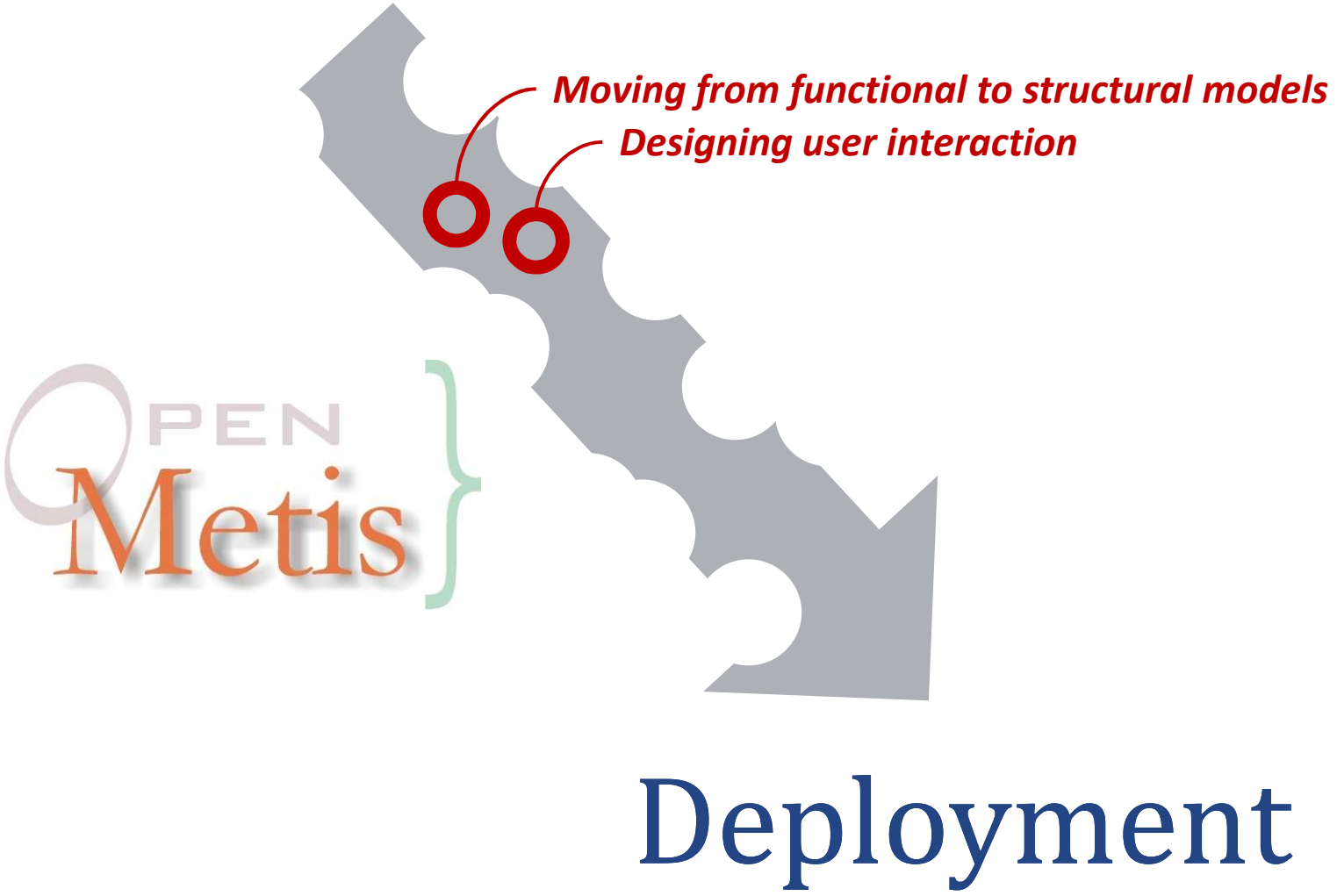
Filling the Voids: From Requirements to Deployment with OPEN/Metis

César González-Pérez

The Heritage Laboratory (LaPa)
Spanish National Research Council (CSIC)

Requirements



Deployment

The Integral Object-Oriented Software Development Framework

OPEN/Metis

What is OPEN/Metis?

A methodology for the development of business information systems

Based on the ISO/IEC 24744 metamodel

Describes the process to follow and the products to use and generate

Iterative, incremental lifecycle

Definitely not “agile”, but very capable

Brief bio of OPEN/Metis

Started up in the early 1990s in academia

Largely inspired by the Fusion method

In the late 1990s it is transferred to industry

Improved through multiple inputs

Internal usage and consulting work since then

Over 20 small and mid-sized projects

How are we supposed to use the system?

Designing User Interaction

35%

of effort on a typical project*
is spent on user interface-related tasks

* Aggregated internal, unpublished data of LAFC and Neco TI.

Common Questions from Developers

"How do I describe what happens inside the system during the performance of a use case?"

"How do I associate system behaviour to dialog boxes and screens?"

"How do I link UI-related classes, attributes, etc. to other, non-UI-related ones?"

"UML is useless here!"

Observations

UI modelling should not be an isolated task.

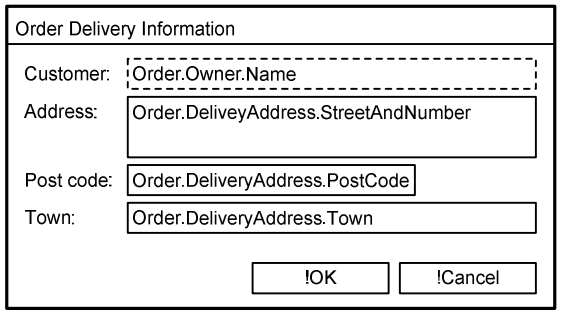
Very complex dynamics occur at the UI boundary.

UI structure is far from trivial.

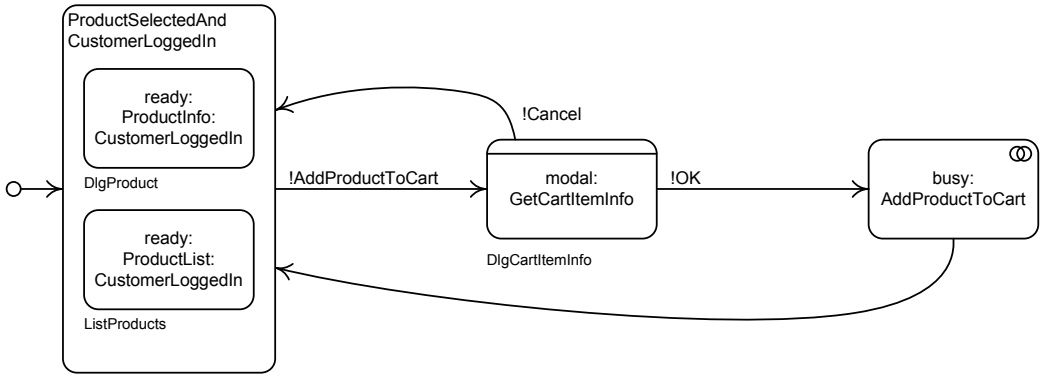
We need a language plus process guidance on UI modelling.

