

Using aspect-oriented programming to promote reuse across domains in software product lines

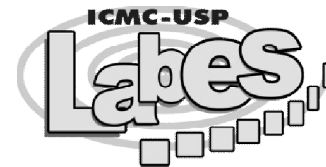
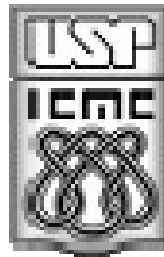
Rosana Teresinha Vaccare Braga

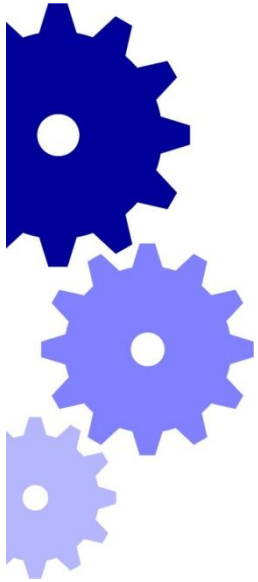
Paulo Cesar Masiero

Carlos A. F. Pereira Jr.

Universidade de São Paulo

Instituto de Ciências Matemáticas e Computação – ICMC – São Carlos



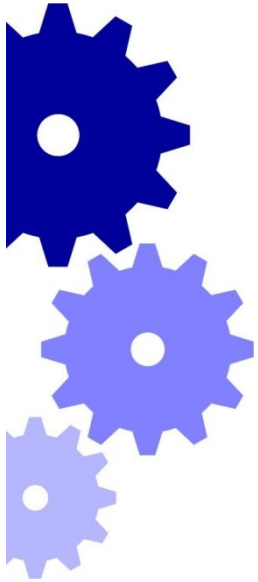


Introduction

- Objective:

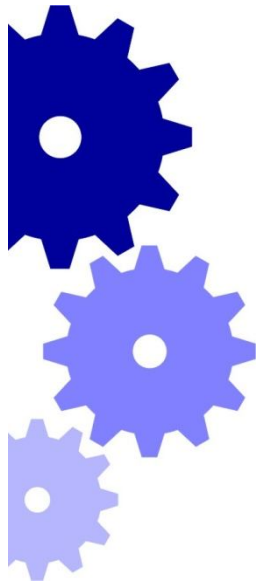
- Combined use of AOP and SPL to improve reuse across different domains
 - Crosscutting domains and crosscutting features
- Benefits: avoid tangling and scattering of features, increase reuse in other SPLs, higher cohesion, better system maintenance and evolution

