Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação

Design Issues in a Component-based Software Product Line

Paula M. Donegan, Paulo C. Masiero

{donegan, masiero}@icmc.usp.br

Fundação de Amparo à Pesquisa do Estado de São Paulo

LabES – Laboratório de Engenharia de Software

Summary

- Introduction and objectives
- SPL to control Electronic Transportation Cards (ETC-SPL)
- Development process of SPLs
- Development of the ETC-SPL
- Design decisions for features of the SPL
- Using a code generator
- Final considerations

Introduction

- The design of an SPL can use various design techniques that facilitate reuse:
  - components, code generators, features diagrams, etc.

- Difficulty of gathering, representing and implementing variabilities in SPLs

Objectives

- Illustrate different solutions based on components to represent variabilities of an SPL
- Discuss how these solutions are influenced by:

  i) the adopted development process

  ii) the decision to use black-box or white-box components

  iii) the decision of automating the composition process

The Electronic Transportation Cards Software Product Line (ETC-SPL)

- Integration and automation of the transport network
- Maintain the data of passengers, cards, routes, buses and journeys
- Validator on bus reads a card and communicates with the central system to debit the fare on the passenger's card
- There may be a bus integration system so that the user can pay a single fare for multiple trips
- Analysis of 3 ETC systems in Brazilian cities:
  - São Carlos (São Paulo)
  - Fortaleza (Ceará)
  - Campo Grande (Mato Grosso do Sul)

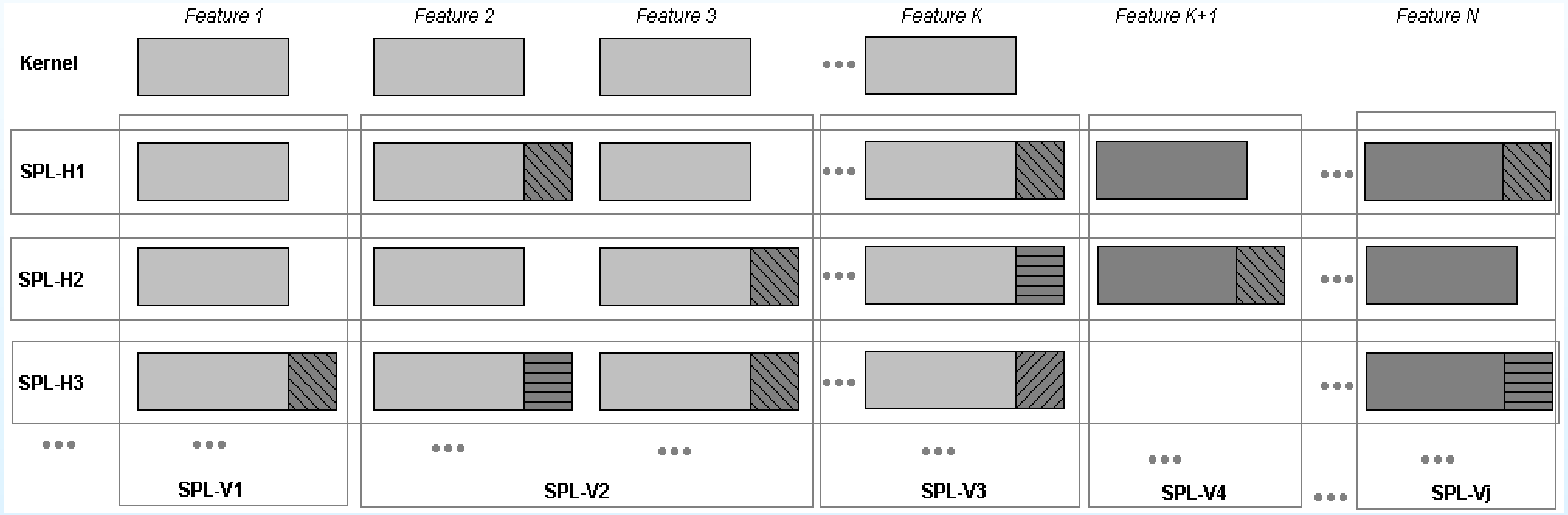
Development Process of SPLs

- Begin with the domain analysis
- Then there are 2 alternatives:

  1) Elaborate the design for the <u>entire</u> domain and implement afterwards (in one version or in various increments)

  2) Design and implement the SPL in a version only with kernel features, and then <u>increment</u> the design and implementation of <u>subgroups</u> of optional and alternative variabilities