OPEN/Metis Solution

A **User Interface Model** describes the structural basics



A **Service Model** describes the dynamics

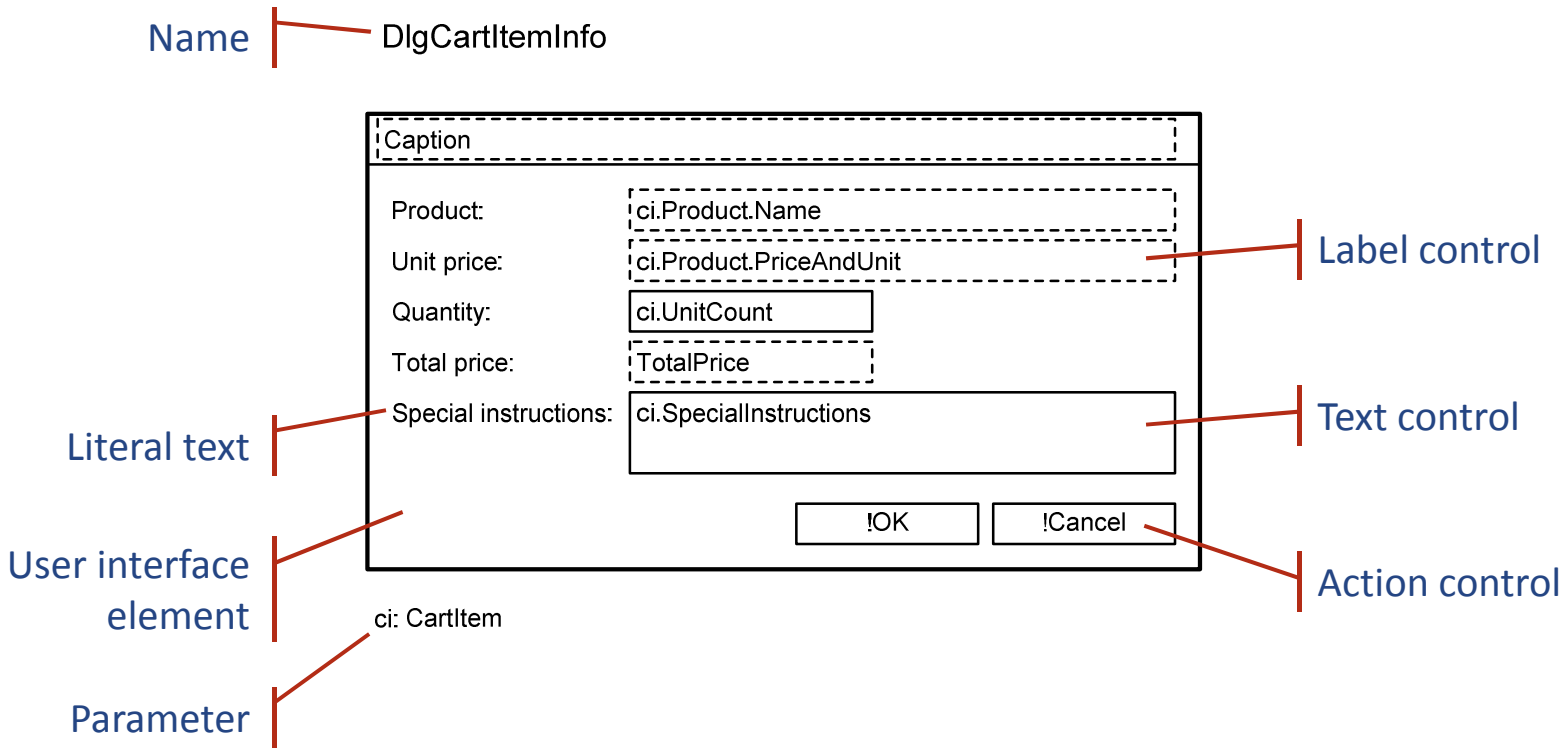


User Interface Model

Describes the basic building blocks of the user interaction from a structural perspective