- Context: Code Generator Captor

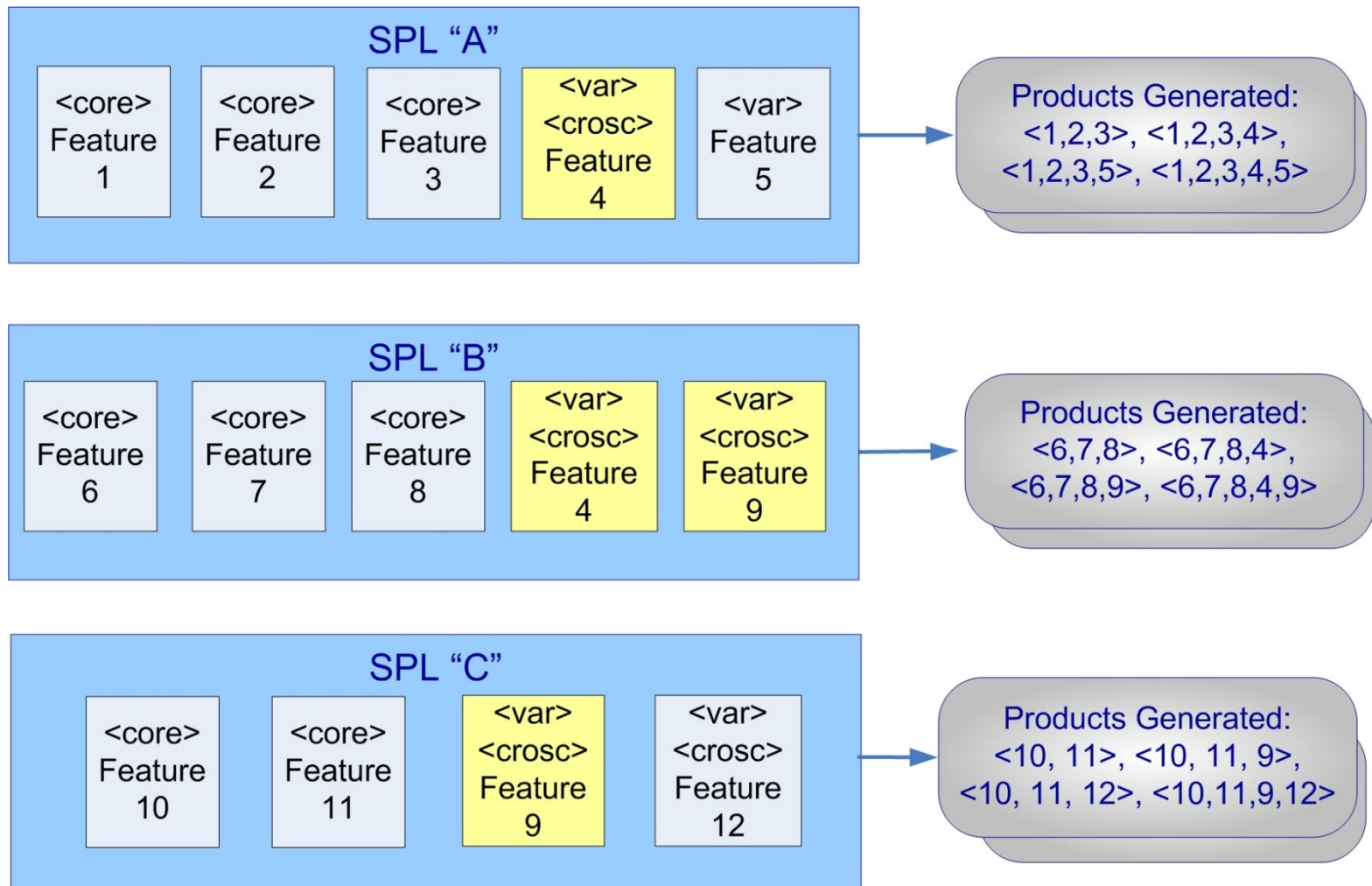


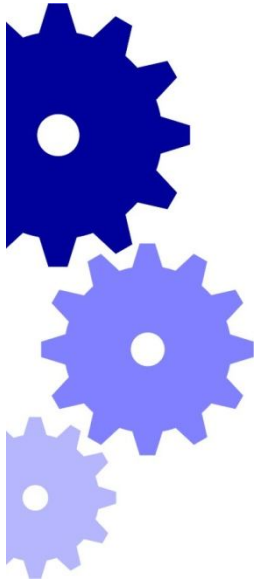
SPL development considering aspects

- What happens when we have several SPLs (for several domains)?
 - Features (functional or non-functional) begin to appear in more than one domain
- Example: Electronic commerce
 - Authentication
 - Persistence
 - Payment



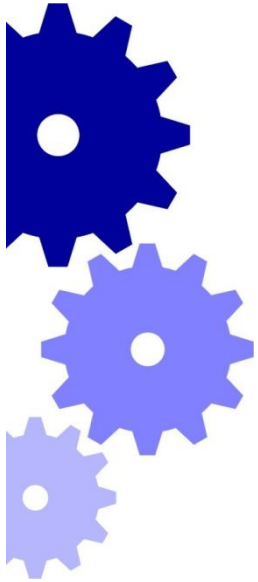
SPL development considering aspects





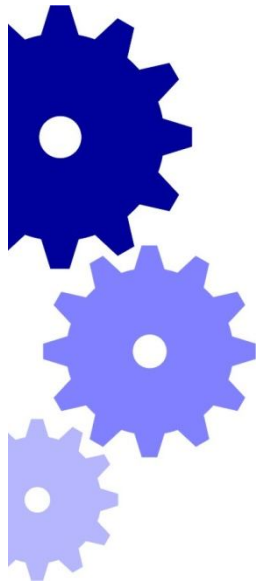
SPL development considering aspects

- What happens when we have several SPLs (for several domains)?
 - Crosscutting features are still designed to be reused in each individual domain
 - Pointcuts are usually fixed: they represent points in the SPL domain where the aspect will intercept the code
 - After developing several SPLs, redundant code begins to be produced
 - But more reuse could be provided if crosscutting features are intended to be used in several domains!!!
 - Pointcuts could vary depending on the base domain

