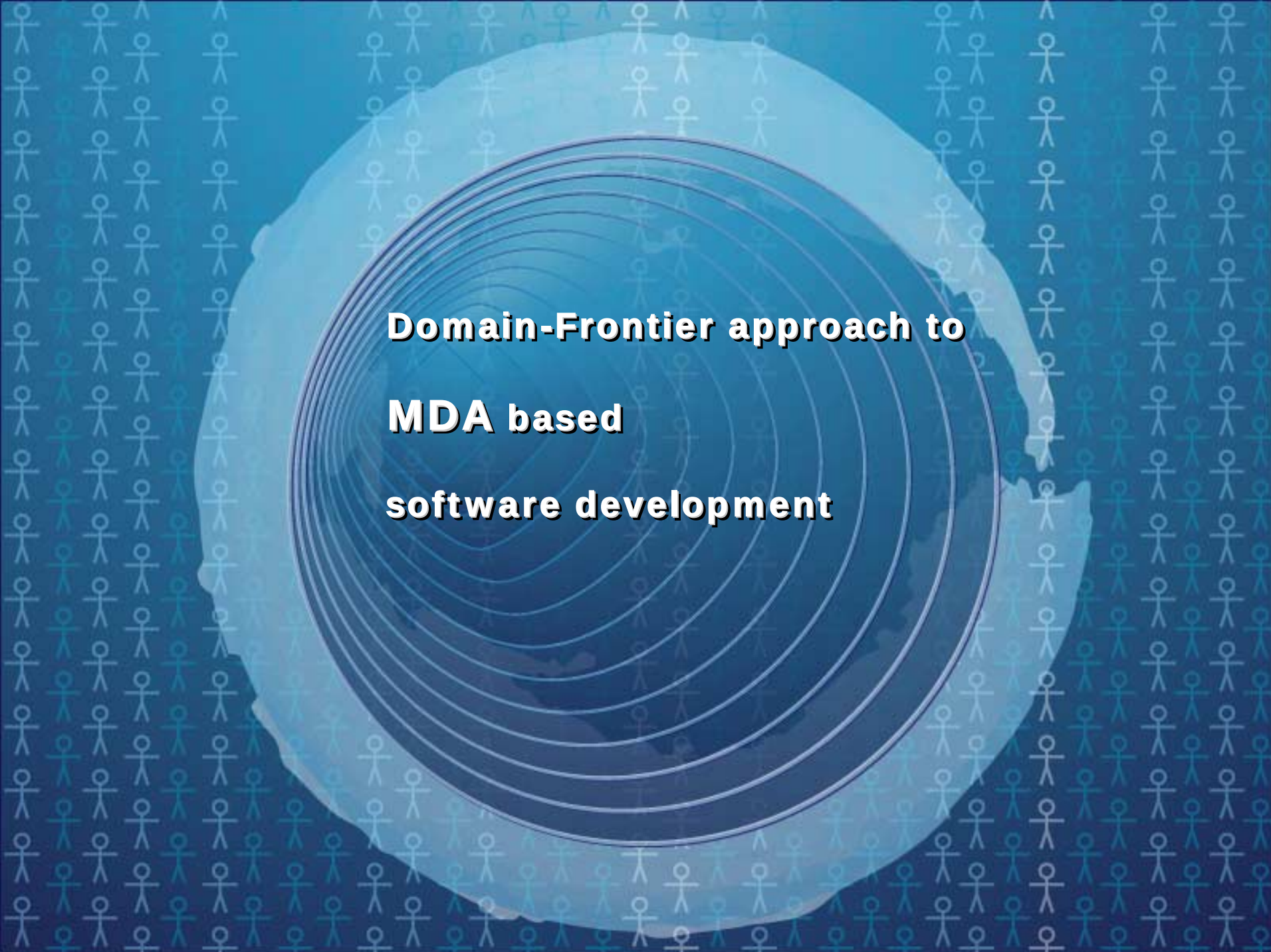
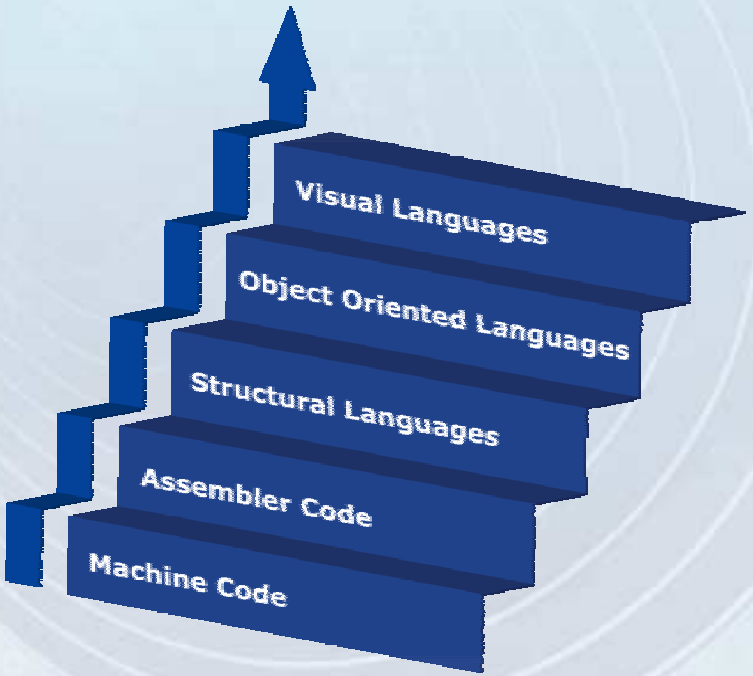

Domain-Frontier approach to
MDA based
software development

# Contents

- Software Development
  Yesterday, Today and Tomorrow
- Domain-Frontier
  Paradigms and Cornerstones
- Domain-Frontier
  Development Process Map
- Example – Rosa's Breakfast Service
- Future Work
- Bibliography

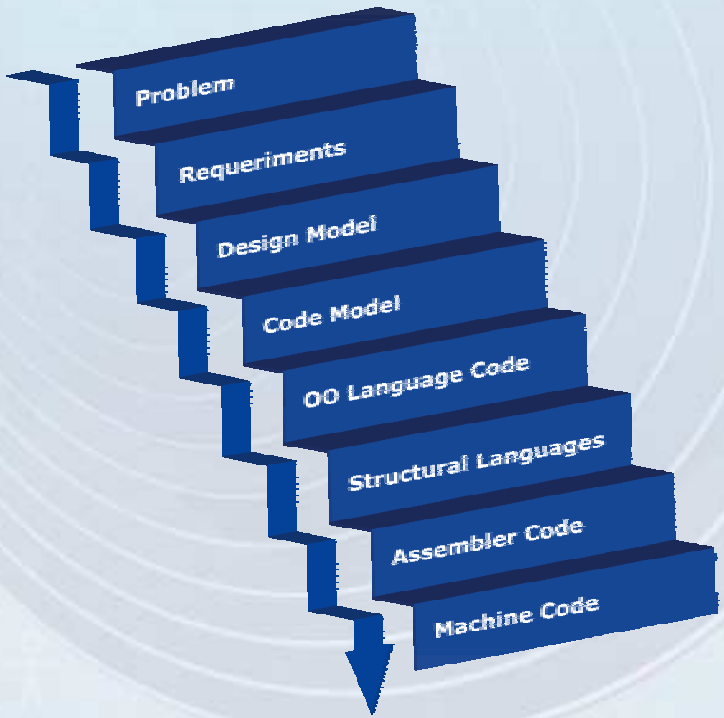
# Software Development Yesterday, Today and Tomorrow

- **Software Engineering History – Continual Raise of Abstraction**



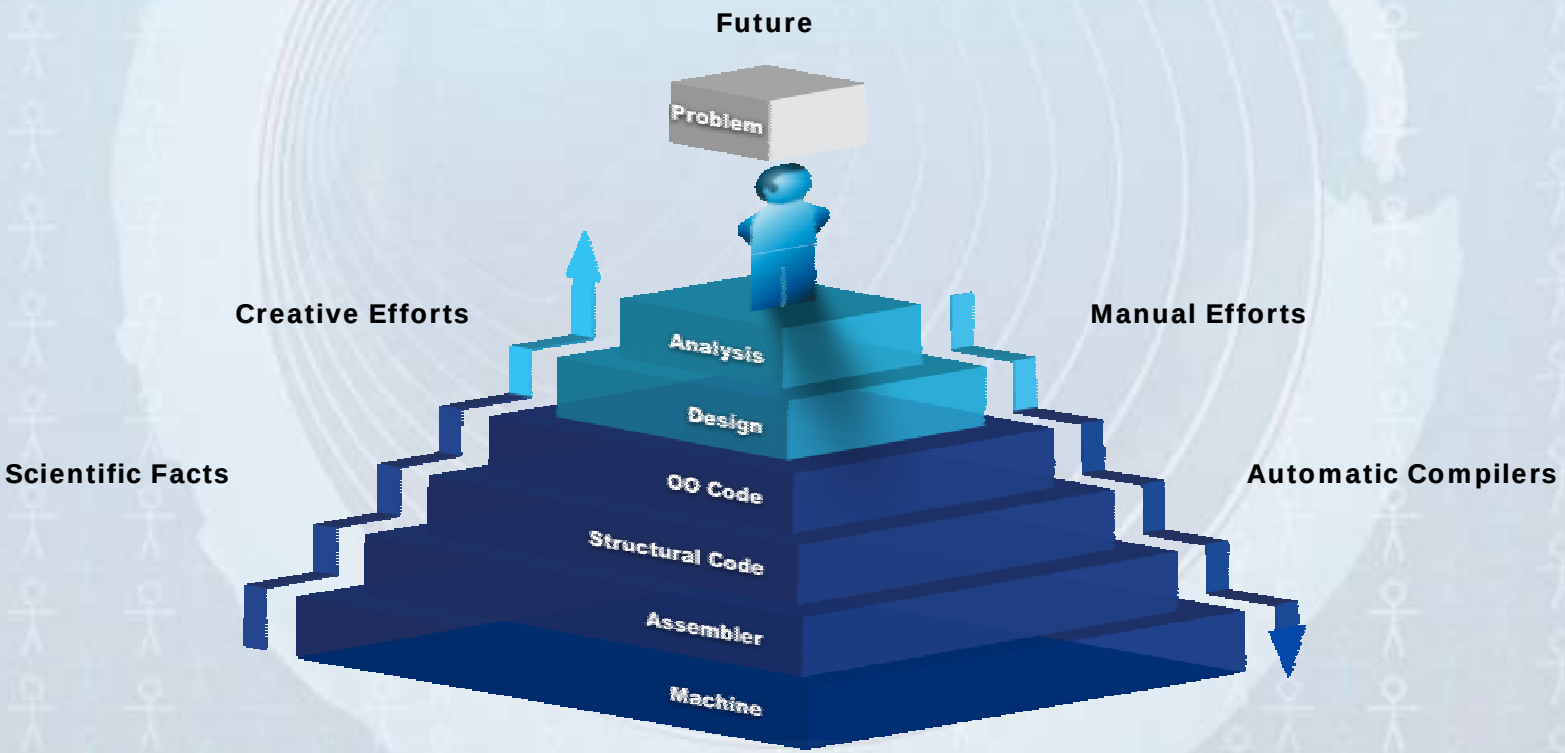
# Software Development Yesterday, Today and Tomorrow

- **Software Project Development – Gradual Lowering of Abstraction**



# Software Development Yesterday, Today and Tomorrow

## Software Engineering Evolution and Software Product Evolution Follow the Same Paths, but in Opposite Directions



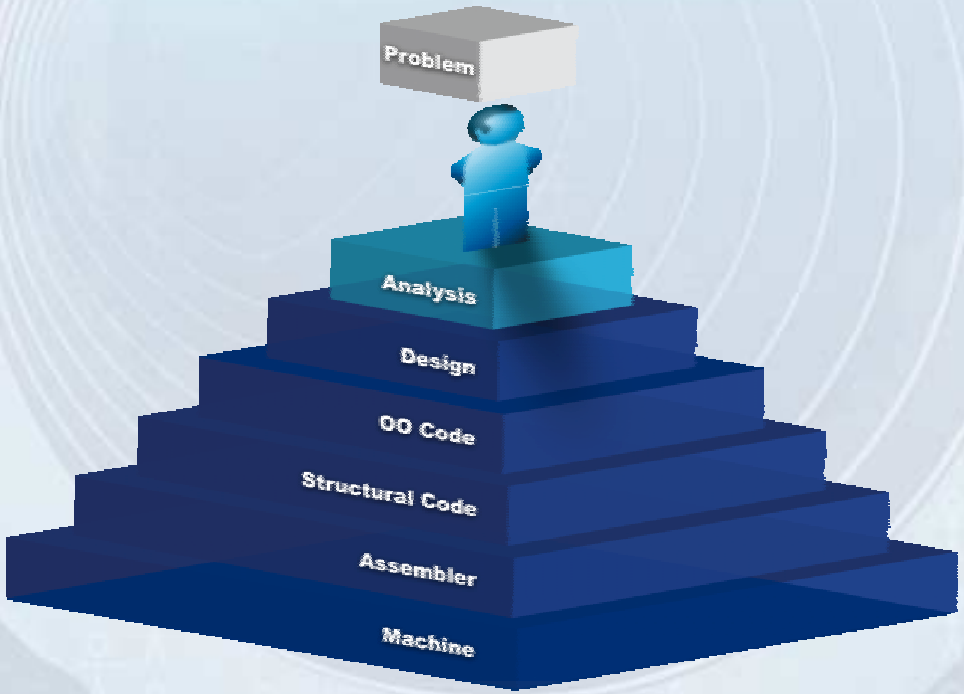
# Software Development Yesterday, Today and Tomorrow

- **Software Trends Today**

  - Visual Programming Languages
  - Frameworks and Patterns
  - Pervasive Services Solutions
  - Partial Code Generators
  - Specific Domain Complete Generators, etc.

