Enhancing Feature Interfaces for Supporting Software Product Line Maintenance

Bruno B. P. Cafeo

LES | DI |PUC-Rio - Brazil

OPUS Group

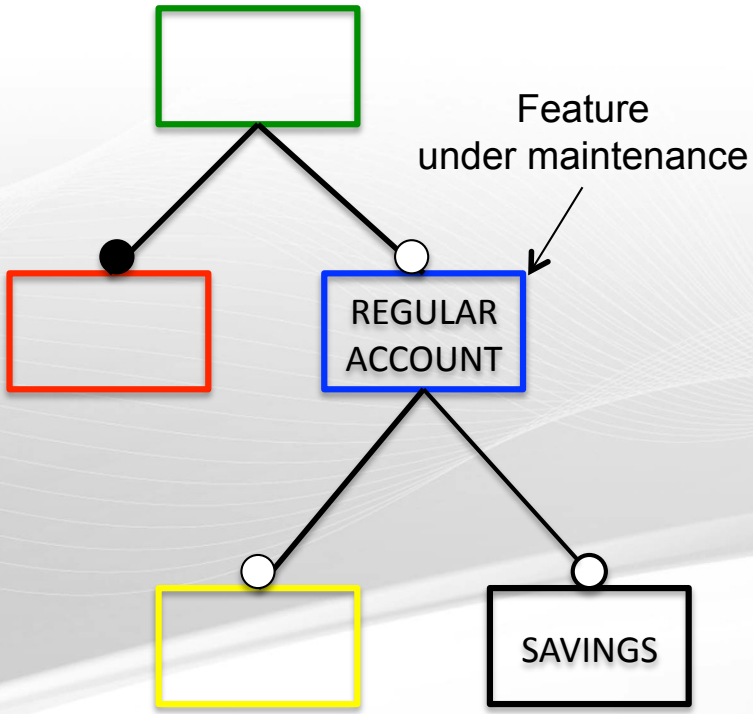
Introduction

- Features carve software into meaningful and manageable pieces
- In the source code, the boundaries of a feature are often not the same as the module in a program
- A challenged faced in the maintenance of SPLs is the identification and understanding of feature dependencies

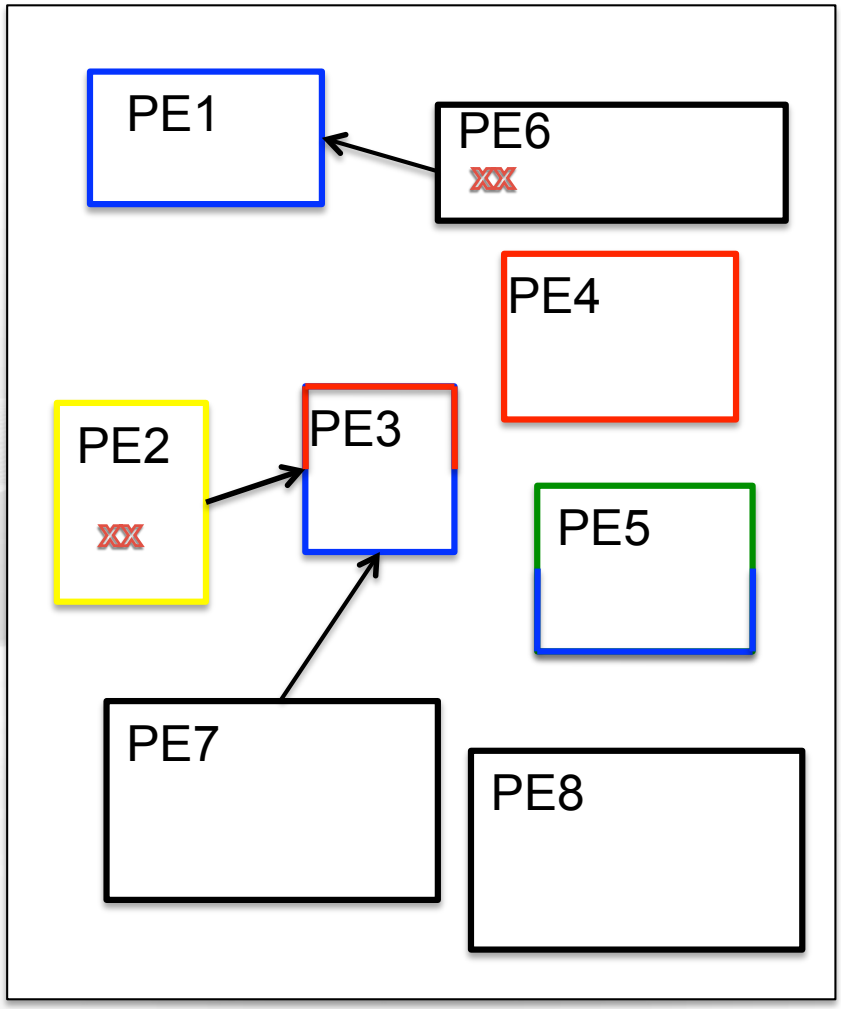
Motivation

Feature Dependencies in the source code of a SPL

Legend: PE – Program Element XX Maintenance problems related to the presence of a feature dependency