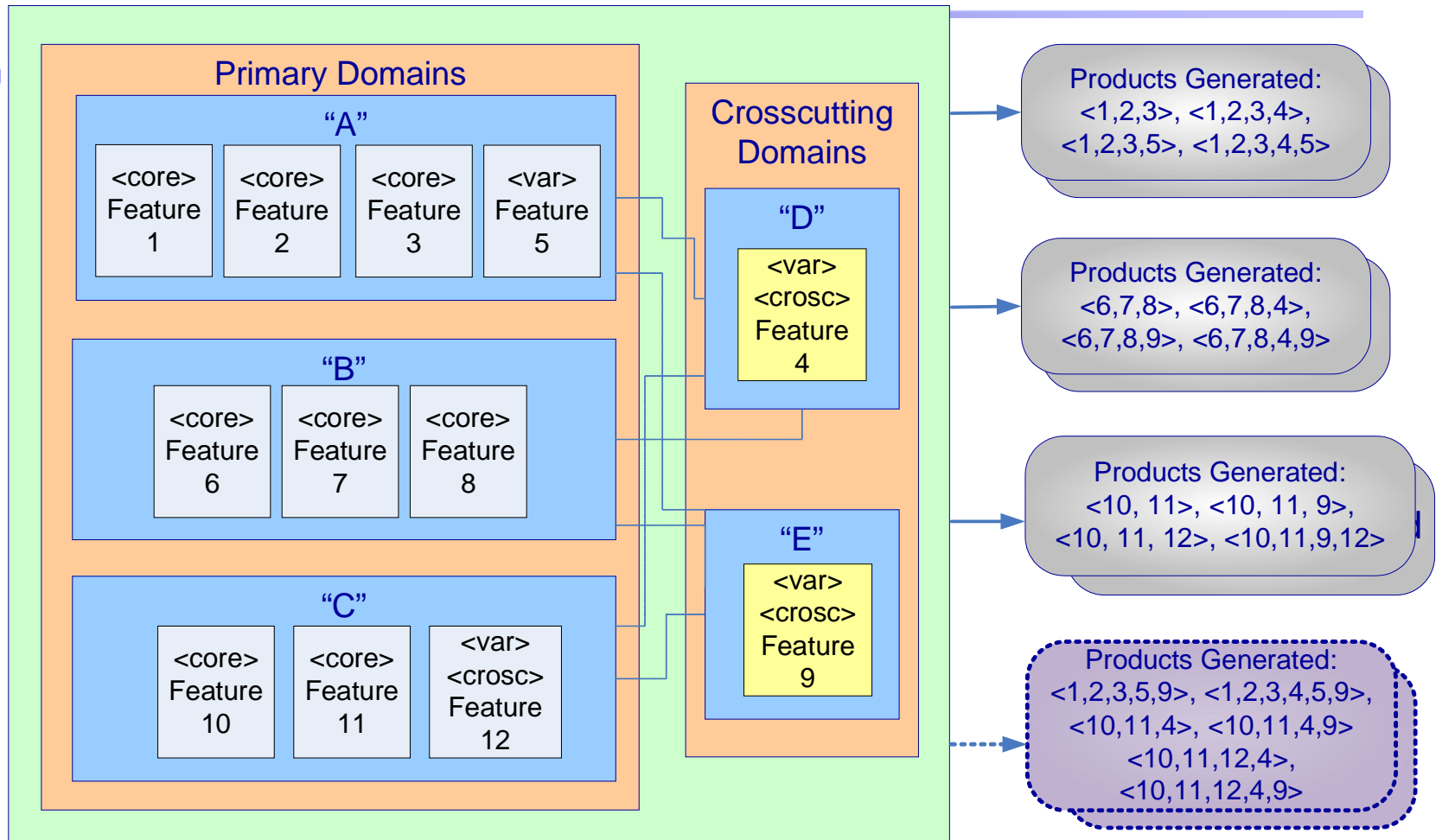


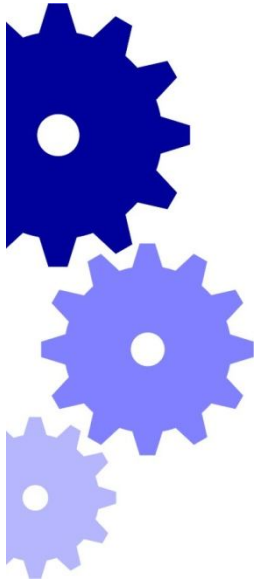
SPL development considering aspects

- Crosscutting features are isolated into “crosscutting domains”
 - They can be both functional or non-functional features
- Domains that represent the core of the product line, and that will be crosscut by aspects, are named “primary domains”
 - They can be combined with zero or more crosscutting domains
- A crosscutting domain can be reused across different primary domains



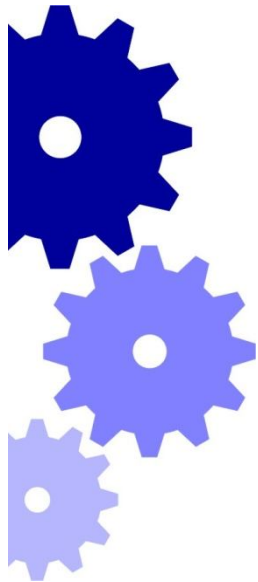
Primary and Crosscutting Domains



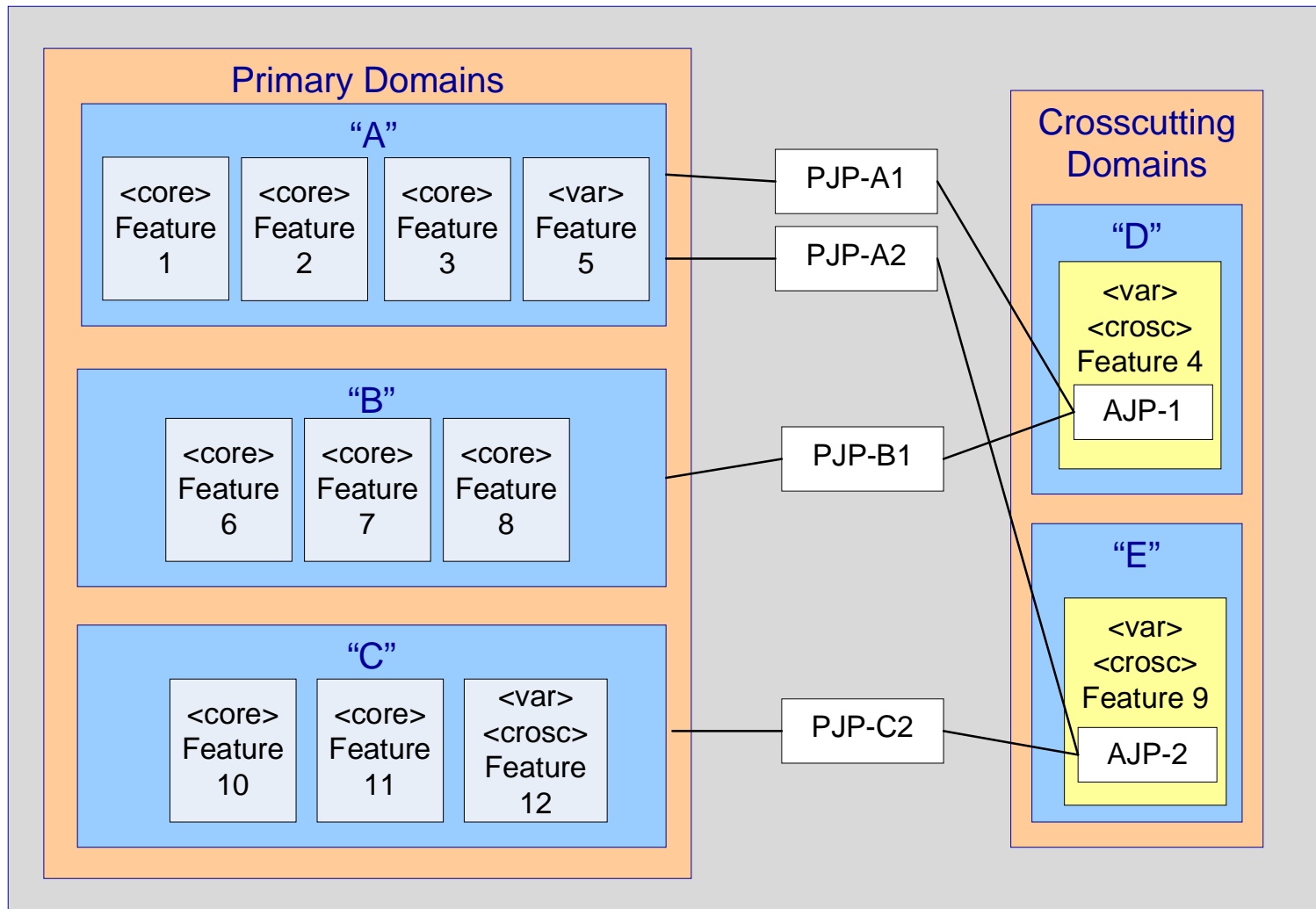


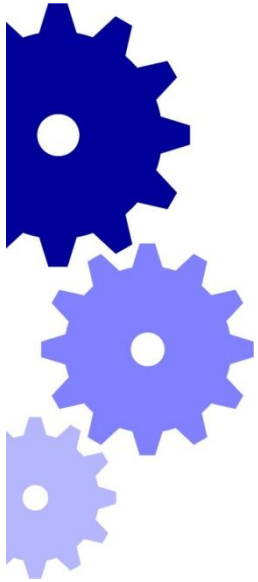
AJPs and PJPs

- Abstract Join Points (AJPs) are defined in crosscutting domain
 - They represent the points where the aspect code will intercept the base code
- Predefined Join Points (PJPs) can be defined, if possible
 - They represent predefined instantiations of AJPs for certain combinations of primary and crosscutting domains