# Software Development Yesterday, Today and Tomorrow

- MDA Unifies and Standardizes Modern Trends in Software Development
- Formalized System Analysis Method?



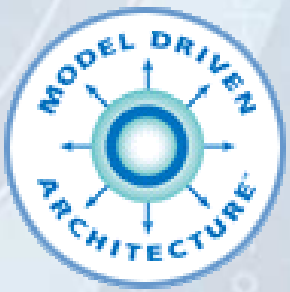
# Software Development Yesterday, Today and Tomorrow

- **A Glimpse to the Future**

  - Formal Problem (Requirements) Definition Language?
  - Will Domain Experts Be Able to Specify and Produce Software?

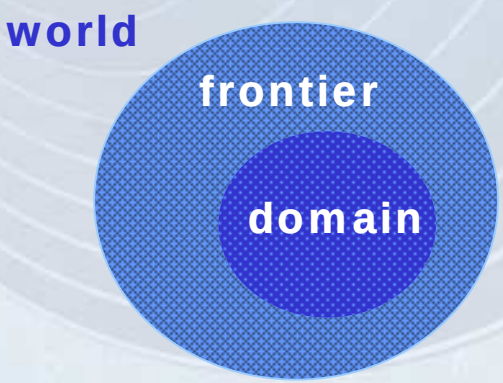
# Domain-Frontier Aproach Paradigms

- **MOF and UML – Unified Modeling Language**
  - Standard Notation
  - Standard Models Repository
  - Executable UML
- **Aspect Oriented Analysis and Design**
  - Overall Separation of Concerns
    - Problem Level: Domain and Frontier
    - System Level: Domain and Pervasive Services
- **Model Driven Architecture**
  - Model Centric Development
  - Model Transformations, etc.



# Domain-Frontier Aproach Cornerstones

- **Abstraction Management**
  - Clear Abstraction Milestones:
    - Problem →Architecture → Solution
    - Early Milestones without ambiguity
- **Dynamics at High Level Model**
  - Capture Behaviour from Begining
- **Focused on System External Interface**
  - Low Domain-Frontier Correlation
  - User Interface Deserves Special Attention!



# Domain-Frontier Models

- **Problem Model (“CIM”) – UML with OCL**
  - Business Process Model
  - Use Case Model
  - Conceptual Model with States
  - Sequence Diagrams
  - OCL Code

- **Architecture Model (“PIM”) – UML with Actions**
  - Analysis Model (Boundary, Control, Entity) with States
  - Sequence Diagrams
  - Action Semantics Code

- **Solution Model (“PSM”) – UML with Target Language**
  - Class Model
  - Component Model
  - Target Platform Code

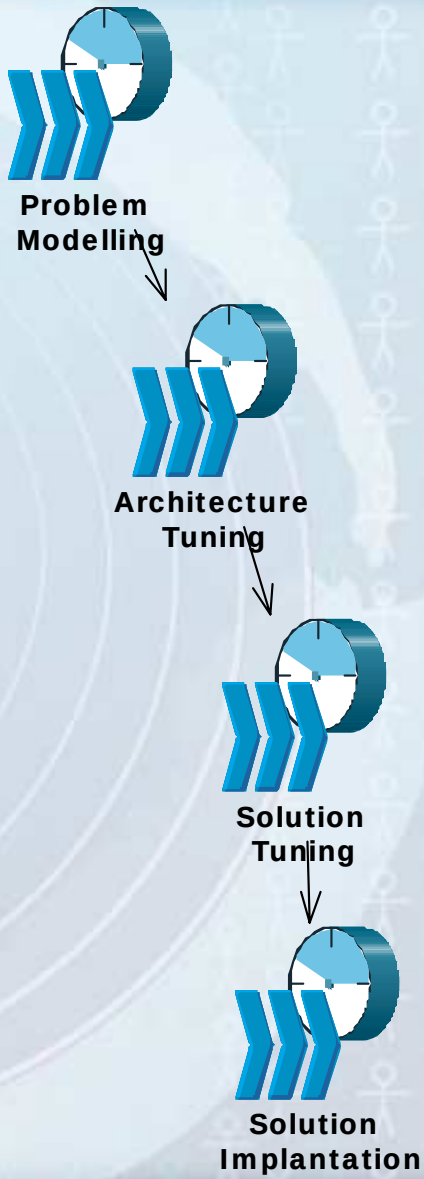
# Development Process Map

- Phases and Milestones
- Activities
- Roles
- Process Models (Artifacts)
- Transformations

# Development Process Map (cont.)

**Milestones:**

- **Problem Modelling**
  - Requirements Understood
  - Architecture Generated
- **Architecture Tuning**
  - Design Generated
  - Scenarios Tested
- **Solution Tuning**
  - Solution Generated and Tested
- **Solution Implantation**
  - Solution Installed



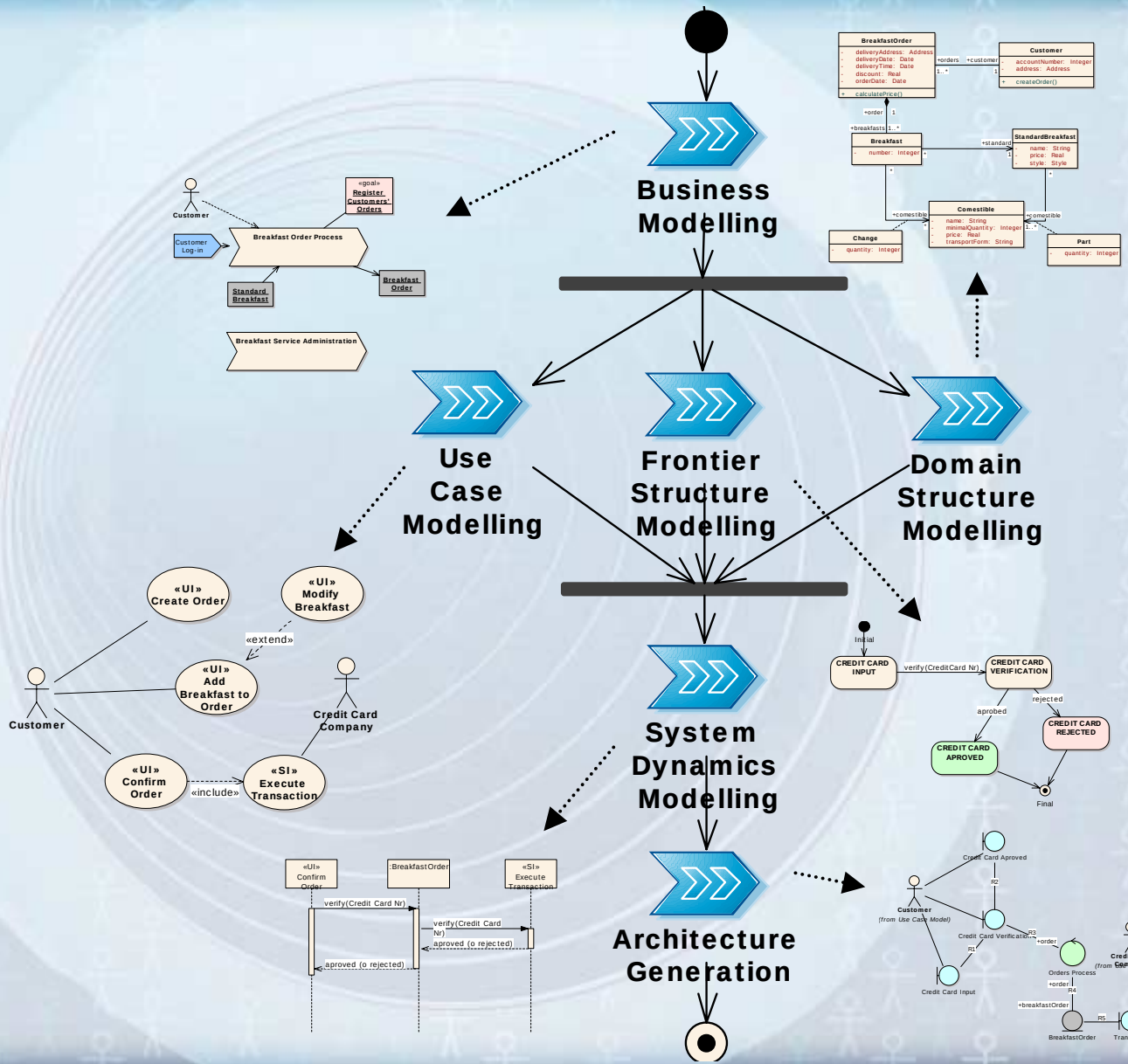
# Problem Modelling Phase

Business Analyst

System Analyst

User Interface Designer

Architect



# Architecture Tuning Phase

Action Implementer

Pervasive Services Specialist

Tester

Architect

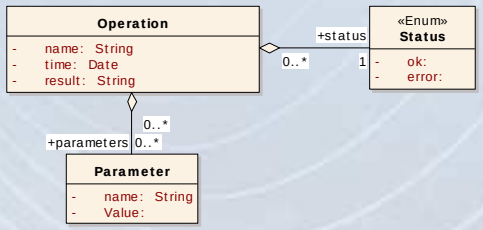
```
self.name = rcvd_evt.name

select many parameter from instances of rcvd_evt.parameters
   create object instance newParameter of Parameter
      newParameter.name = parameter.name
      newParameter.Value = parameter.value
   relate self to newParameter across R10;
```

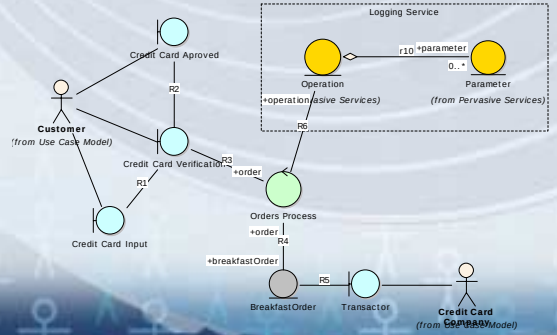
```
self.time = Now
self.status = rcvd_evt.status
self.status = rcvd_evt.result
```

