

## **UML4SPM: Combining Aspect and Model-Driven Engineering Approaches for Software Process Modeling and Execution**

**Reda Bendraou (a) , Jean-Marc Jez équ é (b), and Franck Fleurey(c)**

**(a) University Pierre & Marie Curie, Paris, France**

**[reda.bendraou@lip6.fr](mailto:reda.bendraou@lip6.fr)**

**(b) INRIA-Rennes Bretagne Atlantique, France**

**(c) SINTEF, Oslo, Norway**

# UML4SPM: Context

- **OMG's SPEM2.0 Standardization efforts**
  - **RFP:** executable UML process models
- **ModelWare and ModelPlex European Projects**
  - More than 20 M. Euros Projects.
  - Partners: IBM Haifa, Thalès, Softeam, Adaptive, France Télécom, SINTEF, INRIA, UPM, Fokus Fraunhofer, SAP, Schlumberger, etc.
  - **Goal:** MDE techniques for building reliable software
    - UML as a Building block

# Basic Process Modeling Language Requirements

From [Riddle 89] [Kellner 89] [Curtis 92] and [Jaccheri 99]

- Expressiveness (Semantic Richness)
  - Description of basic process elements
    - *Activity, Artifact, Role, Human and Tool* [Humphrey 91] [Dowson\_91] [Conradi 95]
  - Activities and steps sequencing
    - *Proactive control Vs. Reactive control*
- Understandability
  - Code like descriptions Vs graphical representations
  - Multiple perspectives
  - **Project Requirement: Use of the UML standard**
- Modularization
  - The ability to combine different chunks of processes

# UML4SPM Specific Requirements

- **Executability**

- Constructs with operational semantics
- Support of process execution and simulation
- **Obstacle**: UML models are not executable