Roughly corresponds to the appearance of the system

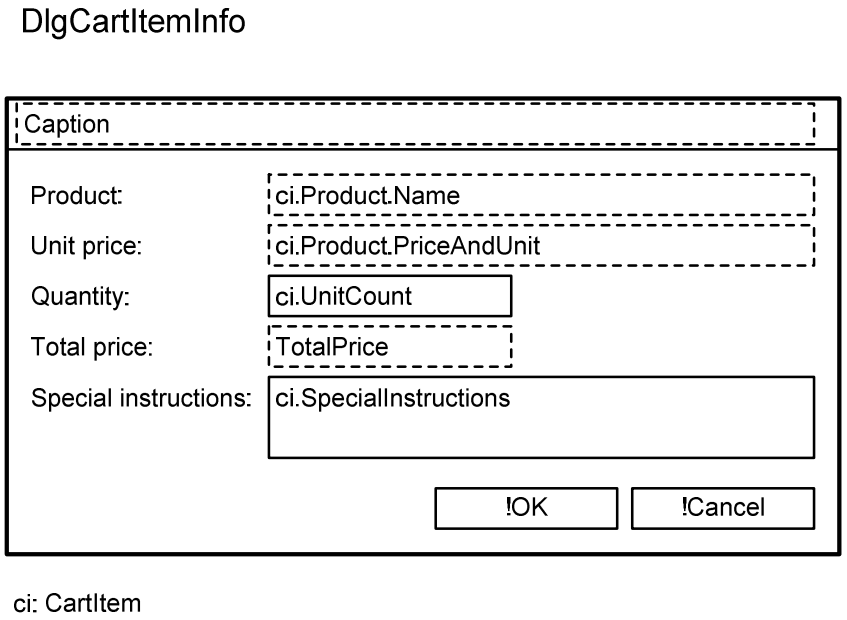
User Interface Model



User Interface Model

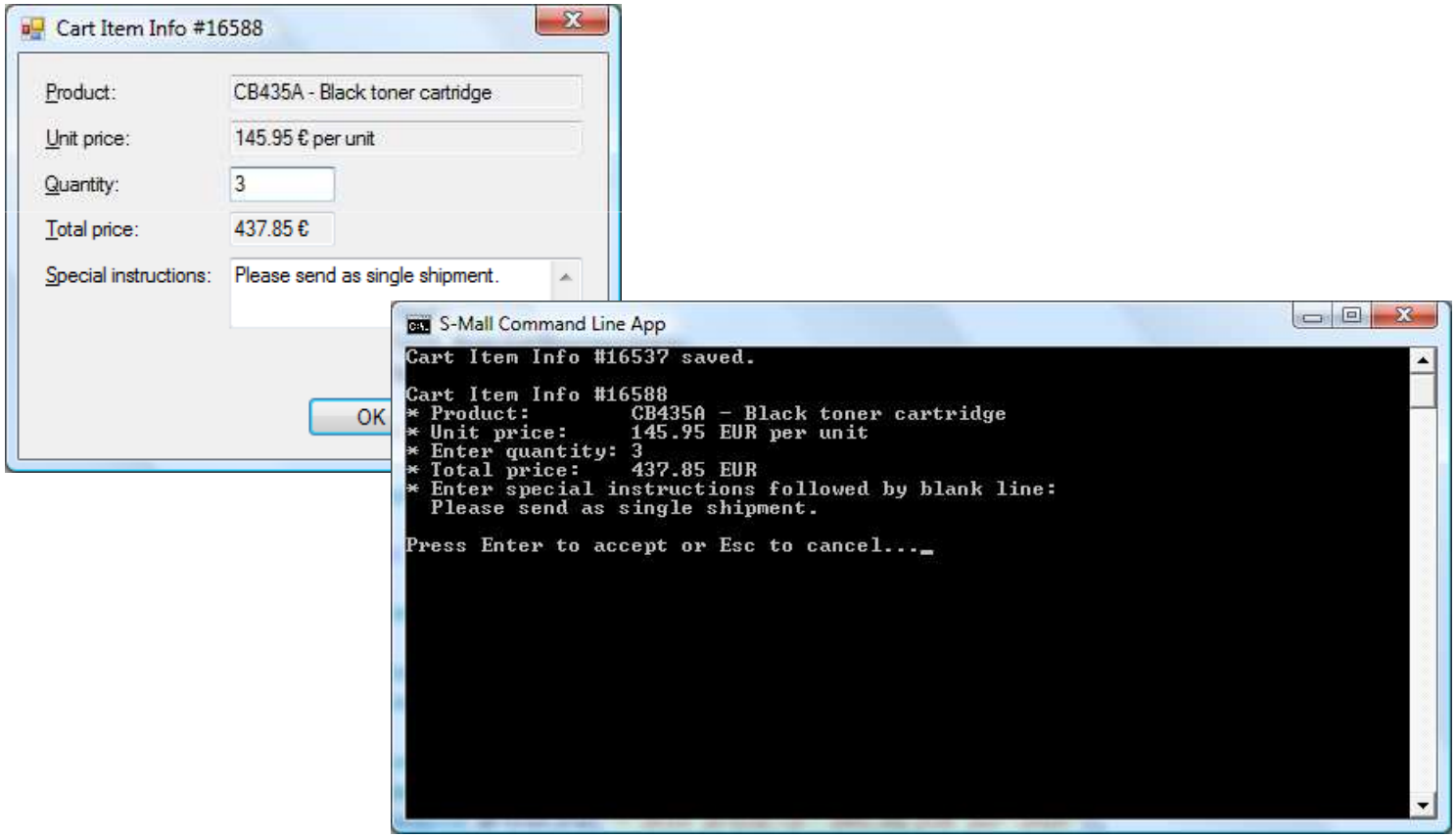
Linked to domain model via parameters

Linked to dynamics via action controls

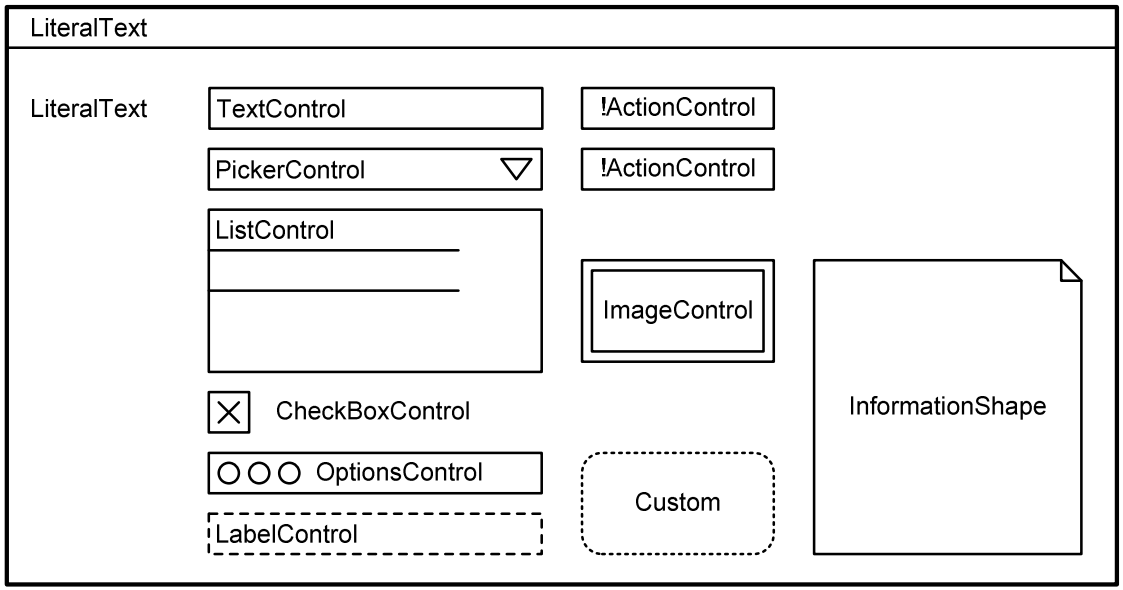


User Interface Model

Abstract specification



User Interface Models Notation



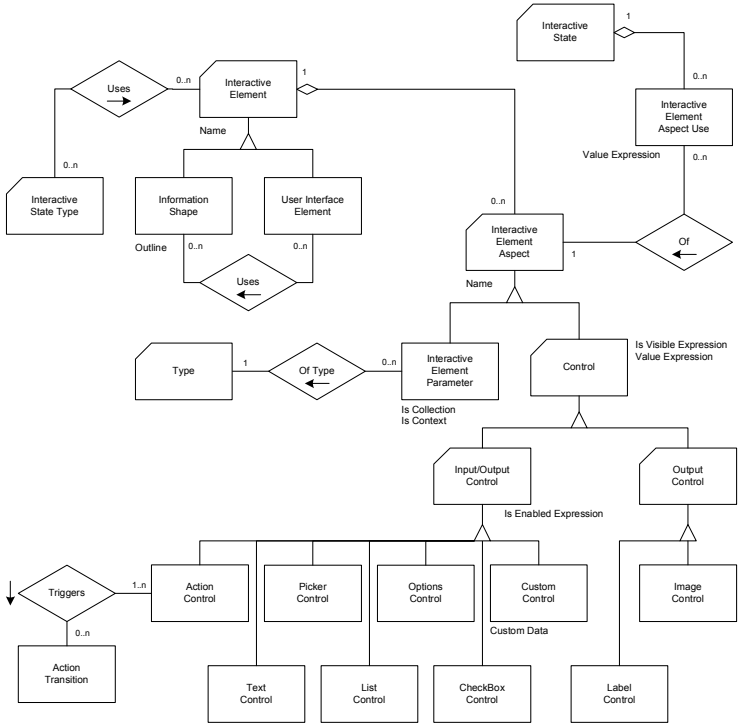
User Interface Models Language

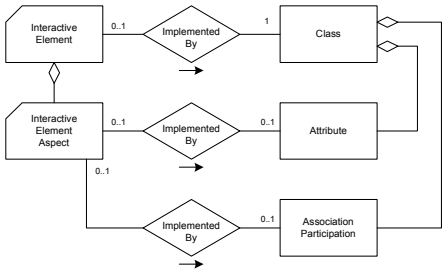
Over 20 specialised classes