Feature Model



Source code

Motivation

- Several parts of the code that are not relevant to the feature maintenance are analysed
- Developers are likely to ignore consciously (or not) feature dependencies while reasoning about a maintenance task
- Other important parts of the code that should be revised might be omitted

Feature Interface

Looking at the source code level

- As in a conventional stand-alone software (i.e. non-SPLs), **interfaces** should **help the understanding** of the **communication** between features in SPLs

- **Program elements** configuring dependencies are part of an implicit **feature interface**
  - Implicit feature interface: elements in the source code that configure a communication between different features

Challenge

Complex feature interfaces

REGULAR_ACCOUNT

account_number
type
sort_code
balance
paid_in[]
paid_out[]
getEndofMonthBalance()
getBalance()
makeAPayment()
getFullStatement()
viewAccountDetailts()
getTaxes()

- Feature interfaces may become **large**
  - **Several elements** are member of an implicit interface
- Implicit feature interfaces are **not cohesive**
  - **Groups** of elements **act together** for the purpose of a dependency

Towards a Solution

- The proposed solution relies on the idea of *Interface Segregation Principle (ISP)* that states that

"clients should not be forced to depend upon interfaces that they do not use"

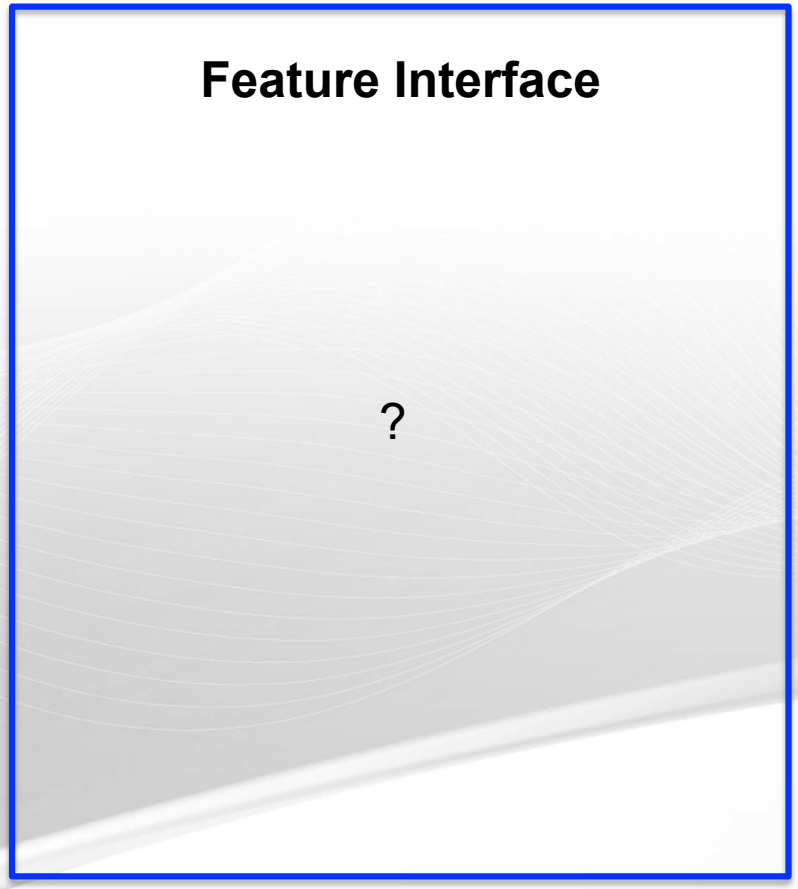
- We observe the idea of ISP from the point of view of SPL maintenance and we argue that

Developers should not be forced to understand parts of an interface that are not useful to their tasks

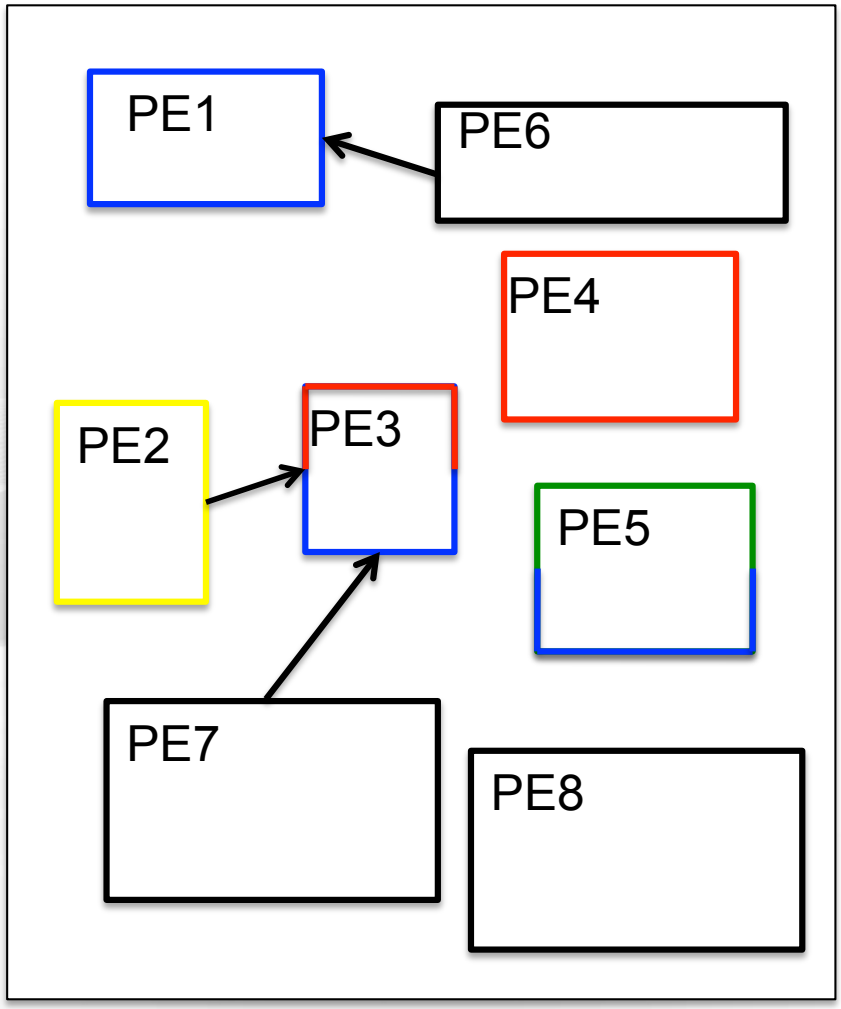
Towards a Solution (simple example)