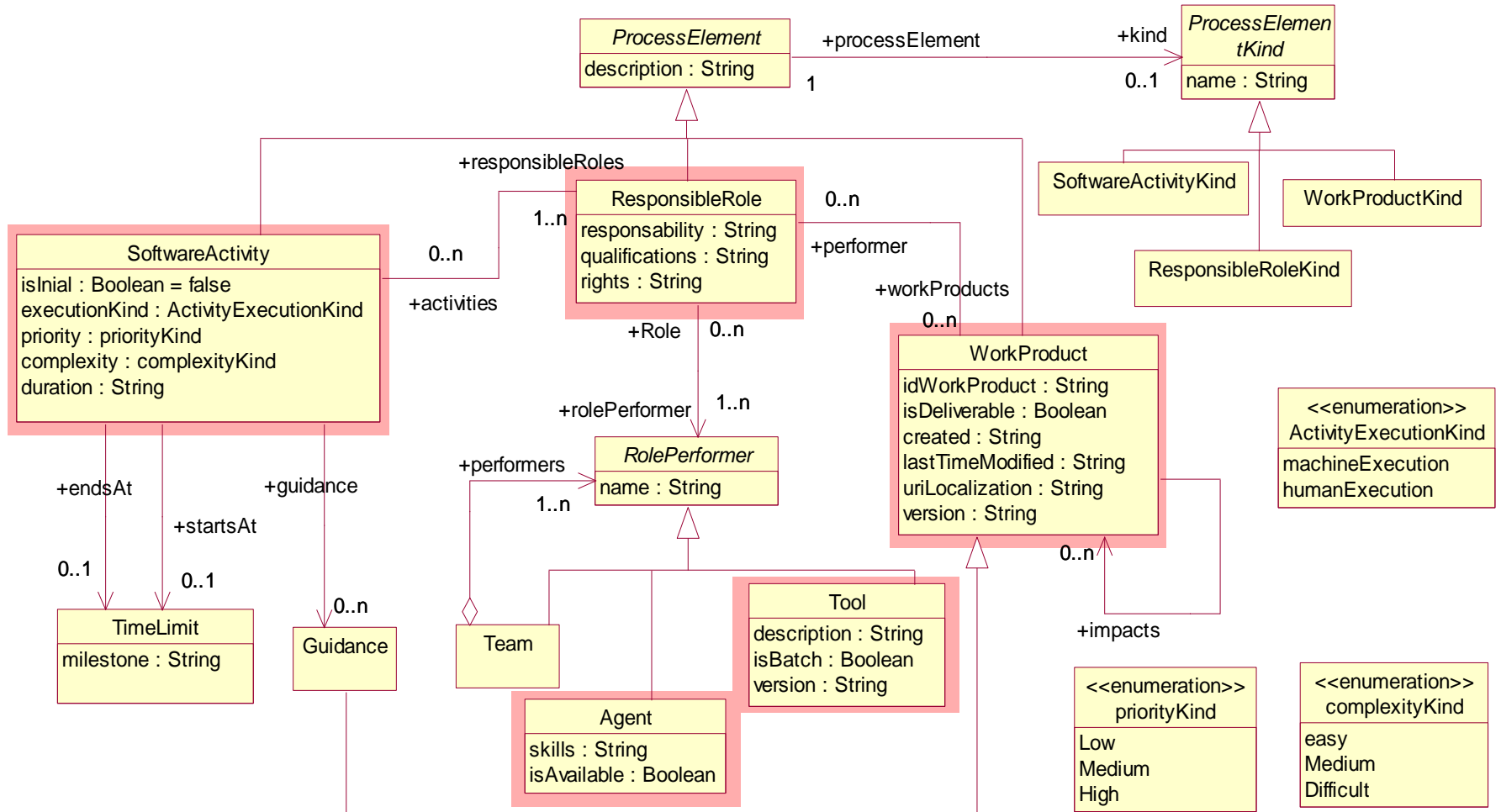
- **Flexibility**

- To be able to adapt and customize your Software Process to specific projects
- Extending the SPML at lower costs
  - No need to build a new interpreter/compiler for the SPML
- **Humans** as deciders for the process workflow
  - **Modification of the process model at runtime**