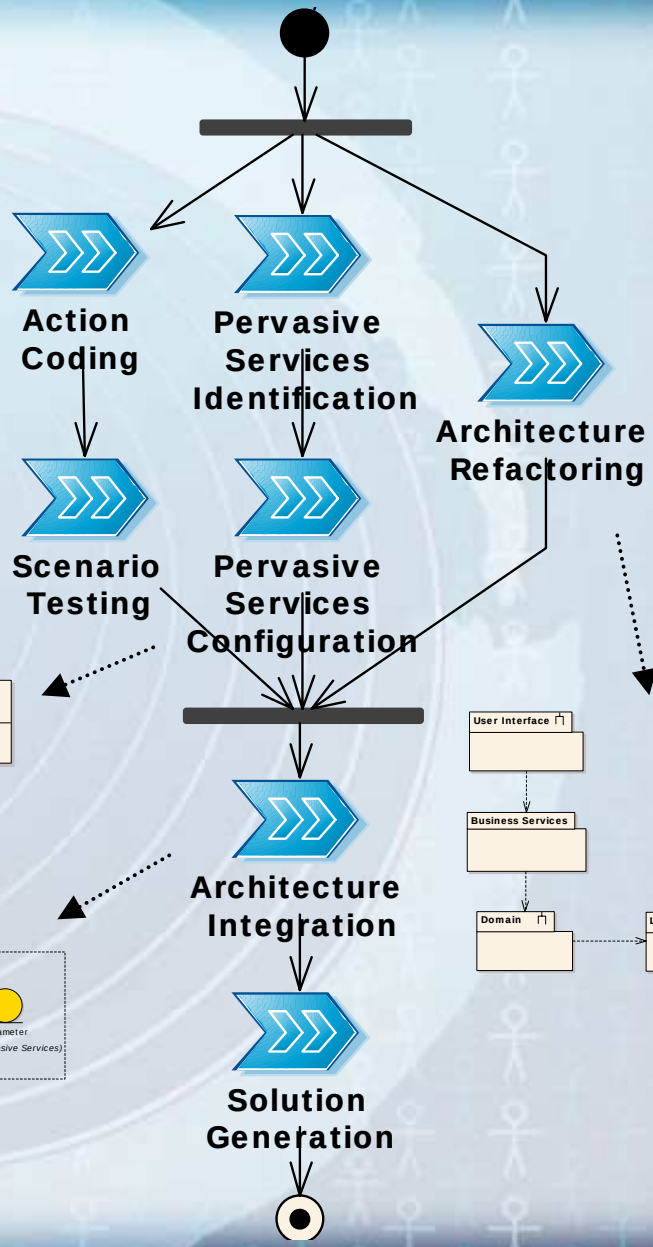
«UI» Confirm Order

«trace»

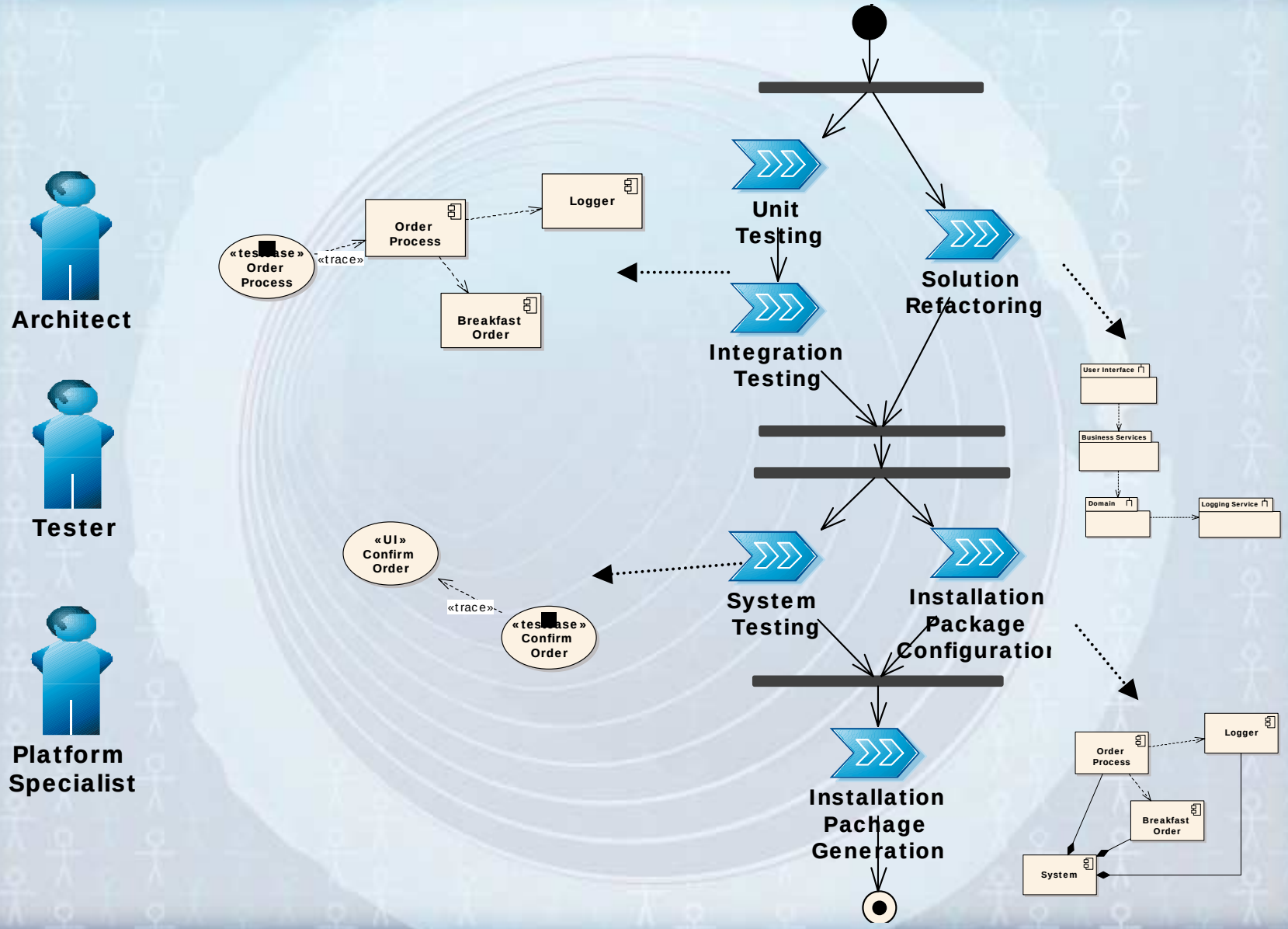
«testcase» Confirm Order



# Solution Tuning Phase



# Process Roles

**Architect**

- Transformations
- Overall Model Structure

**Business Analyst**

- Business Process Model
- Business Rules

**System Analyst**

- Domain Model
- Scenarios

**User Interface Designer**

- User Interface Design
- User Interactions

**Action Implementer**

- Action Programming
- Architecture Model

**Pervasive Services Specialist**

- Pervasive Services Integration

**Tester**

- Scenario Testing
- System Testing

**Platform Specialist**

- Target Platform Integration and Installation

# Example – Rosa's Breakfast Service *

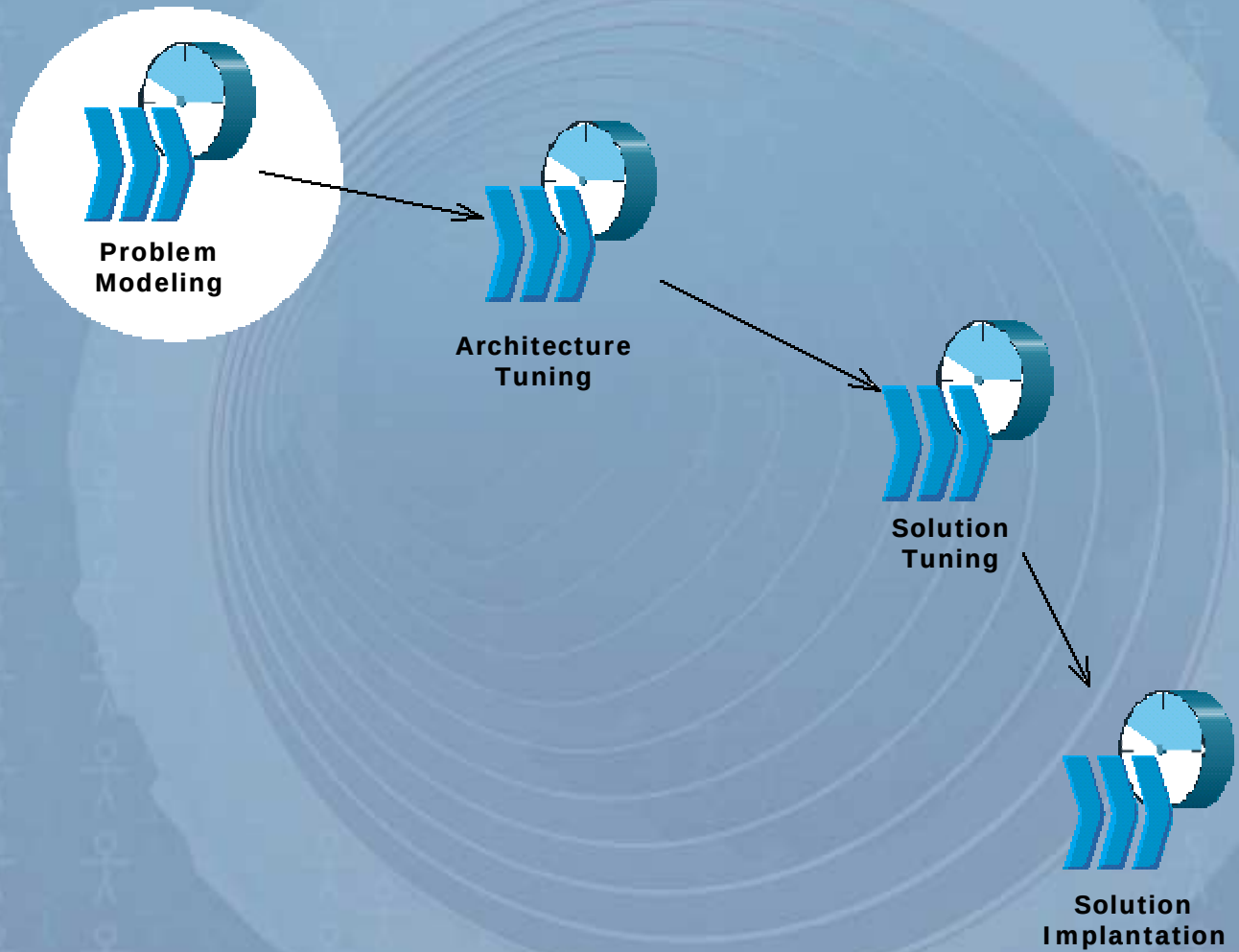
"Rosa has founded a company that supplies a complete **breakfast** delivered to the homes of its **customers**. Customers can **order** from one of the breakfast menus on Rosa's Web site, indicate the hour and the place of delivery, give their credit card number, and the ordered breakfast will be delivered.

...

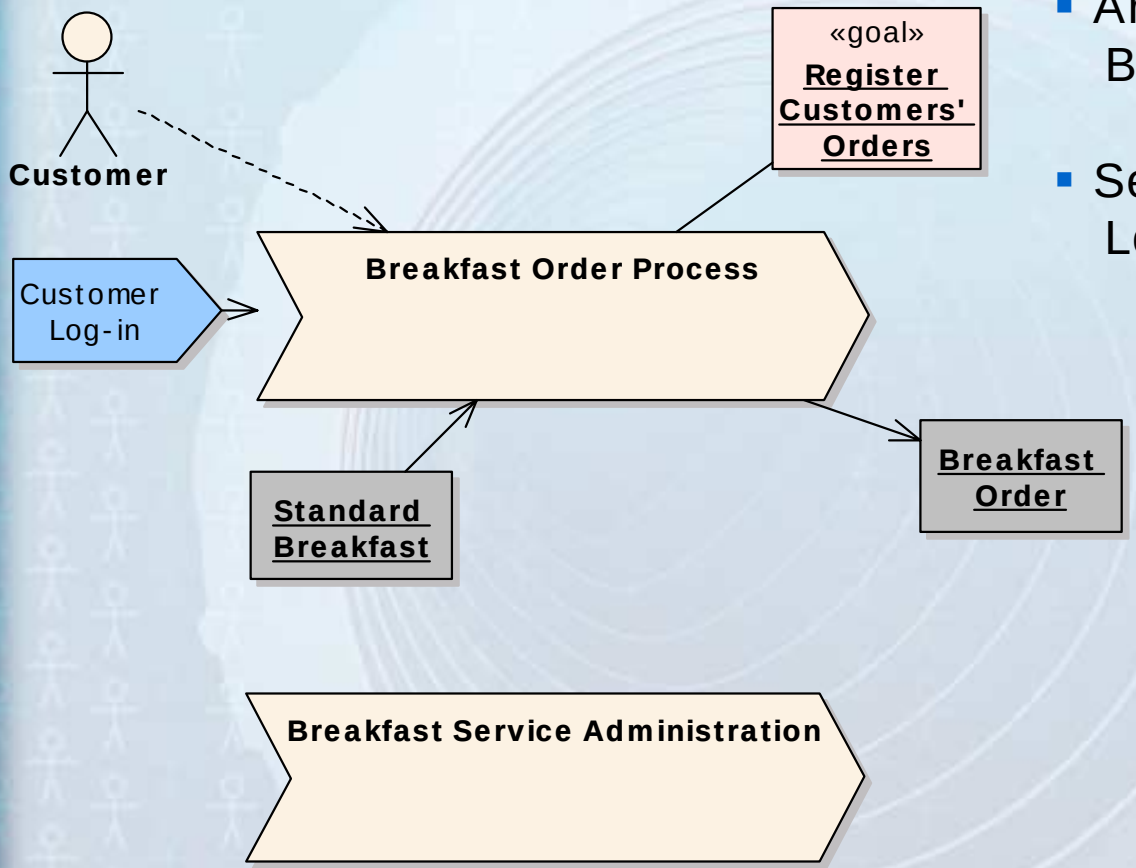
Rosa wants to give to her customers a bit of flexibility. Customers may, after choosing a **standard breakfast** as basis, decide to put in extra **comestibles**, alter the amount of certain **parts**, and even remove parts from the breakfast."

