information...

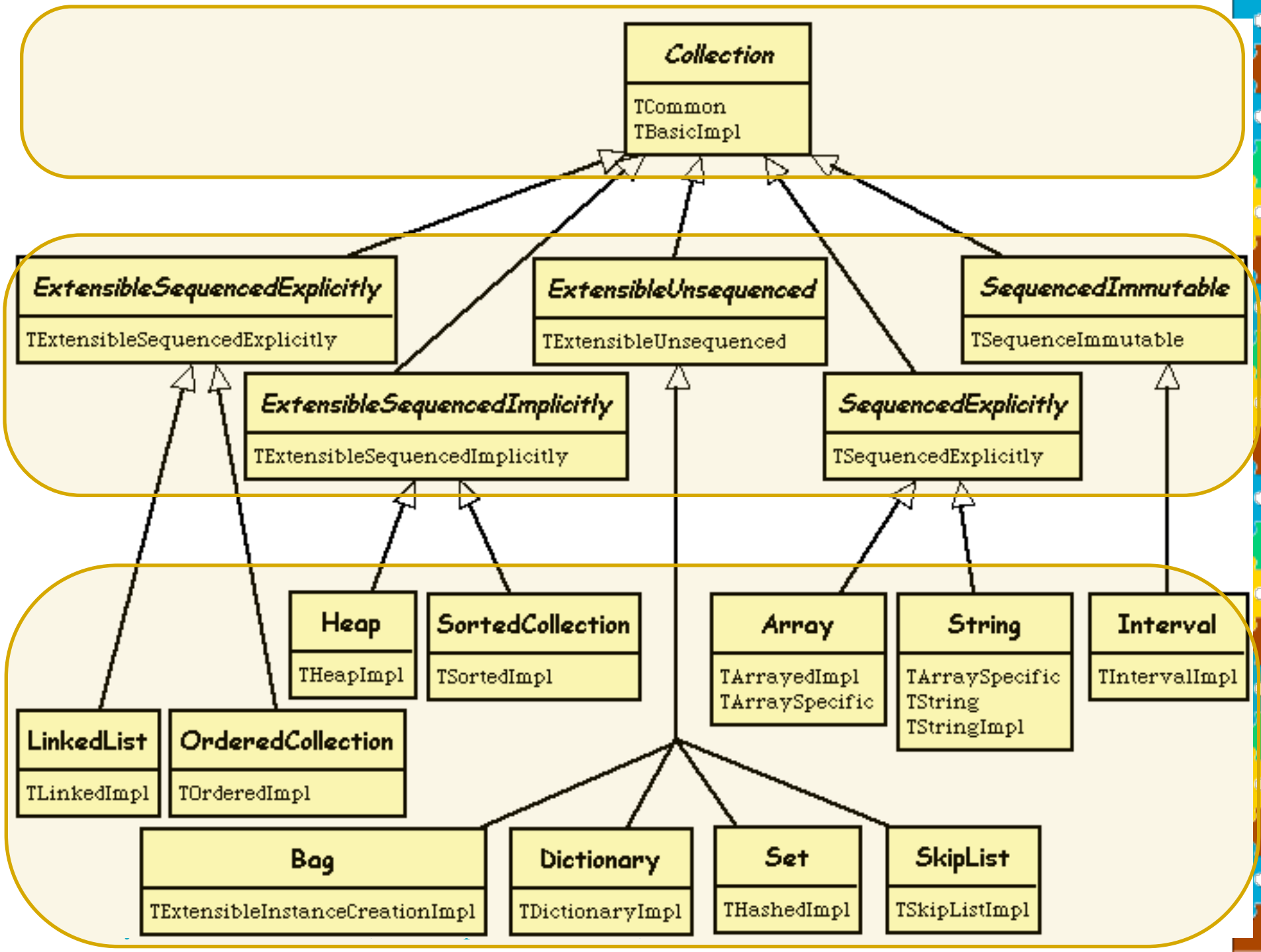
- Alexandre Bergel, Stéphane Ducasse, and Roel Wuyts, *Classboxes: A Minimal Module Model Supporting Local Rebinding*, In Proceedings of JMLC 2003 (Joint Modular Languages Conference), LNCS, Volume 2789, Springer-Verlag, pp. 122--131, 2003
- Andrew P. Black, Nathanael Schärli, Stéphane Ducasse, *Applying Traits to the Smalltalk Collection Hierarchy*, Proceedings of OOPSLA 2003, Springer-Verlag, pp. 47--64, 2003.
- Nathanael Schärli, Stéphane Ducasse, Oscar Nierstrasz, Andrew P. Black, *Traits: Composable Units of Behavior*, Proceedings of ECOOP 2003
- Stephane Ducasse, Nathanael Schaerli, Oscar Nierstrasz, Roel Wuyts and Andrew Black, *Traits: A Mechanism for fine-grained Reuse*, Submittet to TOPLAS.