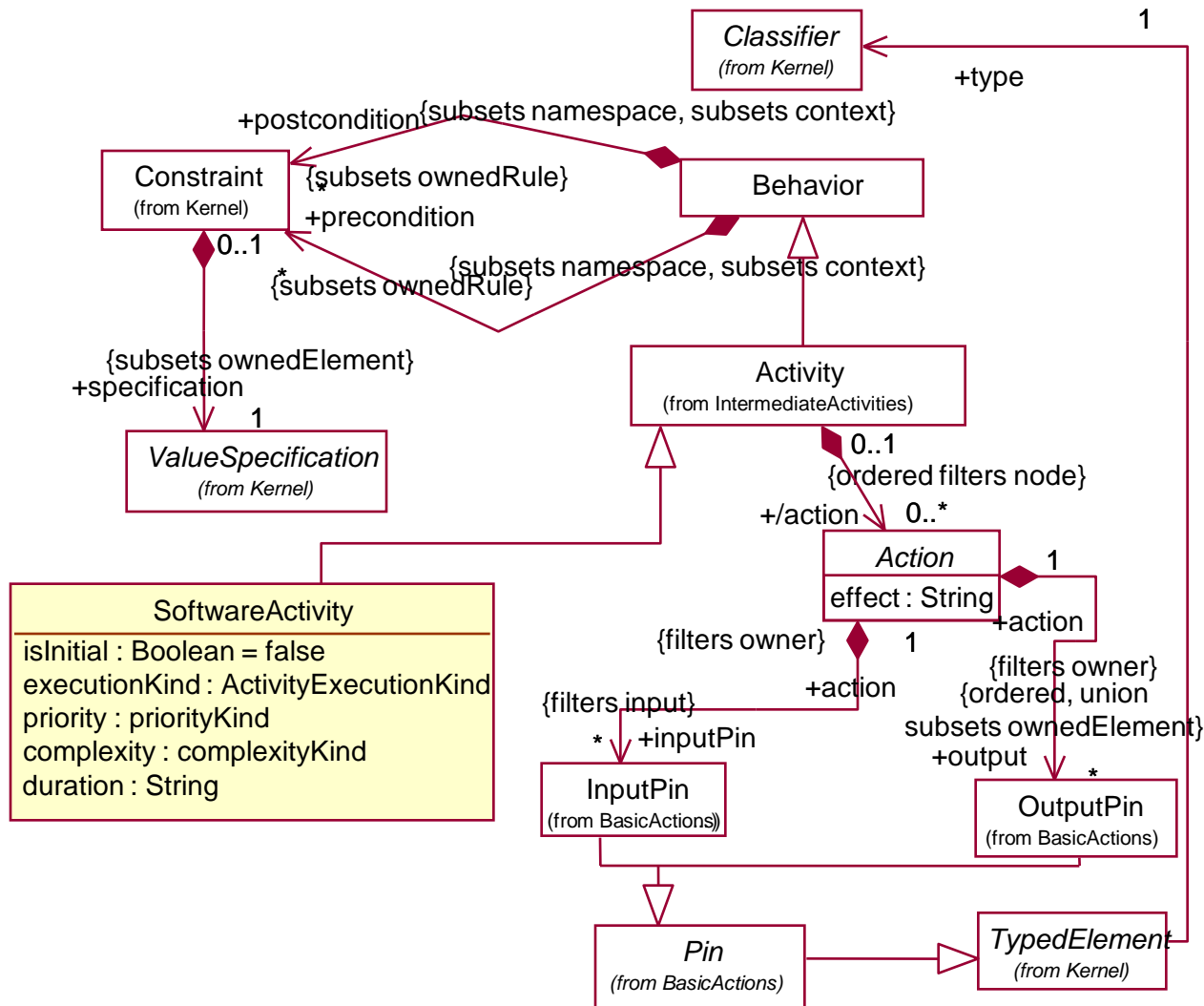
# Outline

- UML4SPM: Context and Requirement
- **The UML4SPM Language**
- Execution of UML4SPM Software Process Models
  - The UML4SPM Execution Model Approach
  - Implementation: the *Kermeta* metaprograming language
- Conclusions & Perspectives

# UML4SPM: Metamodel



# Connecting UML4SPM with UML2.0 Metamodel



# Outline

- UML4SPM: Context and Requirement
- The UML4SPM Language
- **Execution of UML4SPM Software Process Models**
  - The UML4SPM Execution Model Approach
  - Implementation: the *Kermeta* metaprograming language
- Conclusions & Perspectives

# UML4SPM: Executability

## Two Approaches were explored

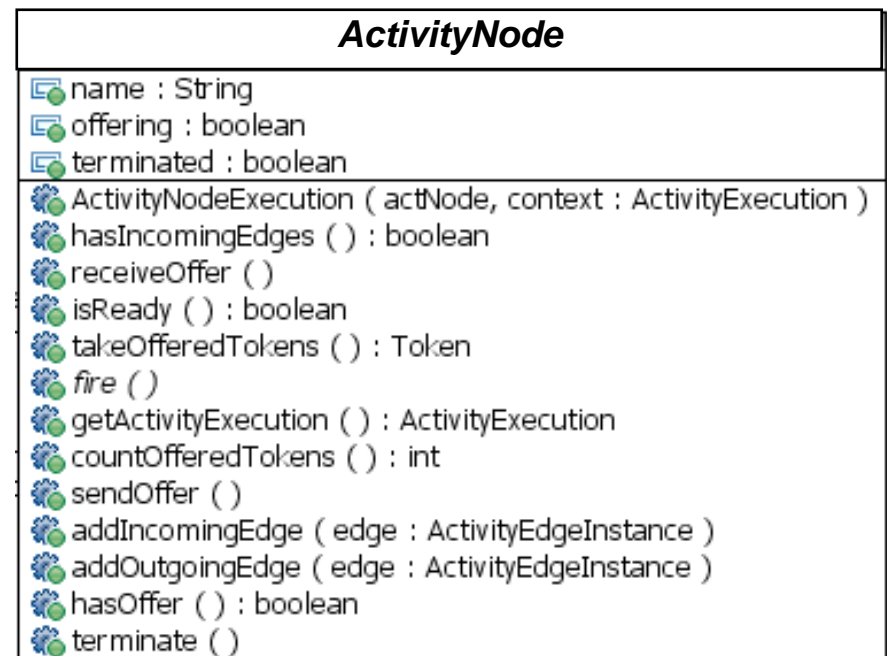
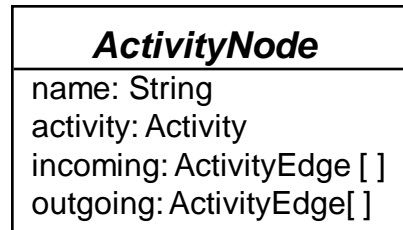
- Implementation of an Interpreter for UML4SPM models : The UML4SPM Execution Model
- Mapping of UML4SPM process models toward a Process Execution Language