Input/output controls

Output controls

Information shapes





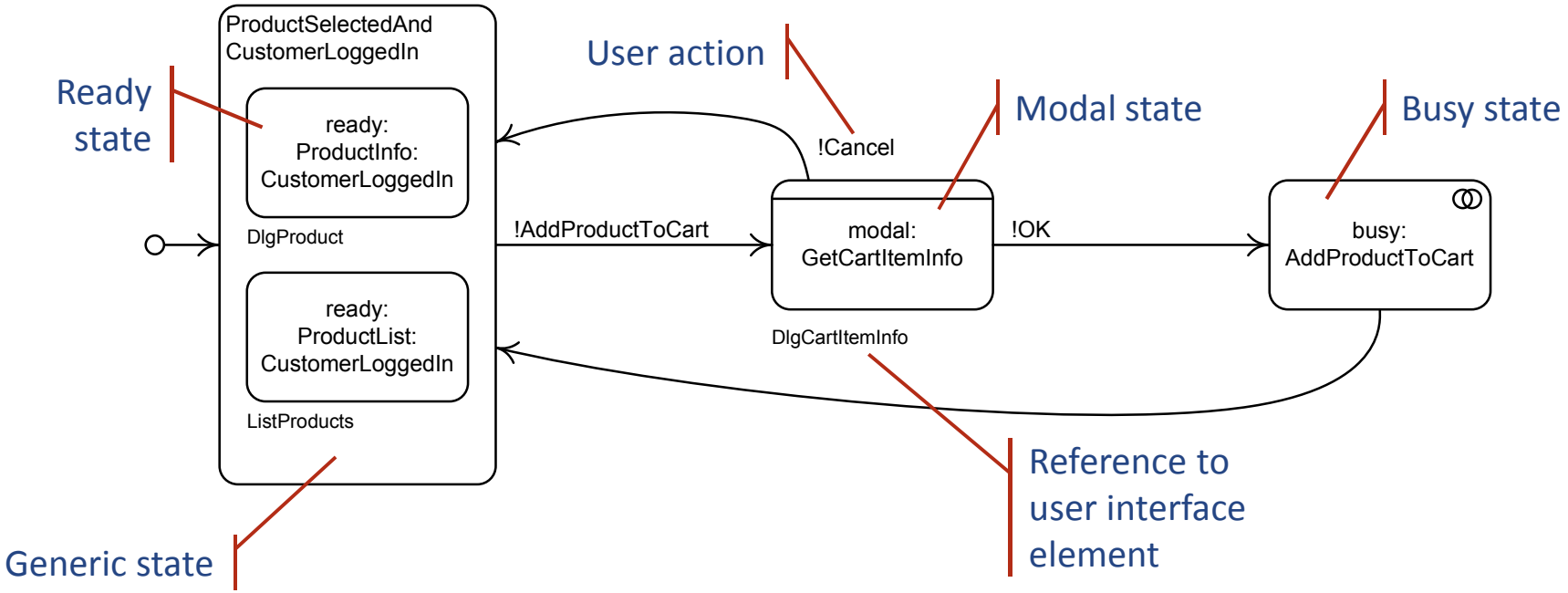
Service Model

Describes the dynamics of the user interaction in full detail

Created from use cases

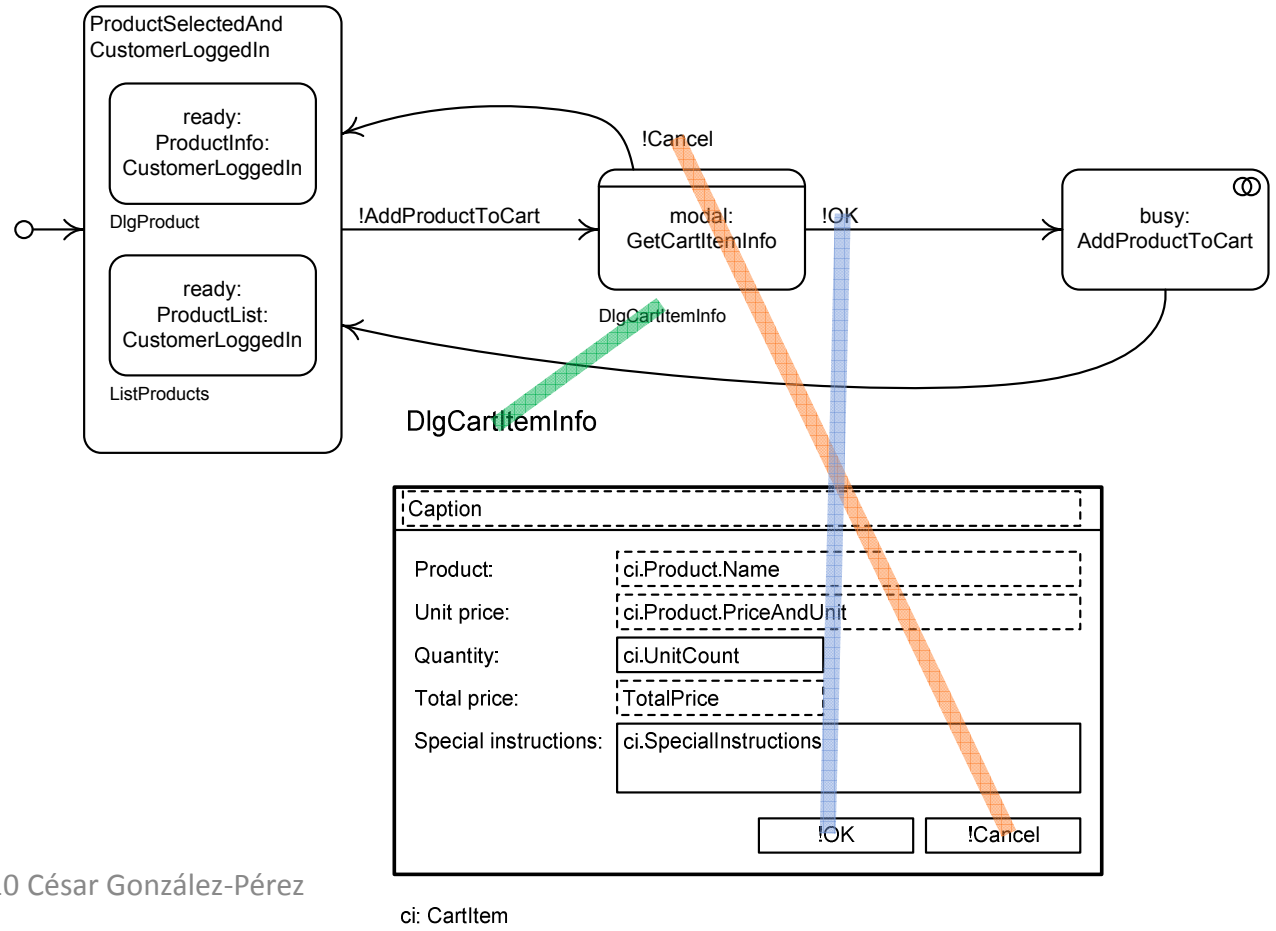
Adds implementation details

Service Model

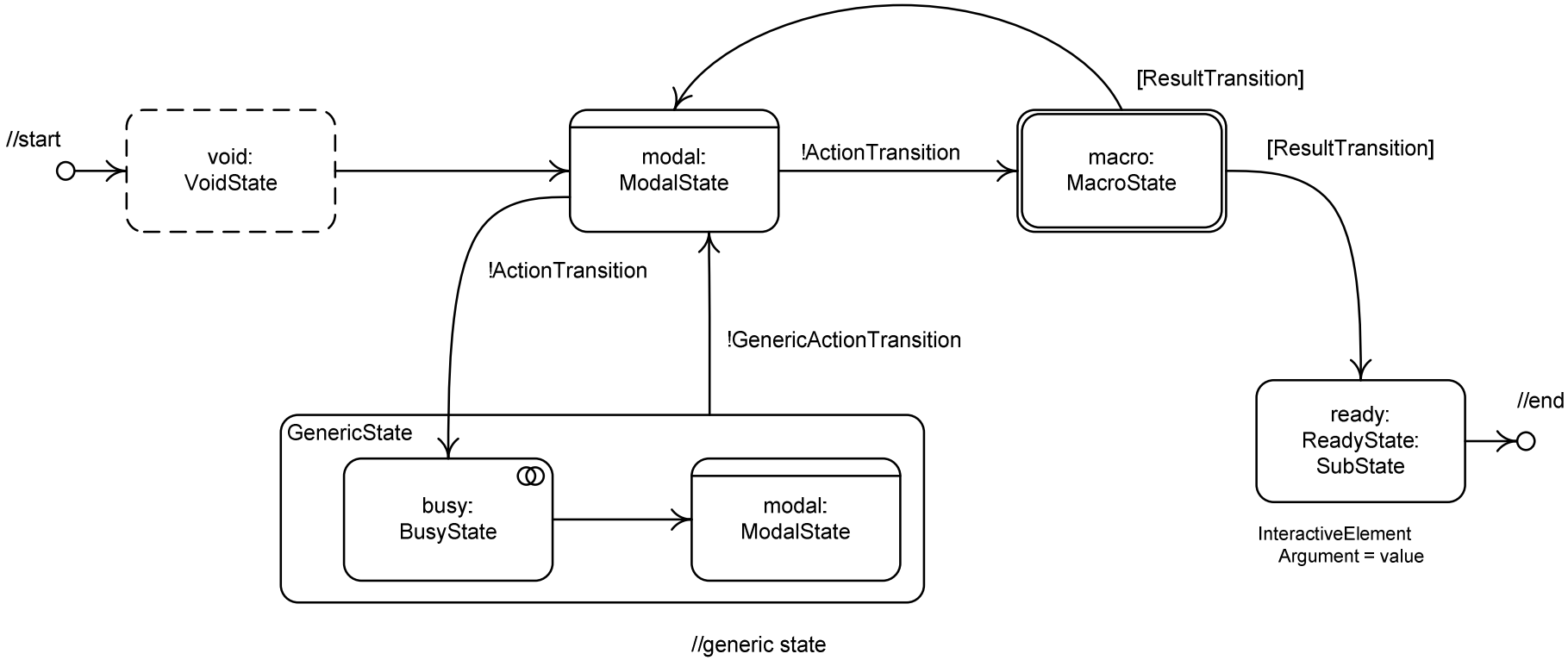


Service Model

Linked to user interface model via element and action references



Service Model Notation

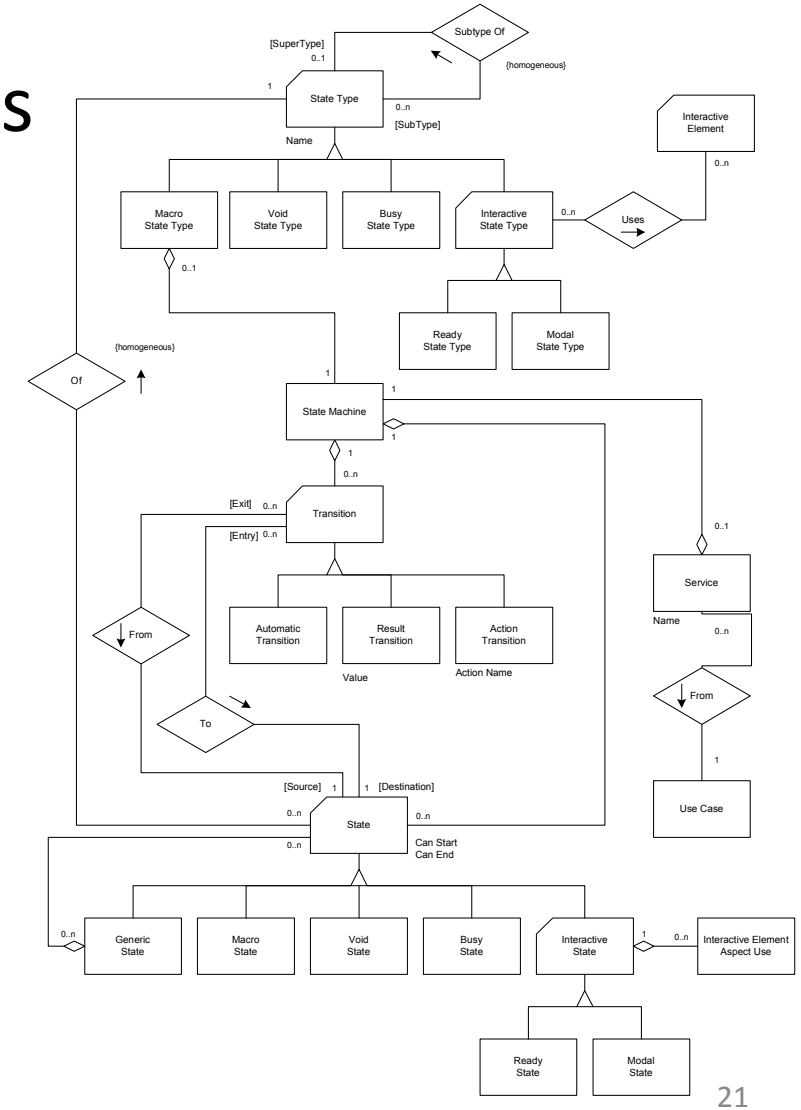


Service Model Language