Identifying feature interface elements

Legend: PE – Program Element



Feature blue

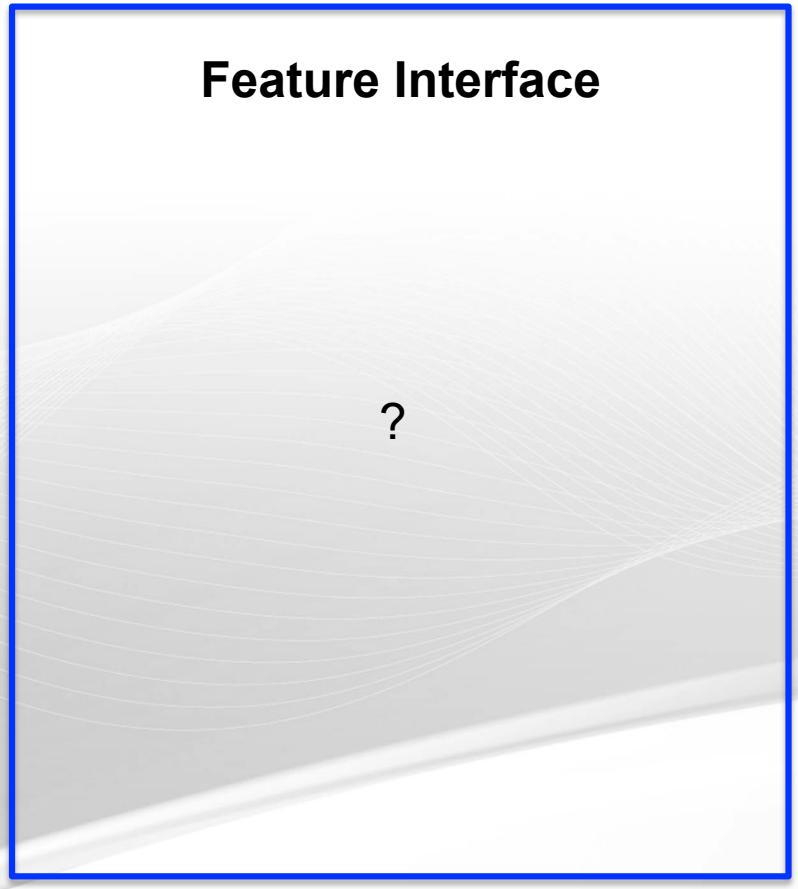


Source code

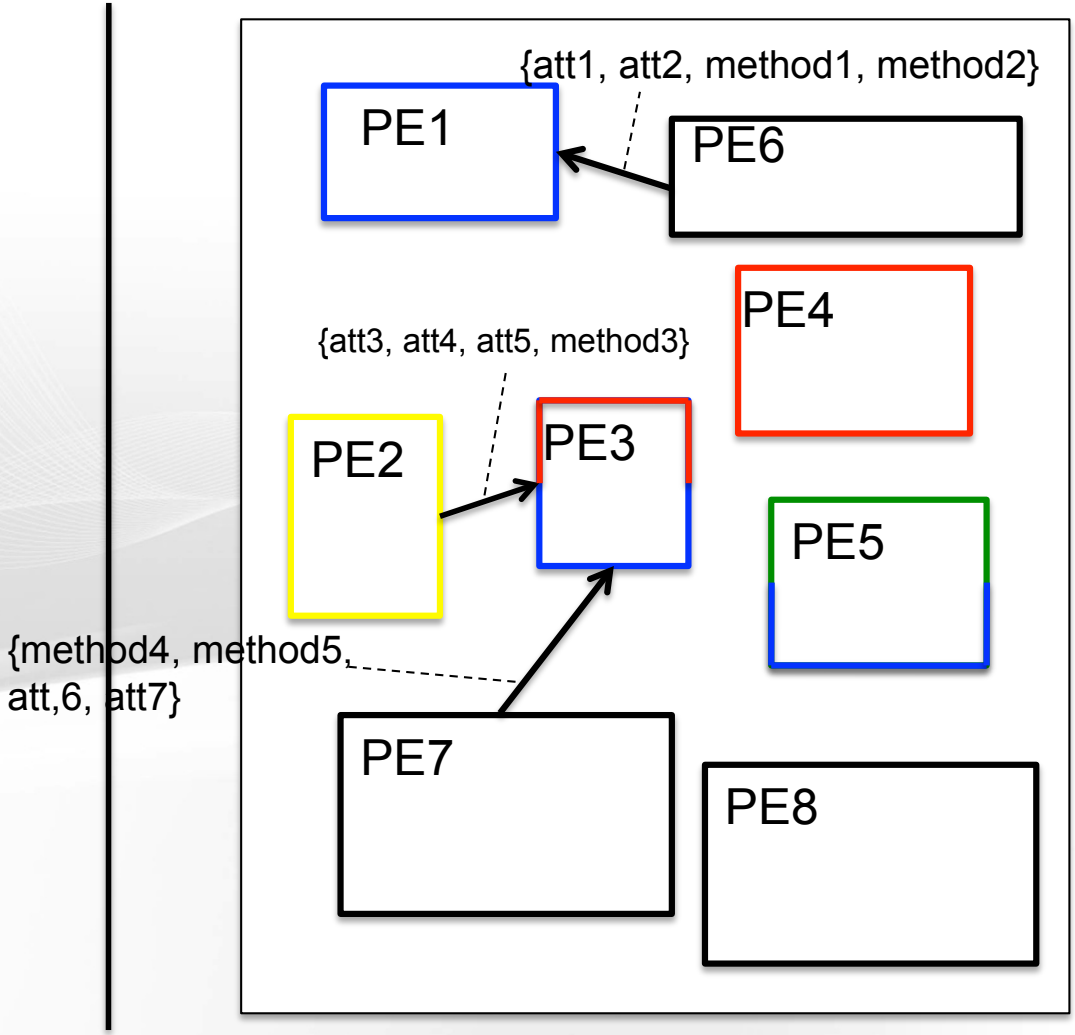
Towards a Solution (simple example)

Identifying feature interface elements

Legend: PE – Program Element



Feature blue



Source code

