OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

FACULTY OF COMPUTER SCIENCE

# Variant-Preserving Refactorings for Migrating Cloned Products to a Product Line

FOSD meeting, March 13-17, 2017

Wolfram Fenske,[*] Jens Meinicke,[*,†] **Sandro Schulze**,[*] Steffen Schulze,[*] Gunter Saake[*]

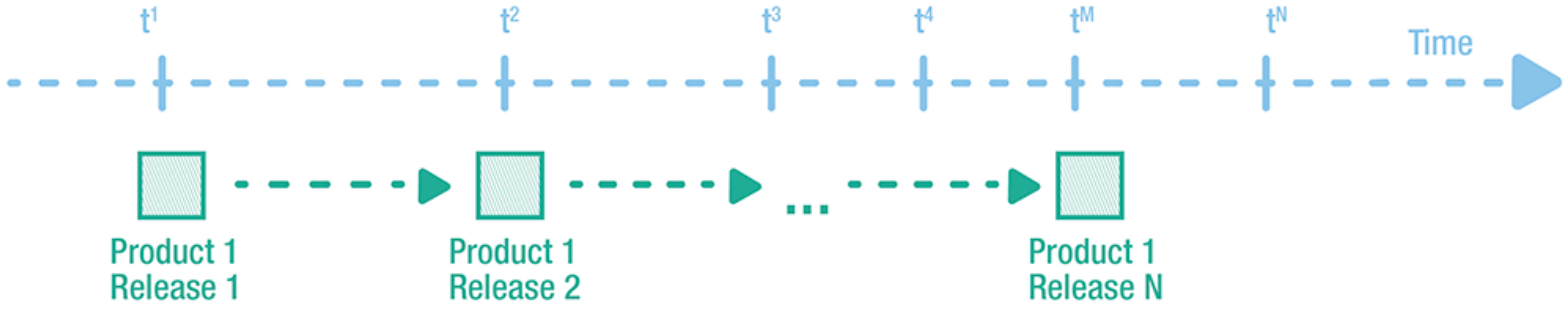
[*] University of Magdeburg, Germany
[†] Carnegie Mellon University, USA

@sanschul

sanschul@ovgu.de

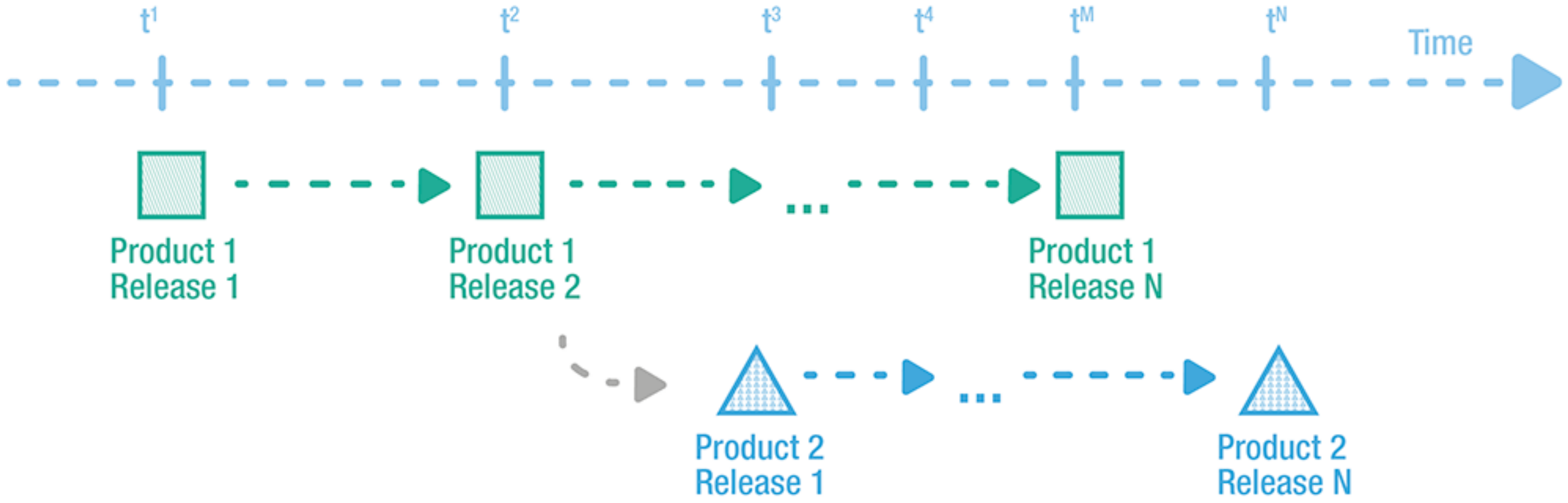
## Motivation: Creating Variants via Clone & Own

- Clone & own (C&O): Copy an existing product and adapt it to new requirements



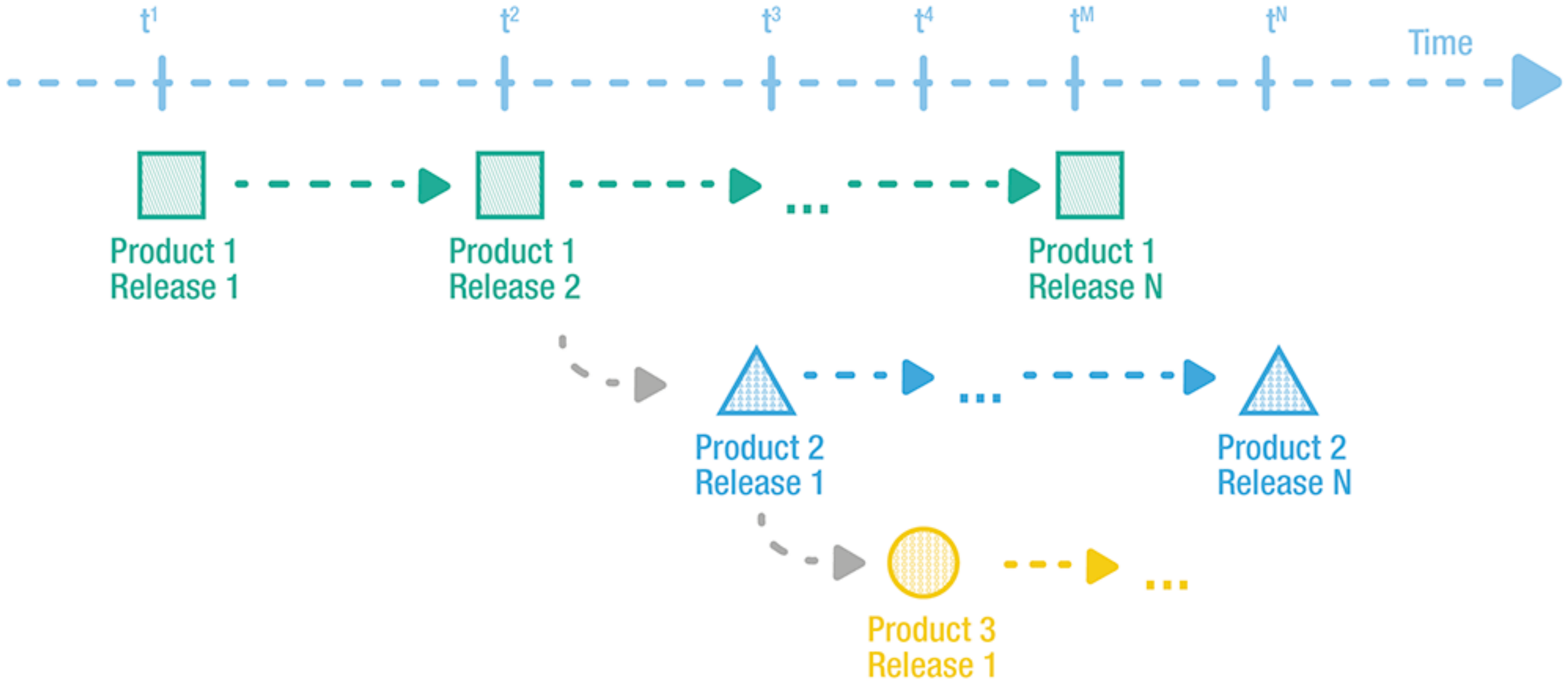
## Motivation: Creating Variants via Clone & Own

- Clone & own (C&O): Copy an existing product and adapt it to new requirements



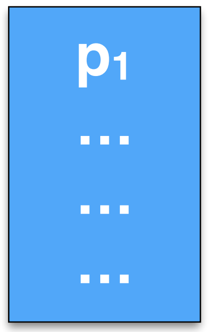
## Motivation: Creating Variants via Clone & Own

- Clone & own (C&O): Copy an existing product and adapt it to new requirements



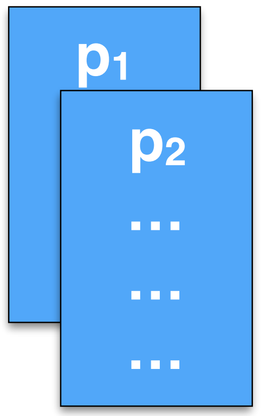
## Unsystematic vs. Systematic Reuse

"Clone & Own" Variant Development



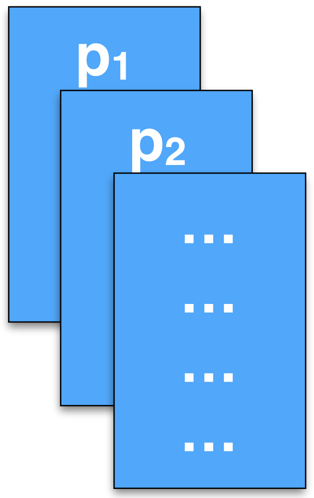
## Unsystematic vs. Systematic Reuse

“Clone & Own” Variant
Development



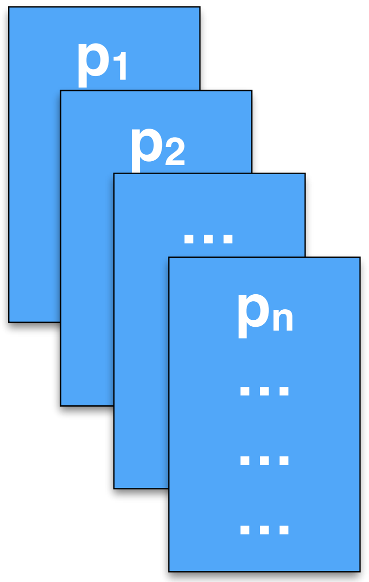
## Unsystematic vs. Systematic Reuse

“Clone & Own” Variant Development



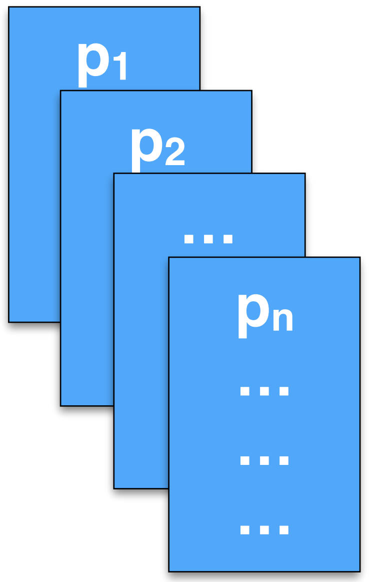
# Unsystematic vs. Systematic Reuse

"Clone & Own" Variant Development



## Unsystematic vs. Systematic Reuse

"Clone & Own" Variant Development

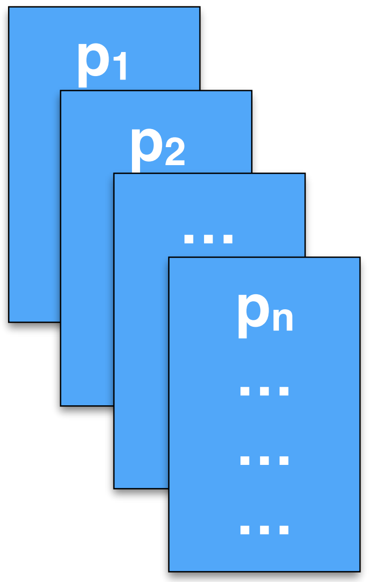


+Initially cheap and easy

- Lack of traceability
- High synchronization effort
- High maintenance & evolution costs

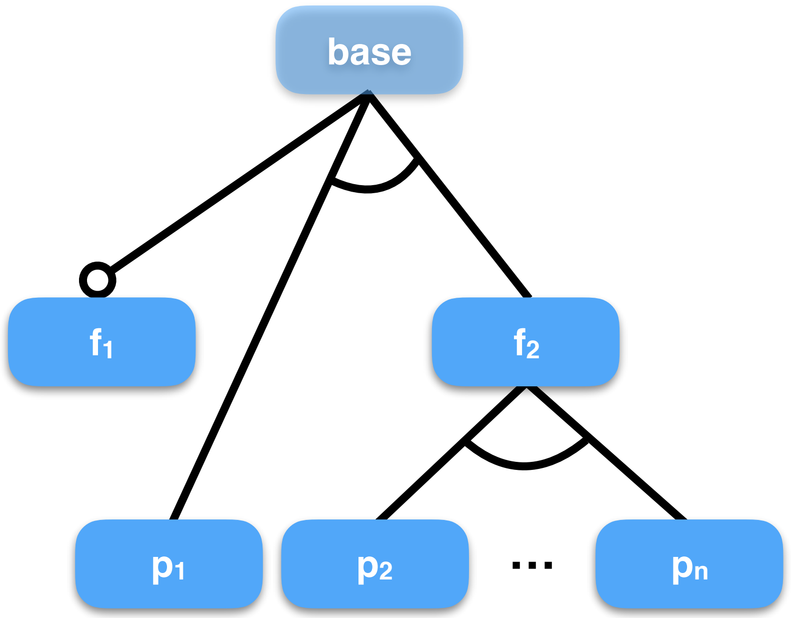
# Unsystematic vs. Systematic Reuse

“Clone & Own” Variant Development



**vs.**

Software Product Line (SPL) Development

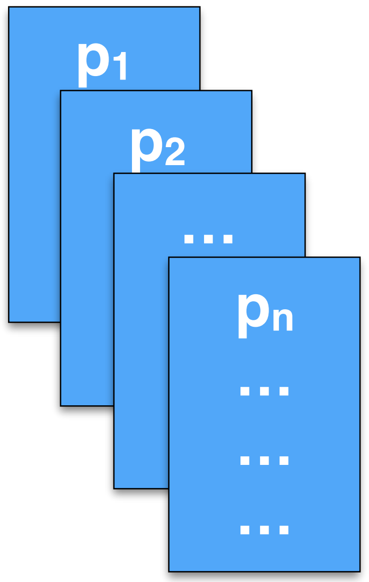


+Initially cheap and easy

- Lack of traceability
- High synchronization effort
- High maintenance & evolution costs

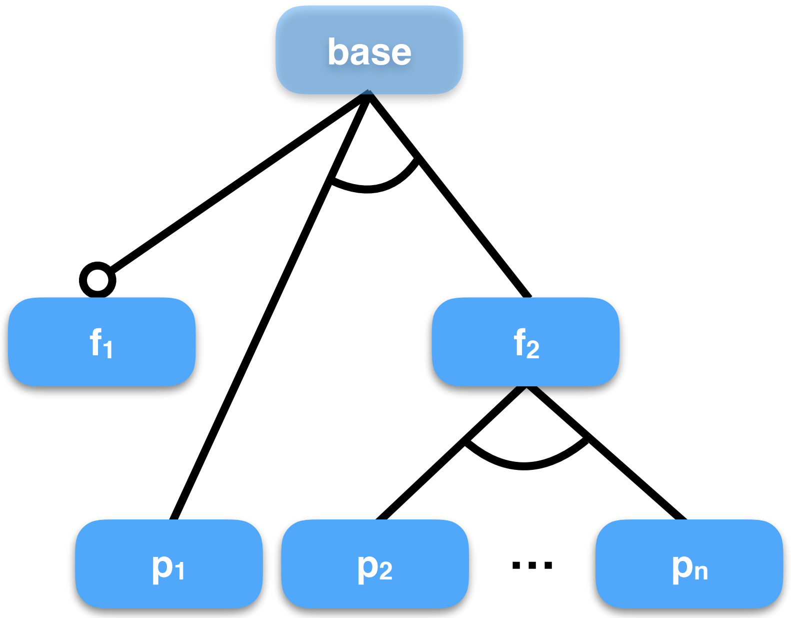
# Unsystematic vs. Systematic Reuse

“Clone & Own” Variant Development



vs.