* *Example problem statement taken, with permission, from "MDA Explained", by Anneke Klepe, Jos Warmer and Wim Bast*

# The First Phase: Problem Modelling



# Business Modelling

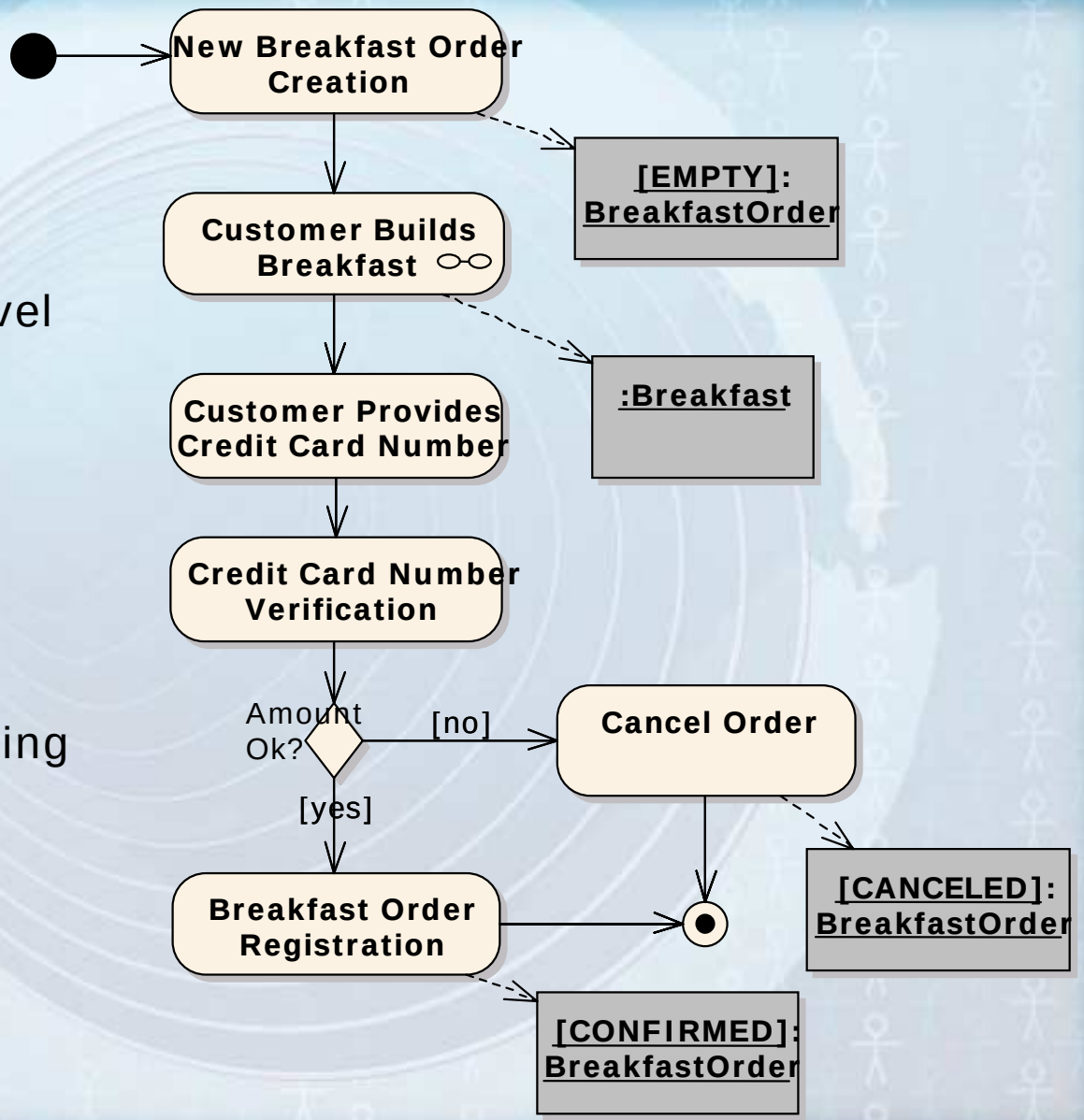



- Artifact:
  Business Process Model
- Semantics:
  Low to High

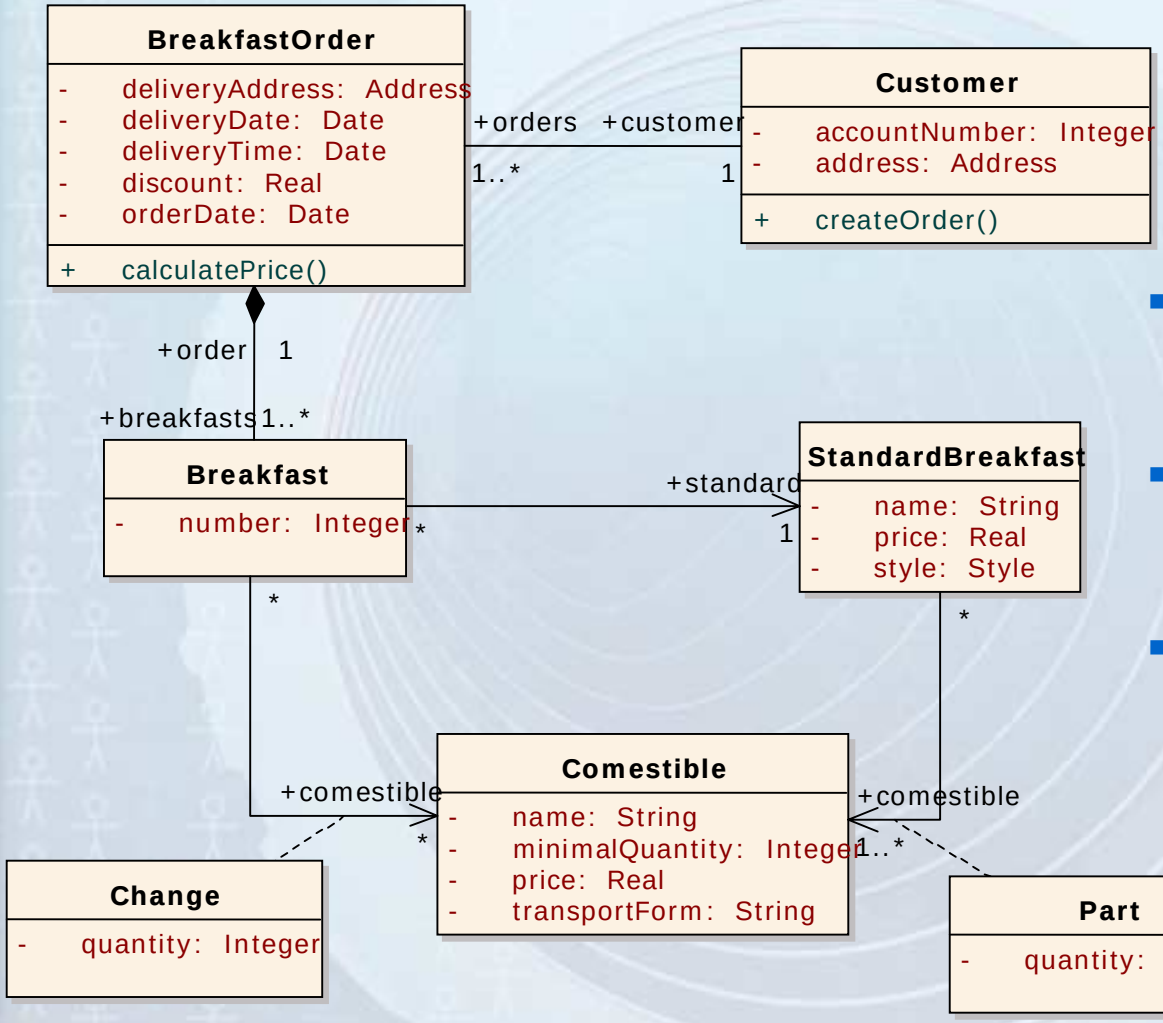
- Purpose:
  Understand Business Processes

# Business Modelling (cont.)

- Variable Detail Level
- Integration with Workflow Engine?
- Early Class/State Identification
- Use Case Secuencing