Towards a Solution (simple example)

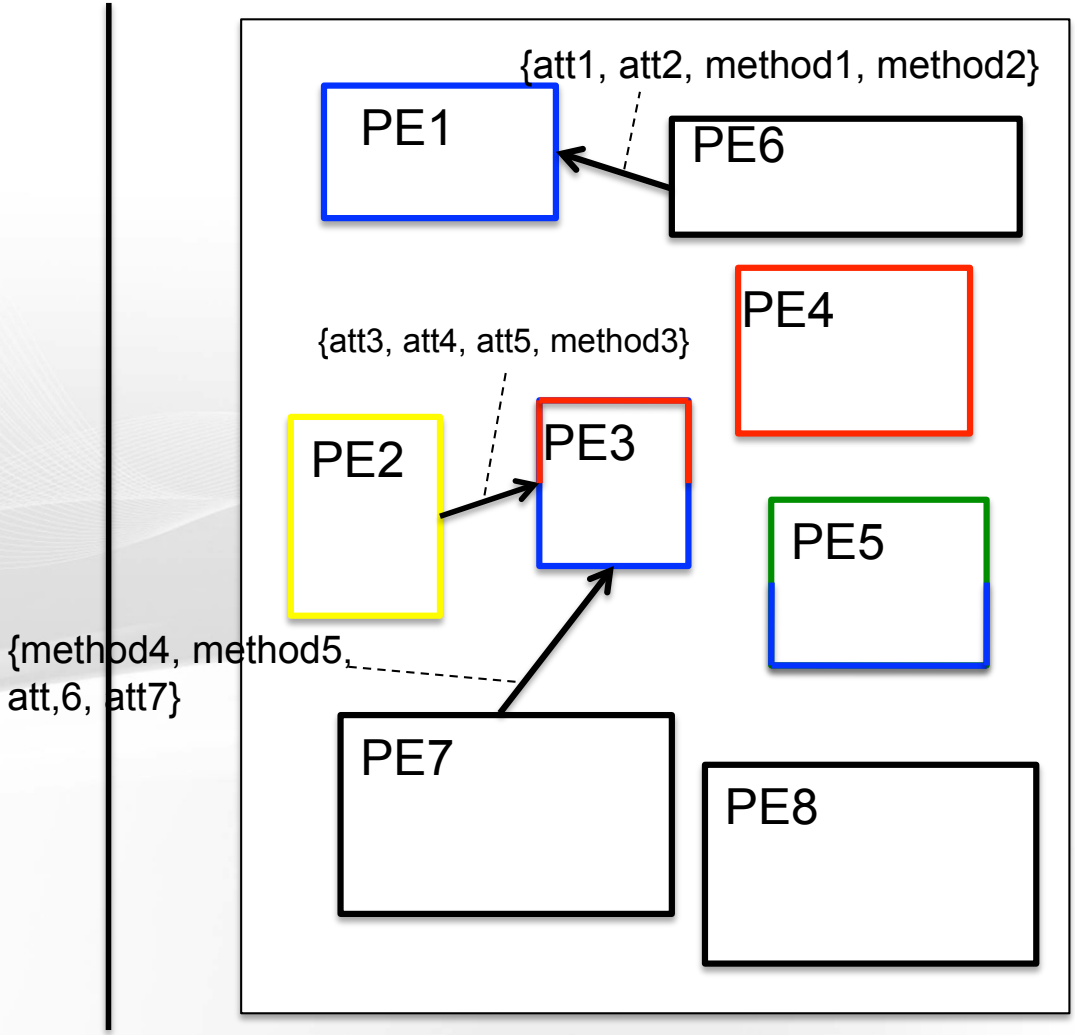
Identifying feature interface elements

Legend: PE – Program Element

Feature Interface

att1
att2
att3
att4
att5
att6
att7
method1
method2
method3
method4
method5

Feature blue



Source code

Towards a Solution (simple example)