Software Product Line (SPL) Development



+Initially cheap and easy

- Lack of traceability

- High synchronization effort

- High maintenance & evolution costs

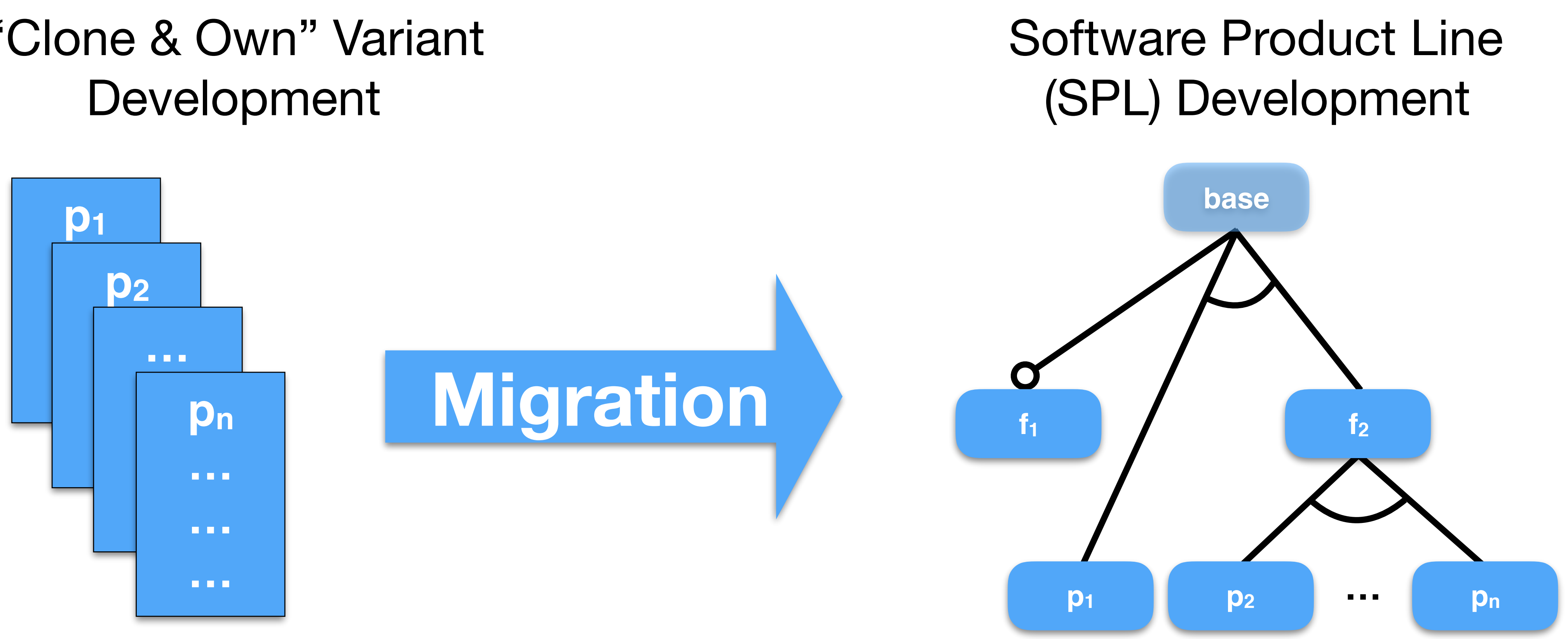
- High initial costs

+Improved traceability

+Minimal synchronization costs

+Low maintenance & evolution costs

## Unsystematic vs. Systematic Reuse


"Clone & Own" Variant Development
p1
p2
...
pn
...
...
...
Migration
Software Product Line (SPL) Development
base
f1
f2
p1
p2
...
pn

+Initially cheap and easy

- Lack of traceability

- High synchronization effort

- High maintenance & evolution costs

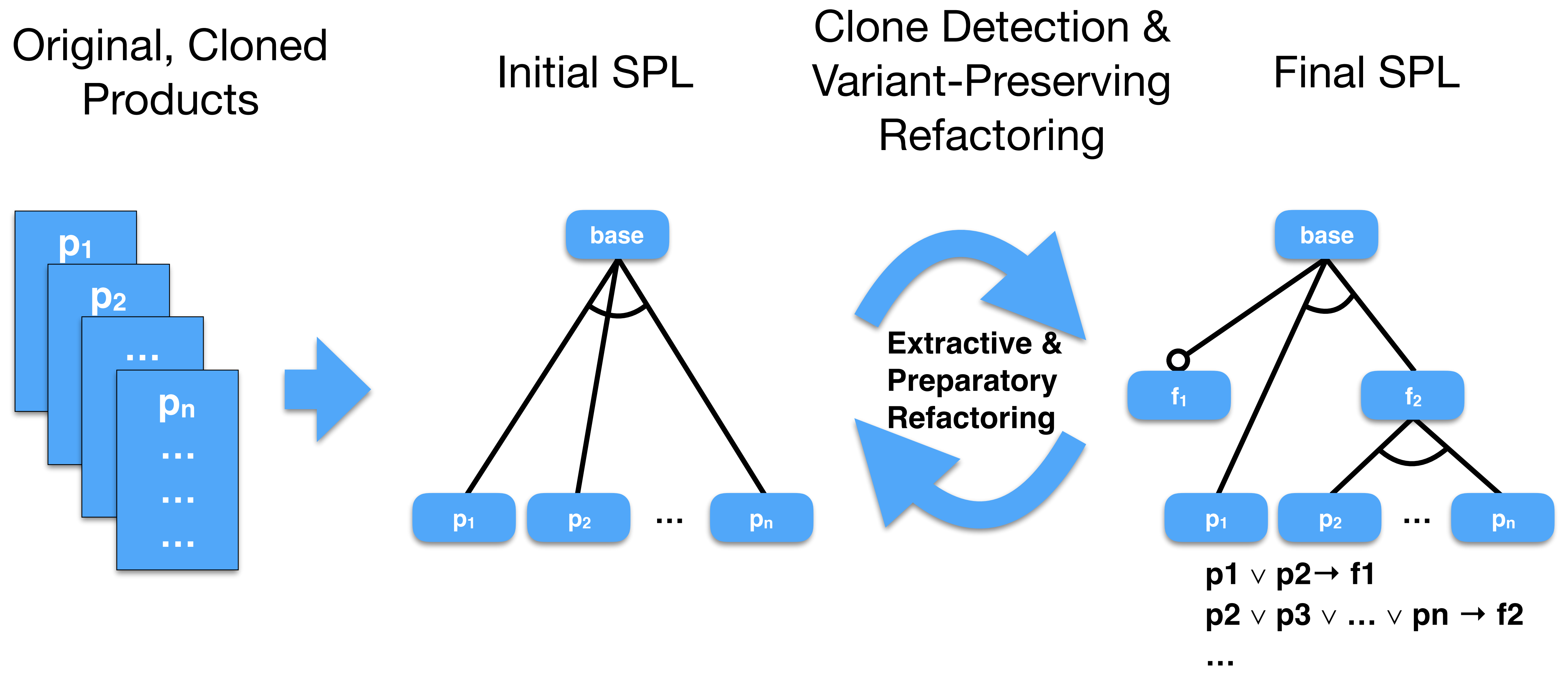
- High initial costs

+Improved traceability

+Minimal synchronization costs

+Low maintenance & evolution costs

# Proposed Step-Wise Migration Process



Original, Cloned Products
Initial SPL
Clone Detection & Variant-Preserving Refactoring
Final SPL
p1
p2
...
pn
...
...
...
base
p1
p2
...
pn
Extractive & Preparatory Refactoring
base
f1
f2
p1
p2
...
pn

$p1 \vee p2 \rightarrow f1$

$p2 \vee p3 \vee \ldots \vee pn \rightarrow f2$

...

**Configurations:**
**C1 = {p1}**
**C2 = {p2}**
**...**
**Cn = {pn}**

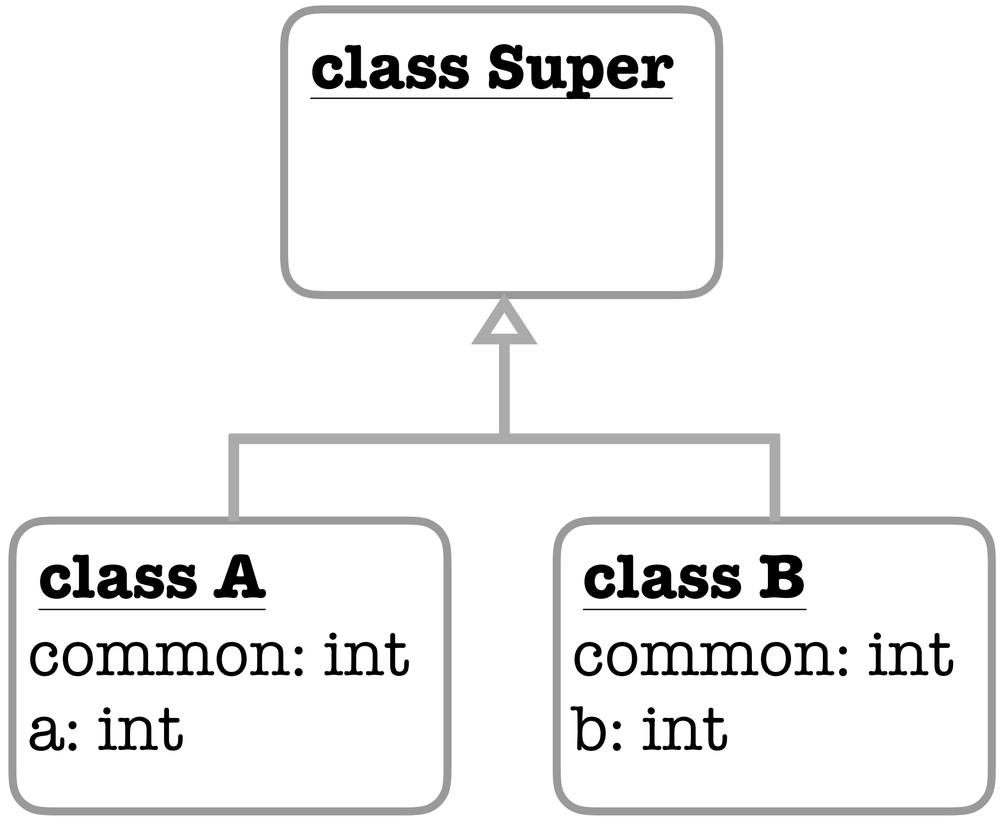
**Configurations:**
**C1 = {f1, p1}**
**C2 = {f1, f2, p2}**
**...**
**Cn = {f2, pn}**

# Key Points of Migration Process

- Step-wise: help w/ the time-consuming, error-prone tasks but leave big design decisions to developers
- Variant-preserving refactoring *[Schulze et al., VaMoS '12]* for clone consolidation
- Preparatory refactoring to align divergent clones
- Feature-oriented programming (FOP) as the variability mechanism
- Integrated tool support

# Refactorings — Clone Consolidation via "Pull Up To Common Feature"

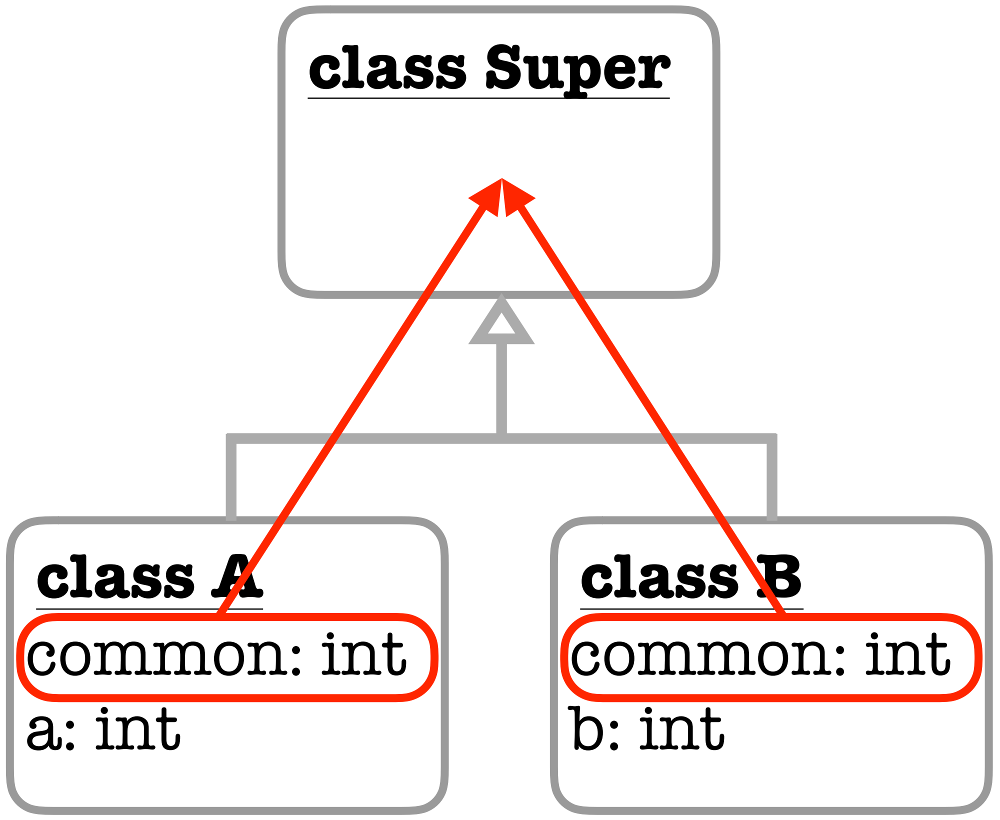
Pull Up (OOP)



Move code within the ***inheritance*** hierarchy

# Refactorings — Clone Consolidation via "Pull Up To Common Feature"

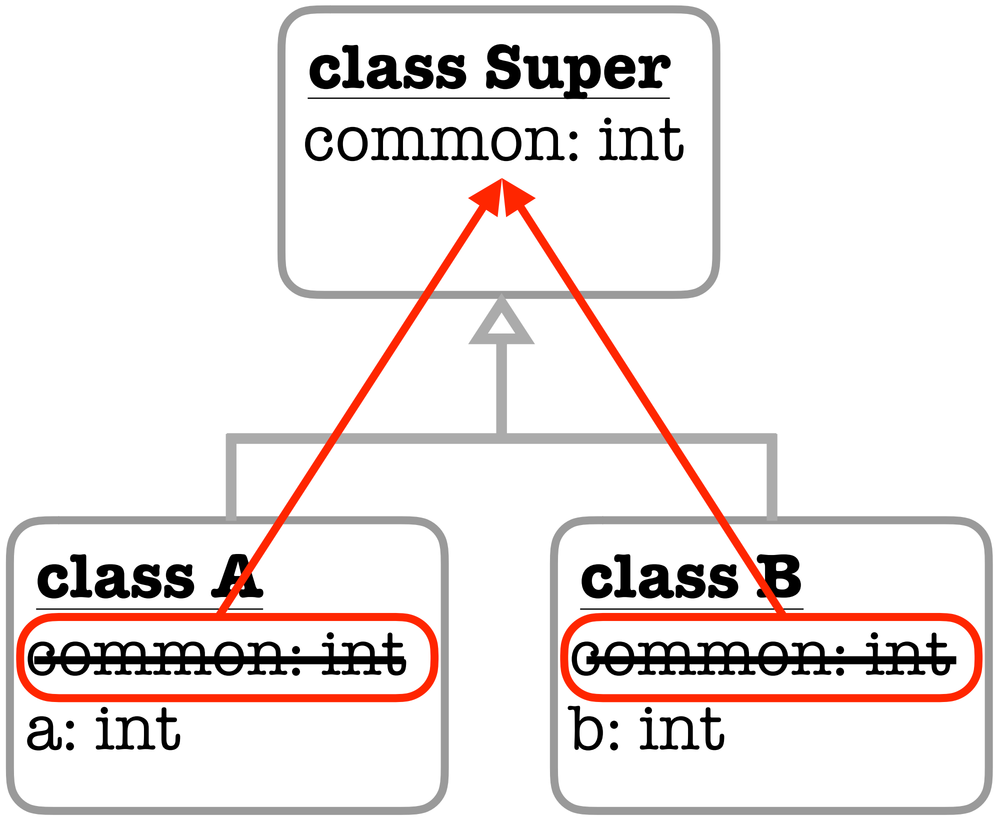
## Pull Up (OOP)



Move code within the ***inheritance*** hierarchy

# Refactorings — Clone Consolidation via “Pull Up To Common Feature”

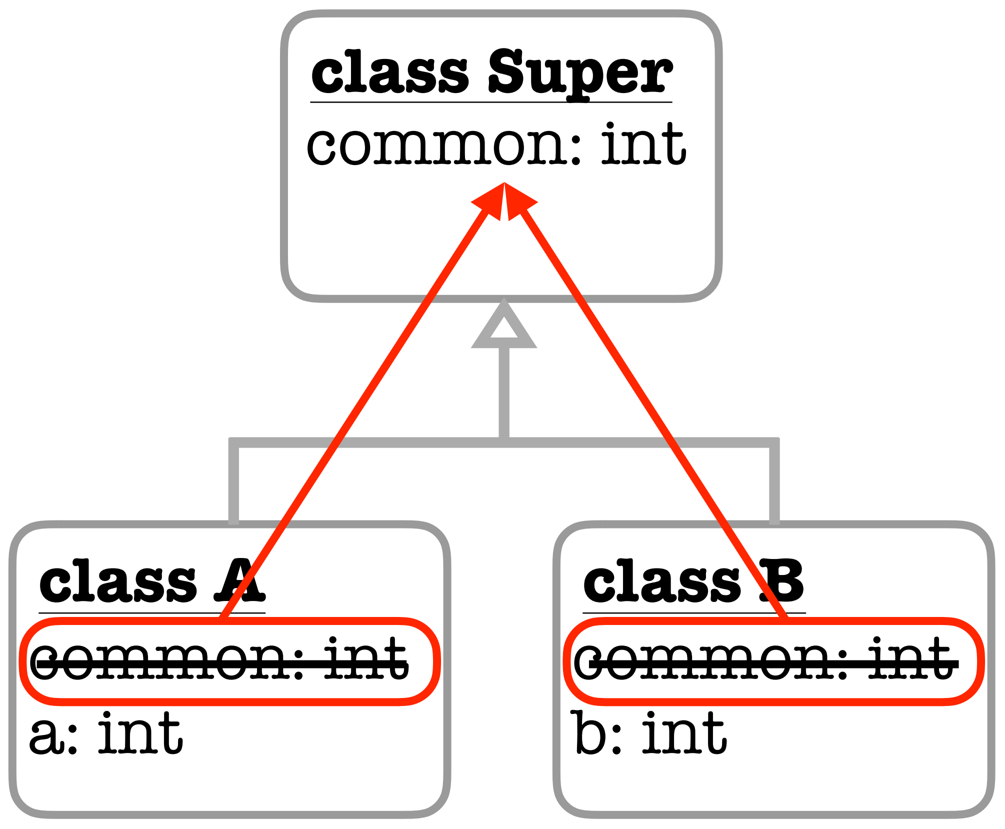
Pull Up (OOP)