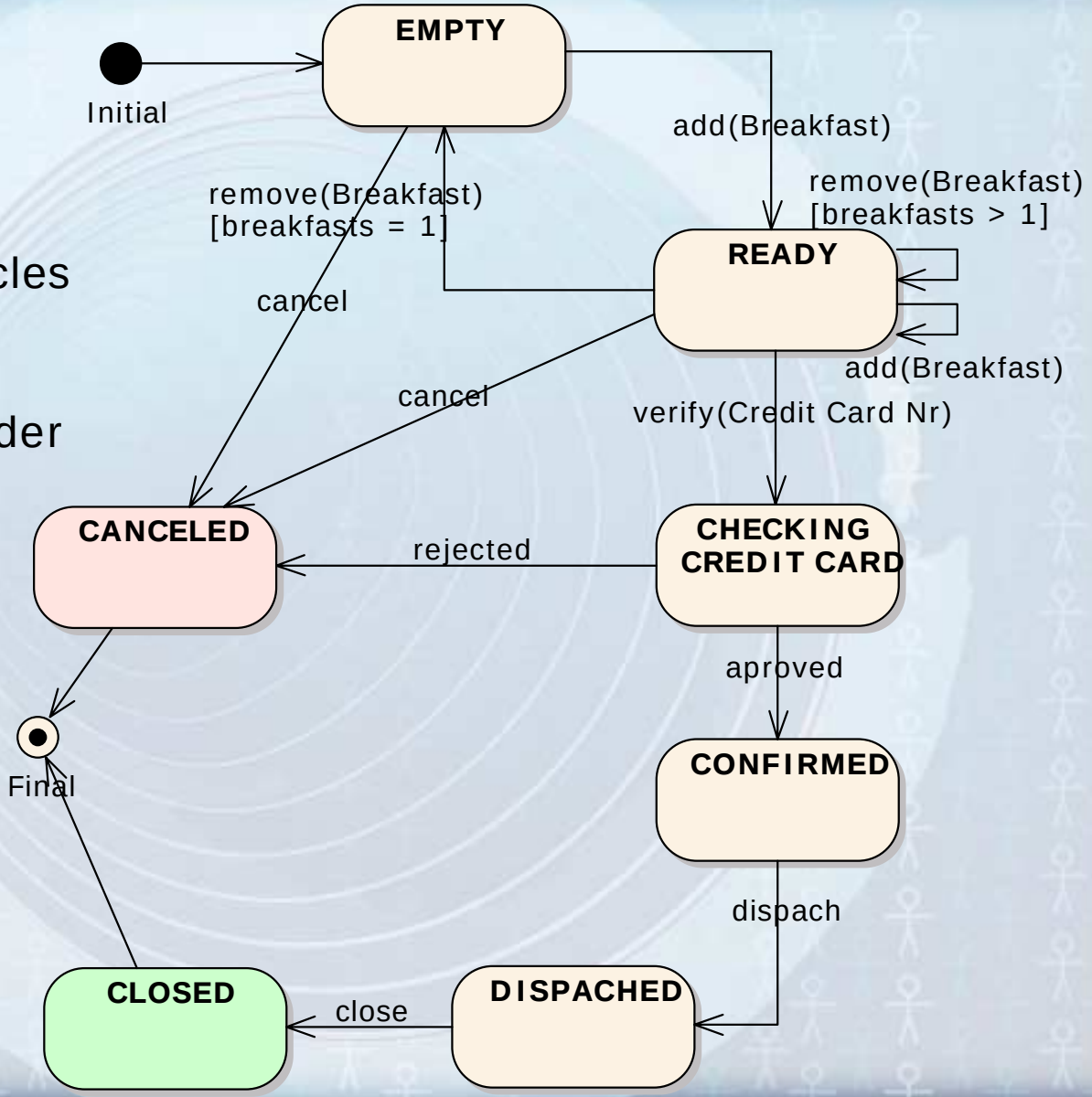
# Domain Modelling



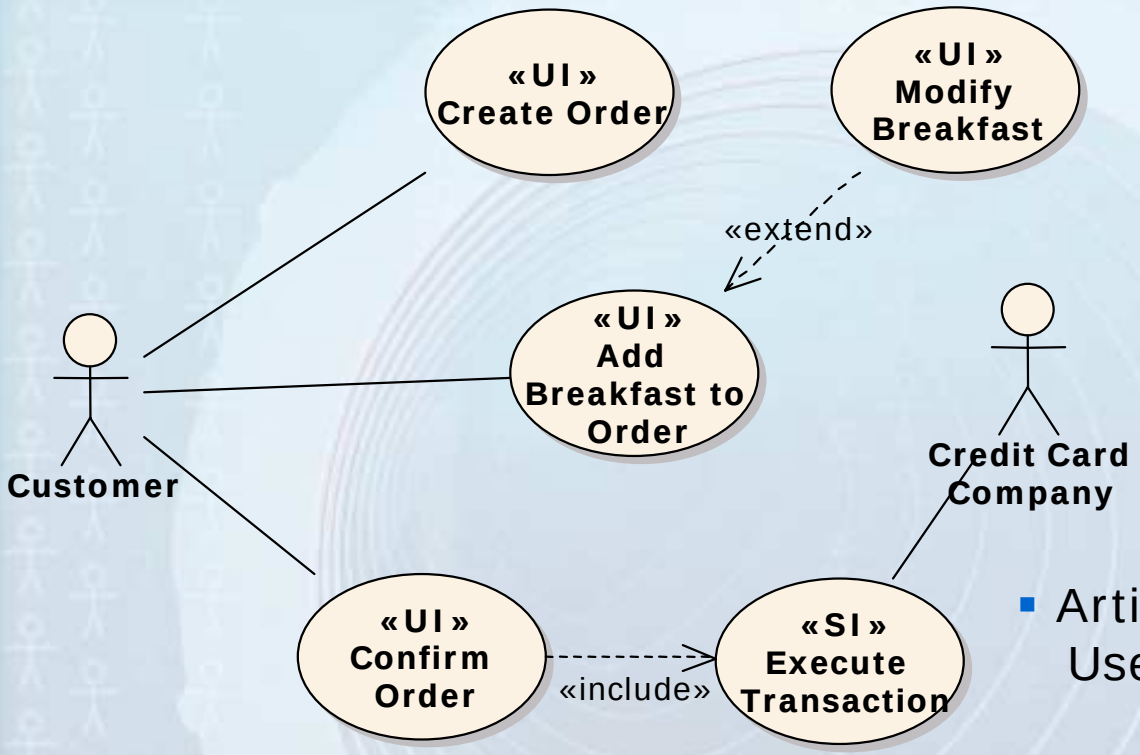

- Artifact:
  Conceptual Model
- Semantics:
  High
- Goal:
  Understand
  Problem Structure

# Domain Modelling – Object States

- Concepts Lifecycles
- Rosa Example:
  - BreakfastOrder

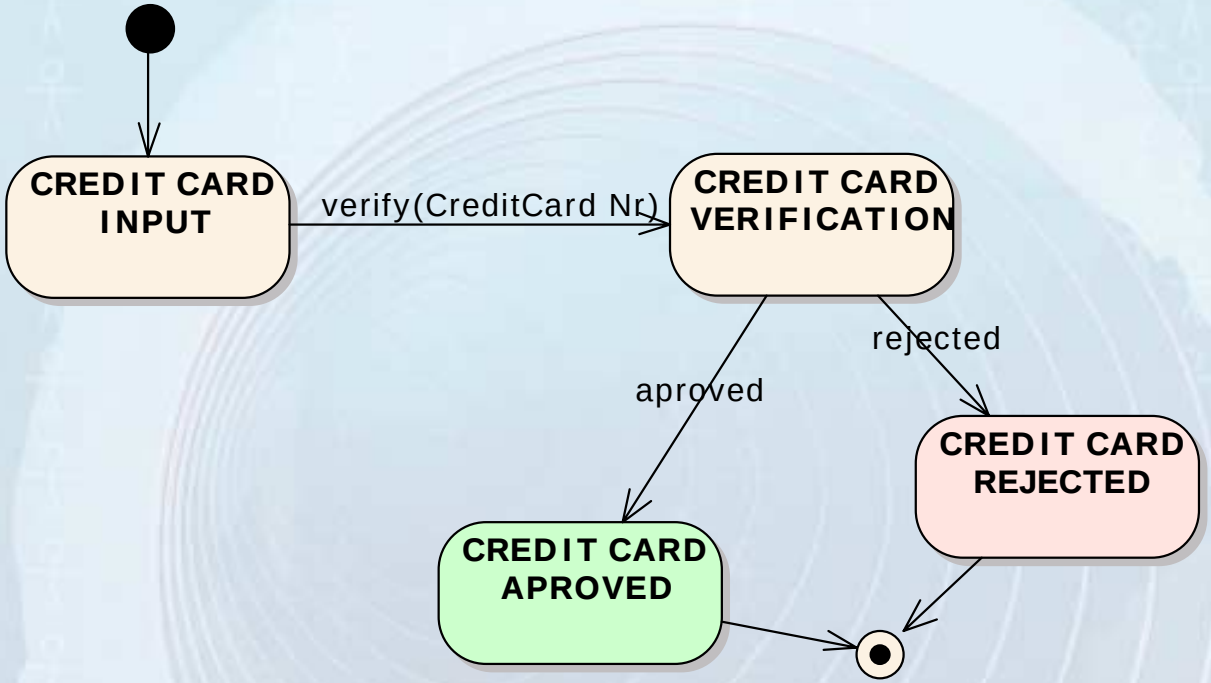


# Use Case Modelling



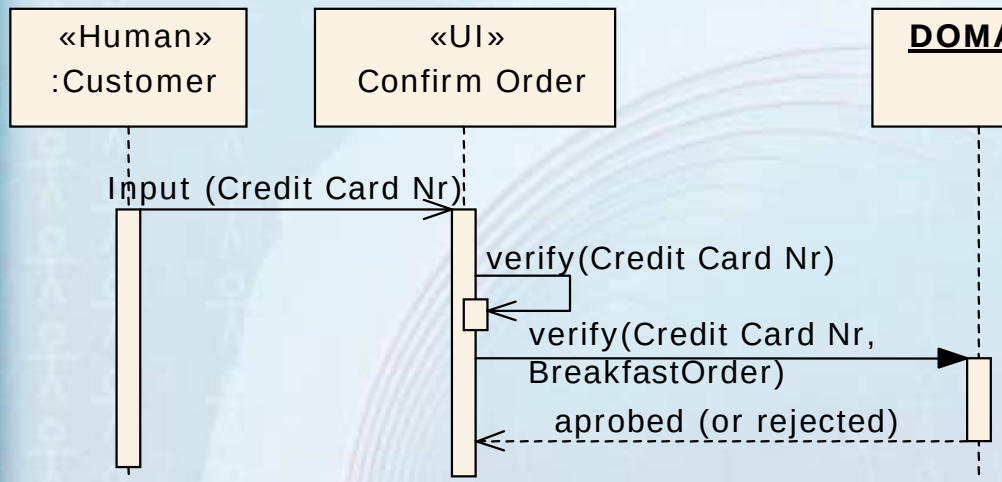

- Artifact:
  Use Case Model
- Semantics:
  Medium
- Purpose:
  Understand
  System Requirements

# Frontier Modelling



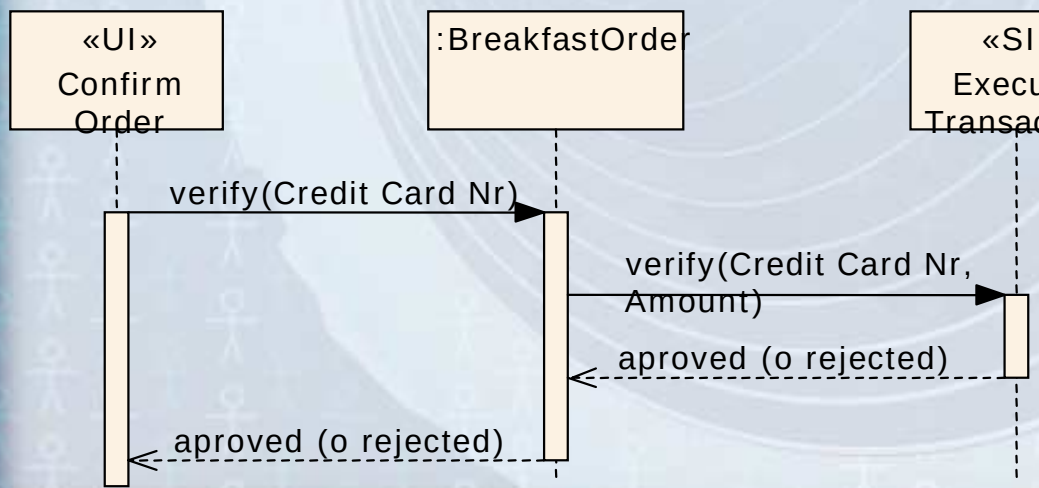

- State Machine per Use Case
- State per User Interface Presentation Unit (Page, Form, Electronic Device Screen, etc.)

# System Dynamics Modelling




- Problem Level Interaction Diagram per Use Case
- All Paths Covered
- Domain Details Ignored
- Role: UI Designer

Requirements




- Domain Level Interaction Diagram per Use Case
- Domain Internal Details
- Role: System Analyst

# Architecture Generation

- **Unified Architecture Metamodel**
  (UML Analysis Model with States and Actions)

- Tasks
  - Select Architecture
    - Client/Server, Enterprise, Embedded, etc.
  - Mark Problem Model (Manual)
  - Generate Architecture Model (Automatic)

- Model Transformations
  - UML Metamodel(Problem) → UML Metamodel (Architecture)
    - Problem Structure → Architecture Structure
    - Problem Dynamics → Architecture Dynamics
  - OCL → Action Language

# Architecture Generation (cont.)

| Source Metamodel |
| --- |
| Element |
| Stereotype |
| Tags (Marks) |
| Additional Conditions (OCL?) |

| Destination Metamodel |
| --- |
| Element |
| Stereotype |
| Tags (Marks) |
| Additional Instructions |

- Rosa Example
  - Architecture: Enterprise
- Transformations:
  - Business Model → Architecture Model (Enterprise Business Services Layer)
  - Use Case Model → Architecture Model (Enterprise Presentation Layer)
  - Domain Model → Architecture Model (Enterprise Business Components Layer)
  - Problem Interaction Models → Architecture Interaction Models
  - OCL → Action Language Instructions