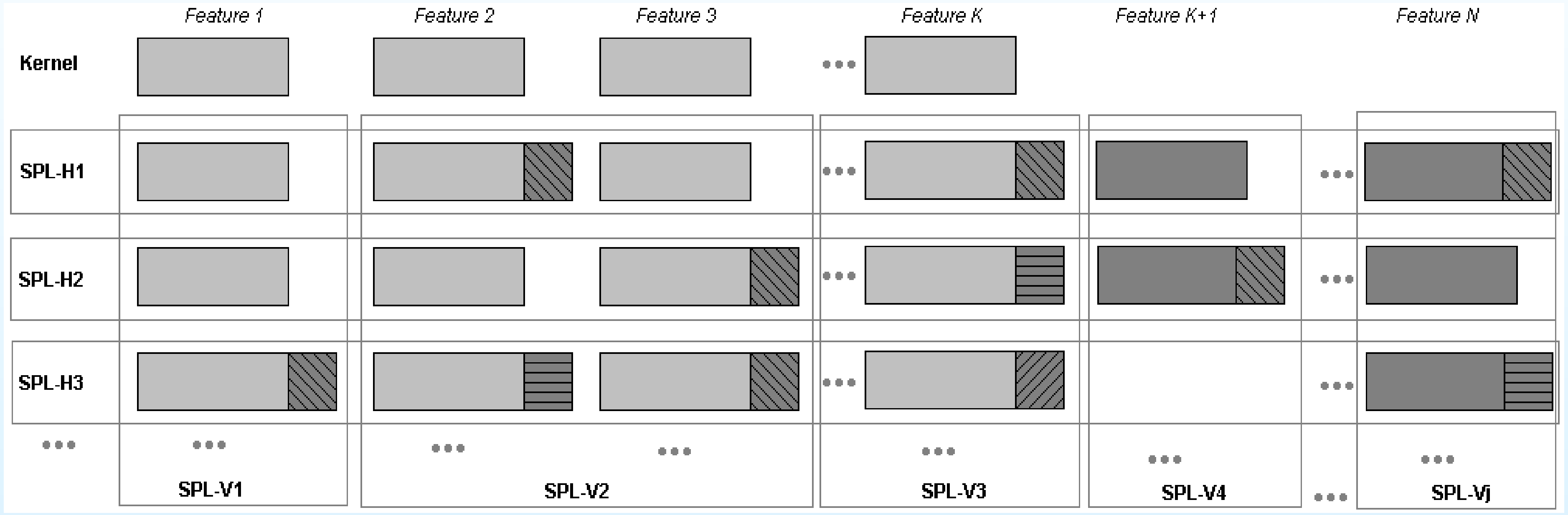
Increments of SPLs



Vertical and horizontal increments

- <u>Horizontal increments</u>
  - Include a subgroup of features that attend to a specific application but do not necessarily contain all possible variabilities of each feature included

Increments of SPLs



Vertical and horizontal increments

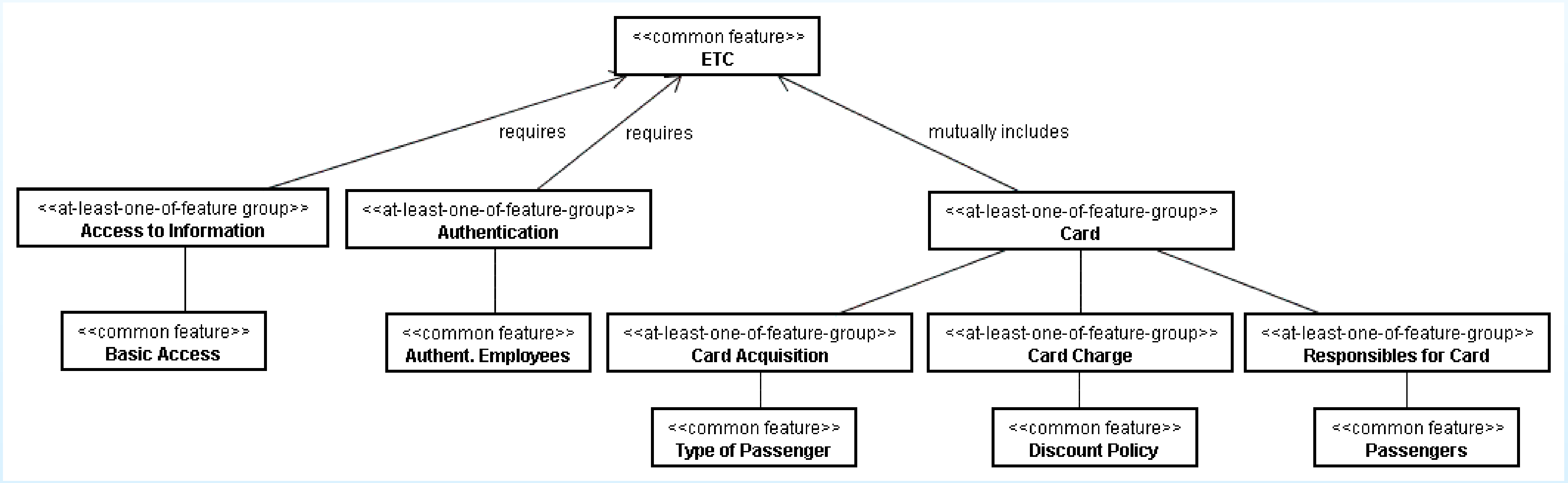
- <u>Vertical increments</u>
    - Implement all the variabilities of a subgroup of chosen features, but do not necessarily produce a specifically desired application

Development of the ETC-SPL

- We considered it important to have a complete application early on:
  - Option of using horizontal iterative cycles generating one application in each increment