About 25 specialised classes

States, transitions

State types



User Interaction Design Recap

Part of the “High-Level Modelling” process kind

Integrated with other aspects of design, such as domain modelling and use case analysis

Abstract enough as to yield different implementations

Where do classes come from?

Moving from Functional to Structural Models

65%

of the classes on a typical project*
are not related to the application domain

* Aggregated internal, unpublished data of LAFC and Neco TI.

Common Questions from Developers

"How do I find the classes in the requirements?"

"How do I associate implementation classes to problem-domain classes?"

"How do I distribute functionality across implementation and problem-domain classes?"

"Use cases and UML don't work together!"

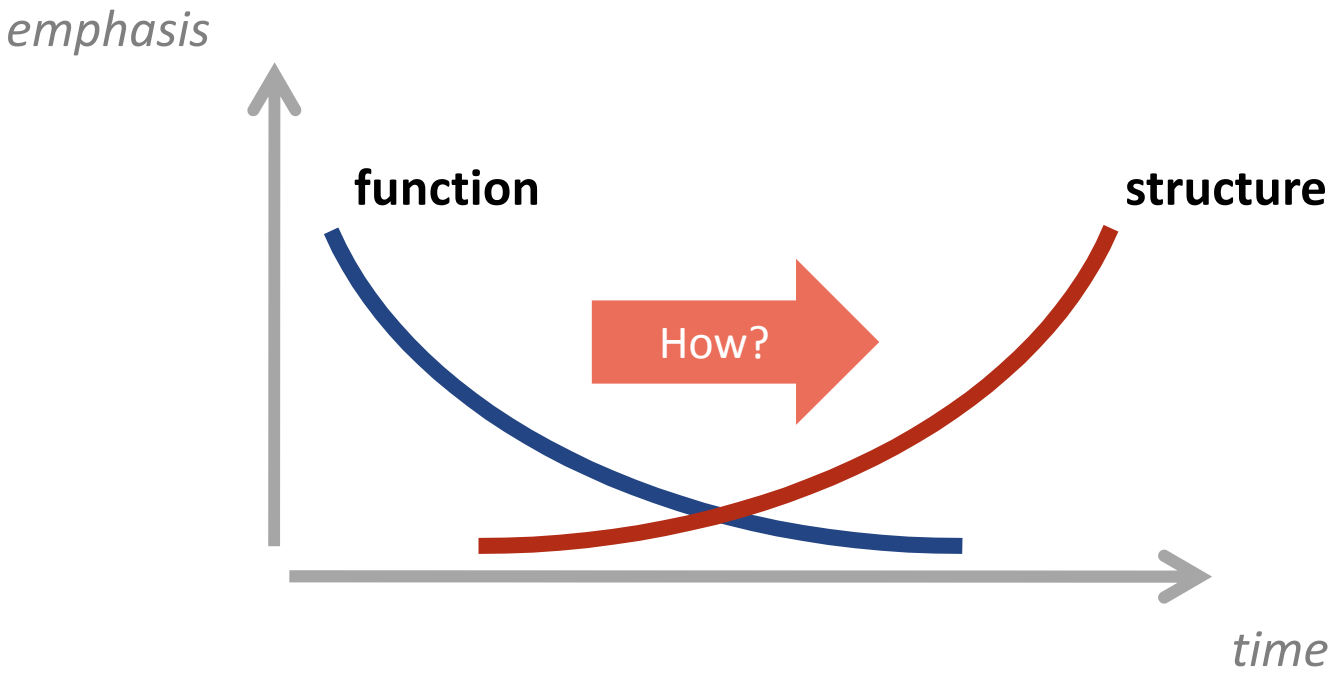
Observations

Functional requirements are, well… *functional*. Use cases are functional too.

Any OO design should be, by nature, structure-focused.

We need a language plus process guidance on how to bridge this gap.