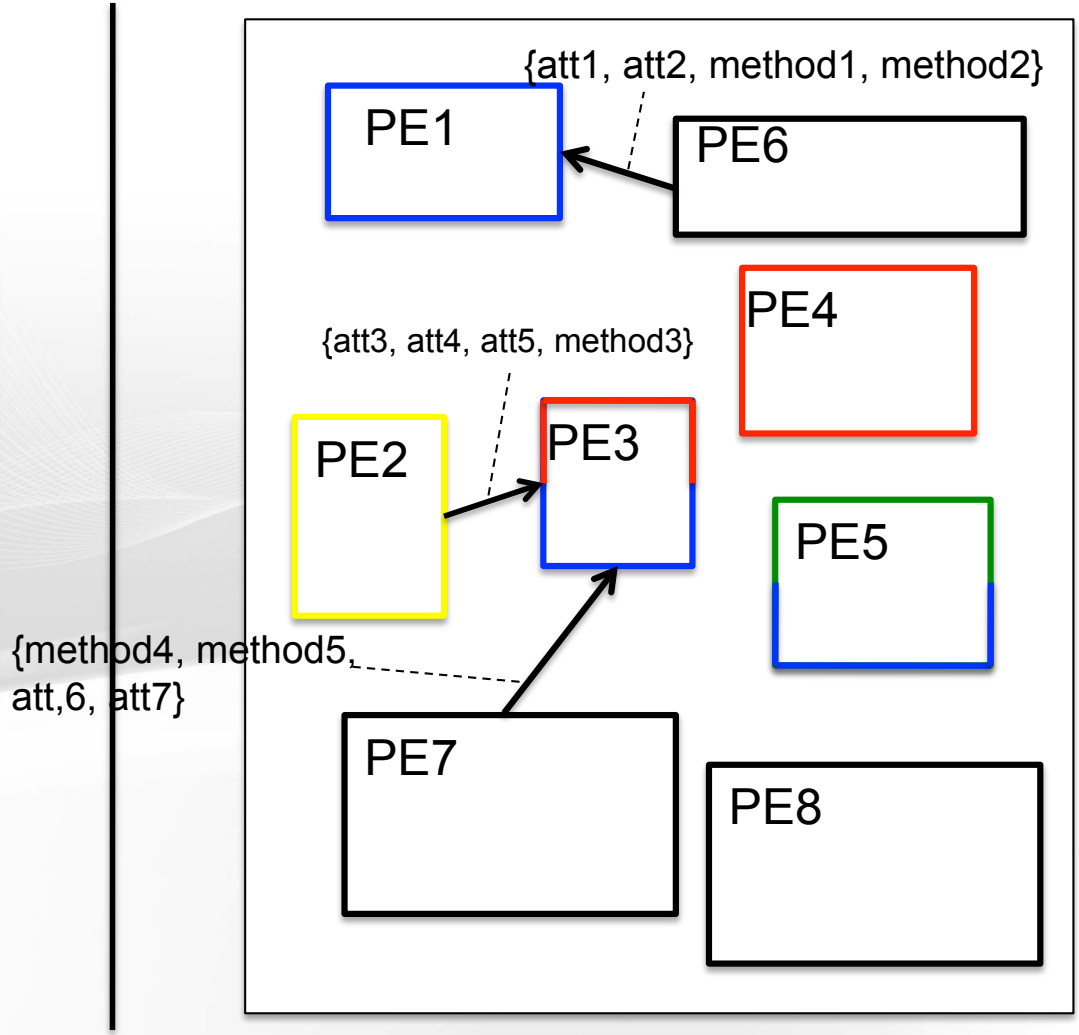
Understanding the feature interface

Legend: PE – Program Element

Feature Interface

att1
att2
att3
att4
att5
att6
att7
method1
method2