Move code within the ***inheritance*** hierarchy

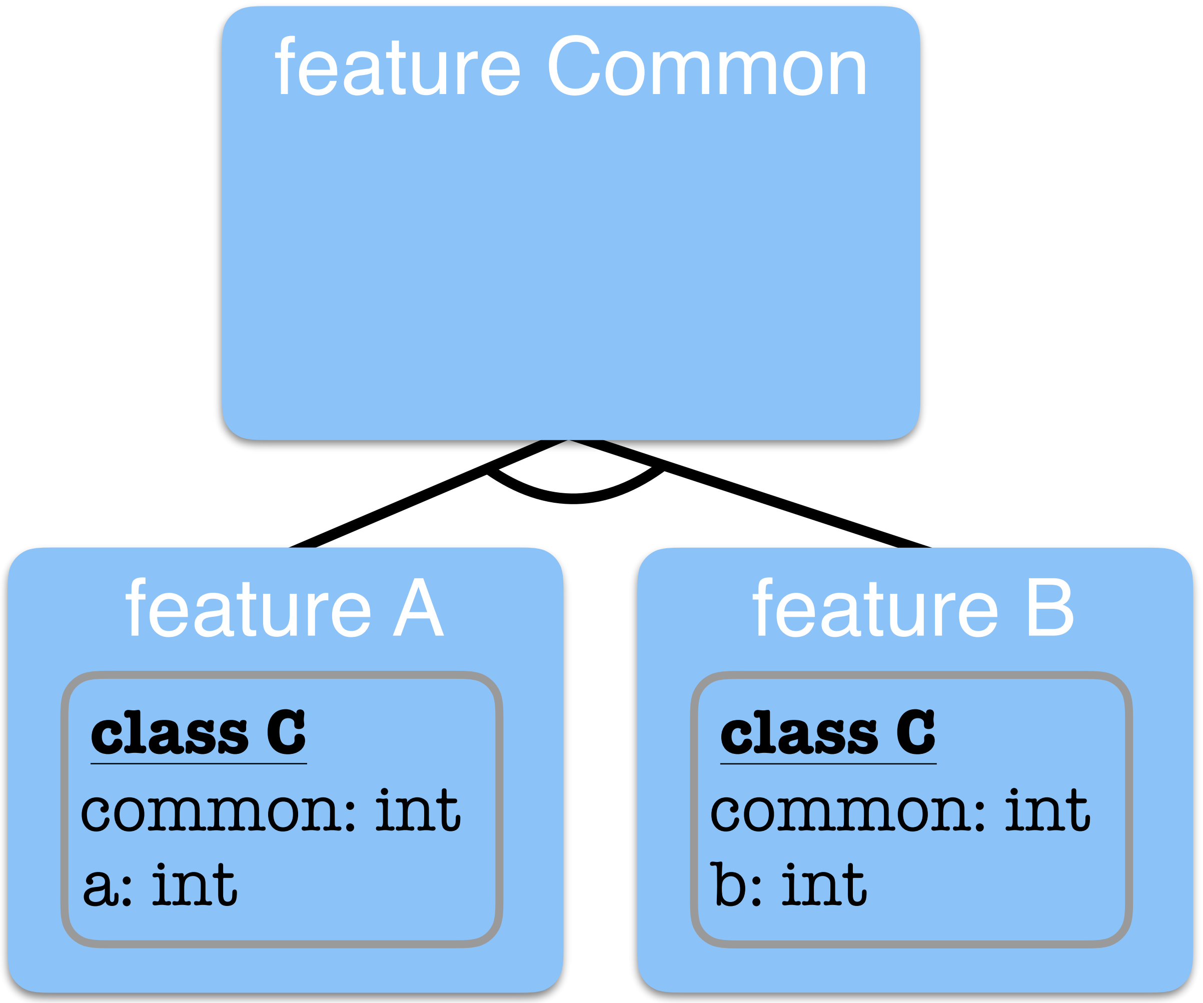
# Refactorings — Clone Consolidation via “Pull Up To Common Feature”

Pull Up (OOP)



Move code within the ***inheritance*** hierarchy

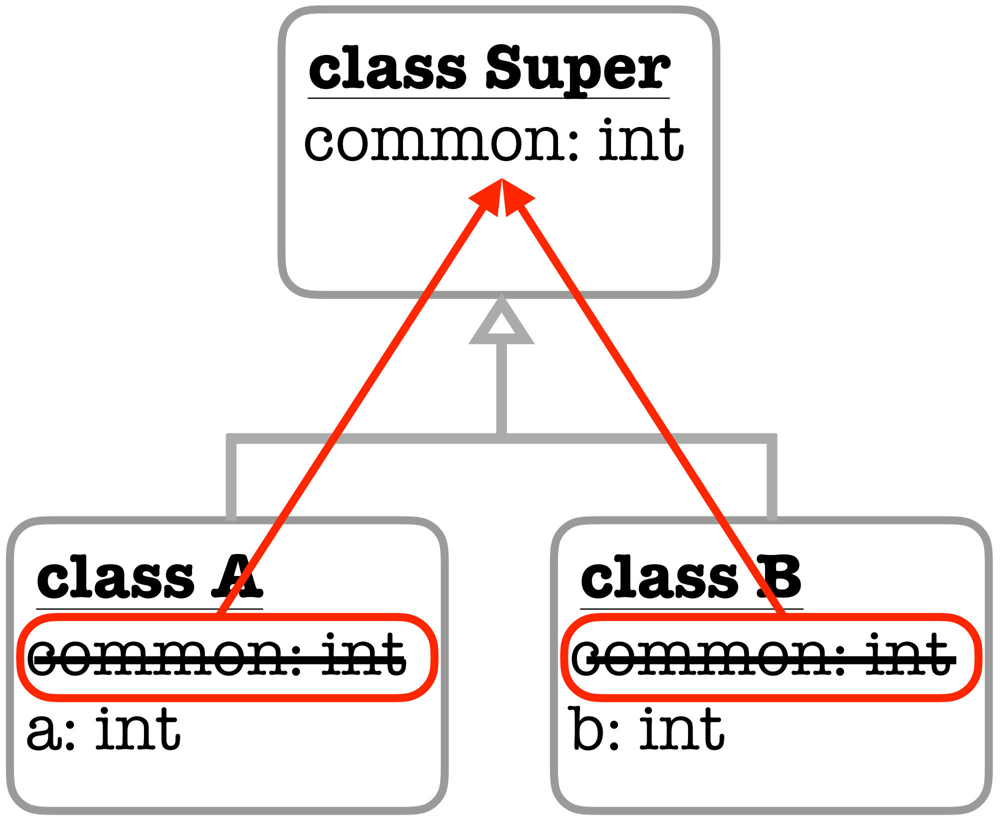
Pull Up To Common Feature (FOP)



Move code within the ***refinement*** hierarchy

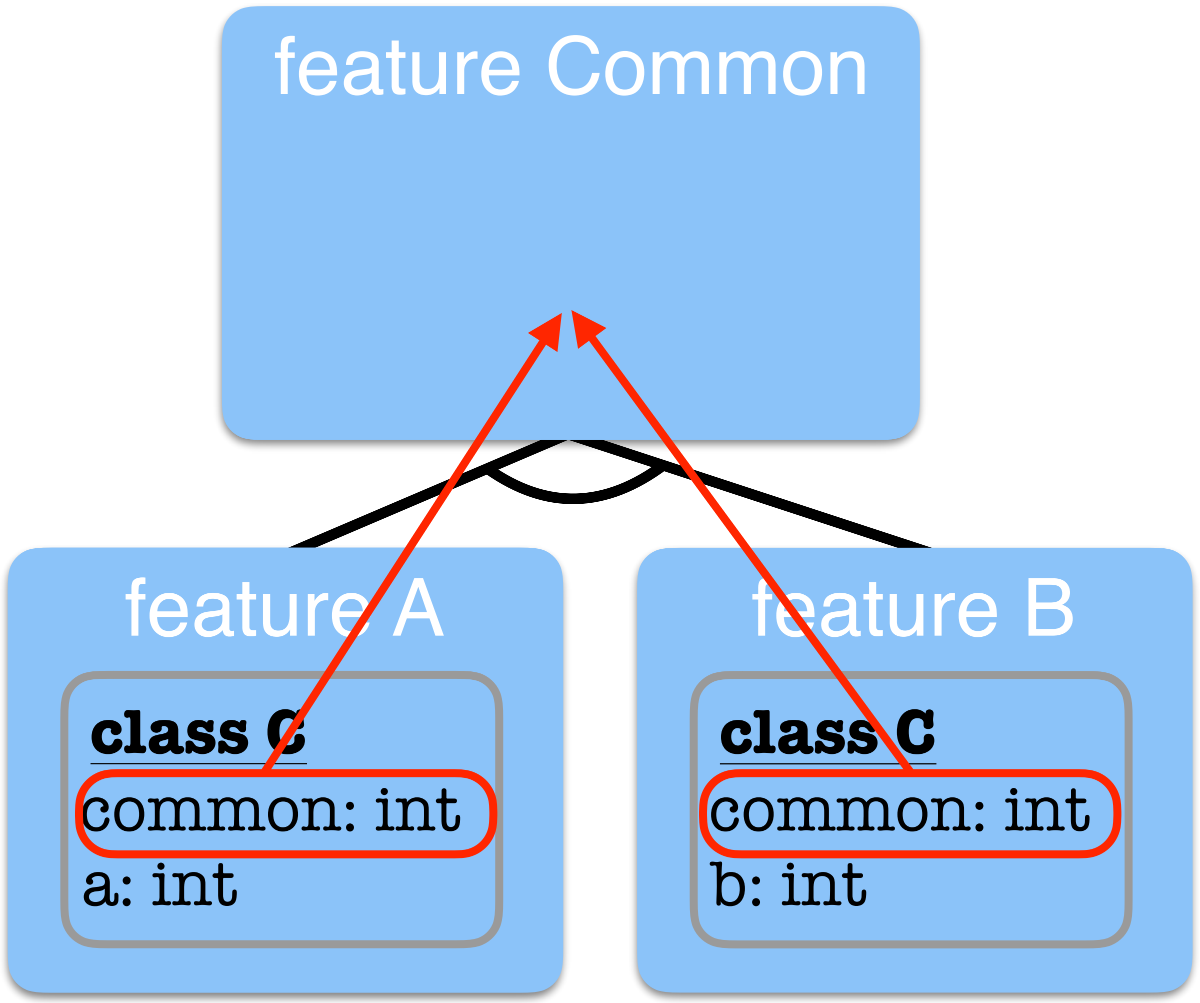
# Refactorings — Clone Consolidation via "Pull Up To Common Feature"

## Pull Up (OOP)



Move code within the ***inheritance*** hierarchy

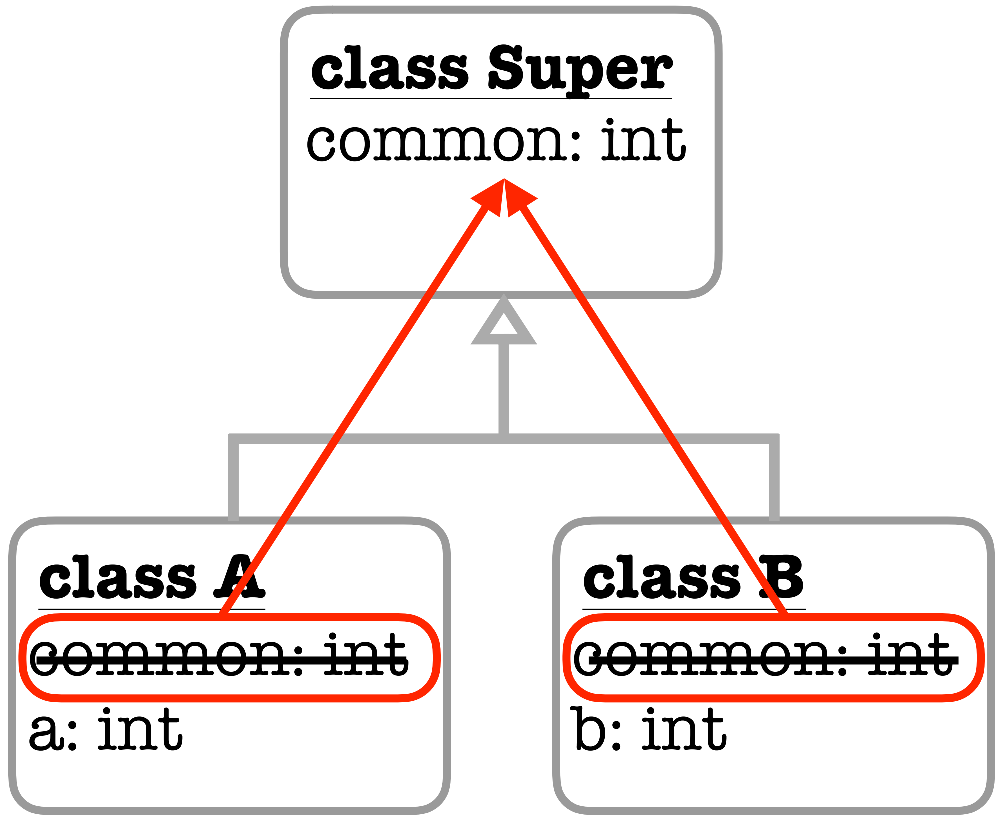
## Pull Up To Common Feature (FOP)



Move code within the ***refinement*** hierarchy

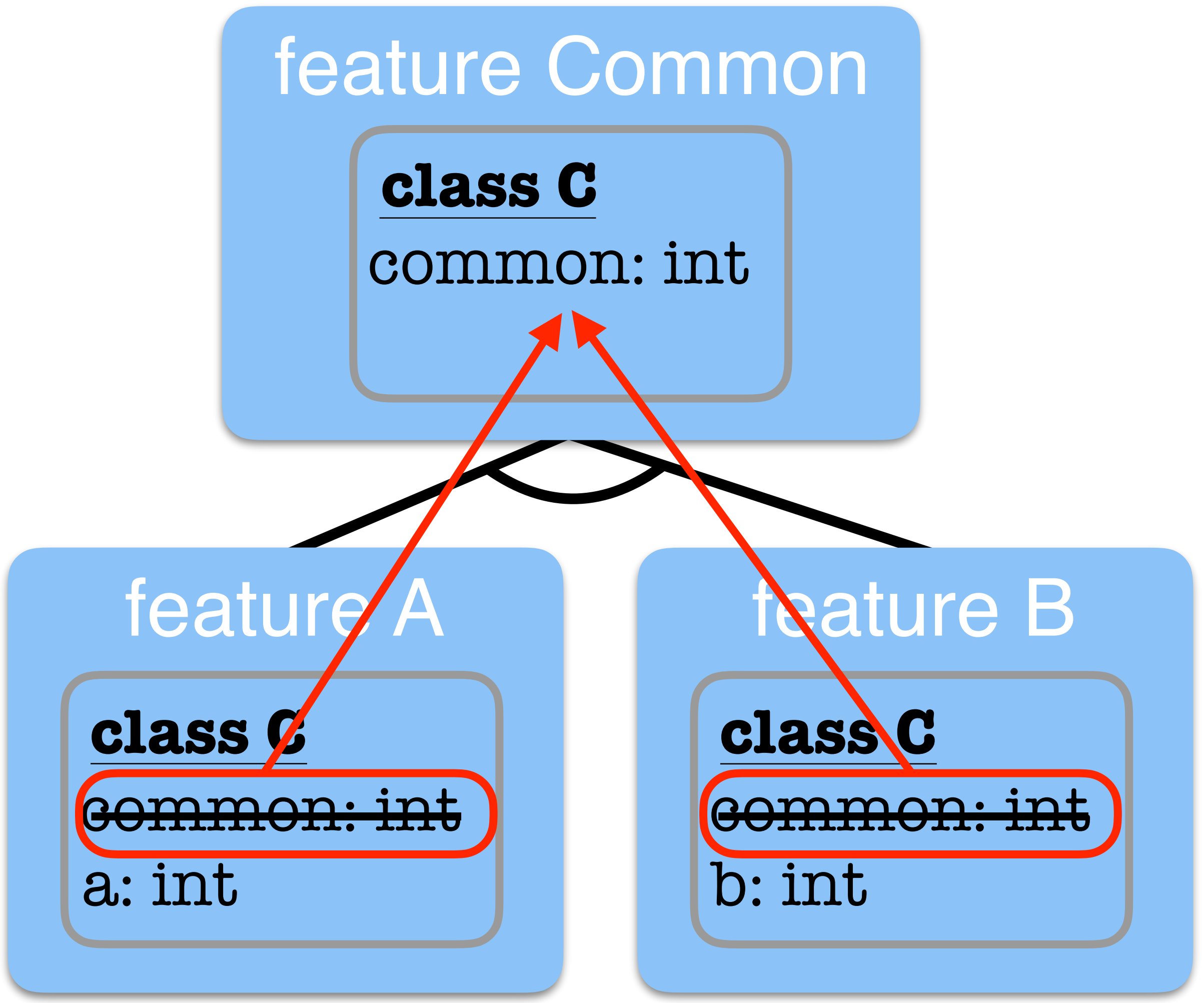
# Refactorings — Clone Consolidation via "Pull Up To Common Feature"

## Pull Up (OOP)



Move code within the ***inheritance*** hierarchy

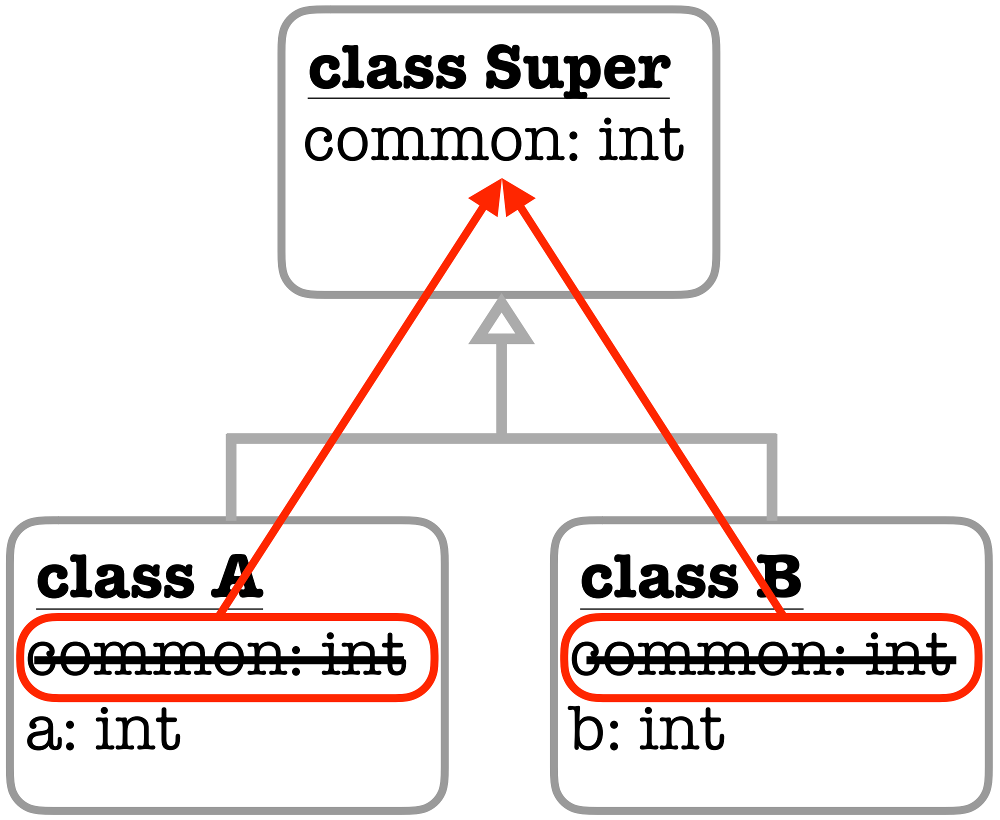
## Pull Up To Common Feature (FOP)