# The UML4SPM Execution Model Approach: Rational

- Definition of an Execution Model for UML4SPM
  - Specifies the precise execution behavior of UML4SPM elements
  - Goal: UML4SPM Process models executed straightforwardly
  
- Comes in form of *classes and operations*
  - The execution behavior respects the UML2.0 *Activity* and *Action* semantics
  - Petri-nets based semantics (offering and consuming tokens)

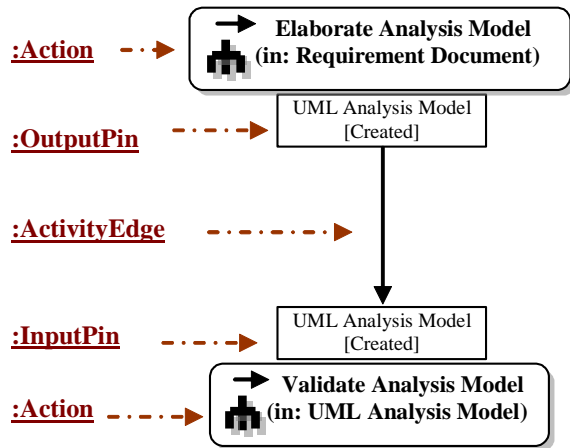
# The UML4SPM Execution Model Approach: Rational

- For each UML4SPM metaclass (element) → we define its execution behavior in the execution model