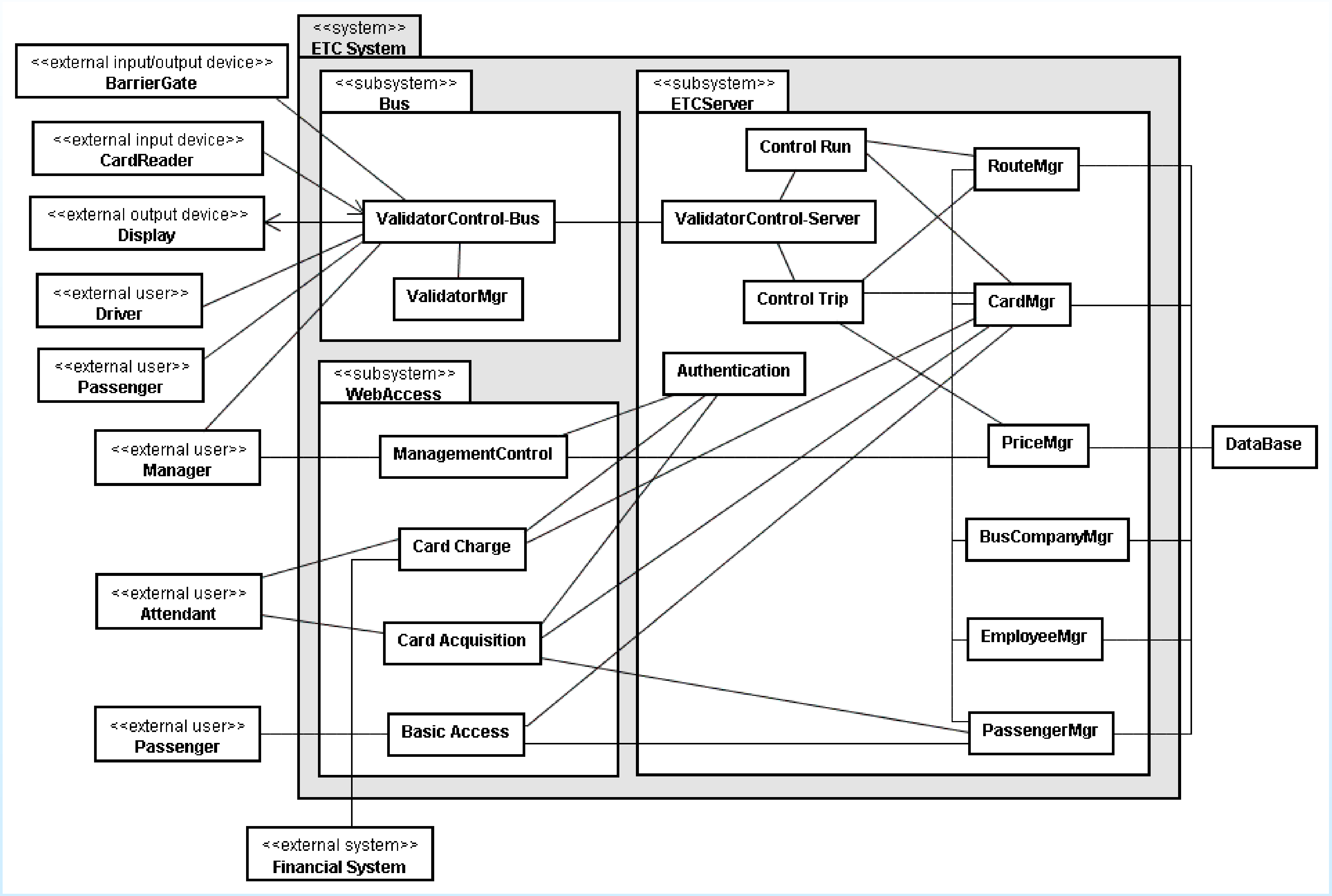
Iteration 1	Comprising only features of the kernel (Version 1)
Iteration 2	Version 1 + features and variabilities of the application of *Fortaleza*
Iteration 3	Version 2 + features and variabilities of the application of *Campo Grande*
Iteration 4	Version 3 + features and variabilities of the application of *São Carlos*
Iteration 5	Version 4 with all variabilities + automatically generated with an *Application Generator*

Development of the ETC-SPL


<<common feature>> ETC
requires
requires
mutually includes
<<at-least-one-of-feature group>> Access to Information
<<at-least-one-of-feature-group>> Authentication
<<at-least-one-of-feature-group>> Card
<<common feature>> Basic Access
<<common feature>> Authent. Employees
<<at-least-one-of-feature-group>> Card Acquisition
<<at-least-one-of-feature-group>> Card Charge
<<at-least-one-of-feature-group>> Responsibles for Card
<<common feature>> Type of Passenger
<<common feature>> Discount Policy
<<common feature>> Passengers

Features diagram for the kernel ETC-SPL

Development of the ETC-SPL



Kernel architecture of the ETC-SPL

Additional Features of the ETC systems

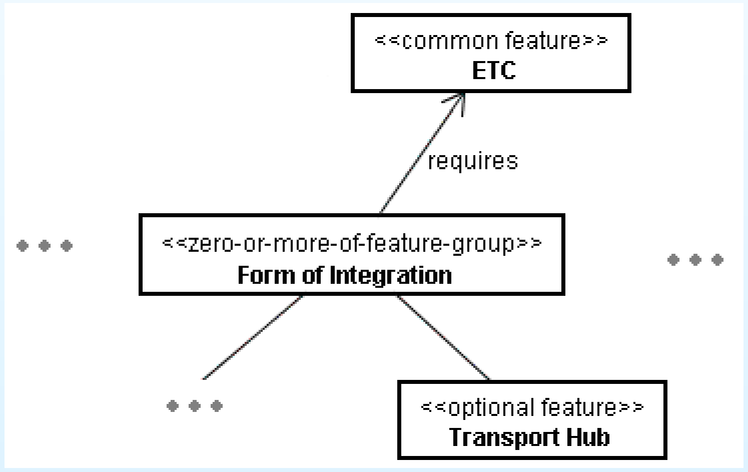
Fortaleza	Campo Grande	São Carlos
	Additional Access	Additional Access
		Passenger Authentication
Form of Integration - Transport Hub	Form of Integration - Transport Hub - Integration * Time * Integration Route * Number of Integration Trips	Form of Integration - Integration * Time * Integration Route
Card Payment		
	Card Restriction - Number of Cards	Card Restriction - Card Combination
User Companies	User Companies	
		Trips Limit