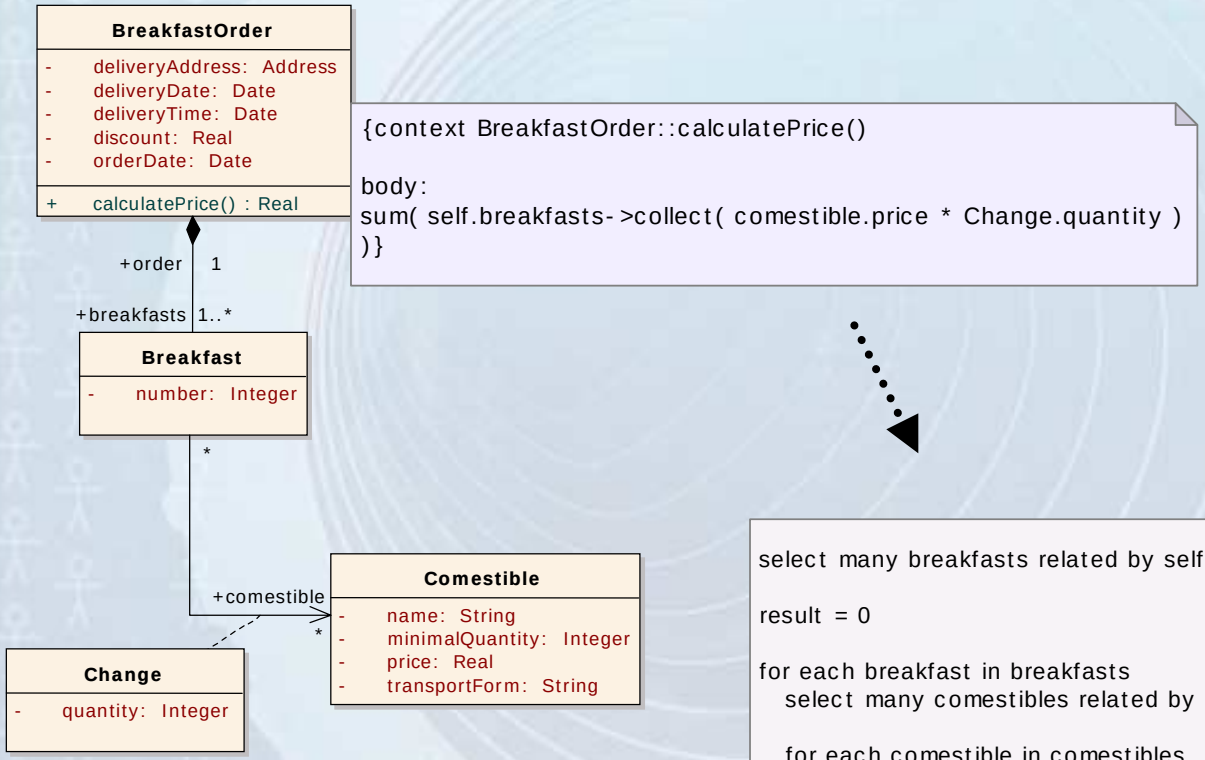
# Architecture Generation – Rosa's Breakfast Service

| Business Model → Analysis Model | | | | | | | |
|---|---|---|---|---|---|---|---|
| Source | | | | Destination | | | |
| Element | Stereotype | Tags | Add. Cond. | Element | Stereotype | Tags | Add. Instr. |
| Activity | Process | | | Class | Control | | |
| Activity | Process | | Is Aggregated | Class | Control | | |
| | | | | Association | | | Aggregate - Agregated |

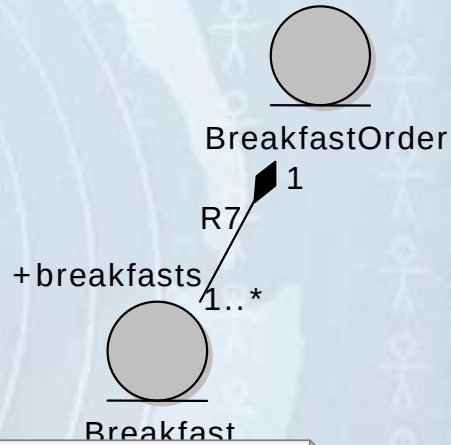
| Use Case Model → Analysis Model | | | | | | | |
|---|---|---|---|---|---|---|---|
| Source | | | | Destination | | | |
| Element | Stereotype | Tags | Add. Cond. | Element | Stereotype | Tags | Add. Instr. |
| State | | | Context UC is UI | Class | Boundary | isUI = true | |
| State | | | | Class | Boundary | isUI = false | |
| State Transition | | | Message is Self (interaction) | Association | Navigate | | Start State Class – End State Class |
| State Transition | | | Message is not Self | | | | State Class – Process Control Class |

# Architecture Generation – Rosa's Breakfast Service

- OCL → Action Language Compiler
- Rosa Example: CalculatePrice() Operation

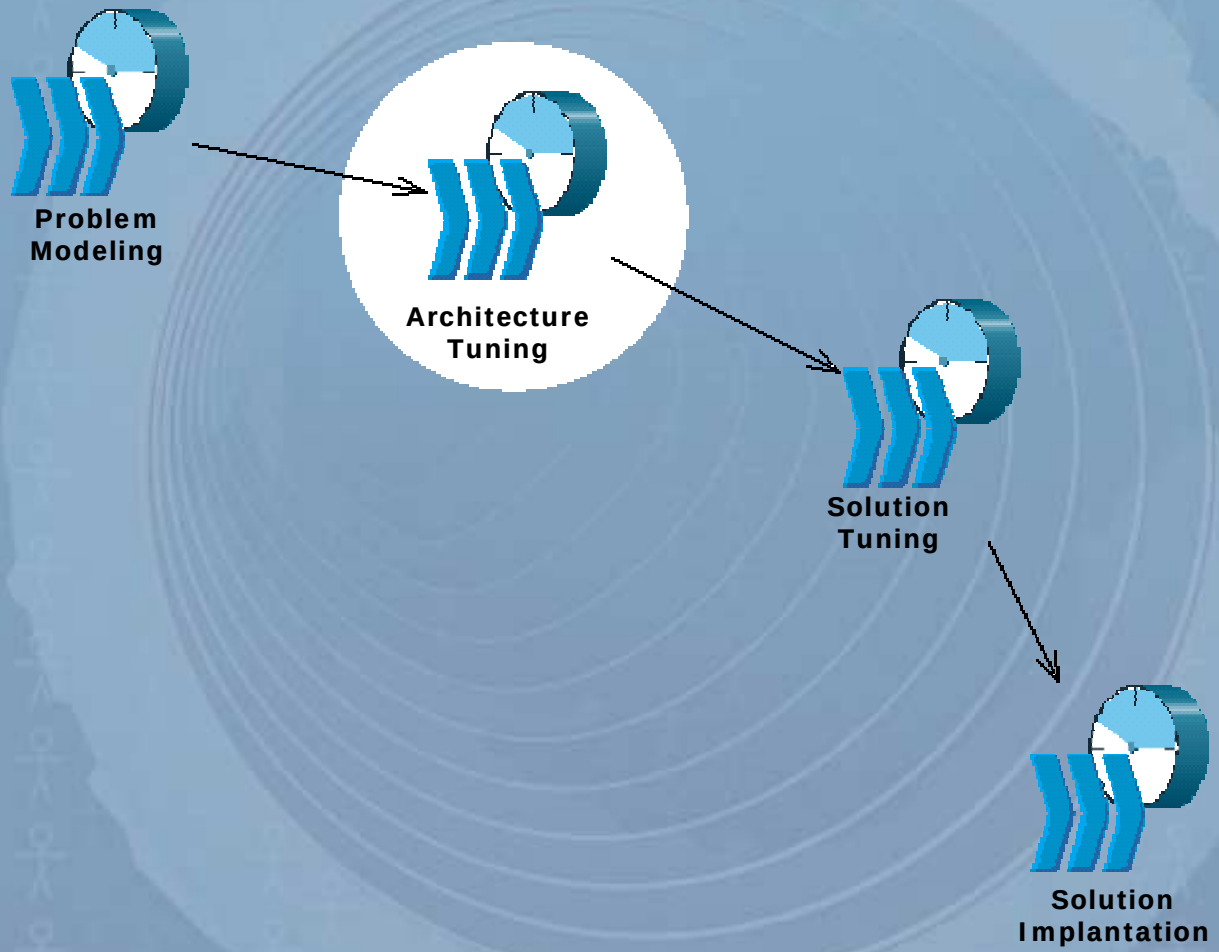



```
{context BreakfastOrder::calculatePrice()

body:
sum( self.breakfasts->collect( comestible.price * Change.quantity )
)}
```

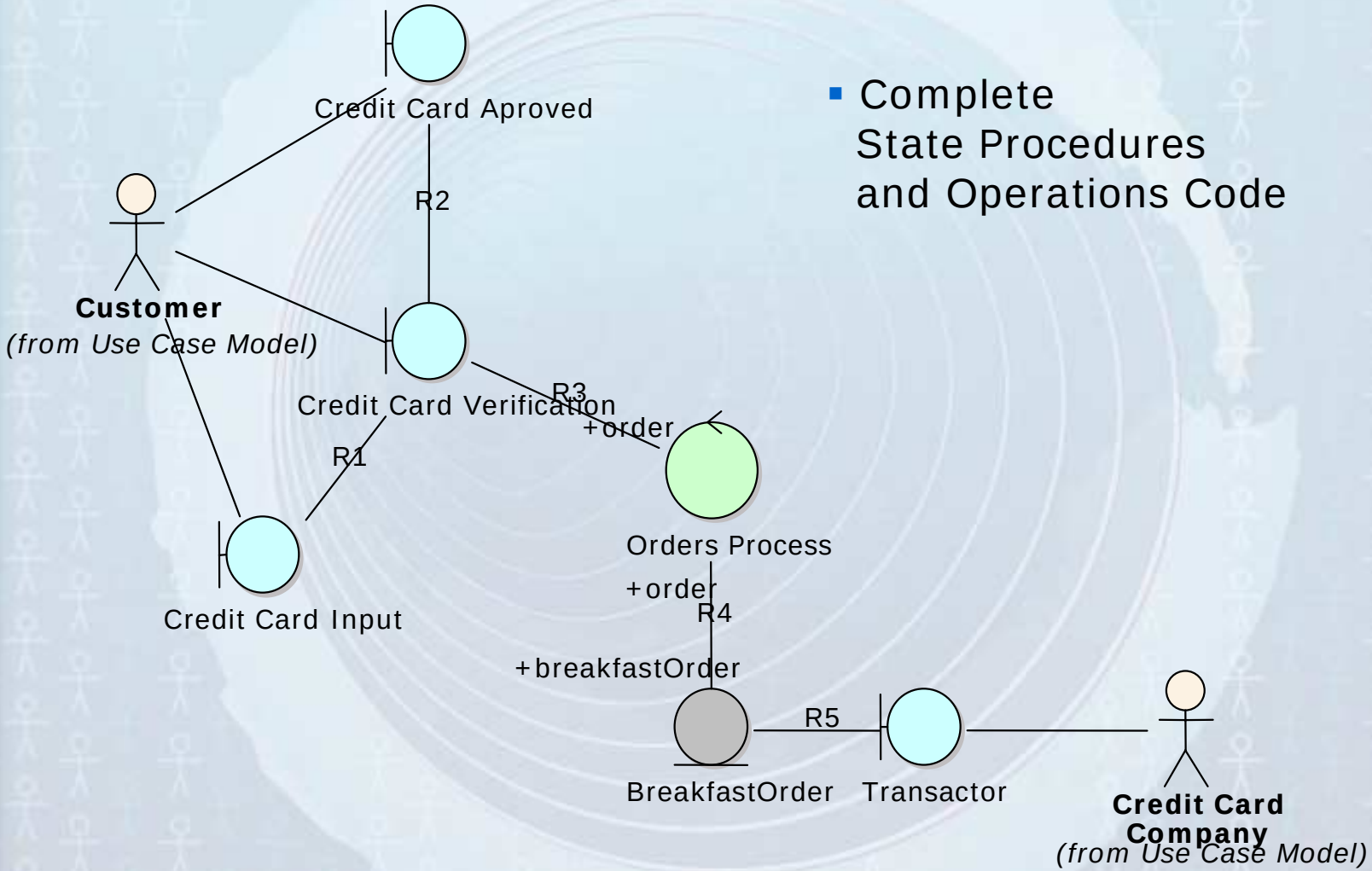



```
select many breakfasts related by self->Breakfast[R7]

result = 0

for each breakfast in breakfasts
  select many comestibles related by breakfast->Comestible[R8]

  for each comestible in comestibles
    select one charge related by comestible->Charge[R8]
    result = result + comestible.price * charge.quantity
  next

next
```