**ActivityNode** (i.e., *Action*, *ControlNode* or *ObjectNode*)

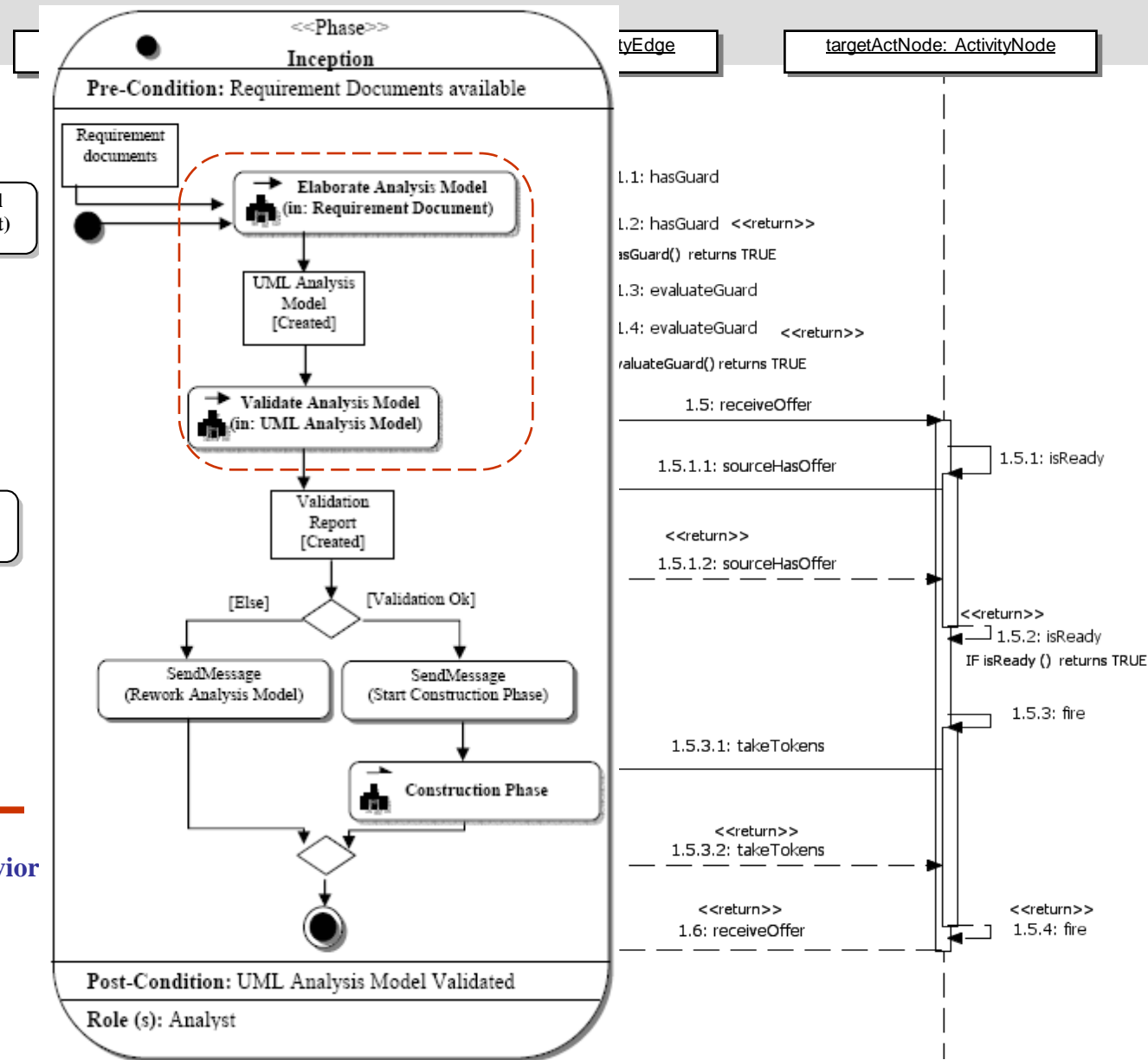
# The UML4SPM Execution Model: Rational



A chunk of the process model



Its operational behavior

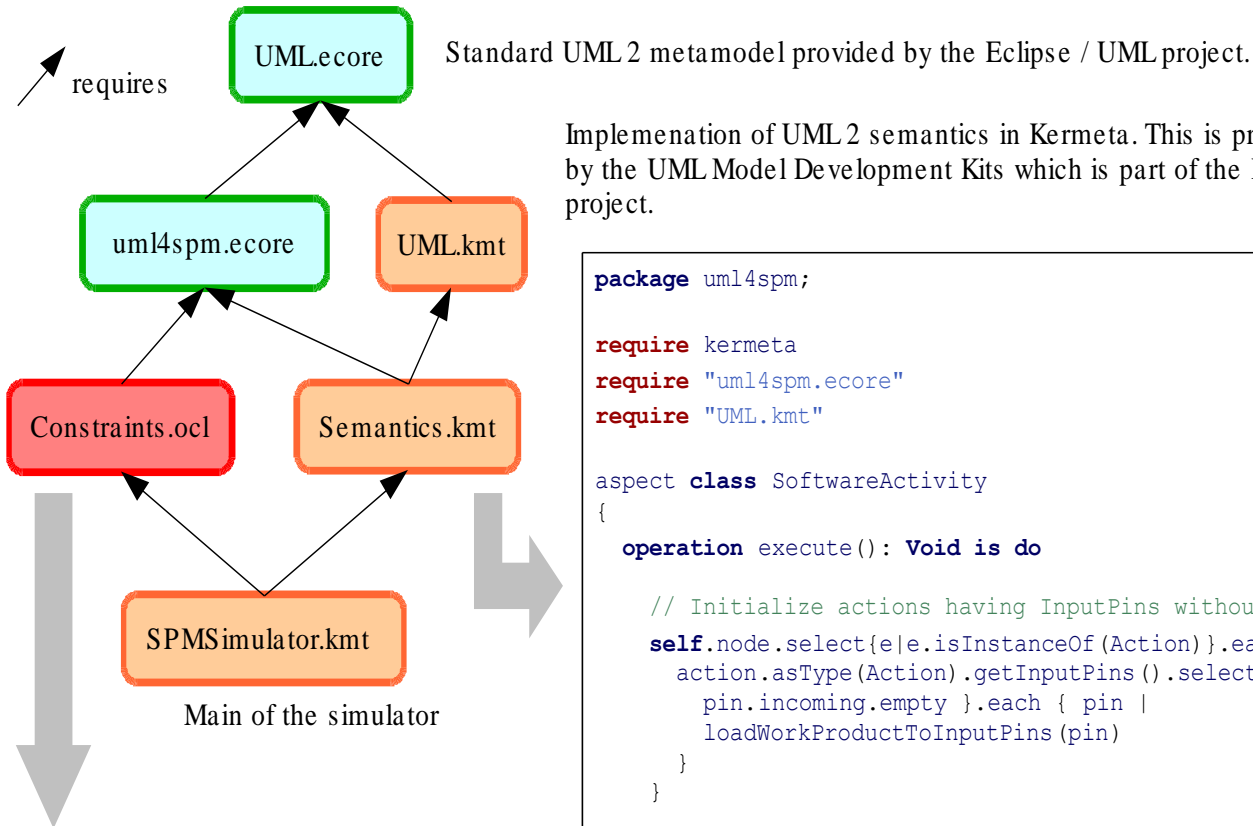


# The UML4SPM Execution Model: Implementation

- UML4SPM operational semantics using *Kermeta*
  - Extension to MOF (Meta Object Facility, a meta-meta model)
  - Imperative OO Language, static typing
  - Support for OCL (Object Constraint Language)
  - A Framework to support aspect modeling
    - Possibility to compose different aspects on the same Metamodel
- An «*Interpretation*» aspect is woven to UML4SPM
  - The UML4SPM metamodel is not affected
    - Weaving at runtime
  - UML4SPM model instances straightforwardly runnable and testable

# The UML4SPM Execution Model :

## The *Kermeta* approach



Implementation of UML 2 semantics in Kermeta. This is provided by the UML Model Development Kits which is part of the Kermeta project.

```

package uml4spm;

require kermeta
require "uml4spm.ecore"
require "UML.kmt"

aspect class SoftwareActivity
{
    operation execute(): Void is do

        // Initialize actions having InputPins without incoming edges
        self.node.select{e|e.isInstanceOf(Action)}.each{ action |
            action.asType(Action).getInputPins().select{ pin |
                pin.incoming.empty }.each { pin |
                    loadWorkProductToInputPins (pin)
                }
            }

        //Initialize Activity's Intial Nodes
        self.node.select{e|e.isInstanceOf(InitialNode)}.each { inode |
            inode.asType(InitialNode).fire ()
        }
    end

    [...]
}

```

```

Context Team inv:
    self.performers->forAll ( roleperformer |
        not roleperformer.isKindOf (Tool)
    )

```

# The UML4SPM Execution Model : Results

## ✓ Executability

- Respects the Execution Semantics defined by the UML2.0 standard
  - + Reusable for UML2.0 Activity Diagram executions
- UML4SPM Execution Model used as a Process Engine
  - UML4SPM Process Models executed through interpretation
    - **No additional steps or configuration are needed**
- Early feedback through simulation and testing
  - Automation of many iterative and no-interacting process's activities

## ✓ Flexibility

- Strong Coupling of UML4SPM Process Models and their Executions
  - Possibility to modify the process definition at runtime without restarting the process execution
    - **Conditions still need to be identified**
- UML4SPM Language and its Execution Model easily extensible
- Process variations through Aspects weaving

# Questions !

- Thank you