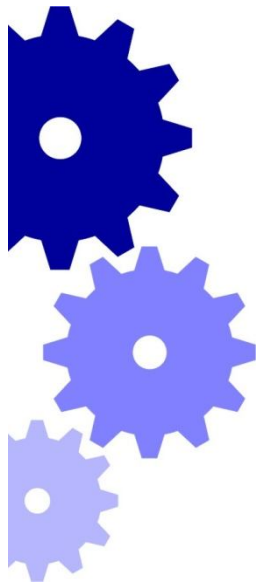
AJPs and PJPs



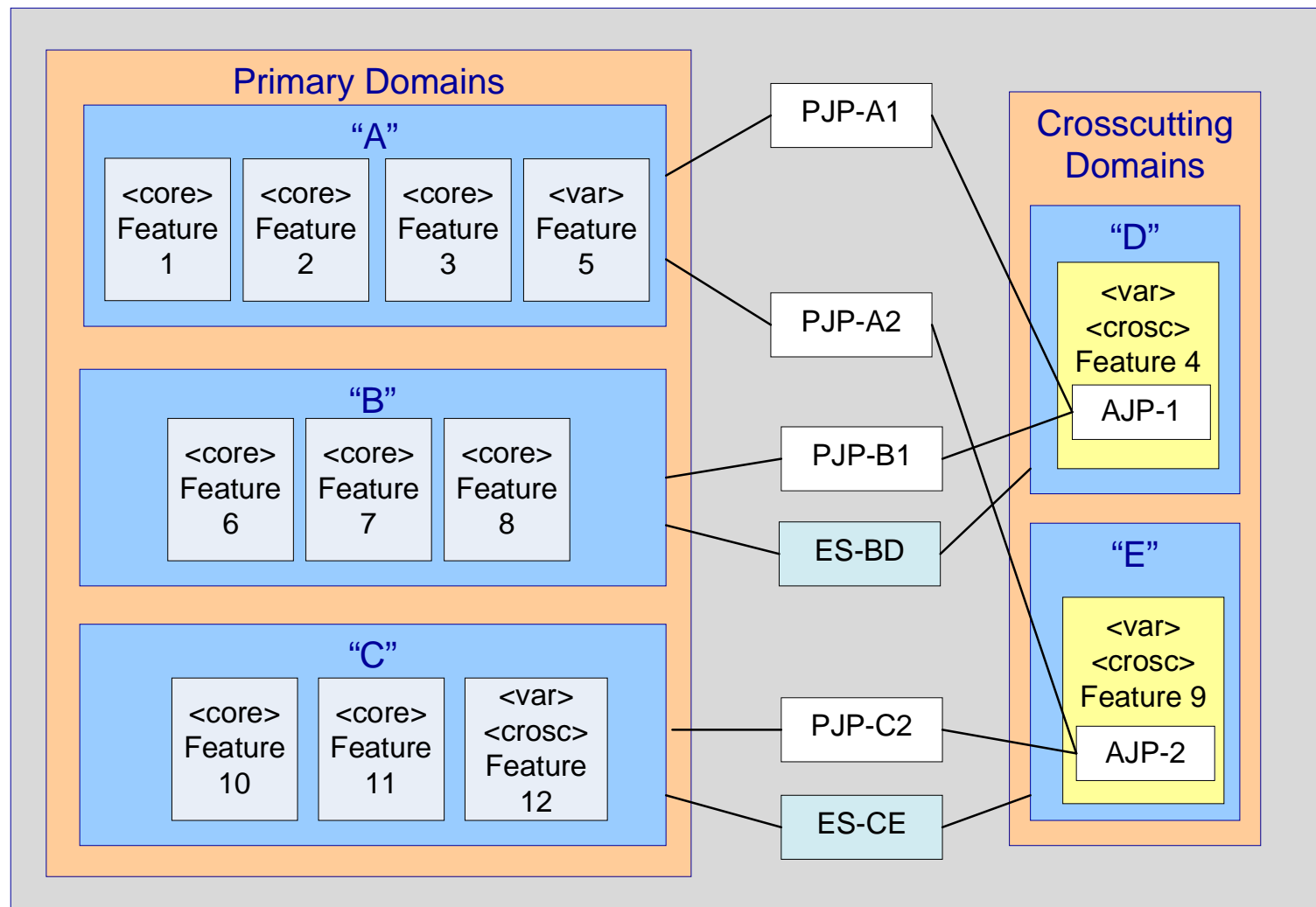


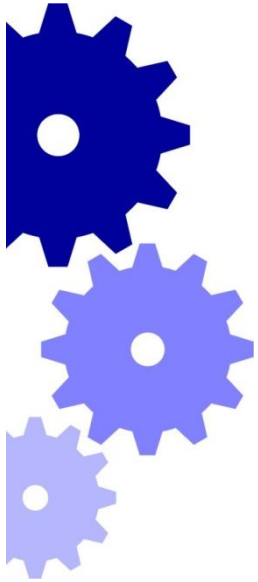
Extension Sets

- Extension Sets (ESs) can be defined in primary domains, according to possible combinations with crosscutting domains
 - They represent pieces of code that will be included depending on the composition
 - They are useful when it is difficult to determine join points in the primary domains
 - Non-obliviousness - Primary domains are aware of the presence of crosscutting domains.