Move code within the ***refinement*** hierarchy

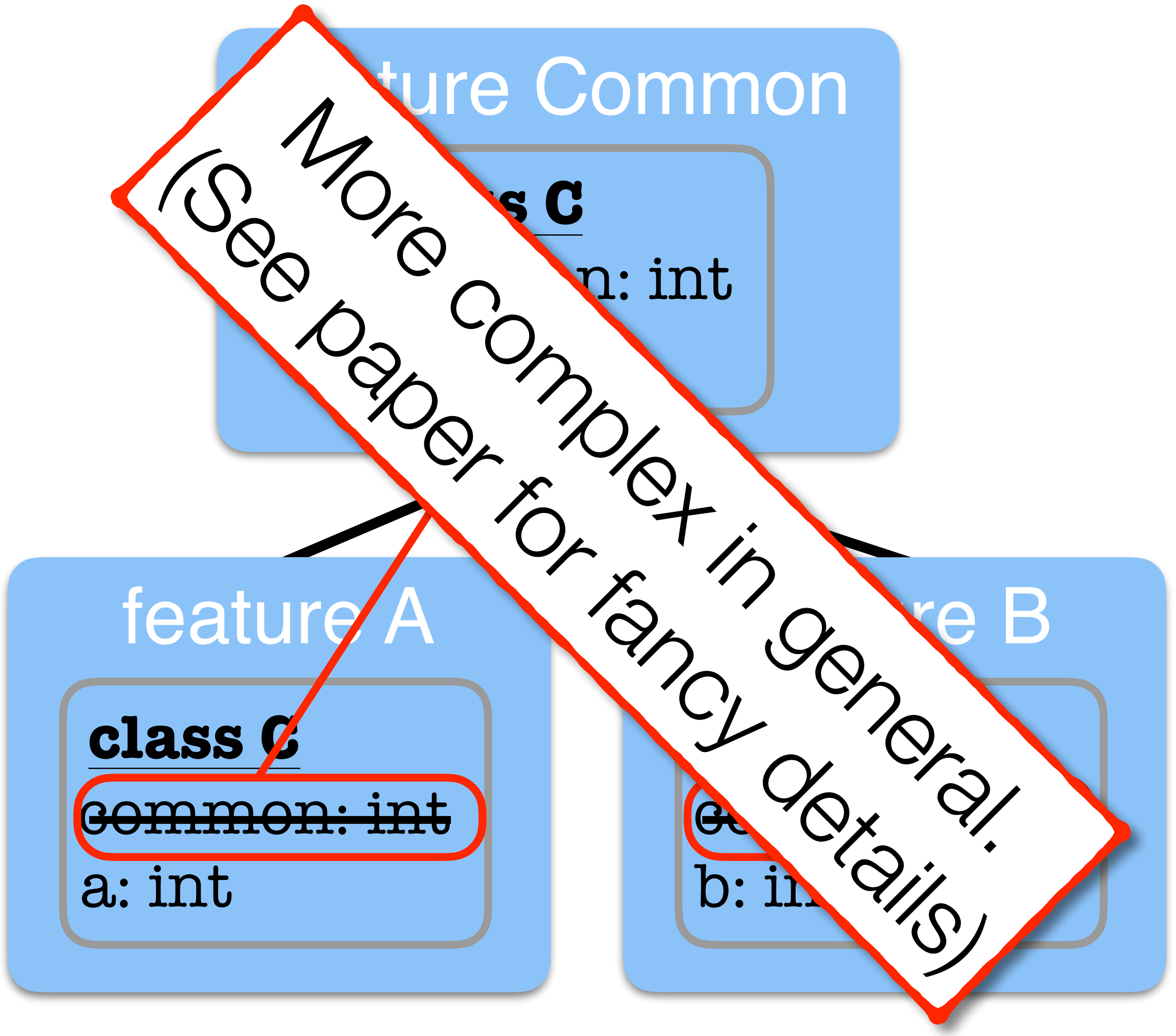
# Refactorings — Clone Consolidation via "Pull Up To Common Feature"

Pull Up (OOP)



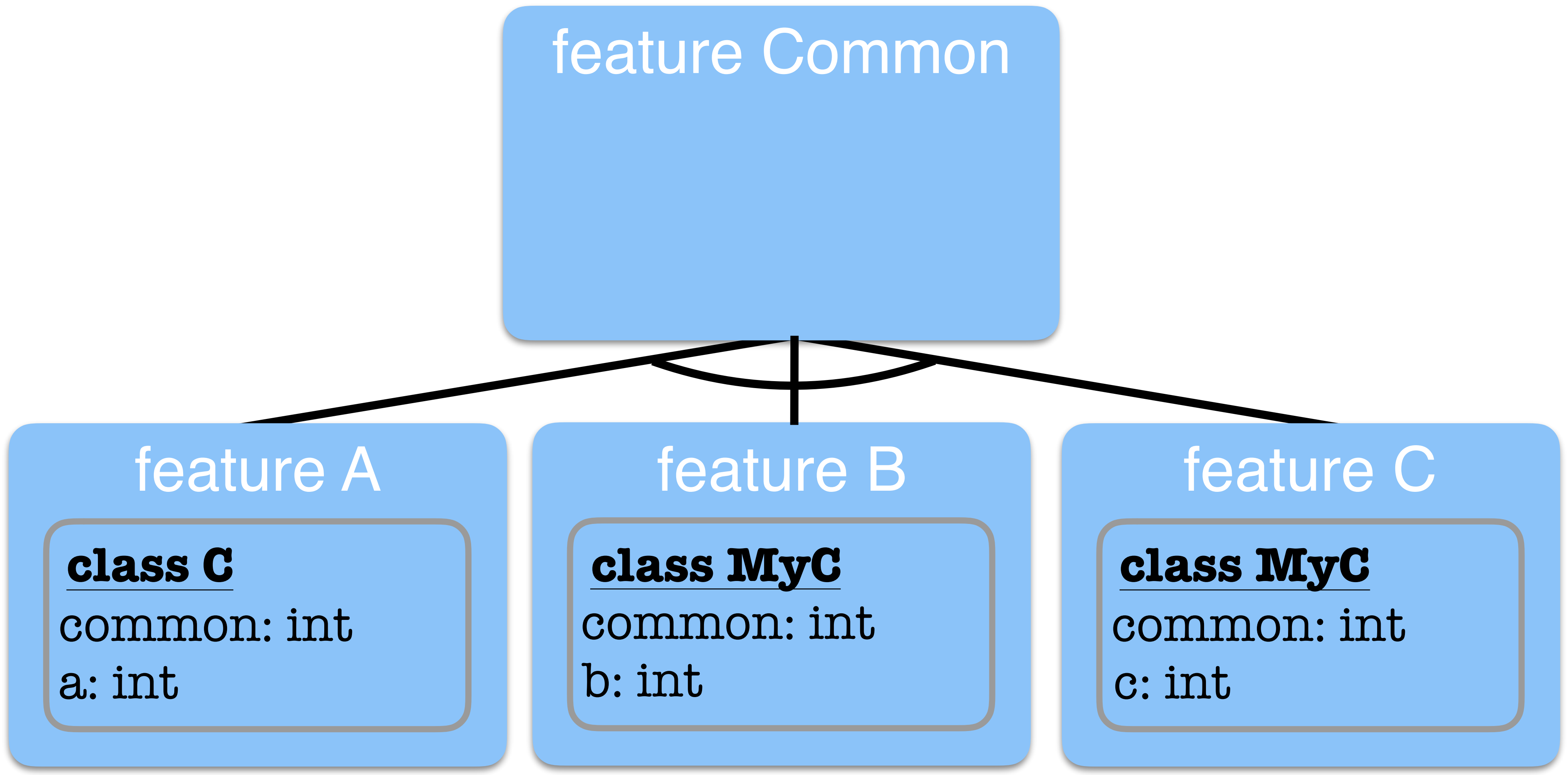
Move code within the *inheritance* hierarchy

Pull Up To Common Feature (FOP)

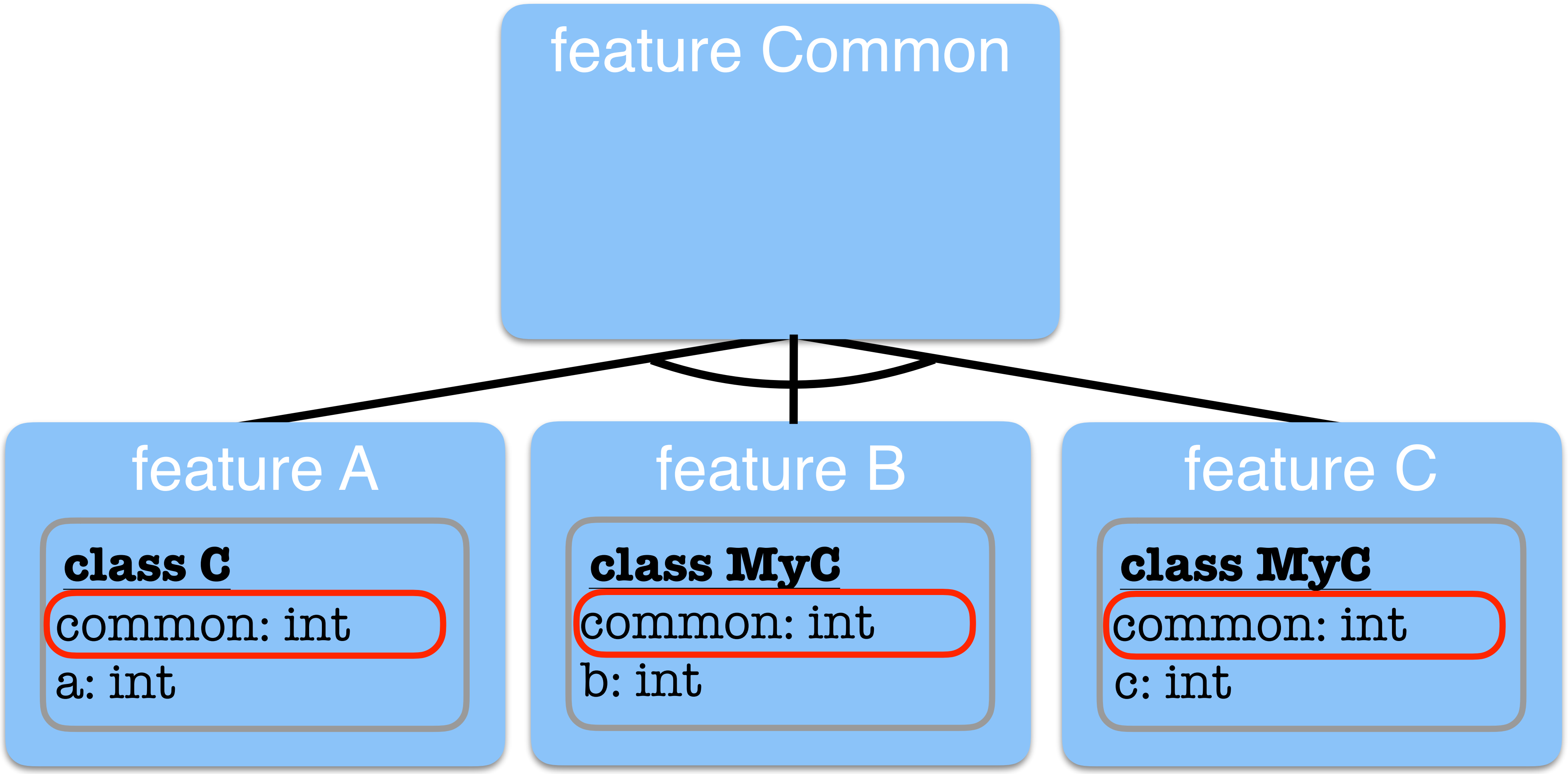


Move code within the *refinement* hierarchy

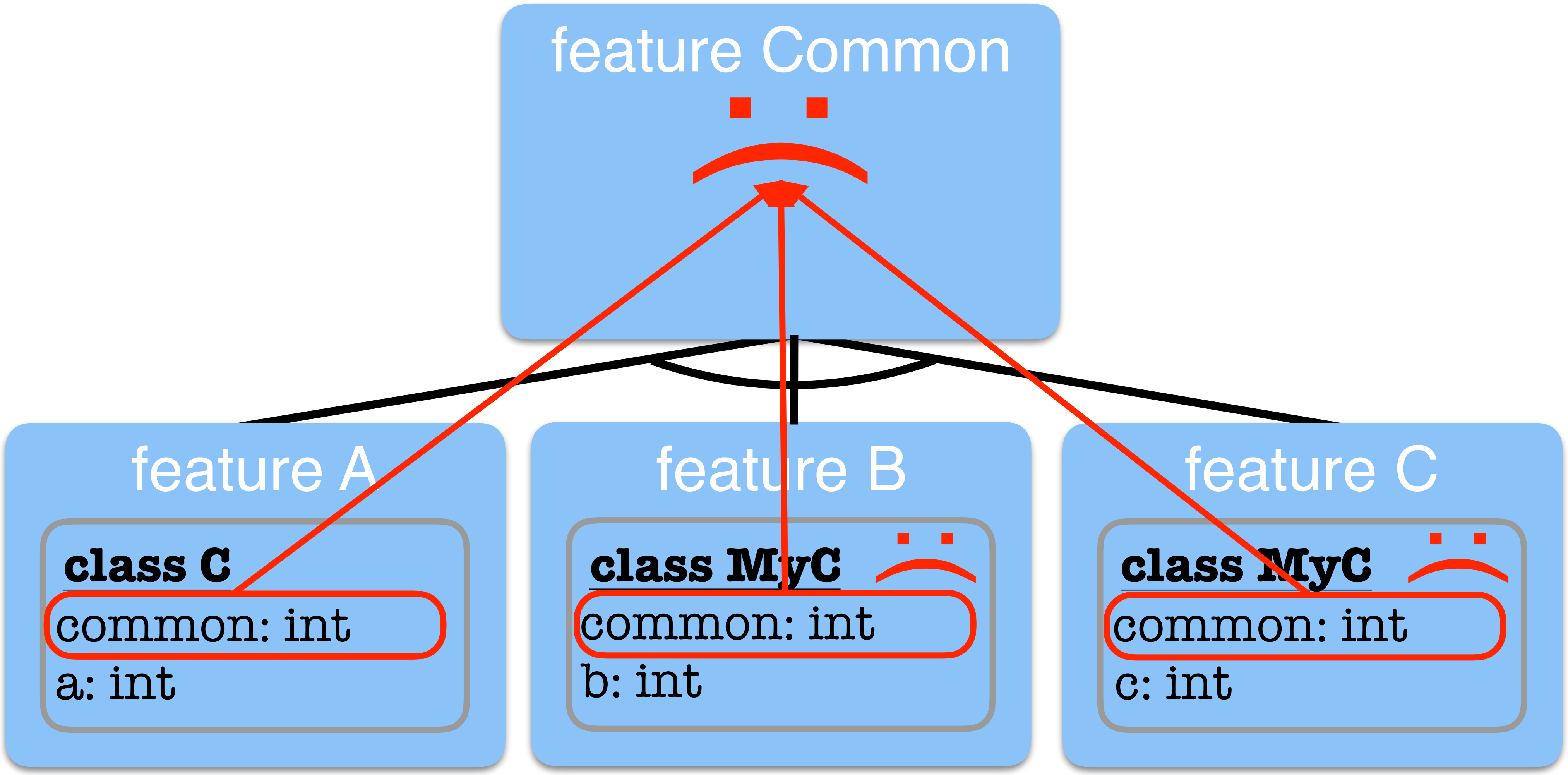
## Refactorings — Aligning Divergencies via "Rename"



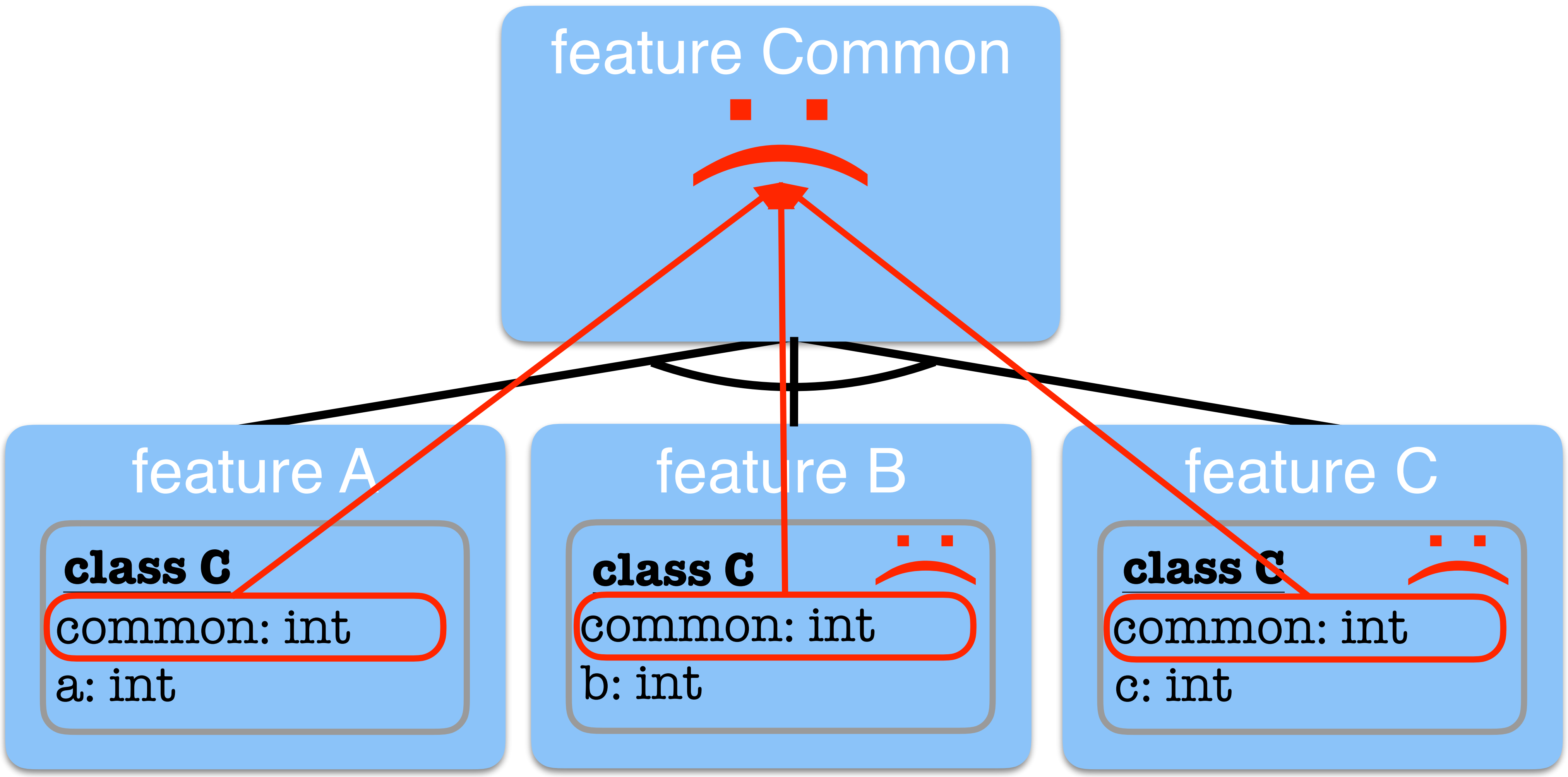
# Refactorings — Aligning Divergencies via "Rename"