Design Decisions for Features of the SPL

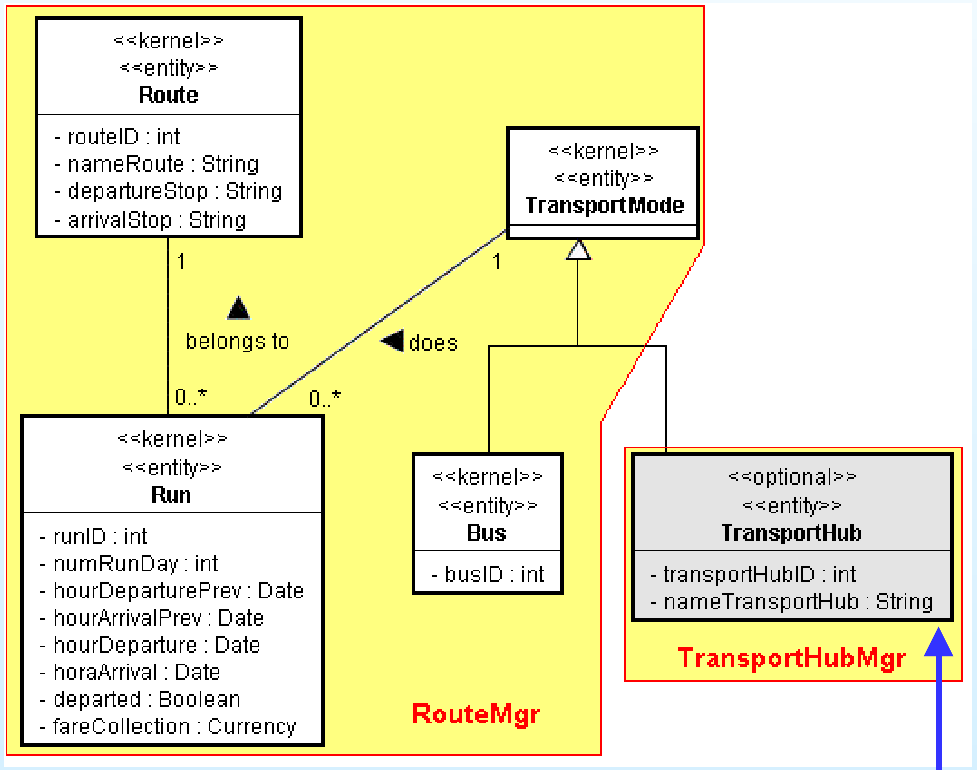
- How design decisions are influenced by:
  - the decisions taken related to the SPL development process adopted
  - the type of component
  - the manner of composition (manual or automated)

- Features
  - *Form of Integration:* uses new classes
  - *Card Payment:* uses subclasses (with new attributes and methods)