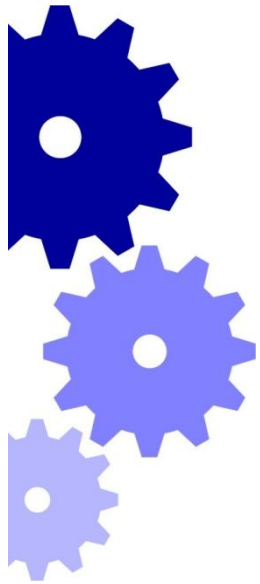
Extension Sets



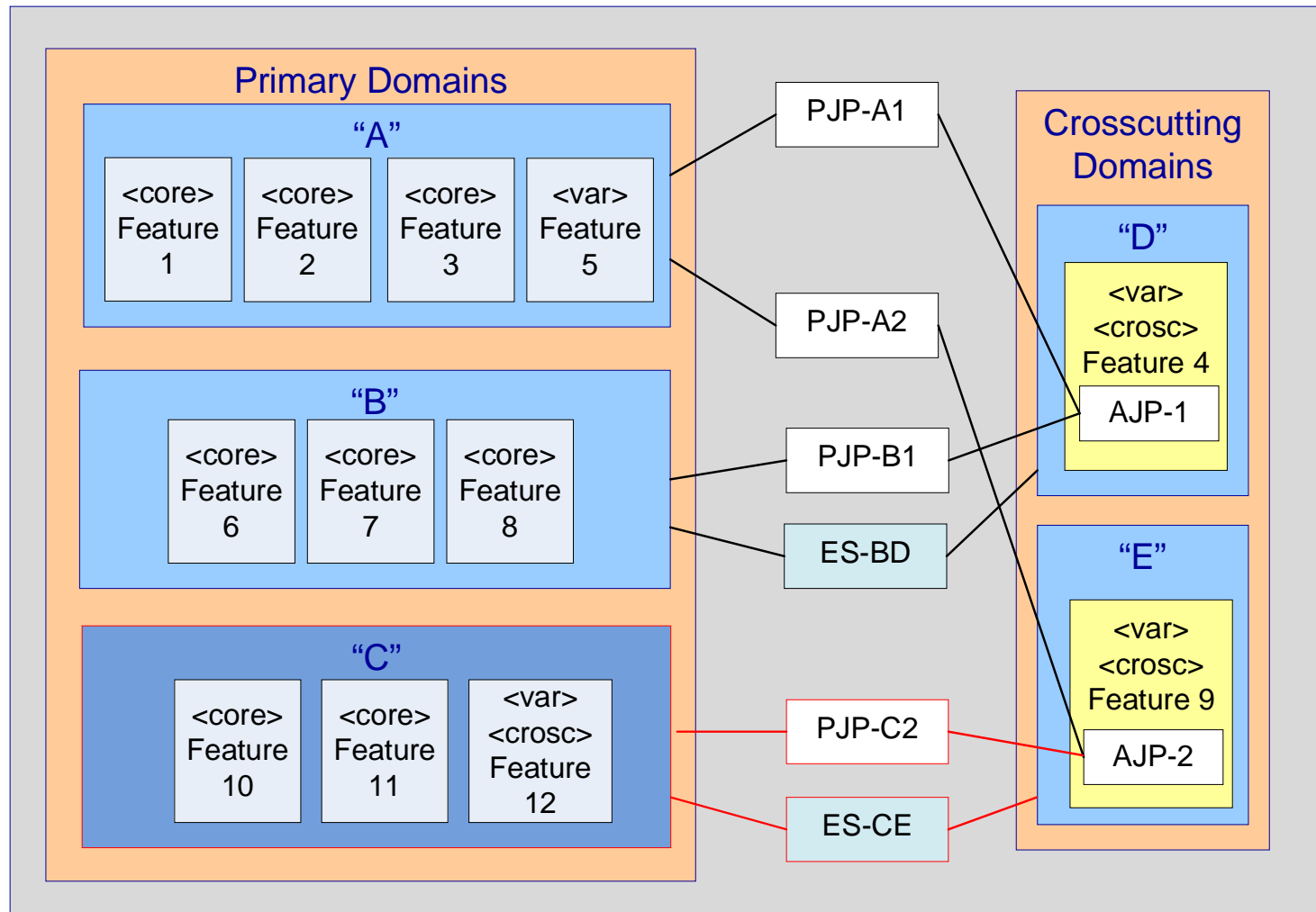


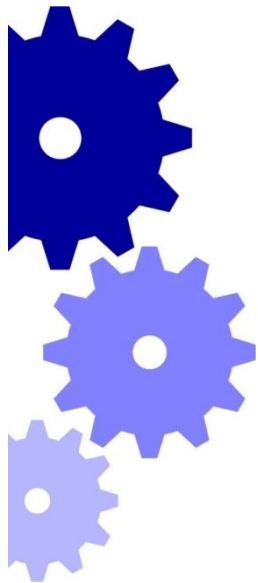
Scenarios of SPL Evolution

- New Primary Domains
- New Crosscutting Domains
- New combinations of Primary X Crosscutting Domains