method3
method4
method5

Feature blue

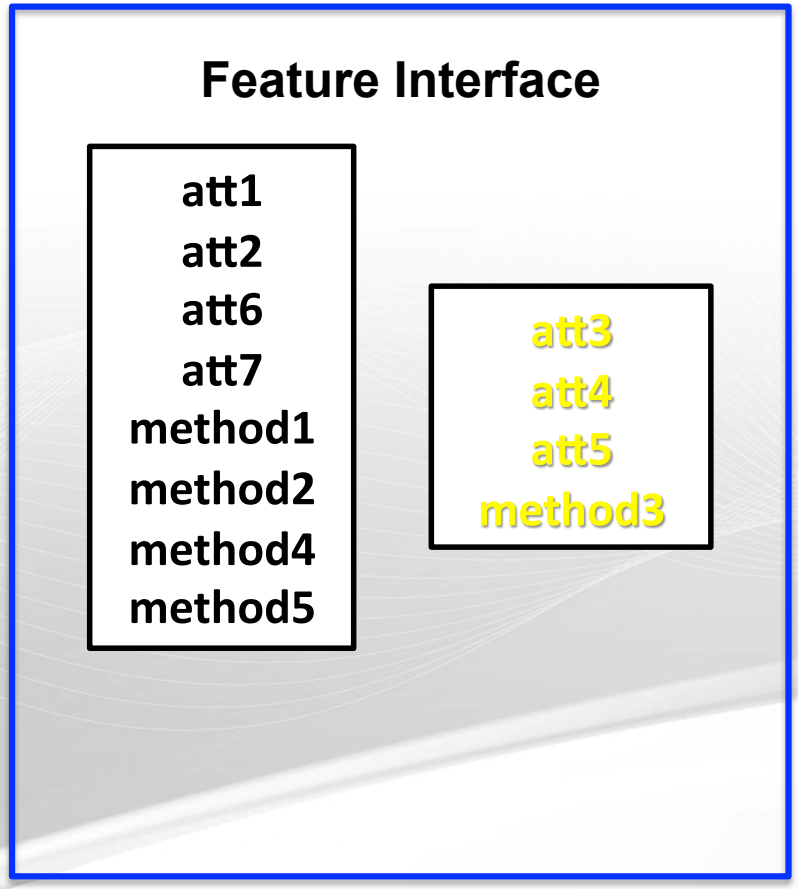


Source code

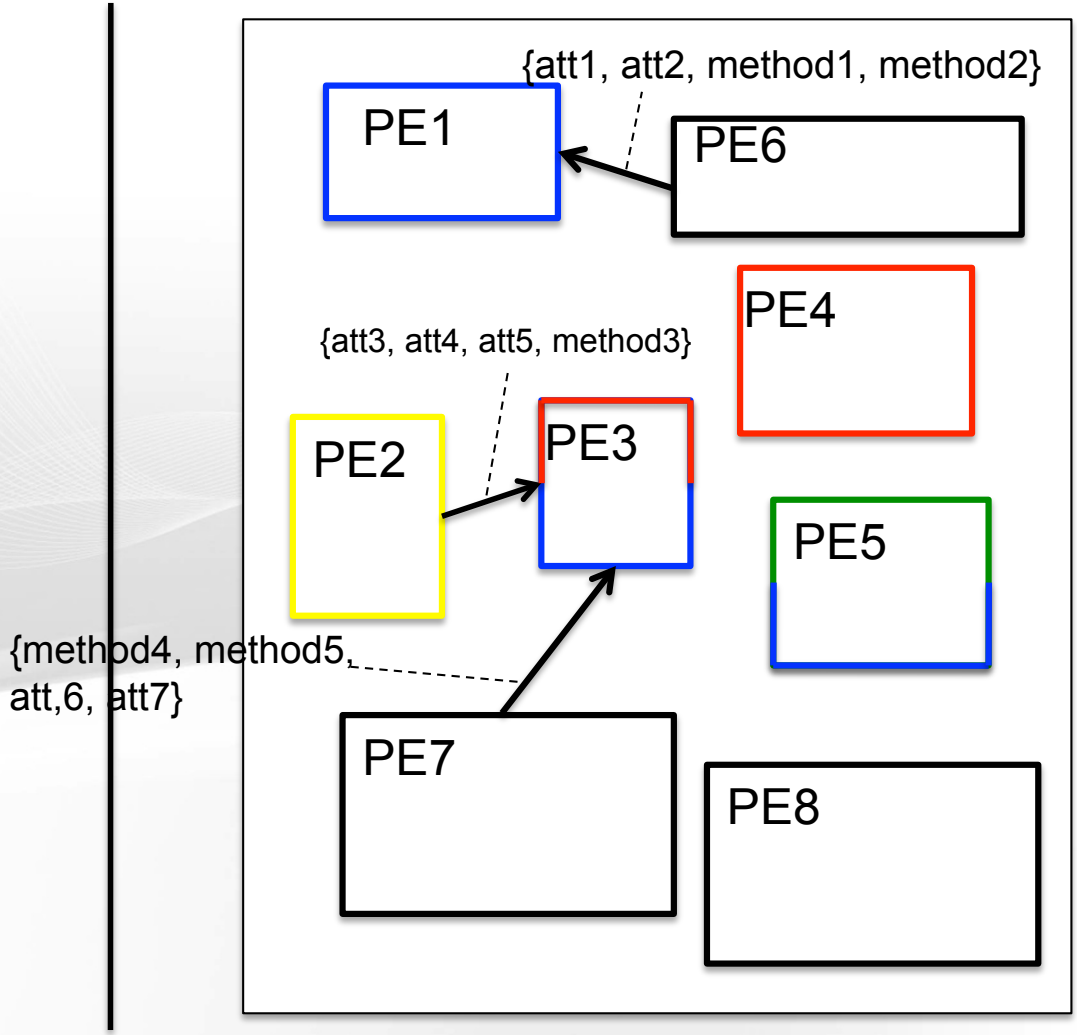
Towards a Solution (simple example)