Observations



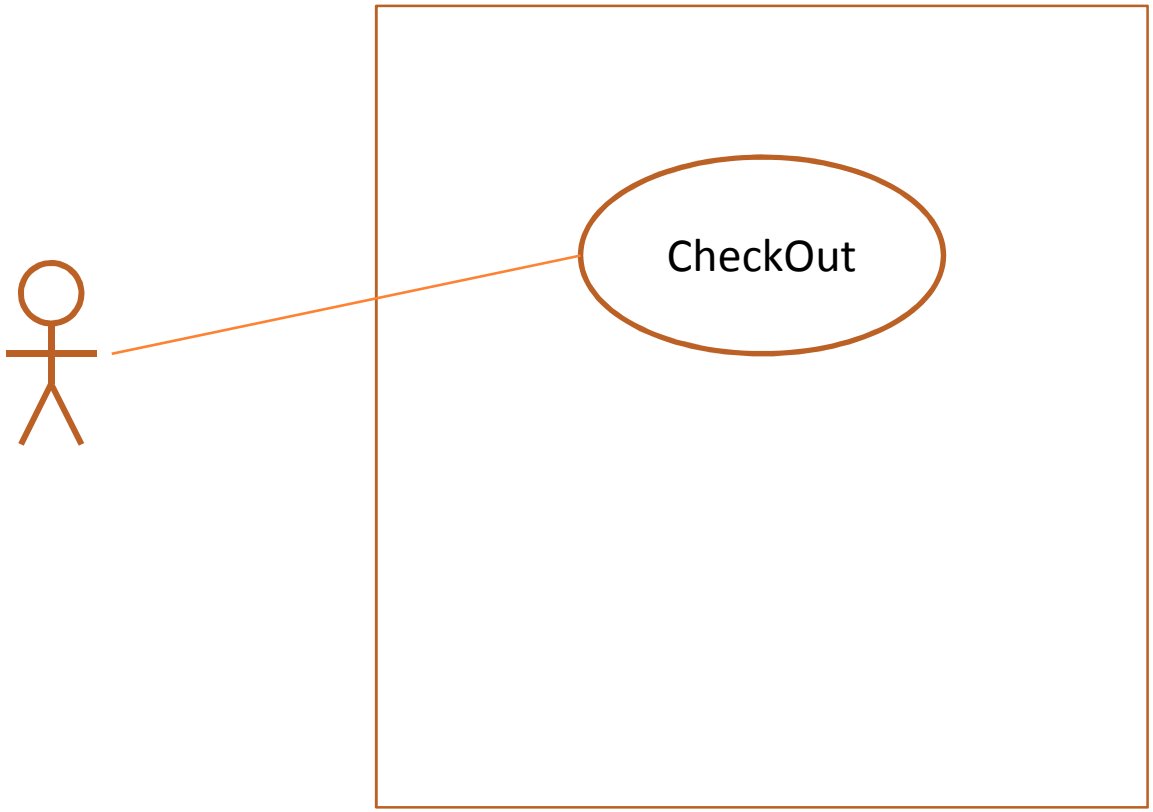
OPEN/Metis Solution

A **Service Model** implements each use case, adding implementation details

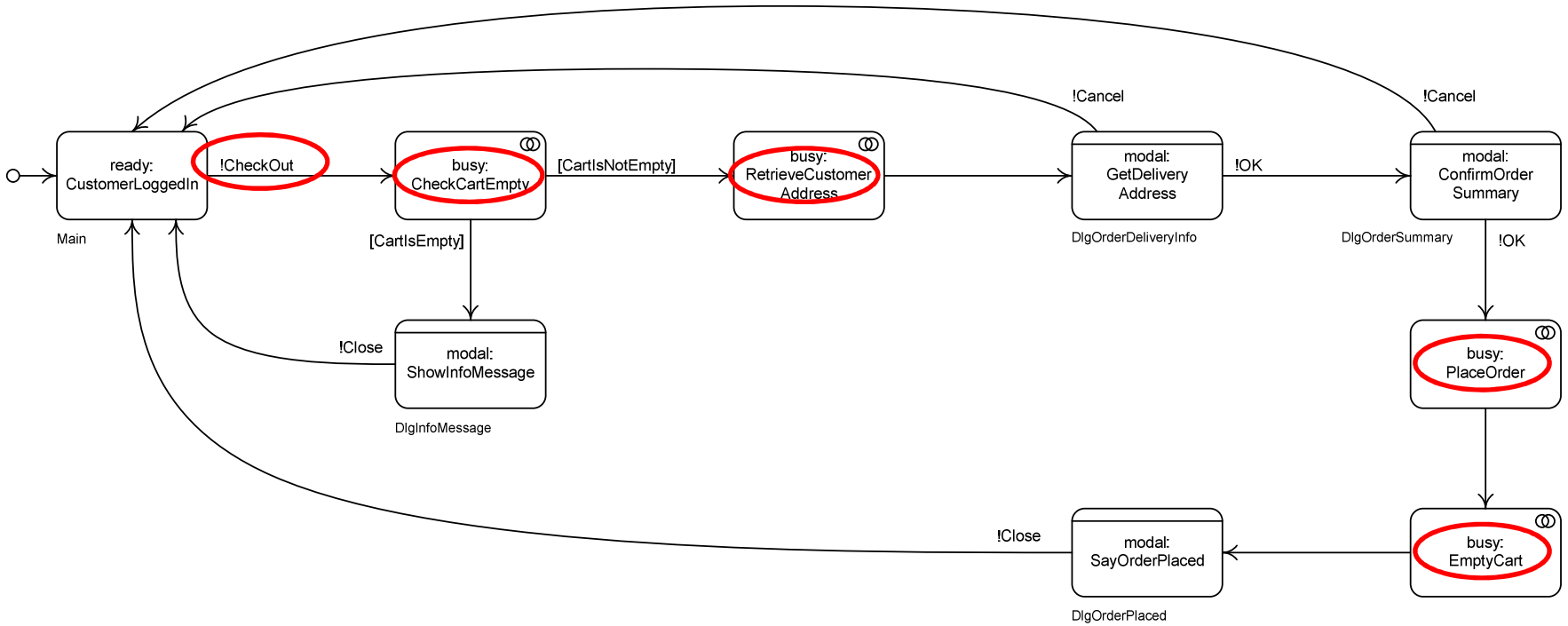
Operations are created for each service and busy state