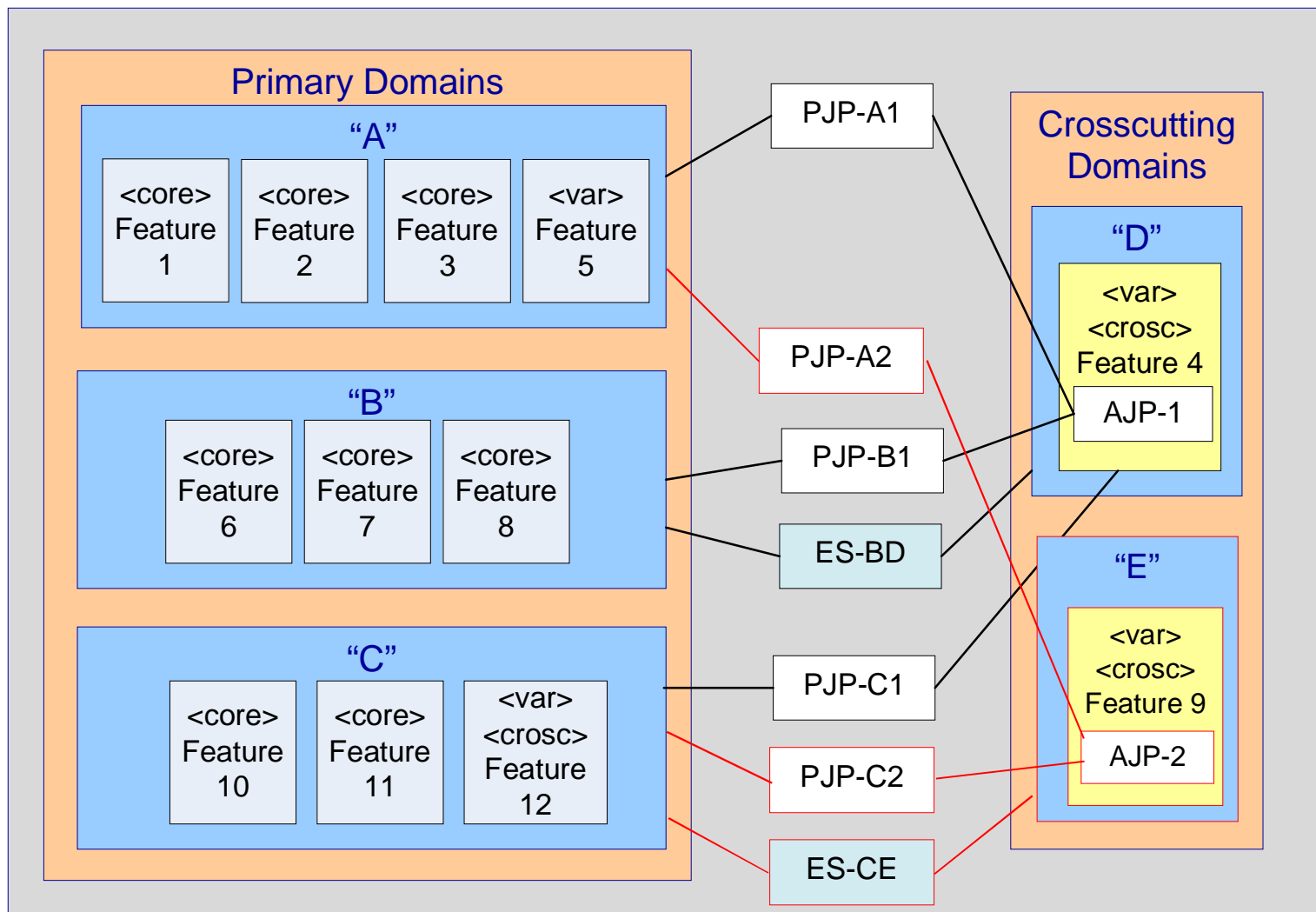


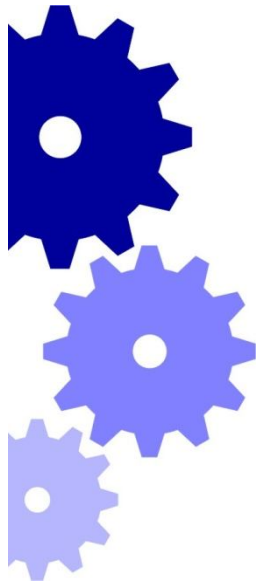
Scenarios of SPL Evolution: New Primary Domains