Feature: *Transport Hub*



Part of the features diagram

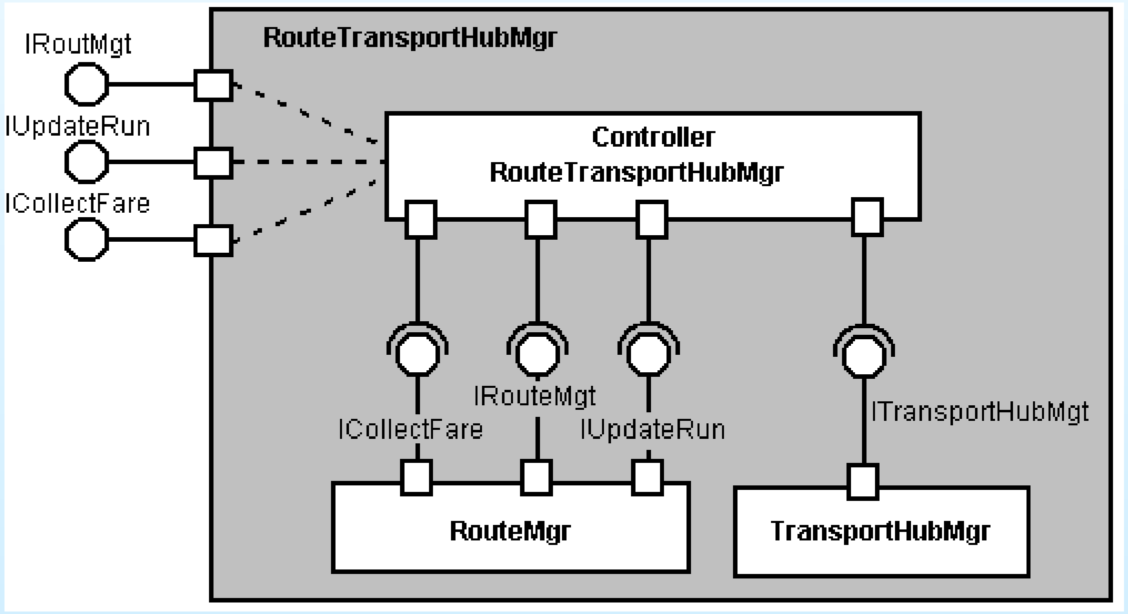


Fragment of the class model

New class required

Feature: *Transport Hub*

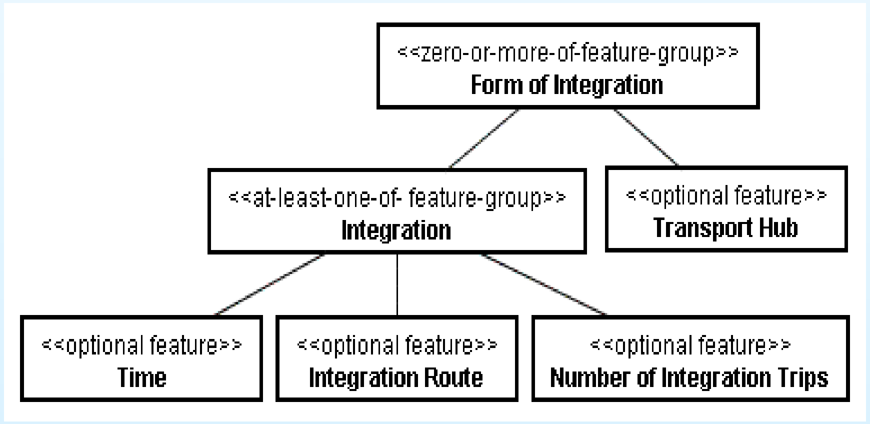
- Without internal access to the implementation of developed components
- *RouteMgr* is reused without any alteration
- Fortaleza version:



Composed component *RouteTransportHubMgr*

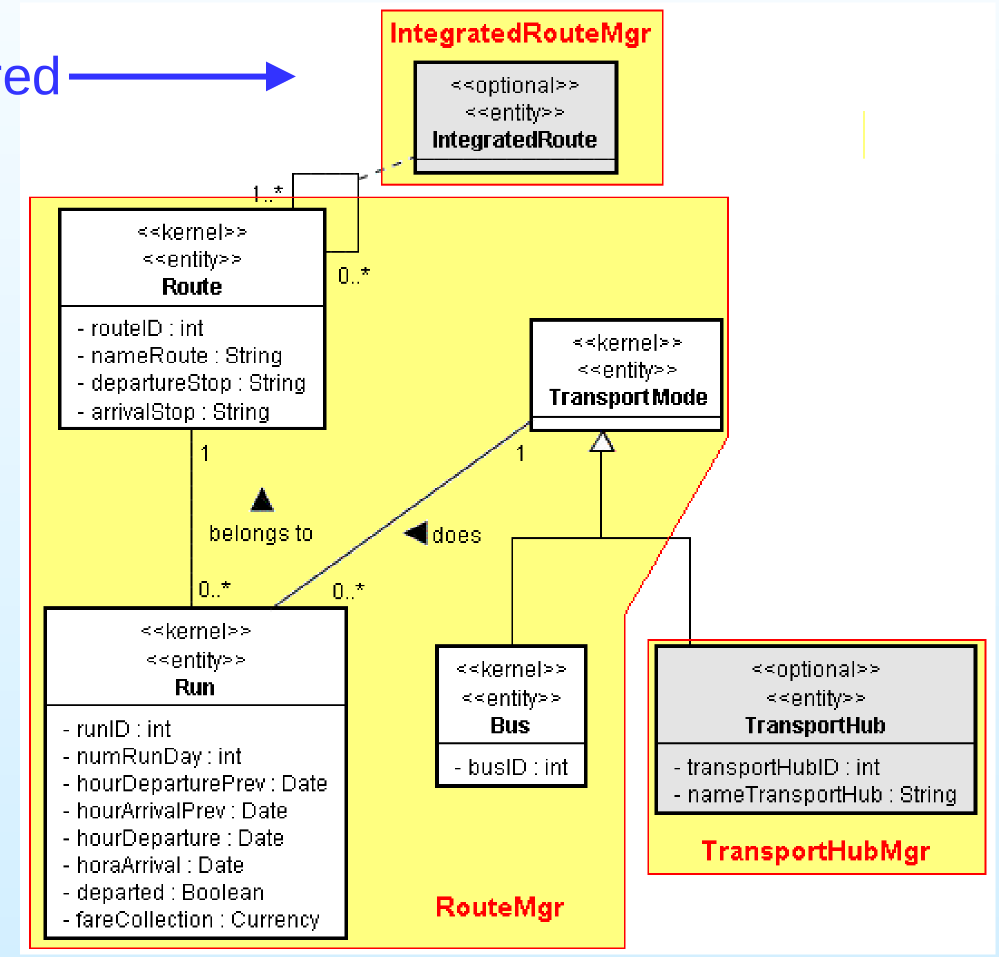
Feature: *Integration Route*

New class required



<<zero-or-more-of-feature-group>> Form of Integration
<<at-least-one-of- feature-group>> Integration
<<optional feature>> Transport Hub
<<optional feature>> Time
<<optional feature>> Integration Route
<<optional feature>> Number of Integration Trips