The **responsible class** for each operation is determined

Example: A Use Case



Implementing the Service Model



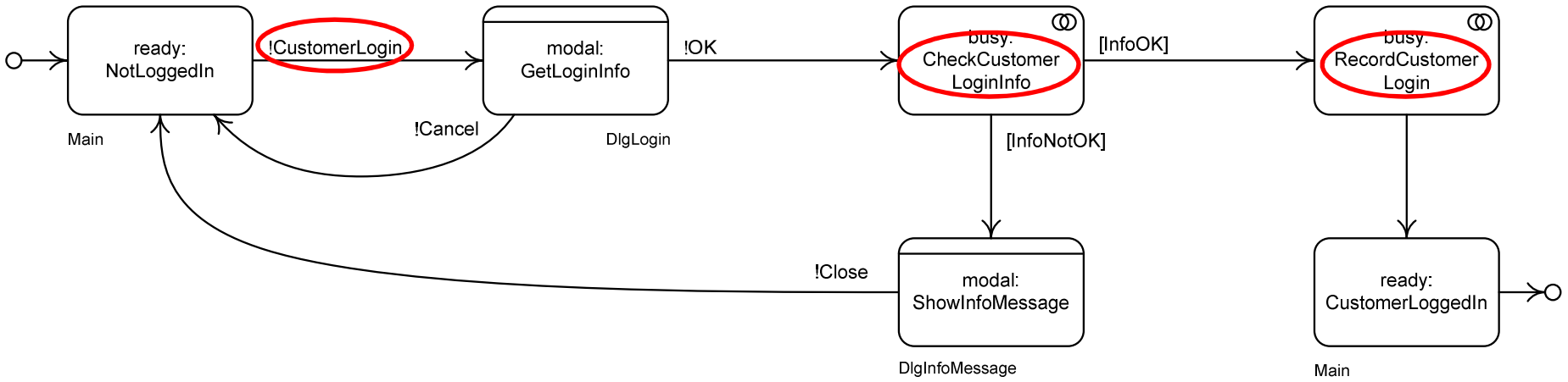
Operations:

CheckOut, CheckCartEmpty, RetrieveCustomerAddress, PlaceOrder, EmptyCart

Determining Responsible Classes

Operation	Responsible Class
CheckOut	OrderManager (new)
CheckCartEmpty	Cart (domain)
RetrieveCustomerAddress	CustomerManager (new)
PlaceOrder	OrderManager (new)
EmptyCart	Cart (domain)

Implementing another Service Model



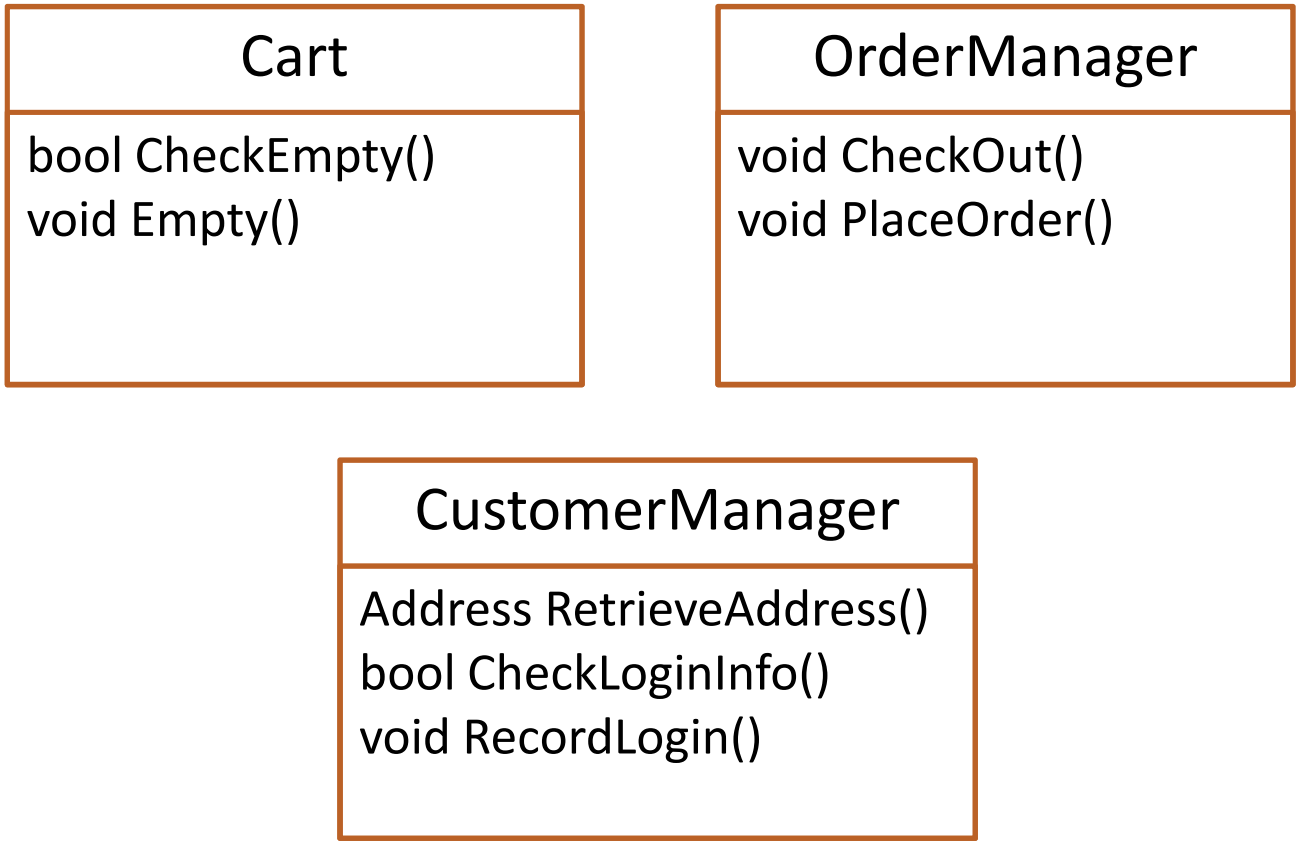
Operations:

CustomerLogin, CheckCustomerLoginInfo, RecordCustomerLogin

Determining Responsible Classes

Operation	Responsible Class
CustomerLogin	UIManager (new)
CheckCustomerLoginInfo	CustomerManager
RecordCustomerLogin	CustomerManager

After a Few Iterations…



Functional to Structural Recap

Classes are populated incrementally from services

Some classes are even introduced during this process

Full traceability from class structure back to use cases

Conclusions

Do not neglect user interaction modelling; it is the lifeblood of the system dynamics

Find a way to derive the system's structural blocks from its functional requirements

Thank you

César González-Pérez
cesar.gonzalez-perez@iegps.csic.es

The Heritage Laboratory (LaPa)
Spanish National Research Council (CSIC)