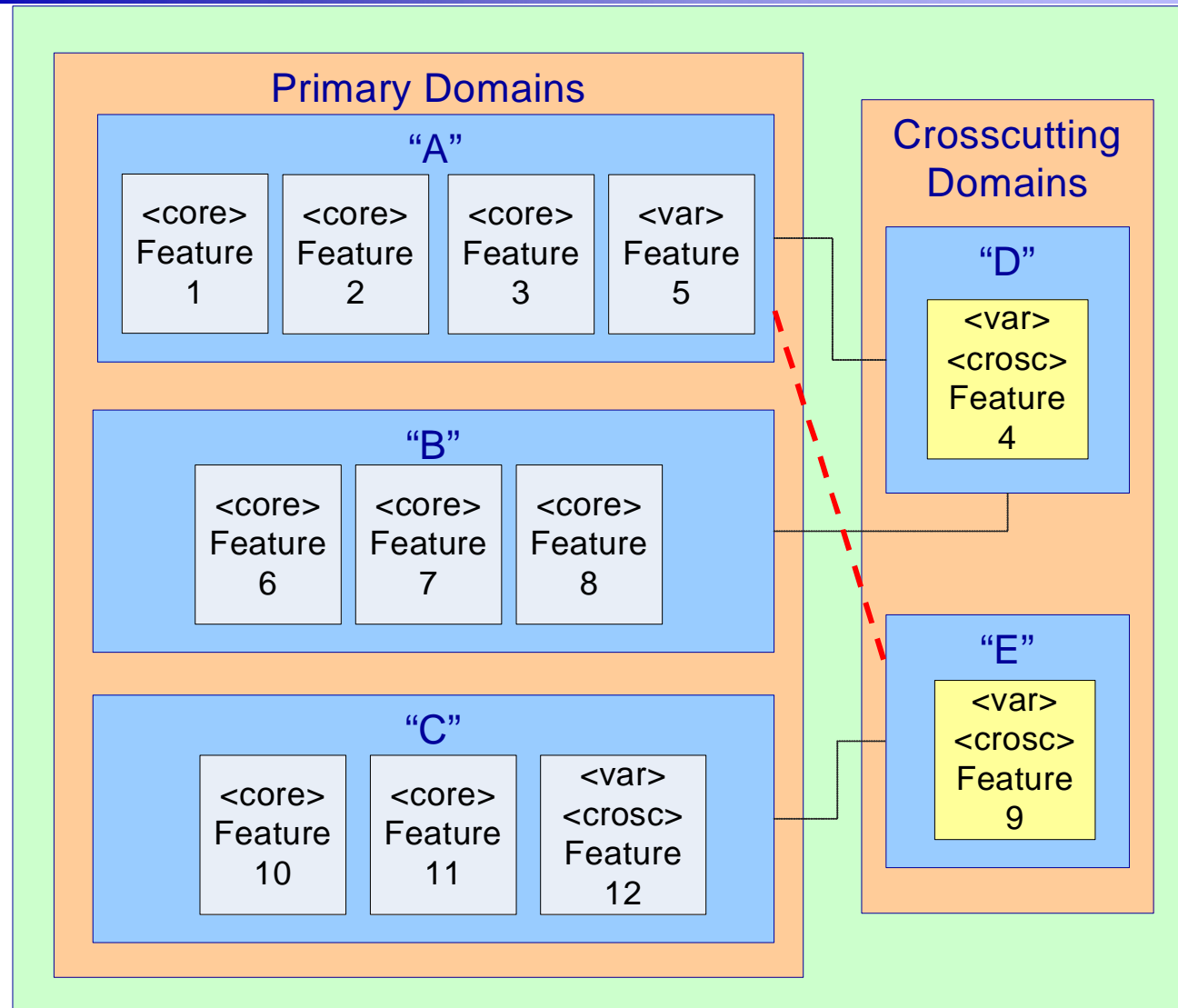


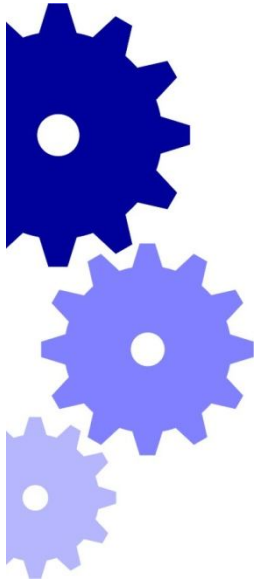
Scenarios of SPL Evolution: New Crosscutting Domains





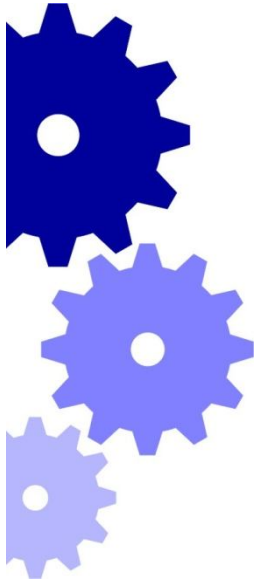
Scenarios of SPL Evolution: New combinations of Prim. X Crossc.Domains





Tool support: Captor-AO

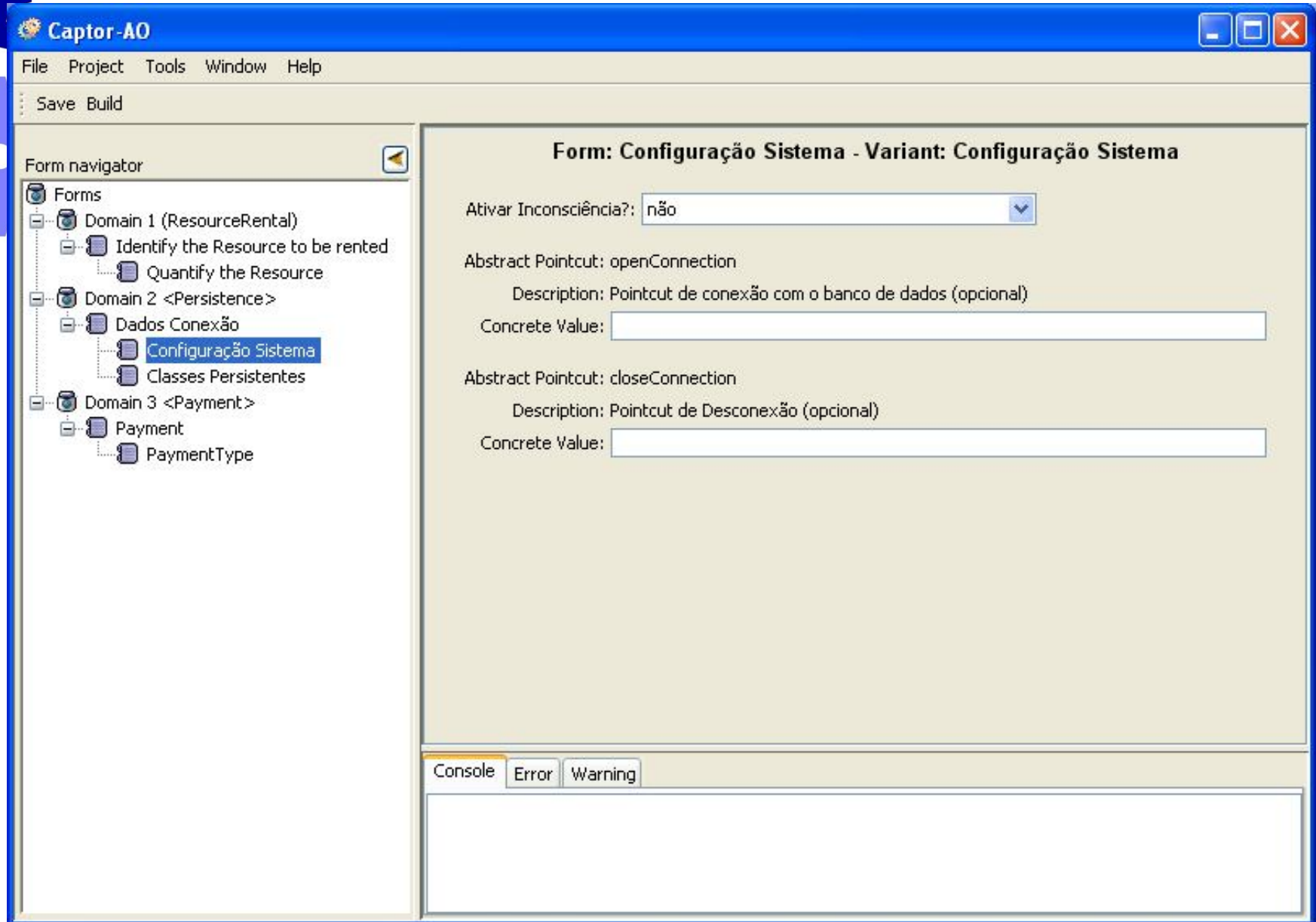
- CAPTOR-AO = Configurable
Application generaTOR – Aspect
Oriented
 - It is an application generator that can be configured to generate applications for different domains, including crosscutting domains
 - Based on templates composition
 - Based on a first version called Captor (Shimabukuro, 2006)