# The Second Phase: Architecture Tuning

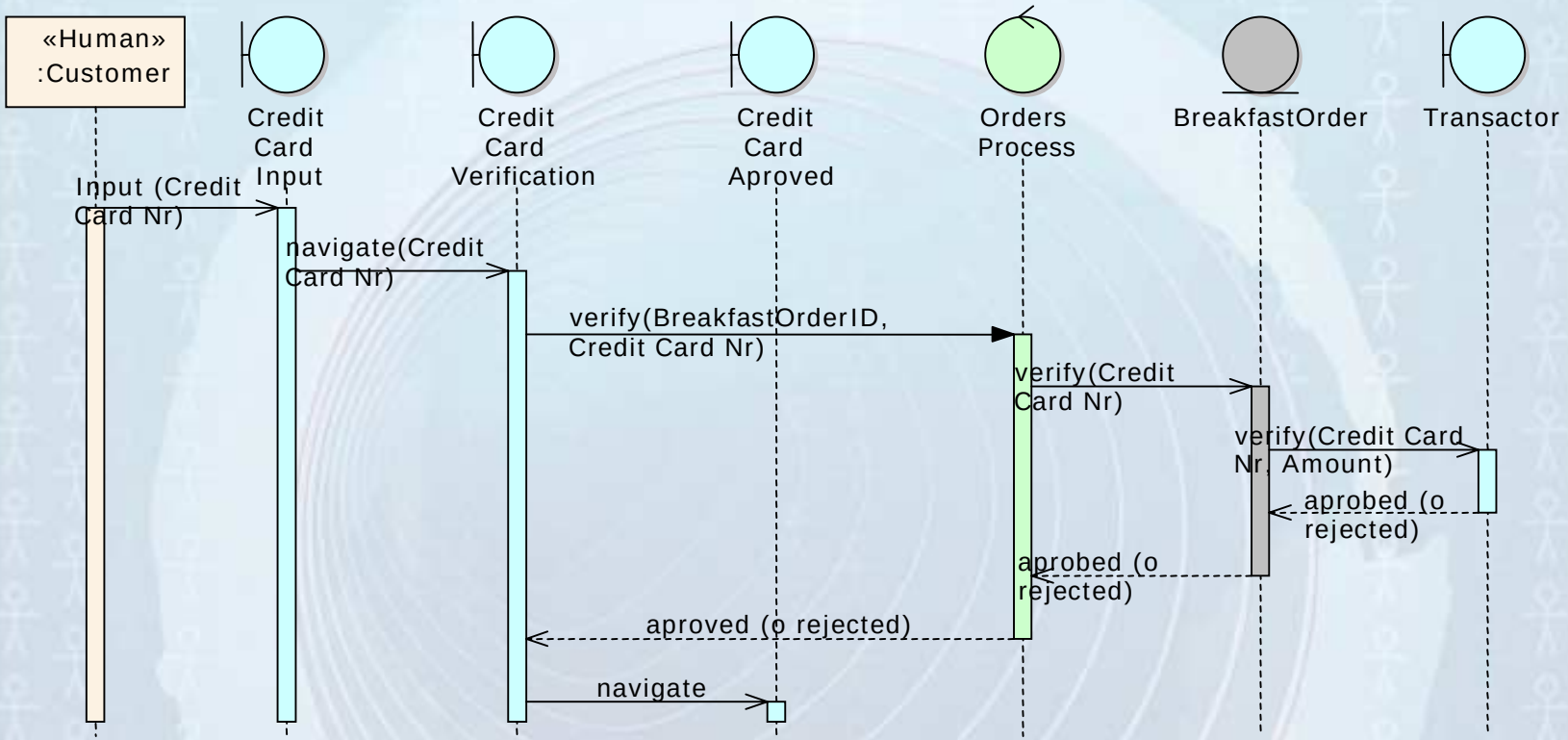


# Action Coding




- Complete State Procedures and Operations Code

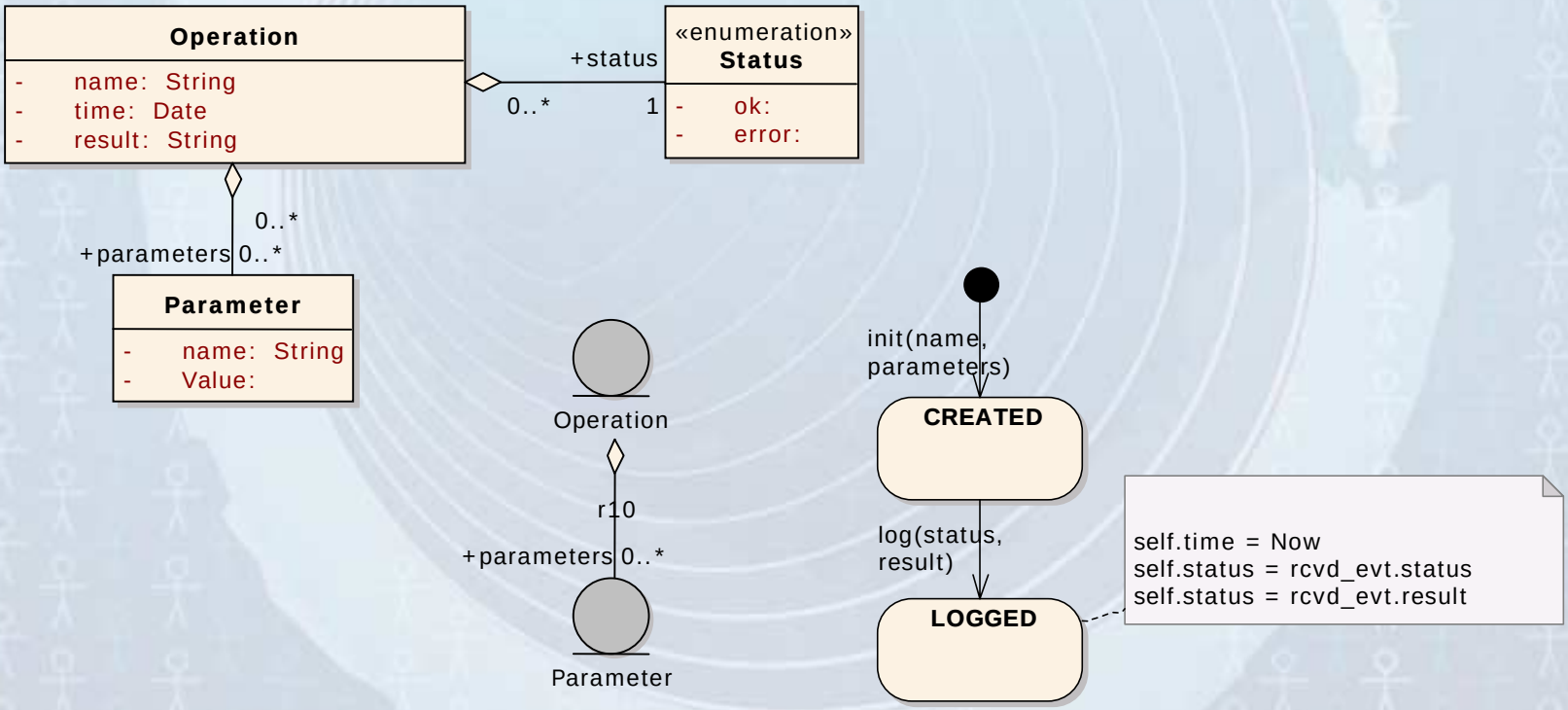
# Scenario Testing




- Architecture Interaction Diagrams Generated from Problem Interaction Diagrams
- Architecture Model Virtual Machine
- Simulations – Execution Traces

# Pervasive Services Identification and Configuration

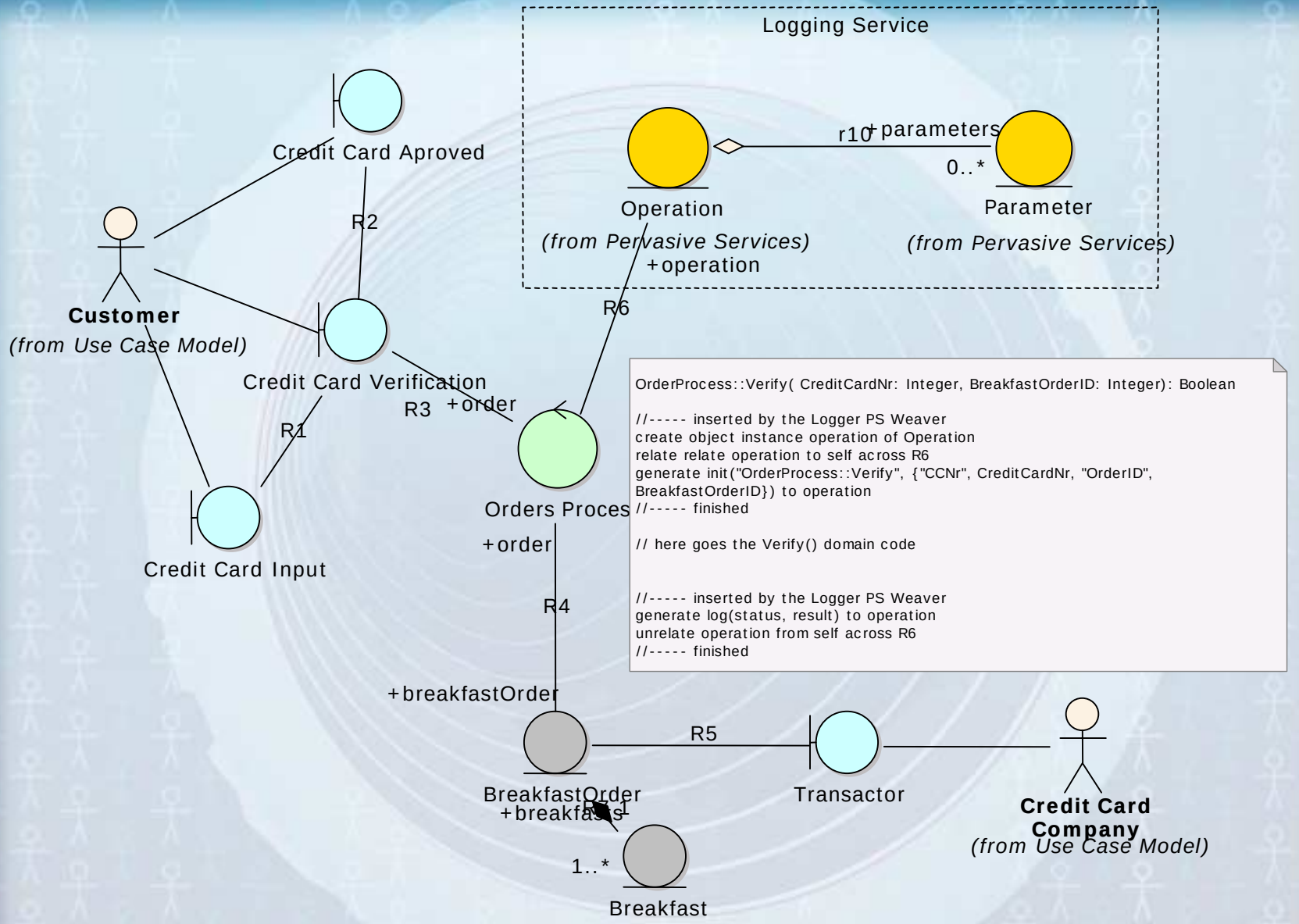
- Identify Needed Pervasive Services (PS)
  - Select From Available PS Modules, or
  - Develop New PS Modules
- Rosa Example: Logging Service



# Architecture Integration

- Integrate System with Pervasive Services
- Tasks:
  - Mark the “Join Points” in the Architecture Model
  - Weave the PS Modules with Architecture Model
- Rosa Example:
  - Requirement:
    All the Executions of theOrderProcess::Verify()
    should be registered
  - Solution:
    Tag the OrderProcess::Verify() with “isLogged = true”
- Mapping Rules are Part of Every Individual PS Module