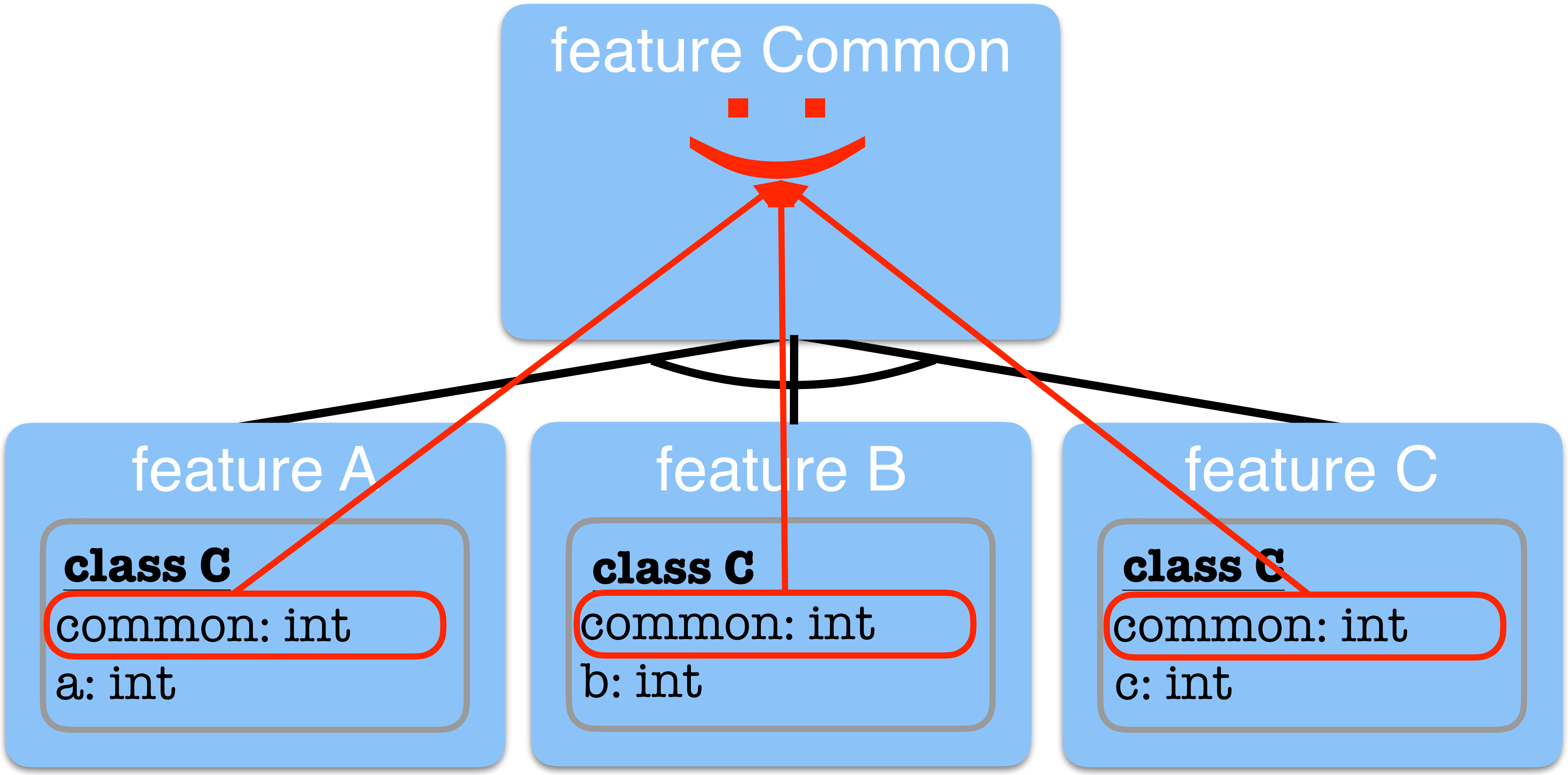
# Refactorings — Aligning Divergencies via “Rename”



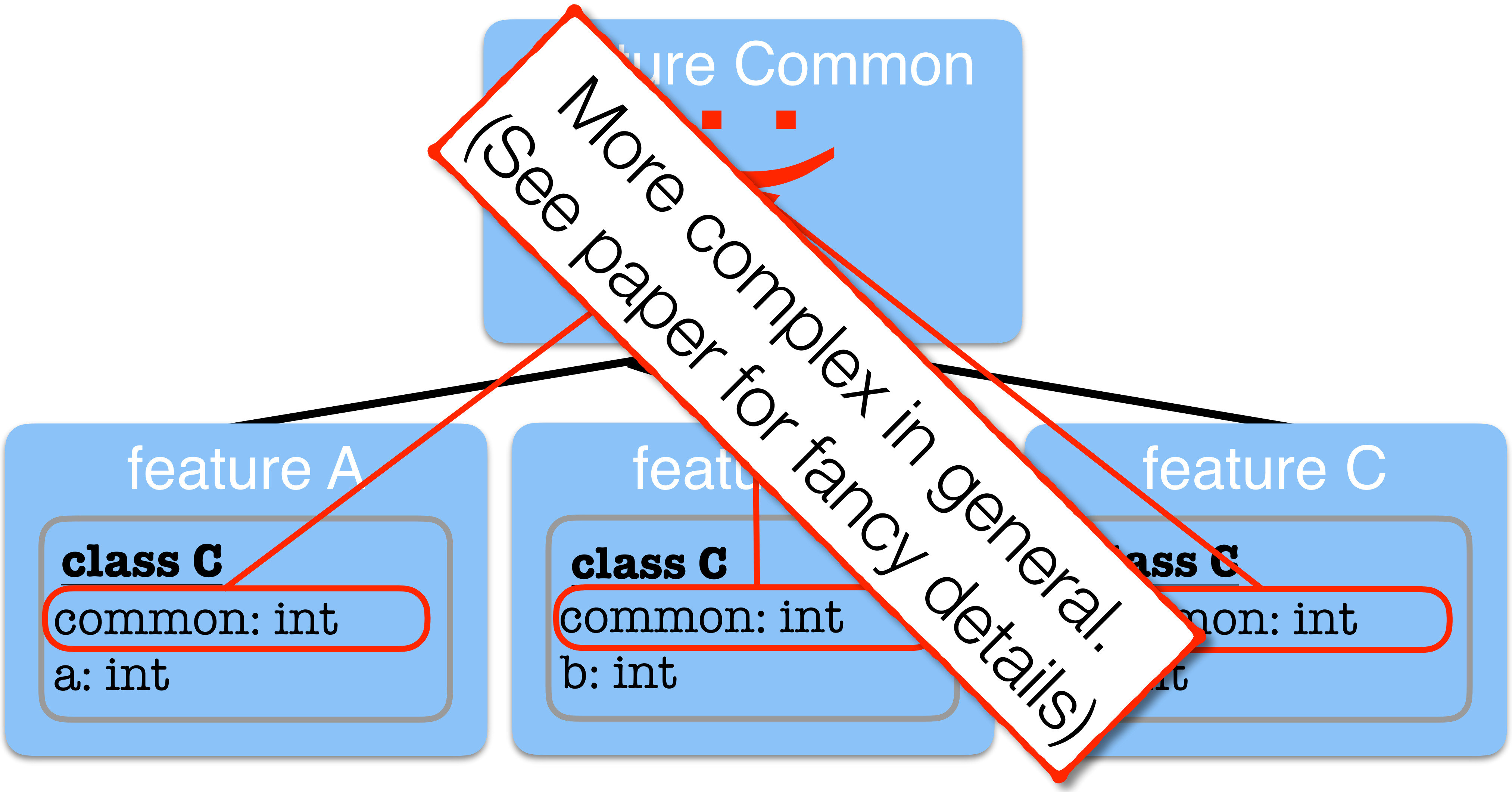
# Refactorings — Aligning Divergencies via "Rename"



## Refactorings — Aligning Divergencies via "Rename"

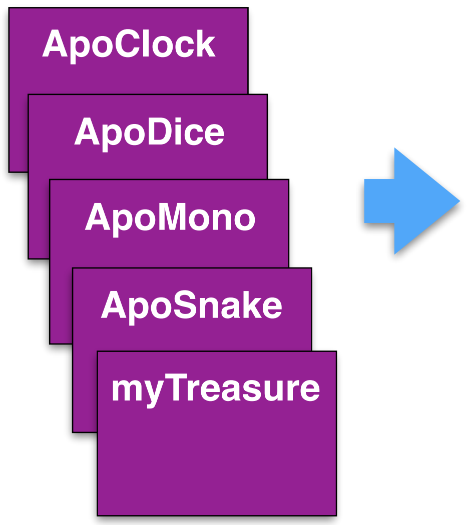


# Refactorings — Aligning Divergencies via "Rename"

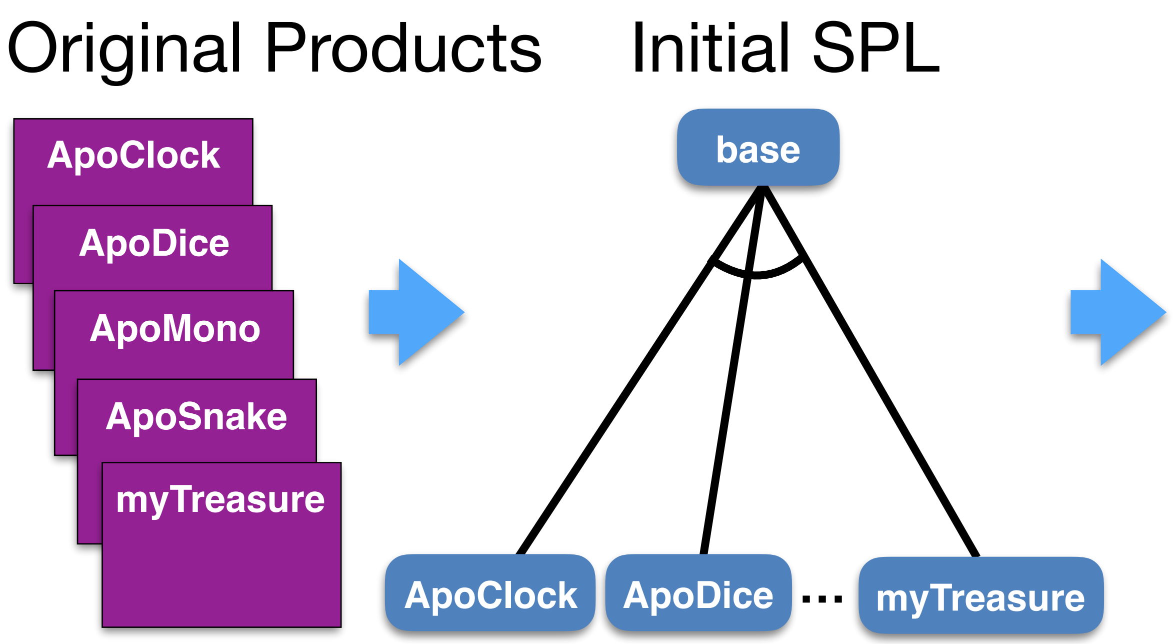


## Evaluation

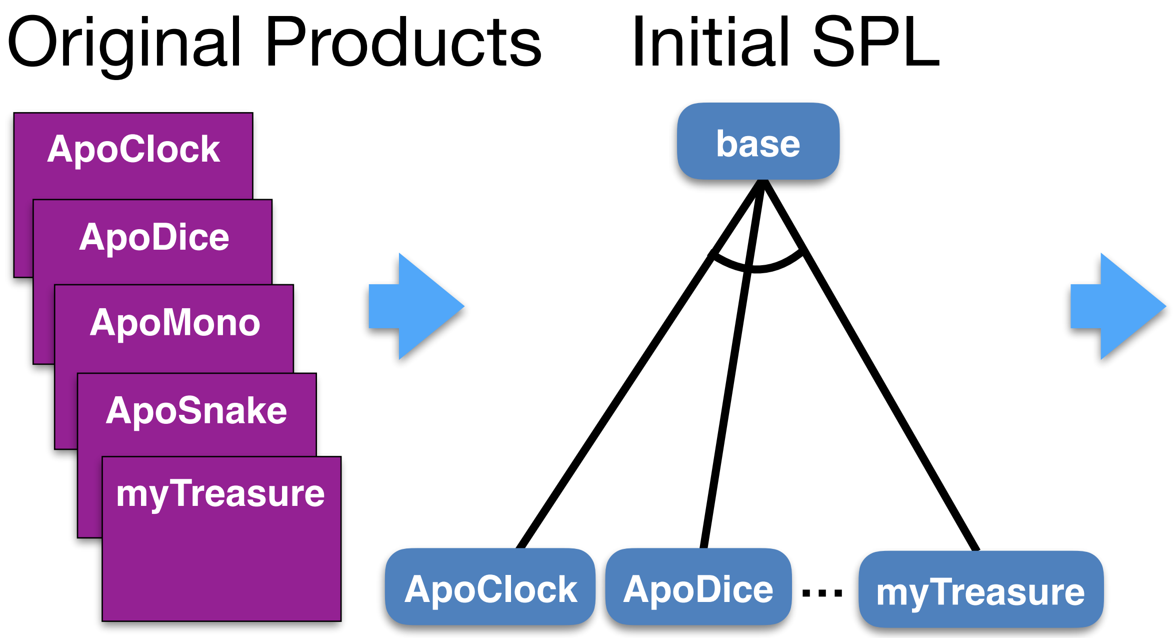
Original Products



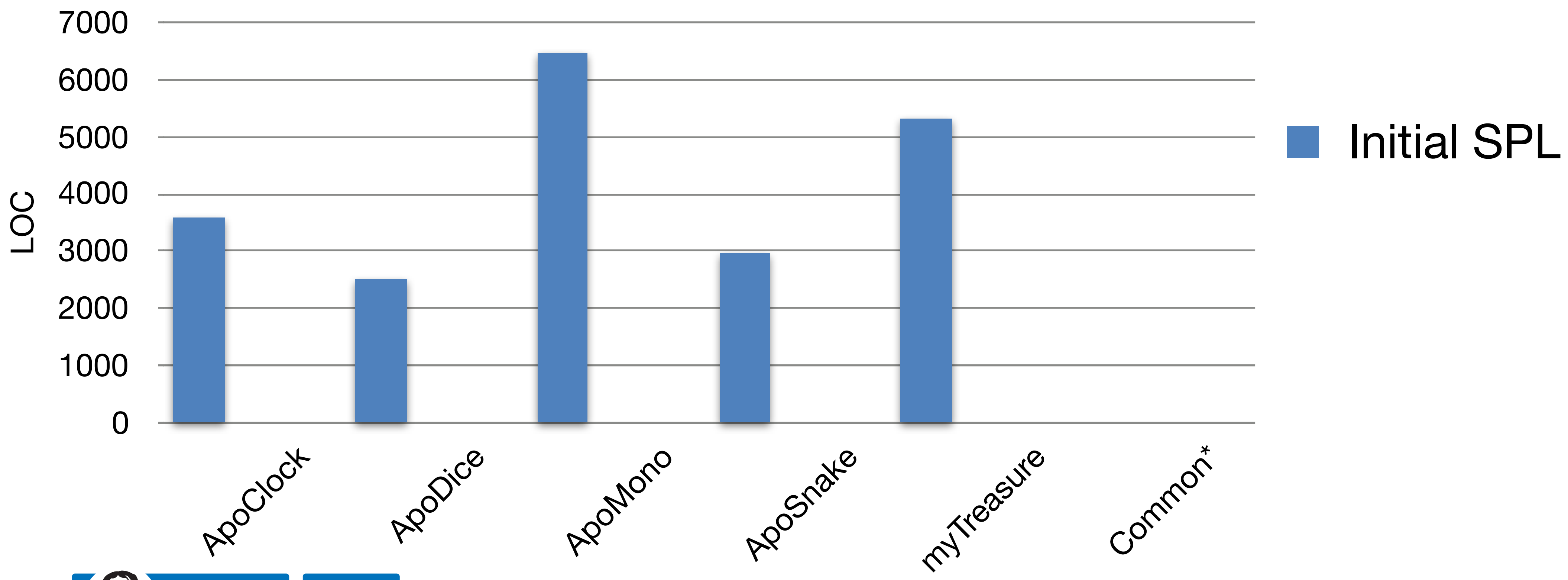
# Evaluation


Original Products
Initial SPL
ApoClock
ApoDice
ApoMono
ApoSnake
myTreasure
base
ApoClock
ApoDice
…
myTreasure