Captor-AO

- CAPTOR-AO = Configurable **AP**plication genera**TOR** – **Aspect Oriented**
 - AML: Application Modeling Language
 - defines the variabilities of the domain
 - focuses on SPL variabilities
 - each domain (primary or crosscutting) has its own AML
 - Software Product Lines are developed by combining **one** primary domain and **zero or more** crosscutting domains
 - Captor-AO automatically generates products based on the AML instances.

Application Engineering



The screenshot displays the Captor-AO application window. The title bar reads "Captor-AO" with standard window controls. The menu bar includes "File", "Project", "Tools", "Window", and "Help". Below the menu bar is a toolbar with "Save" and "Build" buttons. The main interface is divided into two panes. The left pane, titled "Form navigator", shows a hierarchical tree of forms: "Forms" (expanded) contains "Domain 1 (ResourceRental)" (expanded) with "Identify the Resource to be rented" and "Quantify the Resource"; "Domain 2 <Persistence>" (expanded) with "Dados Conexão" (expanded) containing "Configuração Sistema" (selected) and "Classes Persistentes"; and "Domain 3 <Payment>" (expanded) with "Payment" and "PaymentType". The right pane, titled "Form: Configuração Sistema - Variant: Configuração Sistema", contains the following fields: "Ativar Inconsciência?:" with a dropdown menu showing "não"; "Abstract Pointcut: openConnection" with a description "Description: Pointcut de conexão com o banco de dados (opcional)" and a "Concrete Value:" text box; and "Abstract Pointcut: closeConnection" with a description "Description: Pointcut de Desconexão (opcional)" and a "Concrete Value:" text box. At the bottom of the window is a console area with tabs for "Console", "Error", and "Warning", currently showing the "Console" tab.

Captor-AO

File Project Tools Window Help

Save Build

Form navigator

- Forms
 - Domain 1 (ResourceRental)
 - Identify the Resource to be rented
 - Quantify the Resource
 - Domain 2 <Persistence>
 - Dados Conexão
 - Configuração Sistema**
 - Classes Persistentes
 - Domain 3 <Payment>
 - Payment
 - PaymentType

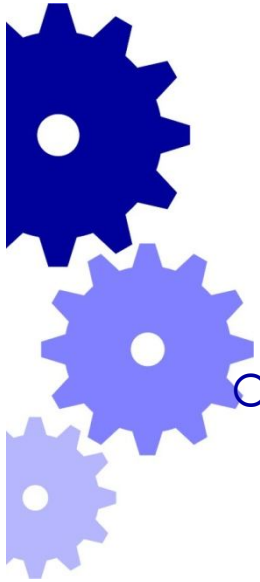
Form: Configuração Sistema - Variant: Configuração Sistema

Ativar Inconsciência?: não

Abstract Pointcut: openConnection
Description: Pointcut de conexão com o banco de dados (opcional)
Concrete Value:

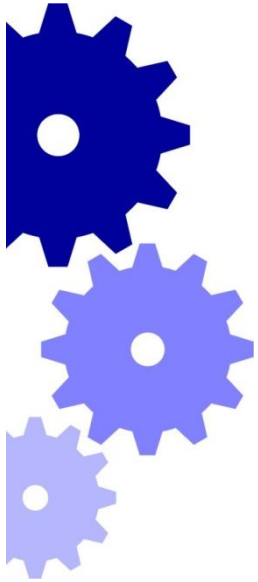
Abstract Pointcut: closeConnection
Description: Pointcut de Desconexão (opcional)
Concrete Value:

Console Error Warning



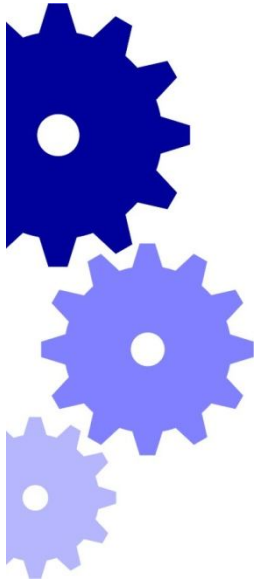
Conclusions

- The use of crosscutting domains allow the reuse of crosscutting concerns present in different SPLs, avoiding the redunding implementation of features.
- The modularization of the joinpoint variabilities eases the combination between aspects of a crosscutting domain and primary domain.
- Captor-AO supports the creation of a repository of primary and crosscutting domains, easing the reuse accross domains.
- SPL evolution can be enhanced, since new features can be included with less impact on existing ones.



Future work

- Allow the use of two or more primary domains in the combination process
- Enhance the process of testing the compatibility among domains
- Create mechanisms to validate the join points where crosscutting domains intercept primary domains
- Conduct more experiments to validate Captor-AO



Contact

- E-mail
 - rtvb@icmc.usp.br (Rosana)

- Web page
 - <http://captor.googlecode.com>