# Architecture Integration (cont.)



# Solution Generation

- **Unified Architecture Metamodel** → Target Platform

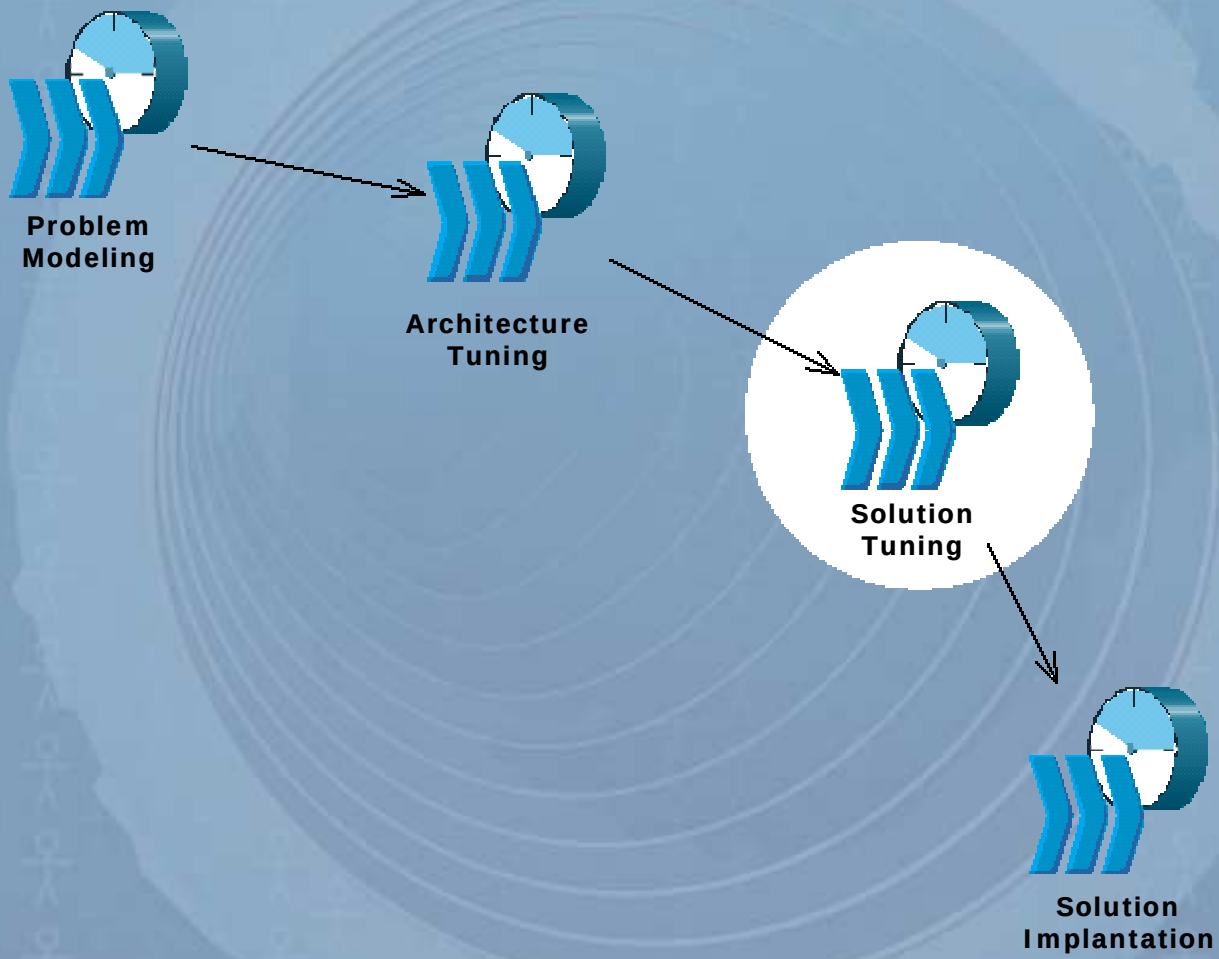
- Tasks:
  - Select the Platform (J2EE, .NET, etc)
  - Mark the Architecture Model (Manual)
  - Generate the Solution Model (Automatic)

- Model Transformations:
  - UML Metamodel (Architecture) → UML Metamodel (Platform)
  - Action Language → Target Platform Language

# Solution Generation – Rosa's Breakfast Service

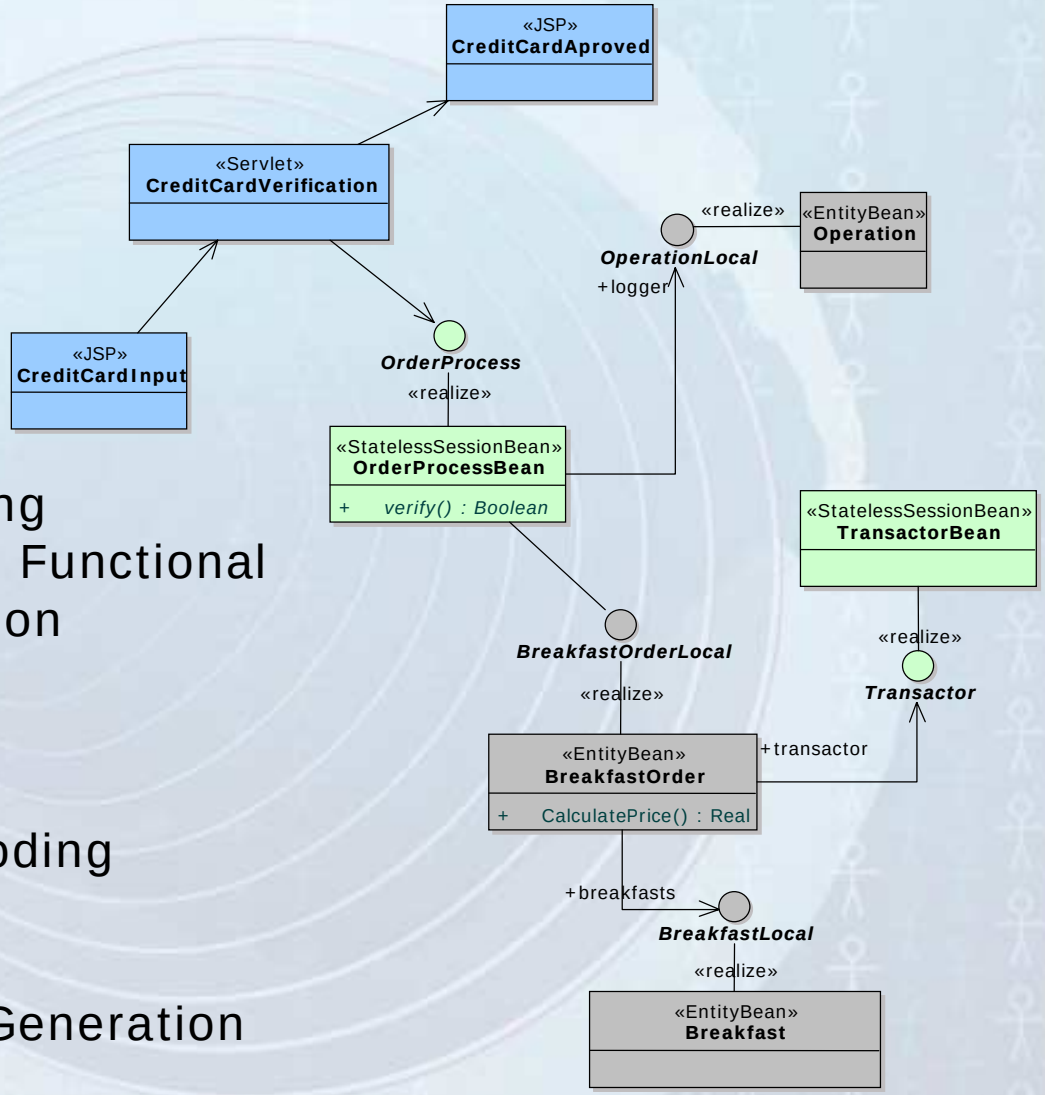
| Analysis Model → Solution Model | | | | | | | |
|---|---|---|---|---|---|---|---|
| Source | | | | Destination | | | |
| Element | Stereotype | Tags | Add. Cond. | Element | Stereotype | Tags | Add. Instr. |
| Class | Boundary | isUI = true | Has input | Class | JSP | | |
| Class | Boundary | isUI = true | Has Control | Class | Servlet | | |
| Class | Boundary | isUI = false | | Class | Stateless SessionBean | | |
| | | | | Interface | Home | | |
| | | | | Interface | Remote | | |
| Class | Control | | | Class | Stateless SessionBean | | |
| | | | | Interface | Home | | |
| | | | | Interface | Remote | | |
| Class | Entity | | | Class | EntityBean | | |
| | | | | Interface | Home | | |
| | | | | Interface | Remote | | |

# The Third Phase: Solution Tuning

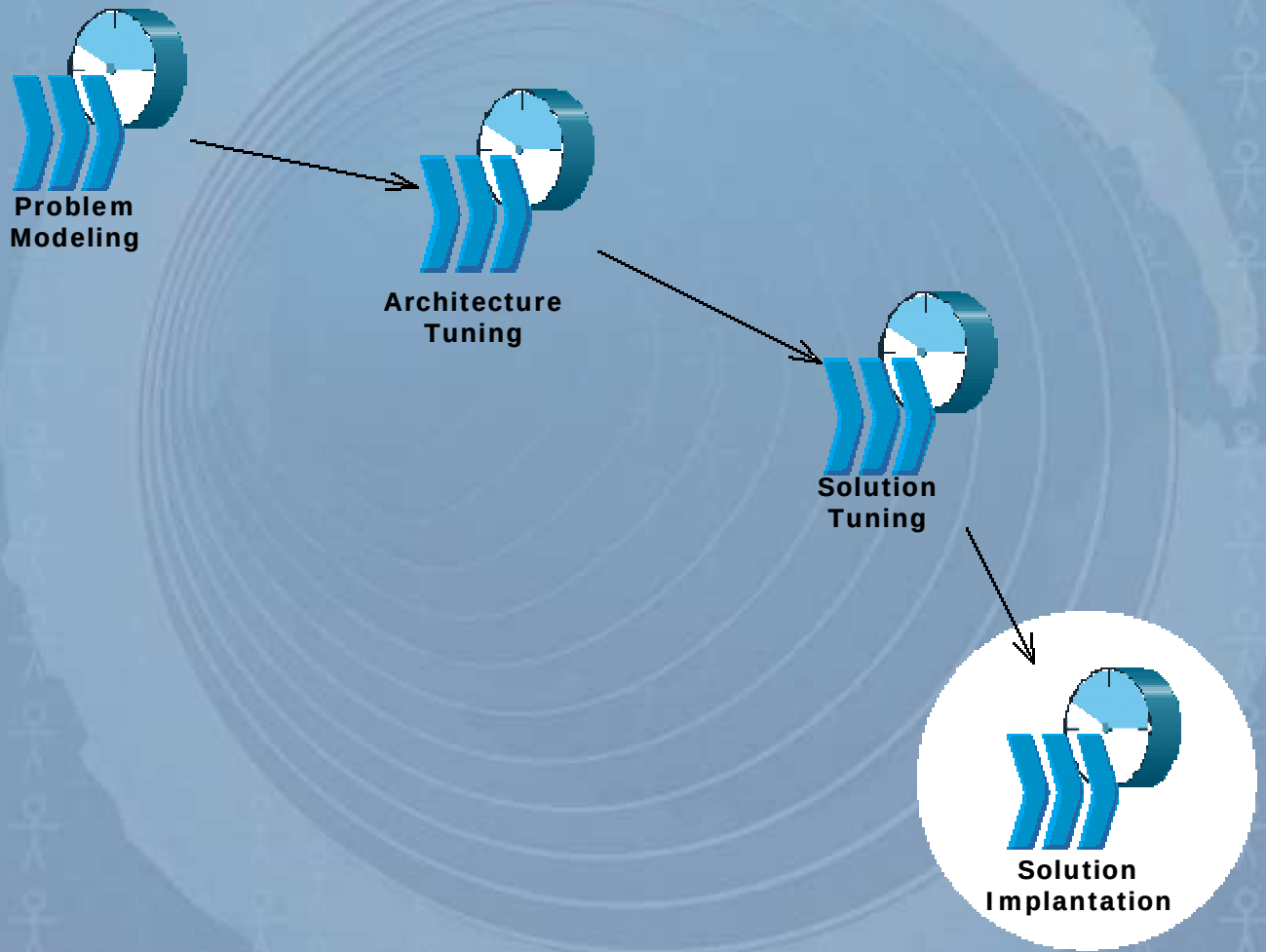


# Solution Generation

- Target Platform Testing
  - Functional and no Functional
  - Unit and Integration
- Refactoring
- Platform Language Coding (if needed)
- Installation Package Generation



# The Last Phase: Solution Implantation



# Future Work

- User Interface Interaction Modelling Metamodel
  - UML Profile v.s. Alternative Metamodel
- Transformations
  - Transformation Definition Metamodel
  - Model Compiler v.s. Archetypes
- Architecture Model Infrastructure
  - Virtual Machine
  - Simulator
- Pervasive Services Integration
  - Aspect Oriented Technics Formalization

# Bibliography


- ***"Executable UML, A Foundation for Model-Driven Architecture",*** *S. J. Mellor, M. J. Balcer*
- ***"MDA Explained, The Model Driven Architecture: Practice and Promise"****, A. Kleppe, J. Warmer, W. Blast*
- ***"MDA Distilled, Principles of Model-Driven Architecture"****, S. J. Mellor, K. Scott, A. Uhl, D. Weise*
- ***"Model Driven Architecture, Applying MDA to Enterprise Computing",*** *D. S. Frankel*
- *"**The Object Constraint Language Second Edition, Getting Your Models Ready for MDA**", J. Warmer, A. Kleppe*
- *"**Mastering AspectJ**", J. D. Gradecki, N. Lesiecki*
- *www.omg.org/mda, www.uml.org*


Thank You!
Aleksandar Orlic
aorlic@craftware.net