## Evaluation

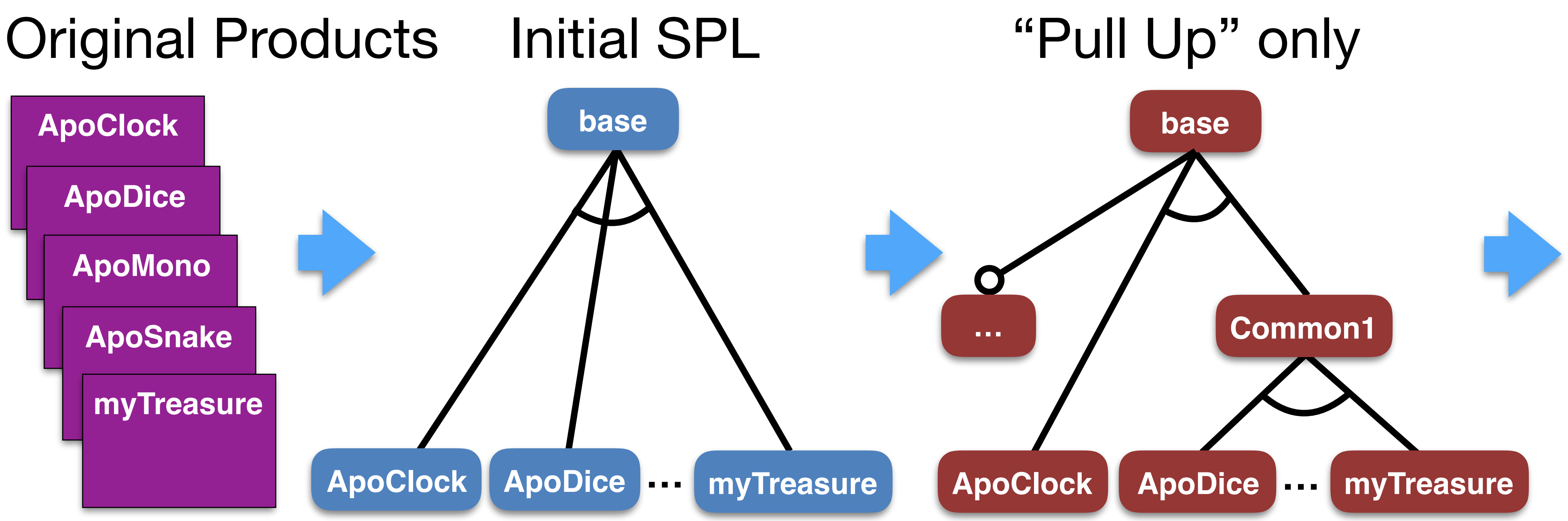


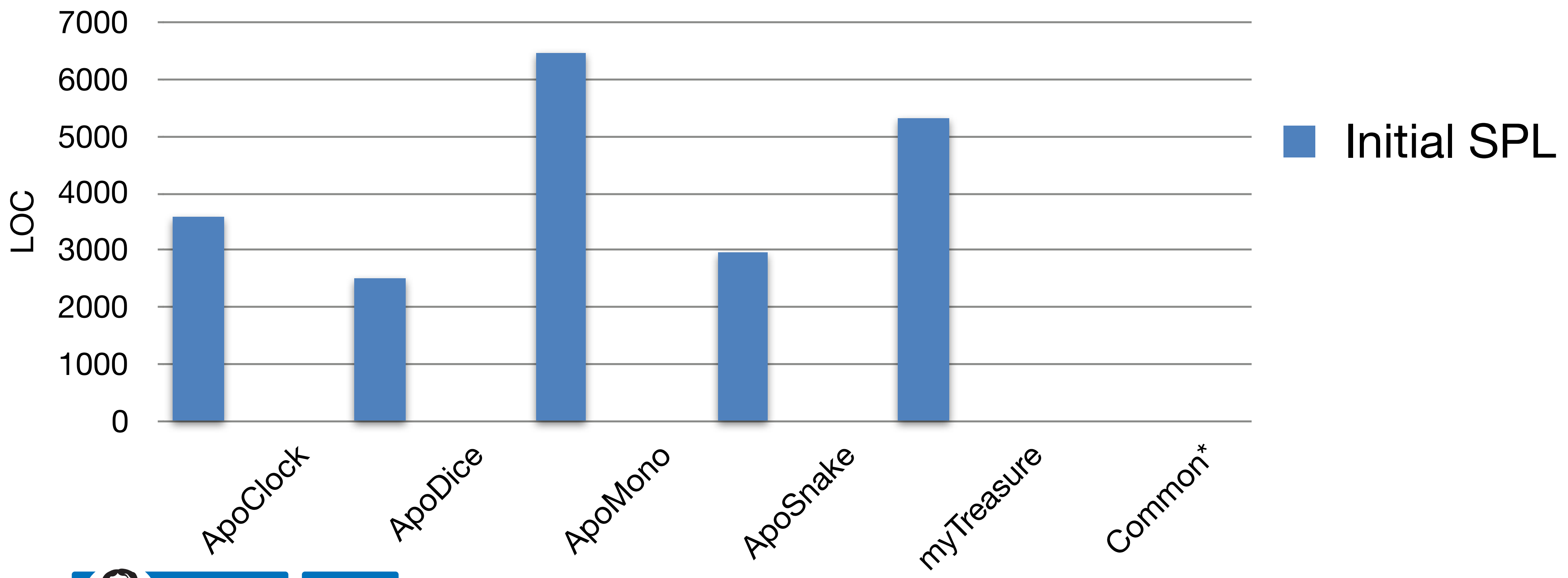
Original Products
Initial SPL
ApoClock
ApoDice
ApoMono
ApoSnake
myTreasure
base
ApoClock
ApoDice
...
myTreasure



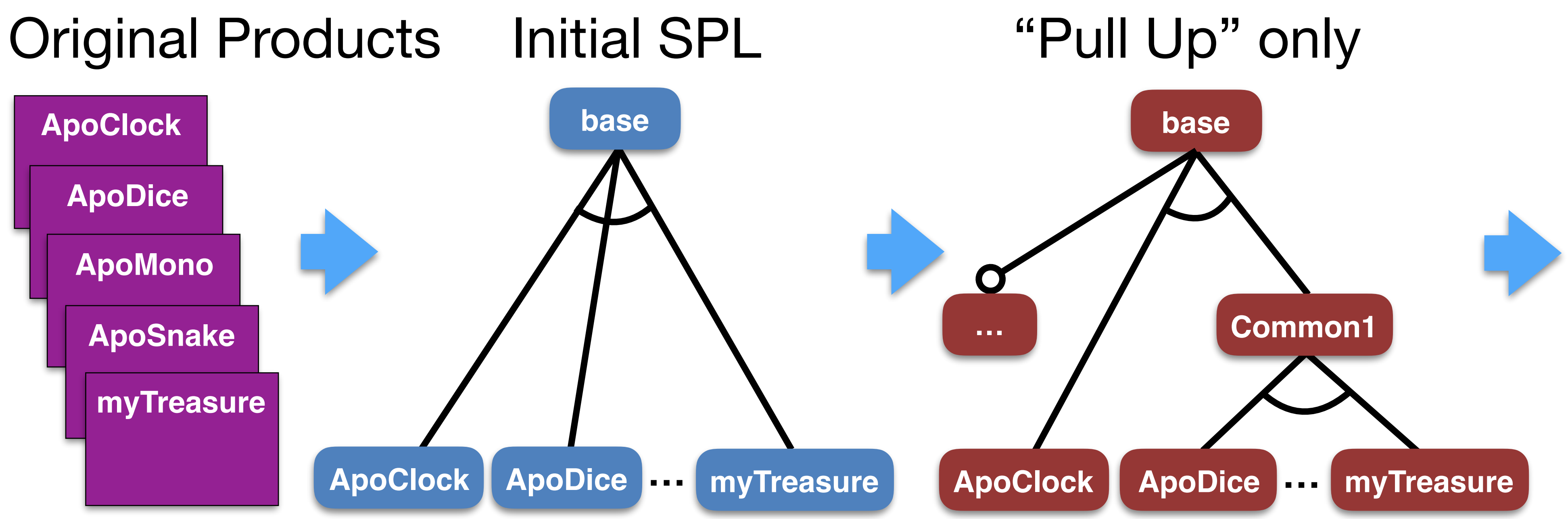
LOC
7000
6000
5000
4000
3000
2000
1000
0
ApoClock
ApoDice
ApoMono
ApoSnake
myTreasure
Common*
■ Initial SPL

## Evaluation

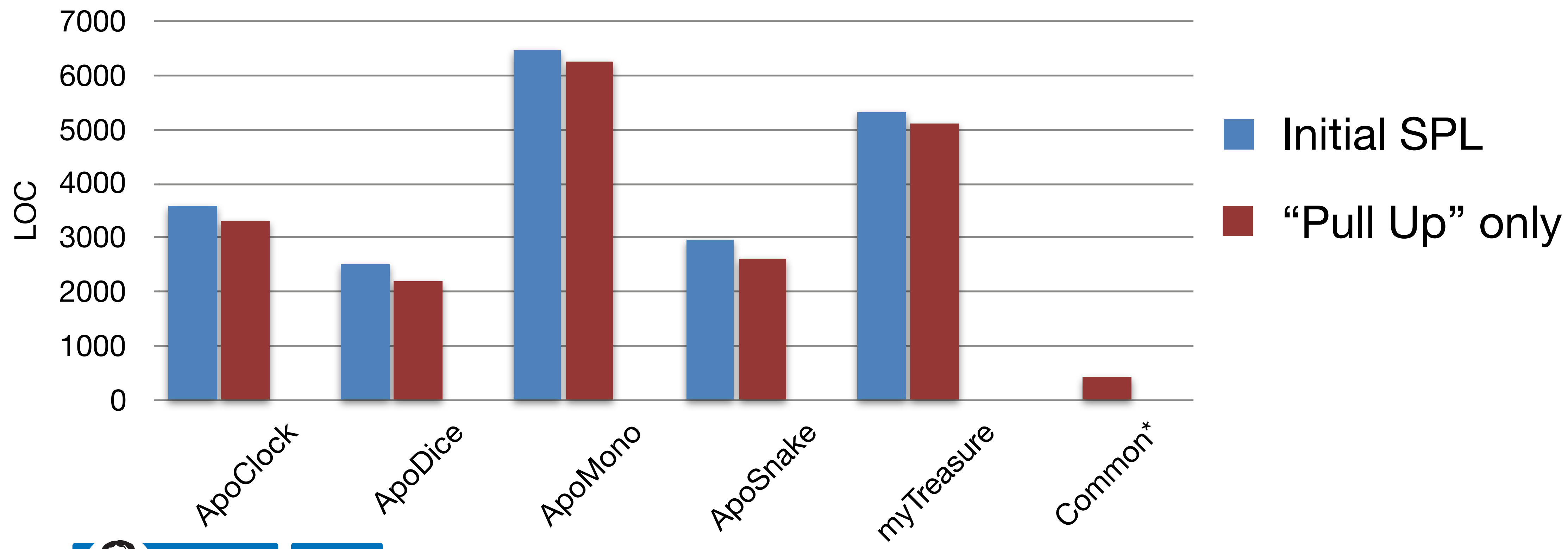

Original Products
ApoClock
ApoDice
ApoMono
ApoSnake
myTreasure
Initial SPL
base
ApoClock
ApoDice
...
myTreasure
"Pull Up" only
base
...
Common1
ApoClock
ApoDice
...
myTreasure


LOC
7000
6000
5000
4000
3000
2000
1000
0
ApoClock
ApoDice
ApoMono
ApoSnake
myTreasure
Common*
Initial SPL

# Evaluation

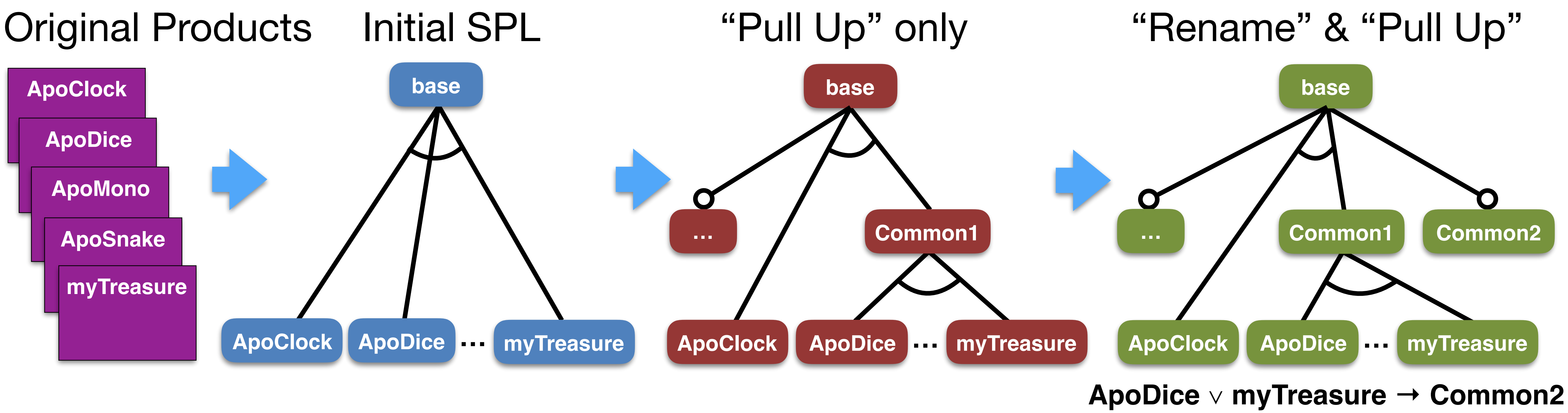


Original Products
ApoClock
ApoDice
ApoMono
ApoSnake
myTreasure
Initial SPL
base
ApoClock
ApoDice
…
myTreasure
"Pull Up" only
base
…
Common1
ApoClock
ApoDice
…
myTreasure



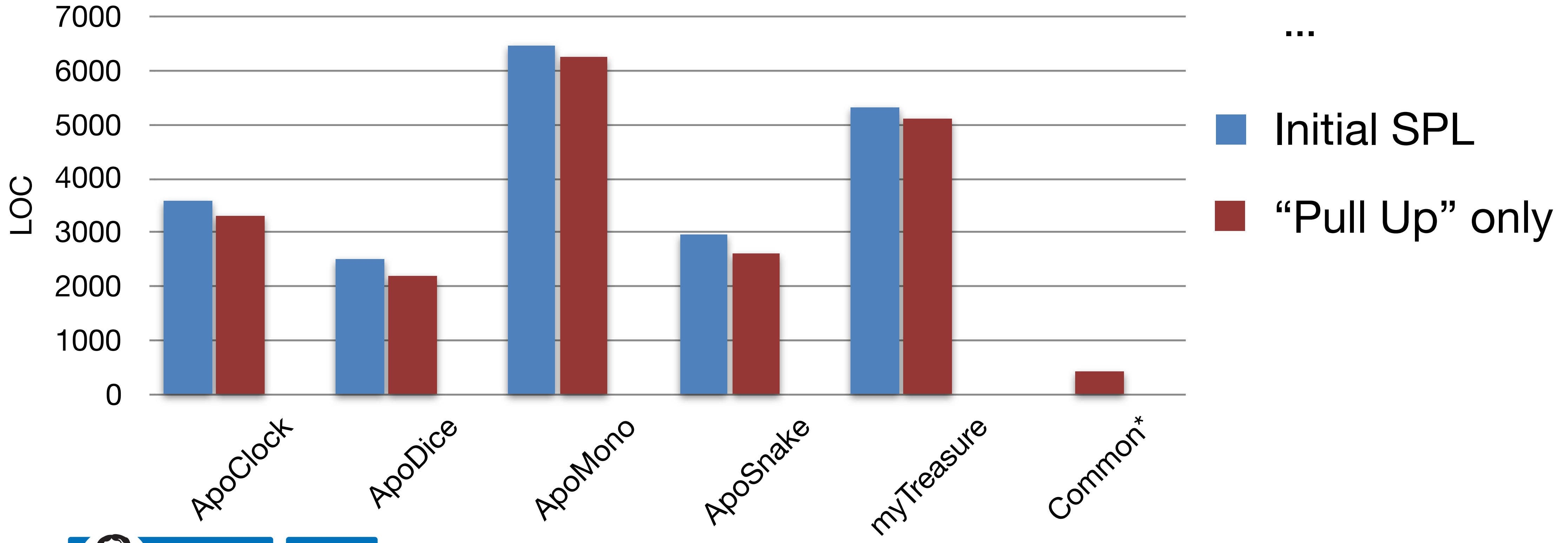
LOC
7000
6000
5000
4000
3000
2000
1000
0
ApoClock
ApoDice
ApoMono
ApoSnake
myTreasure
Common*
Initial SPL
"Pull Up" only