Understanding the feature interface

Legend: PE – Program Element



Feature blue

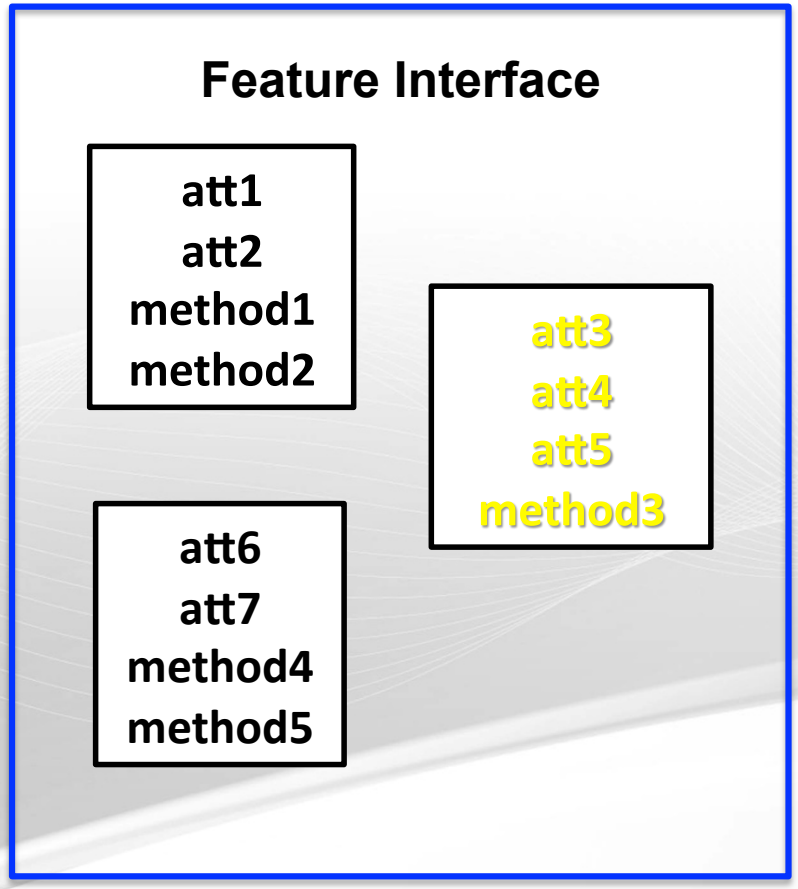


Source code

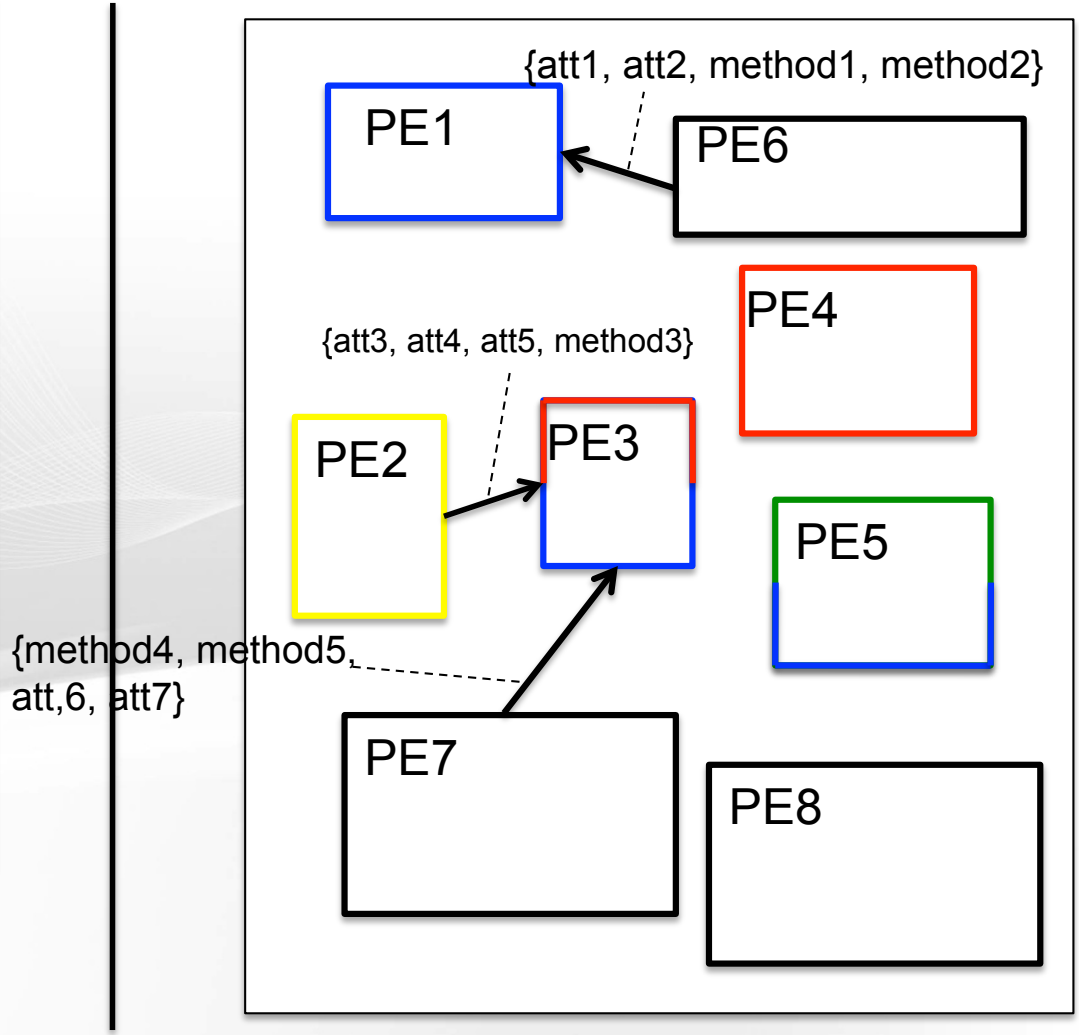
Towards a Solution (simple example)