Meta conclusion



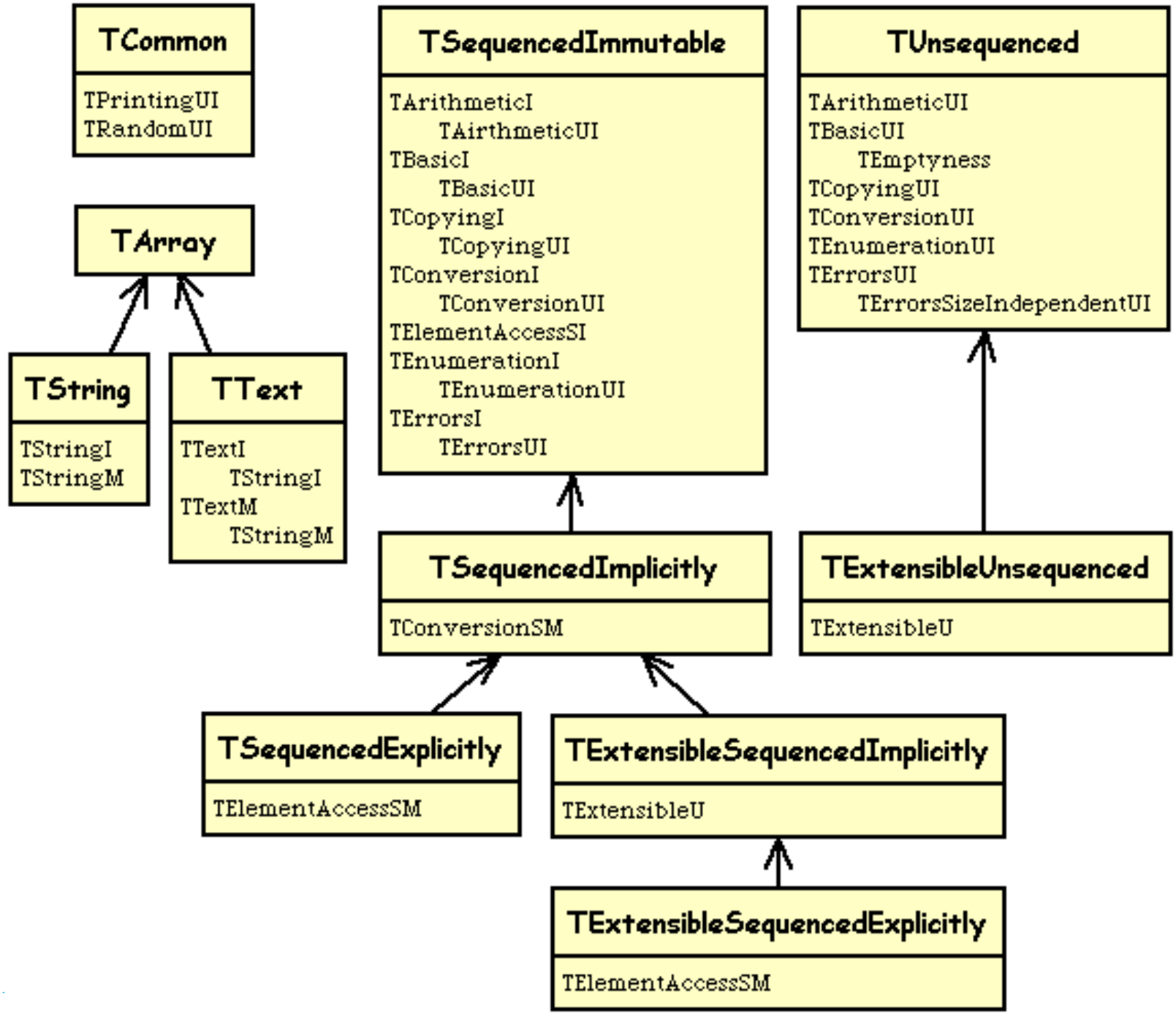
Resulting Class Hierarchy

- Result consists of 3 parts
  - Abstract root class Collection
    - Only contains the methods supported by all collection classes
  - Abstract classes with the public functionality for different kind of collections
  - Layer of concrete collection classes
    - They inherit the public functionality from one of the functionality classes
    - They use a trait that adds a specific implementation


Collection
TCommon
TBasicImpl
ExtensibleSequencedExplicitly
TExtensibleSequencedExplicitly
ExtensibleUnsequenced
TExtensibleUnsequenced
SequencedImmutable
TSequenceImmutable
ExtensibleSequencedImplicitly
TExtensibleSequencedImplicitly
SequencedExplicitly
TSequencedExplicitly
Heap
THeapImpl
SortedCollection
TSortedImpl
Array
TArrayedImpl
TArraySpecific
String
TArraySpecific
TString
TStringImpl
Interval
TIntervalImpl
LinkedList
TLinkedImpl
OrderedCollection
TOrderedImpl
Bag
TExtensibleInstanceCreationImpl
Dictionary
TDictionaryImpl
Set
THashedImpl
SkipList
TSkipListImpl

Resulting Trait Hierarchies

- Two trait hierarchies
  - Functional Traits
  - Implementation Traits
- Very fine-grained
  - Most traits consist of multiple subtraits


TCommon
TPrintingUI
TRandomUI
TArray
TString
TStringI
TStringM
TText
TTextI
TStringI
TTextM
TStringM
TSequencedImmutable
TArithmeticI
TAirthmeticUI
TBasicI
TBasicUI
TCopyingI
TCopyingUI
TConversionI
TConversionUI
TElementAccessSI
TEnumerationI
TEnumerationUI
TErrorsI
TErrorsUI
TUnsequenced
TArithmeticUI
TBasicUI
TEmptyness
TCopyingUI
TConversionUI
TEnumerationUI
TErrorsUI
TErrorsSizeIndependentUI
TSequencedImplicitly
TConversionSM
TExtensibleUnsequenced
TExtensibleU
TSequencedExplicitly
TElementAccessSM
TExtensibleSequencedImplicitly
TExtensibleU
TExtensibleSequencedExplicitly
TElementAccessSM