# Evaluation

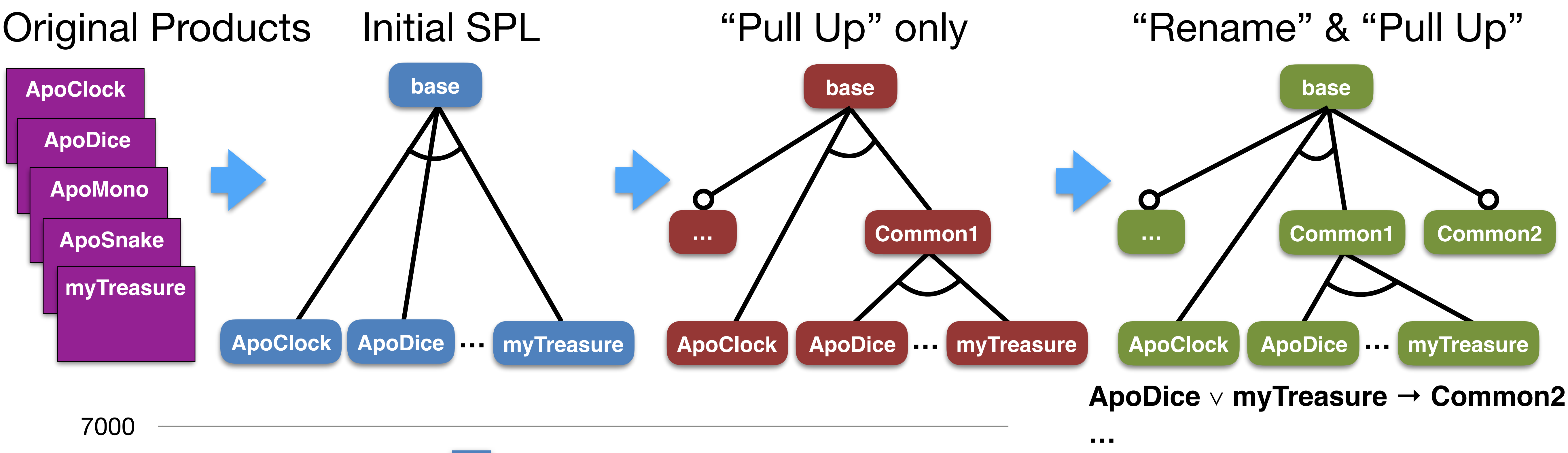


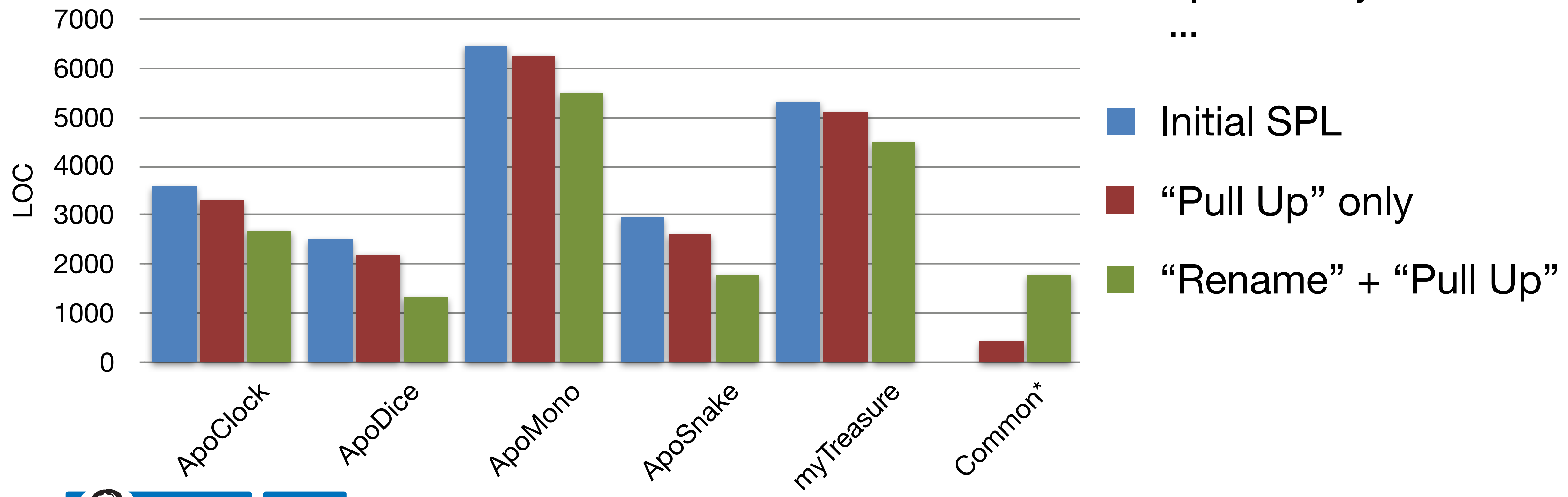
**ApoDice ∨ myTreasure → Common2**

**…**



# Evaluation


Original Products
ApoClock
ApoDice
ApoMono
ApoSnake
myTreasure
Initial SPL
base
ApoClock
ApoDice
…
myTreasure
“Pull Up” only
base
…
Common1
ApoClock
ApoDice
…
myTreasure
“Rename” & “Pull Up”
base
…
Common1
Common2
ApoClock
ApoDice
…
myTreasure
ApoDice ∨ myTreasure → Common2
…


7000
6000
5000
4000
3000
2000
1000
0
LOC
ApoClock
ApoDice
ApoMono
ApoSnake
myTreasure
Common*
Initial SPL
“Pull Up” only
“Rename” + “Pull Up”

## Evaluation — Discussion

- Naming in case study exaggerates efficacy of "Rename" (e.g. class `ApoClockMenu` in ApoClock vs. class `ApoDiceMenu` in ApoDice)
- Why do clones remain?
  - Long similar, but not identical methods (Type-3 clones) — more preparatory refactorings needed
  - Differing releases of 3rd-party libraries w/ conflicting APIs

## Conclusion & Future Work

- Step-wise process to migrate from clone & own to SPL
- Variant-preserving refactorings (Pull Up and Rename)
- Case study shows feasibility
- Future work:
  - Further case studies
  - More (preparatory) refactorings (e.g., “Extract {Method, Field, Constant …}”)
  - Make code similarities more understandable
  - Support for other languages (e.g., C)

## Limitations

- We force developers
  - to choose a variability mechanism of our choice —> no further mechanism is supported
  - to migrate the whole project —> Risky!
- We take only text-based (syntactical) information into account
- We omit possibilities of alternative features

## Mission & Vision

## Mission & Vision

….extract information by means of flexible and customizable reverse engineering

## Mission & Vision

….extract information by means of flexible and customizable reverse engineering

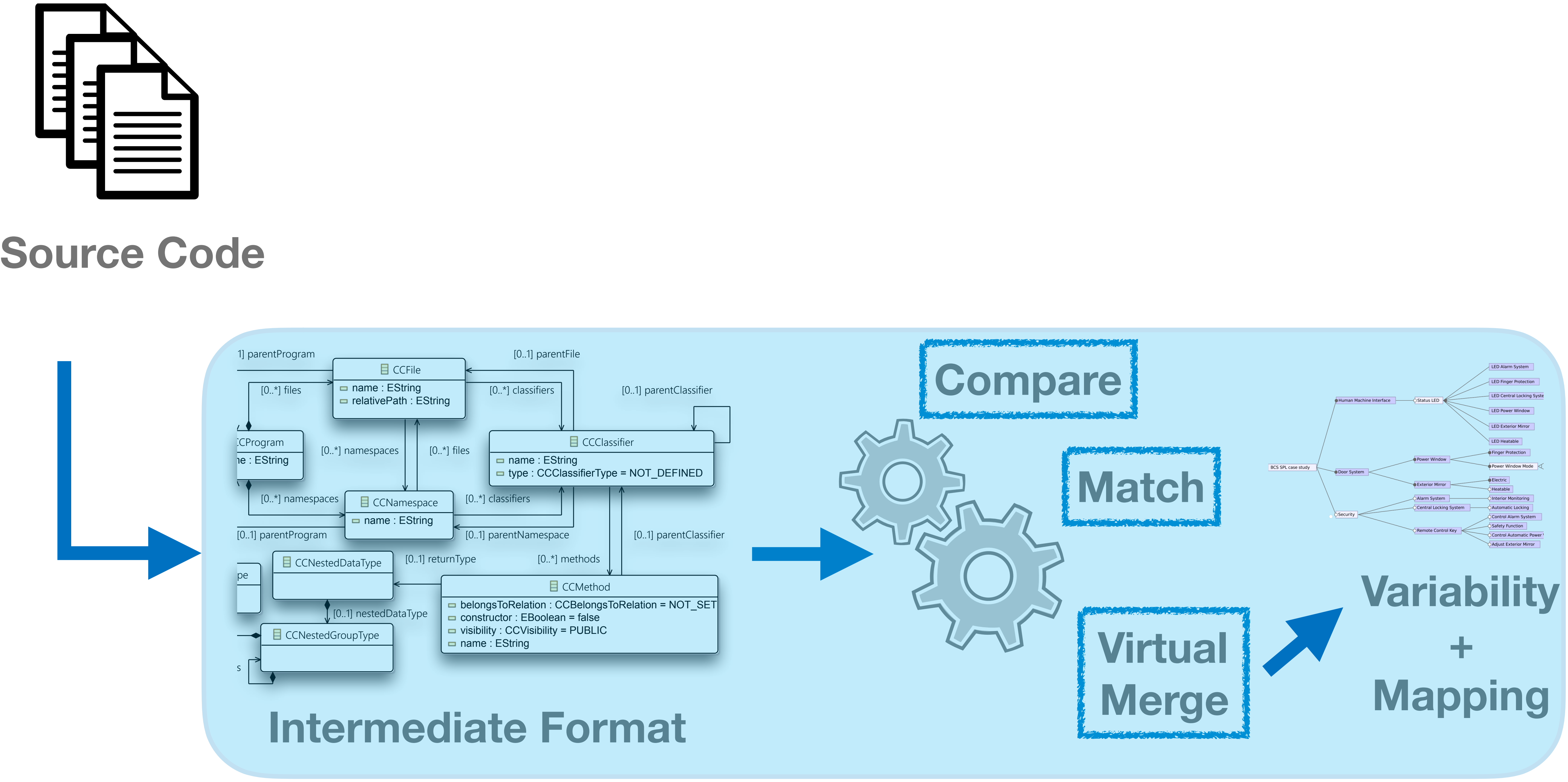
….keep this information in a language-independent format

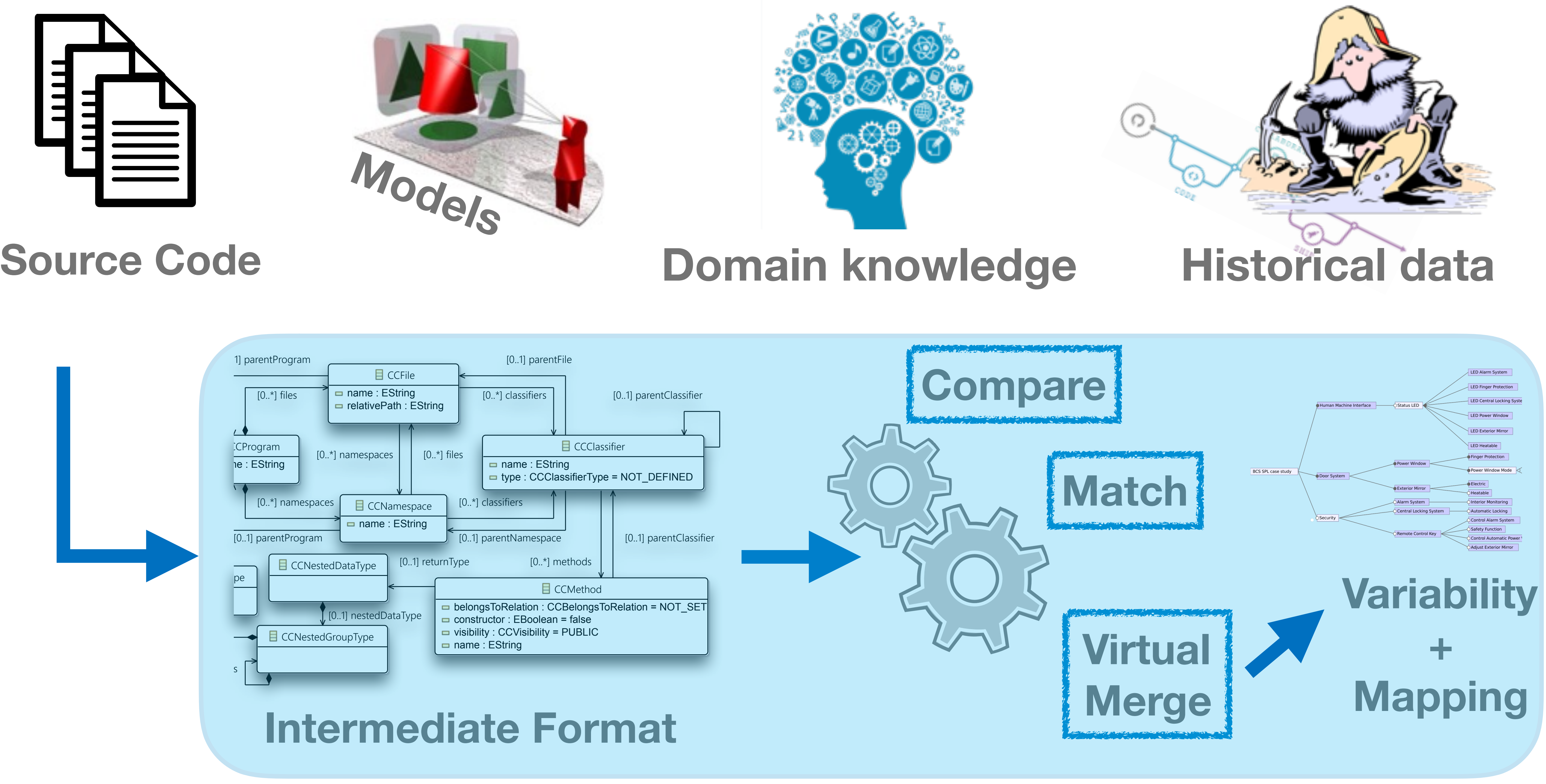
## Mission & Vision

….extract information by means of flexible and customizable reverse engineering

….keep this information in a language-independent format

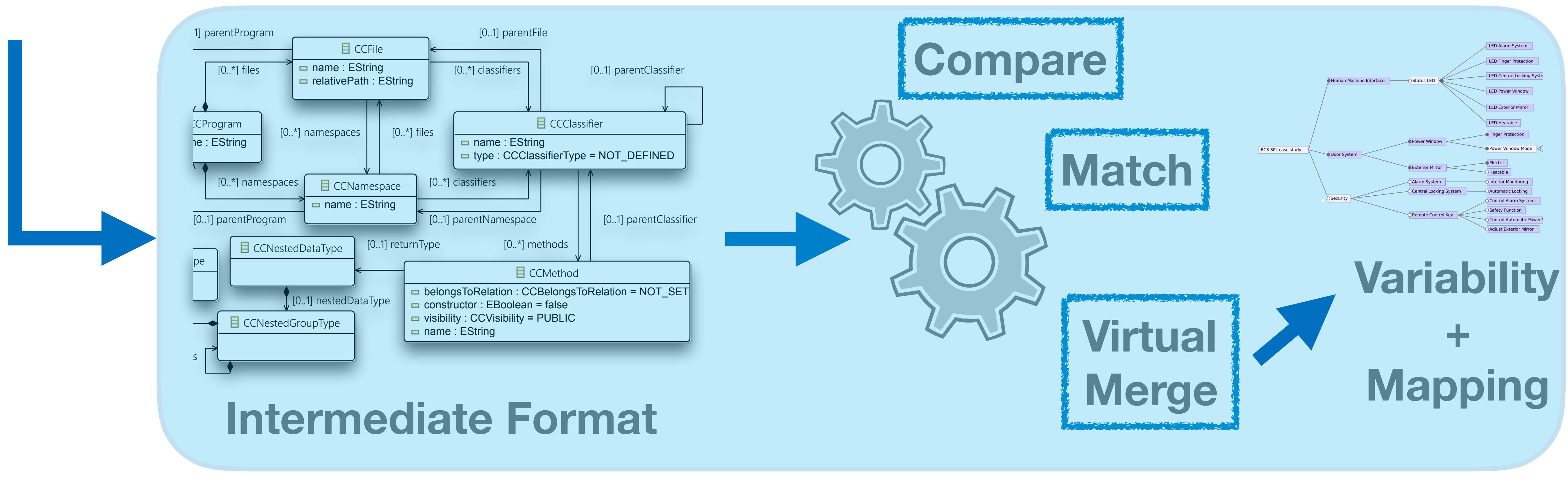
….migration-on-demand by providing reengineering techniques for several variability mechanisms

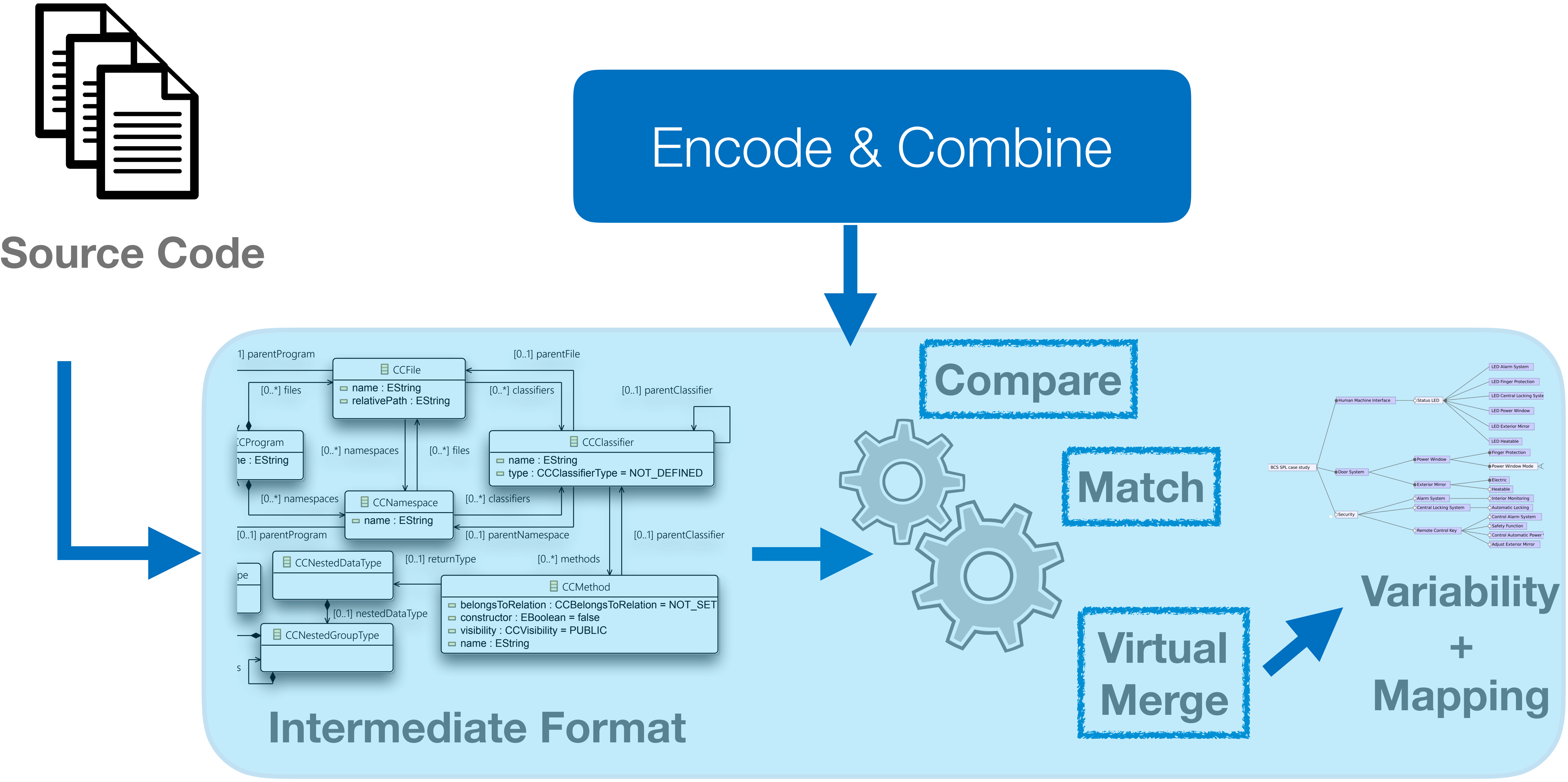

Source Code
Intermediate Format
CCFile
name : EString
relativePath : EString
CCProgram
CCNamespace
name : EString
CCClassifier
name : EString
type : CCClassifierType = NOT_DEFINED
CCNestedDataType
CCNestedGroupType
CCMethod
belongsToRelation : CCBelongsToRelation = NOT_SET
constructor : EBoolean = false
visibility : CCVisibility = PUBLIC
name : EString
[0..1] parentProgram
[0..1] parentFile
[0..*] files
[0..*] classifiers
[0..1] parentClassifier
[0..*] namespaces
[0..1] parentNamespace
[0..1] returnType
[0..*] methods
[0..1] nestedDataType
Compare
Match
Virtual Merge
Variability + Mapping
BCS SPL case study
Human Machine Interface
Status LED
LED Alarm System
LED Finger Protection
LED Central Locking System
LED Power Window
LED Exterior Mirror
LED Heatable
Door System
Power Window
Finger Protection
Power Window Mode
Exterior Mirror
Electric
Heatable
Security
Alarm System
Interior Monitoring
Central Locking System
Automatic Locking
Remote Control Key
Control Alarm System
Safety Function
Control Automatic Power Window
Adjust Exterior Mirror