Understanding the feature interface

Legend: PE – Program Element



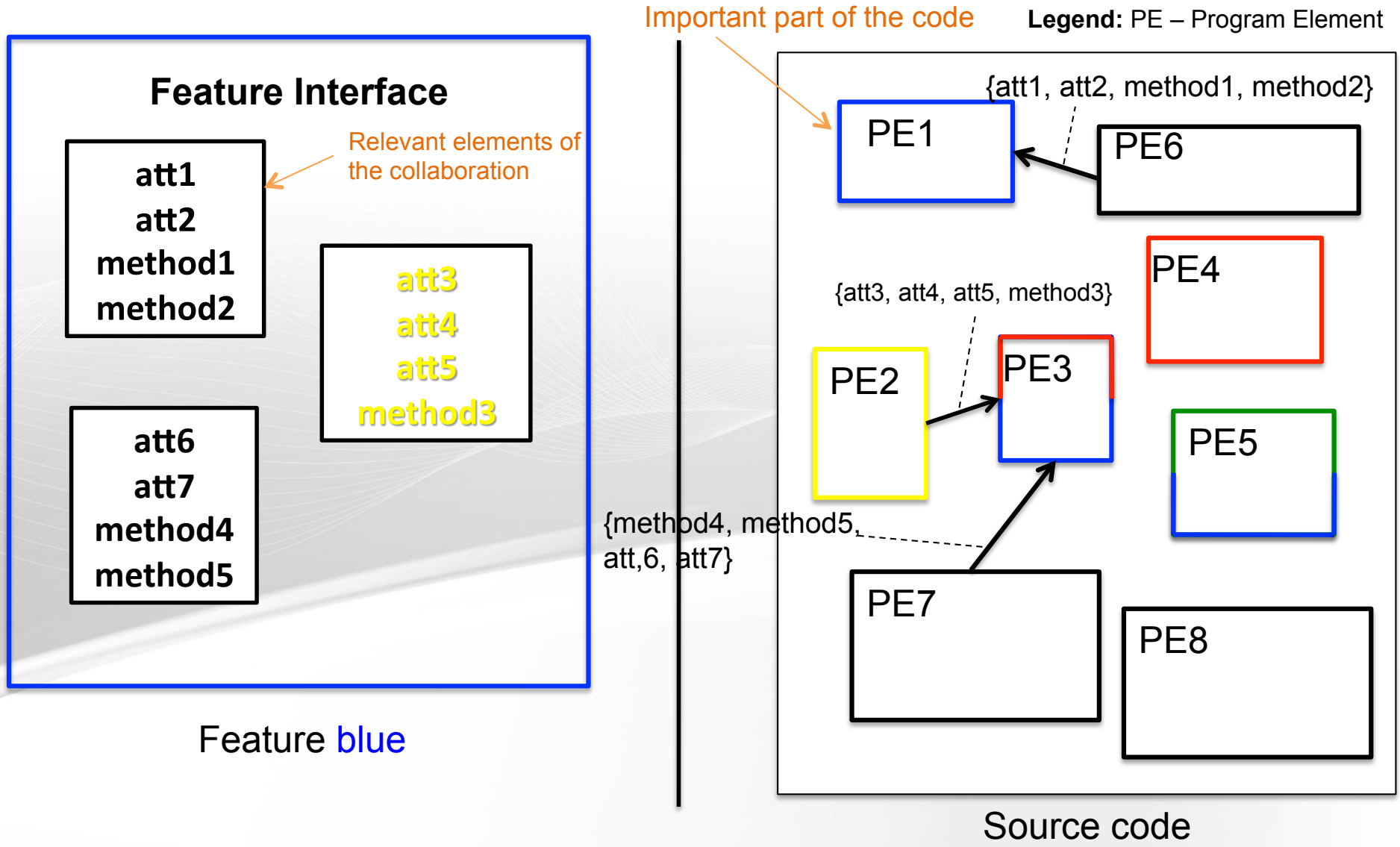
Feature blue



Source code

Towards a Solution (simple example)

Understanding the feature interface



Points for discussion

- Approach to segregate/organize the members of the interface
  - Configuration knowledge
  - Properties and metrics of features and their dependencies
  - ...
- Beyond provided/required interface
  - Information about configurations
  - Preconditions, postconditions
  - ...

Enhancing Feature Interfaces for Supporting Software Product Line Maintenance

Bruno B. P. Cafeo

LES | DI |PUC-Rio - Brazil

OPUS Group