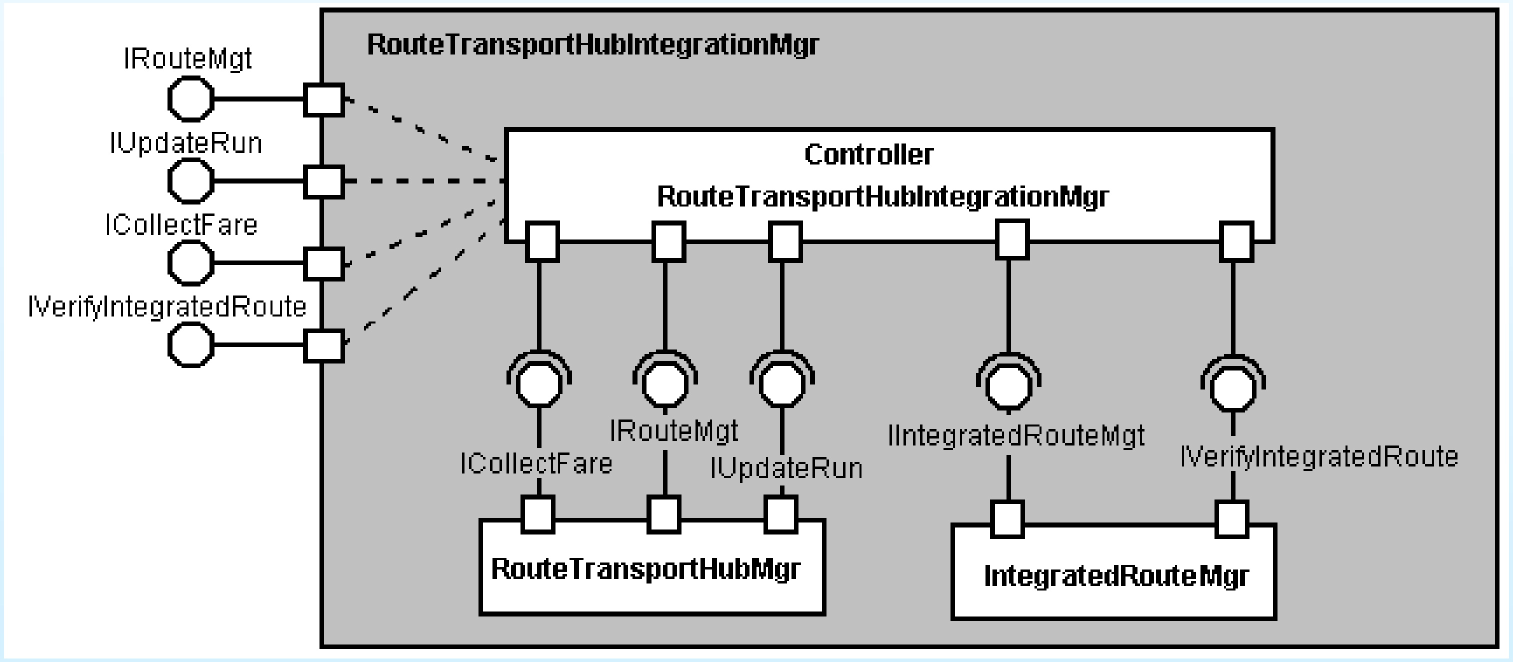
Part of the features diagram



Fragment of the class model

Feature: *Integration Route*

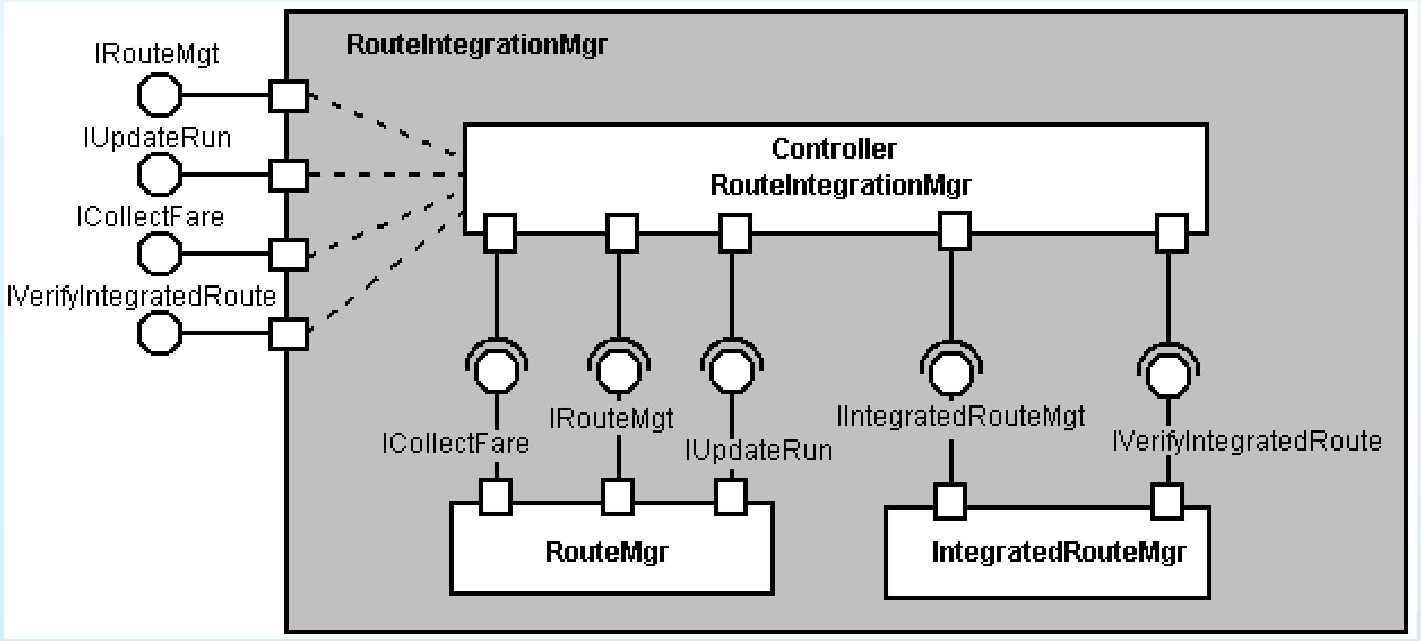
- *RouteTransportHubMgr* developed for Fortaleza is reused
- Campo Grande version:



Composed component *RouteTransportHubIntegrationMgr*

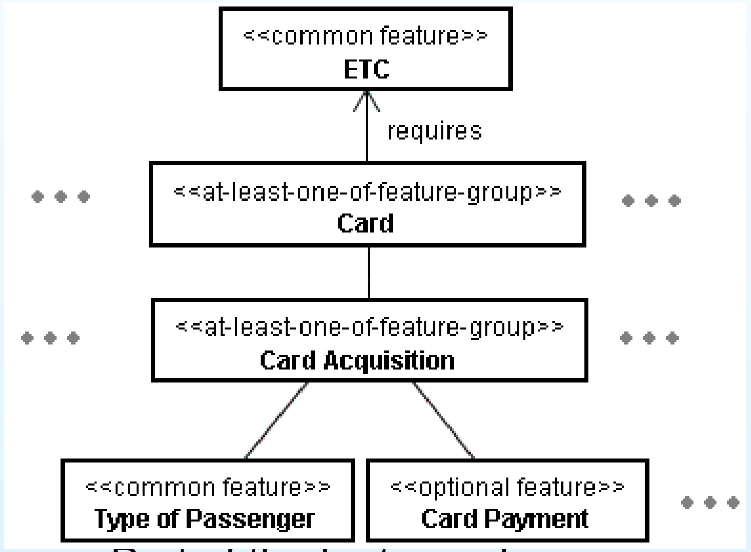
Feature: *Integration Route*

- *RouteMgr* is reused
- *IntegratedRouteMgr* developed for Campo Grande is reused
- São Carlos version:



Composed component *RouteIntegrationMgr*

Feature: *Card Payment*



Part of the features diagram

- Variation points in the classes *PassengerType* and *Payment*
  - Altering attributes and operations of these classes (not necessary to insert a new class into the model)

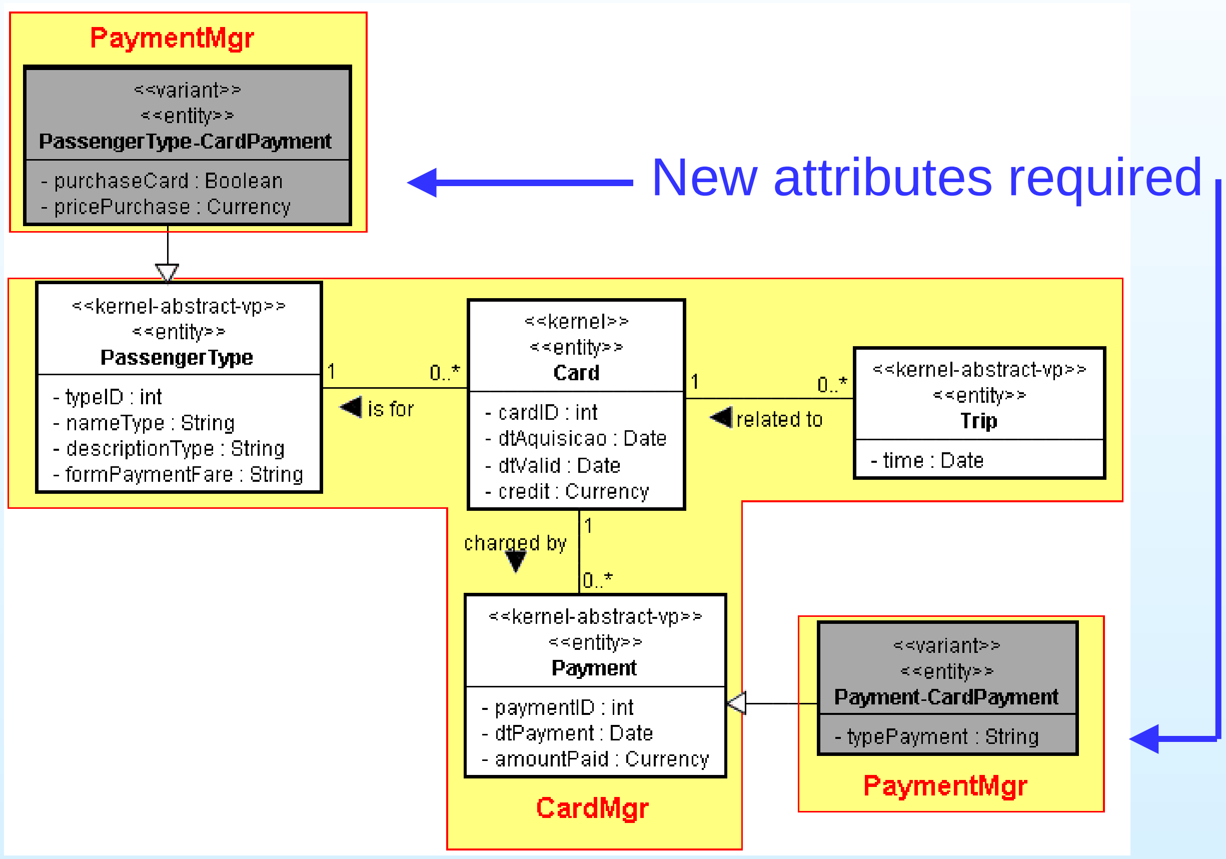
Feature: *Card Payment*

Option 1: Use parameterized classes

Option 2: Use classes with variation points and separate the *Card Payment* feature in a new component called *PaymentMgr*