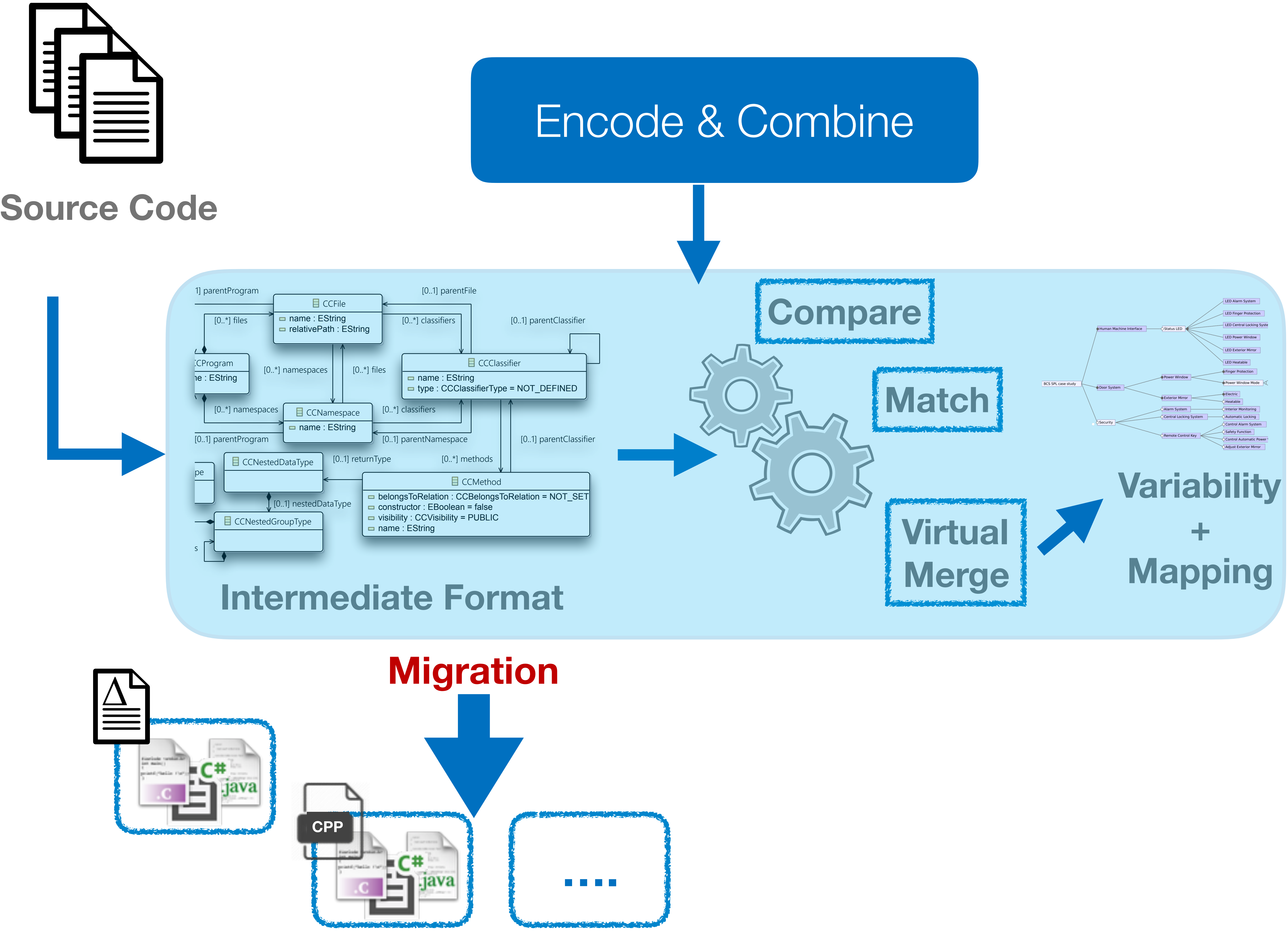

Source Code
Models
Domain knowledge
Historical data
CCFile
name : EString
relativePath : EString
CCClassifier
name : EString
type : CCClassifierType = NOT_DEFINED
CCNamespace
name : EString
CCNestedDataType
CCNestedGroupType
CCMethod
belongsToRelation : CCBelongsToRelation = NOT_SET
constructor : EBoolean = false
visibility : CCVisibility = PUBLIC
name : EString
Intermediate Format
Compare
Match
Virtual Merge
Variability + Mapping

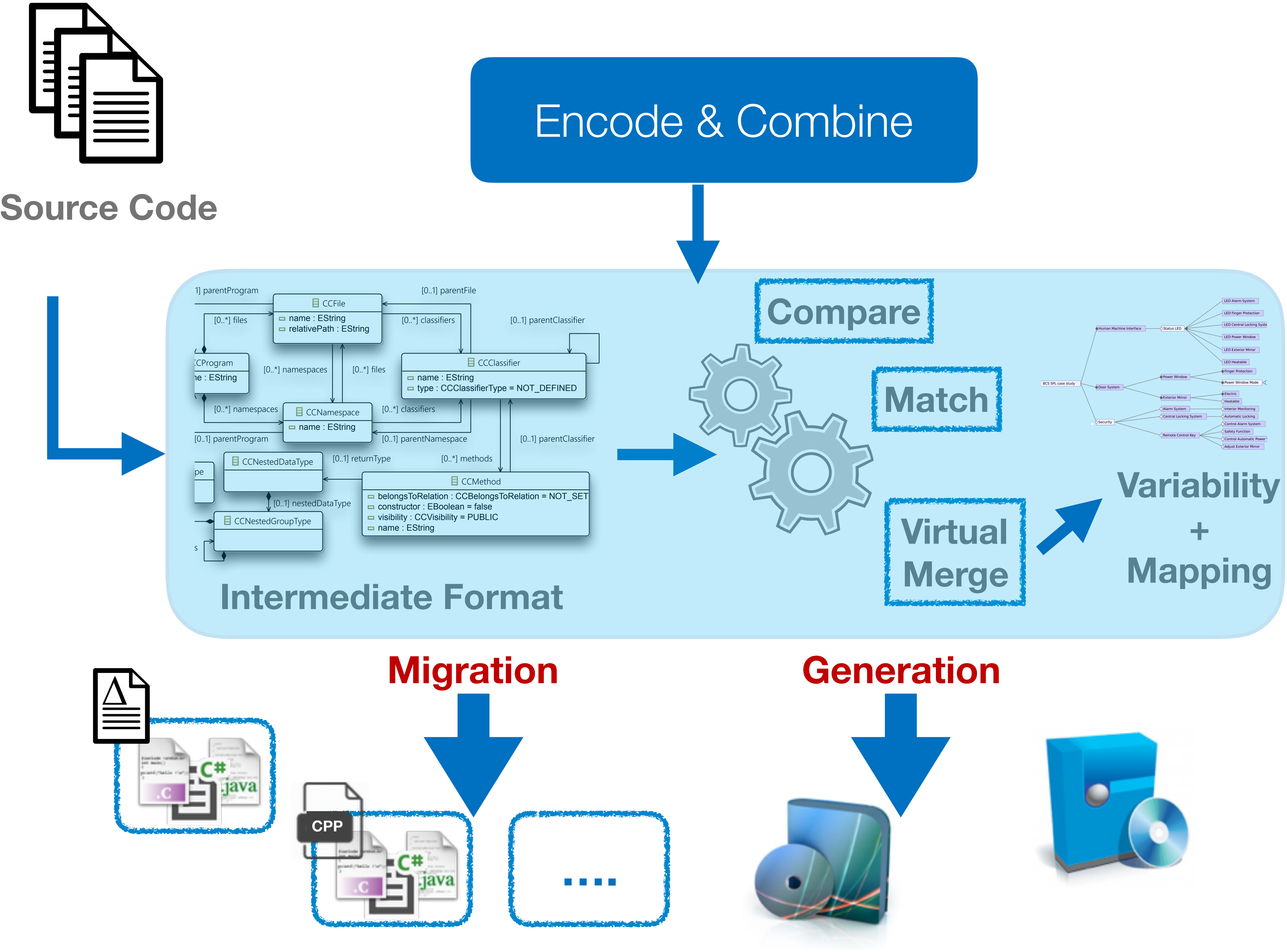
## Encode & Combine

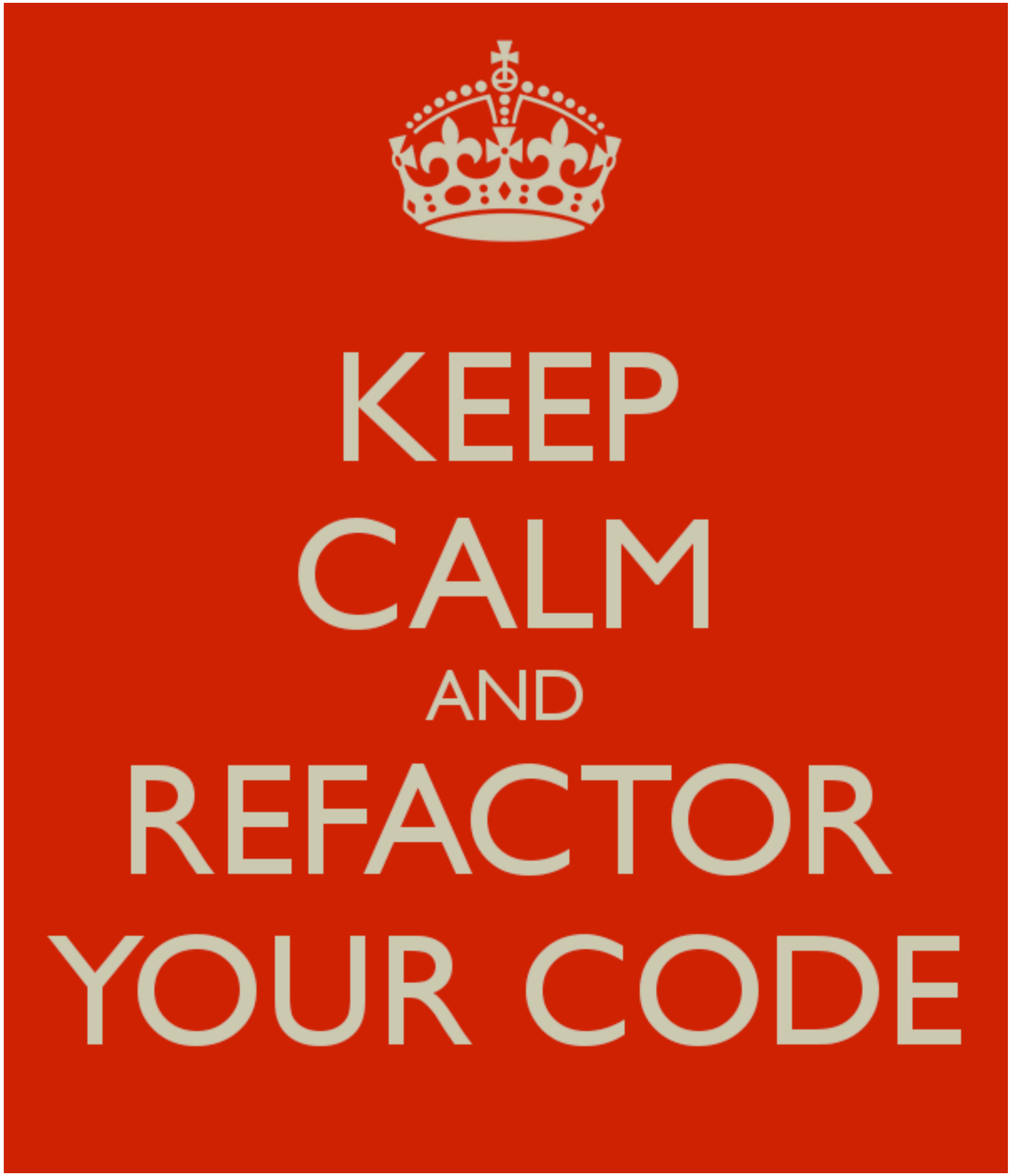
Source Code




Source Code
Encode & Combine
] parentProgram
CCFile
name : EString
relativePath : EString
[0..1] parentFile
[0..*] files
[0..*] classifiers
[0..1] parentClassifier
CProgram
e : EString
[0..*] namespaces
[0..*] files
CCClassifier
name : EString
type : CCClassifierType = NOT_DEFINED
[0..*] namespaces
CCNamespace
name : EString
[0..*] classifiers
[0..1] parentProgram
[0..1] parentNamespace
[0..1] parentClassifier
CCNestedDataType
[0..1] returnType
[0..*] methods
CCMethod
belongsToRelation : CCBelongsToRelation = NOT_SET
constructor : EBoolean = false
visibility : CCVisibility = PUBLIC
name : EString
[0..1] nestedDataType
CCNestedGroupType
Intermediate Format
Compare
Match
Virtual Merge
BCS SPL case study
Human Machine Interface
Status LED
LED Alarm System
LED Finger Protection
LED Central Locking Syste
LED Power Window
LED Exterior Mirror
LED Heatable
Door System
Power Window
Finger Protection
Power Window Mode
Exterior Mirror
Electric
Heatable
Security
Alarm System
Interior Monitoring
Central Locking System
Automatic Locking
Remote Control Key
Control Alarm System
Safety Function
Control Automatic Power
Adjust Exterior Mirror
Variability + Mapping


Source Code
Encode & Combine
Compare
Match
Virtual Merge
Variability + Mapping
Intermediate Format
Migration
CCFile
name : EString
relativePath : EString
CCClassifier
name : EString
type : CCClassifierType = NOT_DEFINED
CCNamespace
name : EString
CCNestedDataType
CCNestedGroupType
CCMethod
belongsToRelation : CCBelongsToRelation = NOT_SET
constructor : EBoolean = false
visibility : CCVisibility = PUBLIC
name : EString
[0..1] parentProgram
[0..1] parentFile
[0..*] files
[0..*] classifiers
[0..1] parentClassifier
[0..*] namespaces
[0..1] parentNamespace
[0..1] returnType
[0..*] methods
[0..1] nestedDataType
BCS SPL case study
Human Machine Interface
Status LED
LED Alarm System
LED Finger Protection
LED Central Locking System
LED Power Window
LED Exterior Mirror
LED Heatable
Door System
Power Window
Finger Protection
Power Window Mode
Exterior Mirror
Electric
Heatable
Security
Alarm System
Interior Monitoring
Central Locking System
Automatic Locking
Control Alarm System
Safety Function
Remote Control Key
Control Automatic Power
Adjust Exterior Mirror
C#
java
.C
CPP
....

 Encode & Combine
Source Code
1] parentProgram
CCFile
name : EString
relativePath : EString
[0..1] parentFile
[0..*] files
[0..*] classifiers
[0..1] parentClassifier
CProgram
e : EString
[0..*] namespaces
[0..*] files
CCClassifier
name : EString
type : CCClassifierType = NOT_DEFINED
[0..*] namespaces
CCNamespace
name : EString
[0..*] classifiers
[0..1] parentProgram
[0..1] parentNamespace
[0..1] parentClassifier
CCNestedDataType
[0..1] returnType
[0..*] methods
CCMethod
belongsToRelation : CCBelongsToRelation = NOT_SET
constructor : EBoolean = false
visibility : CCVisibility = PUBLIC
name : EString
[0..1] nestedDataType
CCNestedGroupType
Intermediate Format
Compare
Match
Virtual Merge
BCS SPL case study
Human Machine Interface
Status LED
LED Alarm System
LED Finger Protection
LED Central Locking Syste
LED Power Window
LED Exterior Mirror
LED Heatable
Door System
Power Window
Finger Protection
Power Window Mode
Exterior Mirror
Electric
Heatable
Security
Alarm System
Interior Monitoring
Central Locking System
Automatic Locking
Remote Control Key
Control Alarm System
Safety Function
Control Automatic Power
Adjust Exterior Mirror
Variability + Mapping
Migration
Generation
C#
.C
java
CPP
....


KEEP
CALM
AND
REFACTOR
YOUR CODE