- Interests separated and black-box components

Feature: *Card Payment*

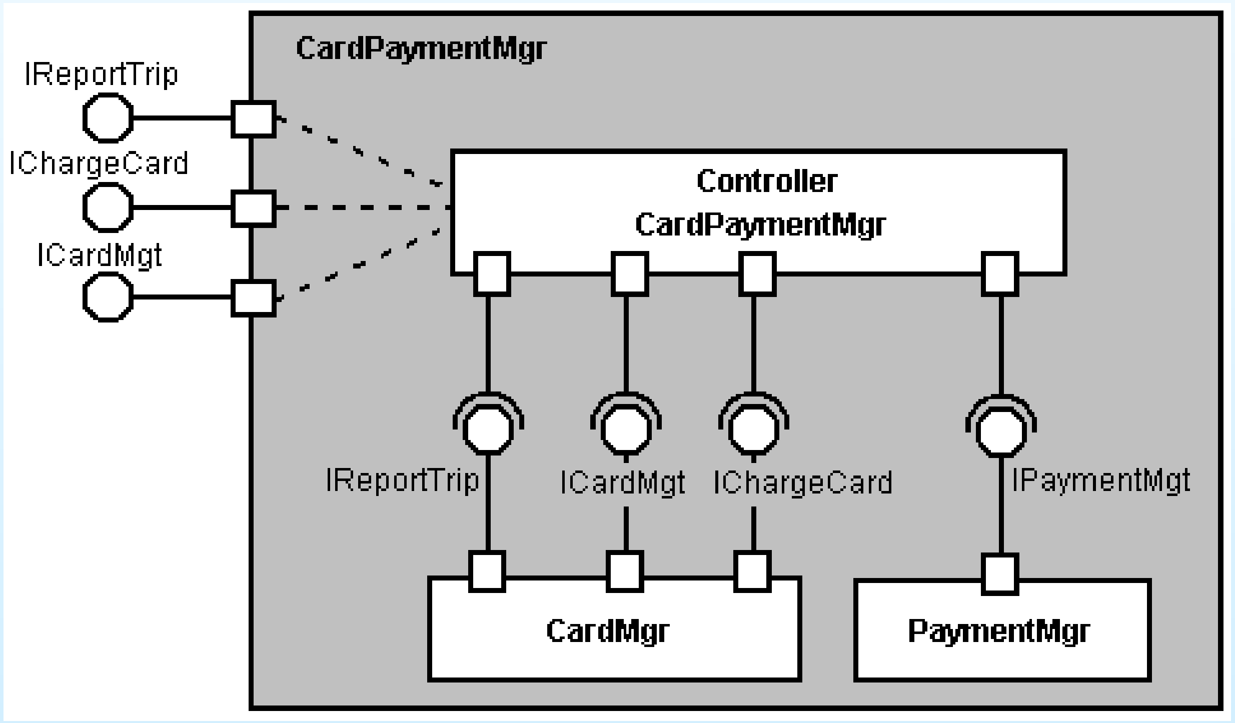


Fragment of the class model

- Both classes stay in one component because they have the same interest and are always used together

Features: *Card Payment*

- *CardMgr* is reused without any alteration
- <u>Fortaleza</u> version:



Composed component *CardPaymentMgr*

Using a Code Generator

- List of features: initial sketch of the Application's Modeling Language (AML)
- <u>White-box components</u>:
  - Generator performs changes inside each component thereby generating additional classes and modifying other elements inside the components
  - The generator would be much more complex and act as a <u>*composer*</u>
- <u>Black-box components</u>:
  - Generator acts like a <u>*configurator*</u>, starting from the kernel architecture, replacing and including necessary components, and generating glue code for components being composed

Using a Code Generator (II)

- Automating the composition process influences the design as well as the moment of introducing the automation in the SPL
  - If automation is used from the first version
    - Design of new versions of the SPL is influenced
    - Each new horizontal iteration requires considerable rework in the generator
- We intend to use the configurable code generator Captor developed by our research group

Final Considerations

- Development of the ETC-SPL:
  - The kernel and version 2 (Fortaleza) have already been designed
    - Some other features have also been designed vertically with the intention of investigating different solutions
    - The implementation of the kernel is ongoing
- Having decided to evolve the line in horizontal iterations:
  - Important to take some time to analyse how feature groups will evolve in the following iterations before committing to a design that cannot be easily modified or reused
- The examples show trade-offs between horizontal and vertical development
- The decision of using black box or white box components is crucial

Thanks for your attention!

Questions?

Contact:

Paula Donegan: donegan@icmc.usp.br

Paulo C. Masiero: masiero@icmc.usp.br