

# A Reinterpretation of Pattern to increase the Expressive Power of Model-driven Engineering

**Matteo Bordin**

AdaCore (F)  
bordin@adacore.com

**Marco Panunzio**

**Carlo Santamaria**

**Tullio Vardanega**

University of Padova (I)

**1<sup>th</sup> Int'l Workshop on Model Based Architecting and  
Construction of Embedded Systems – ACESMB 2008**

September 29<sup>th</sup>, Toulouse, France

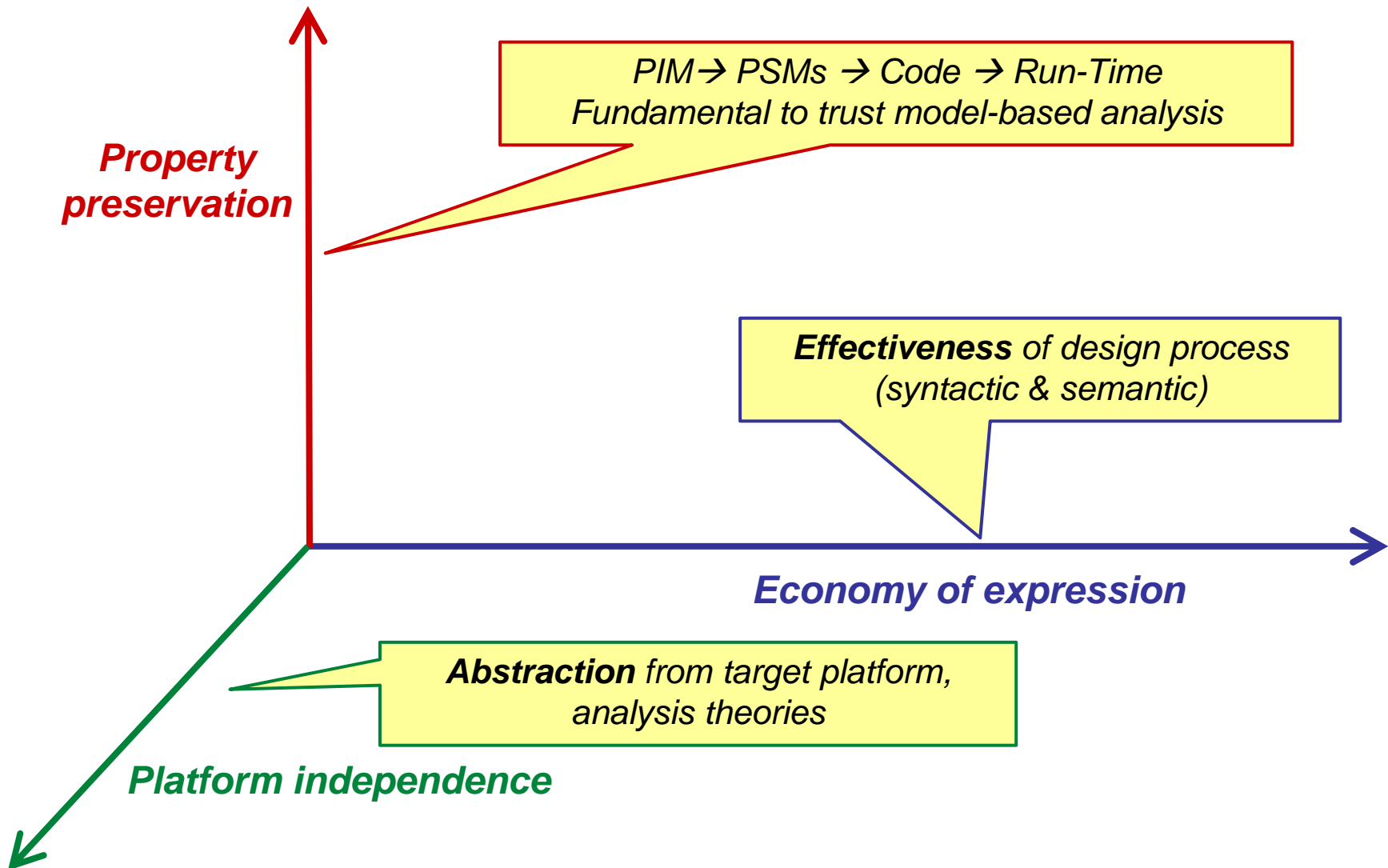


## Summary

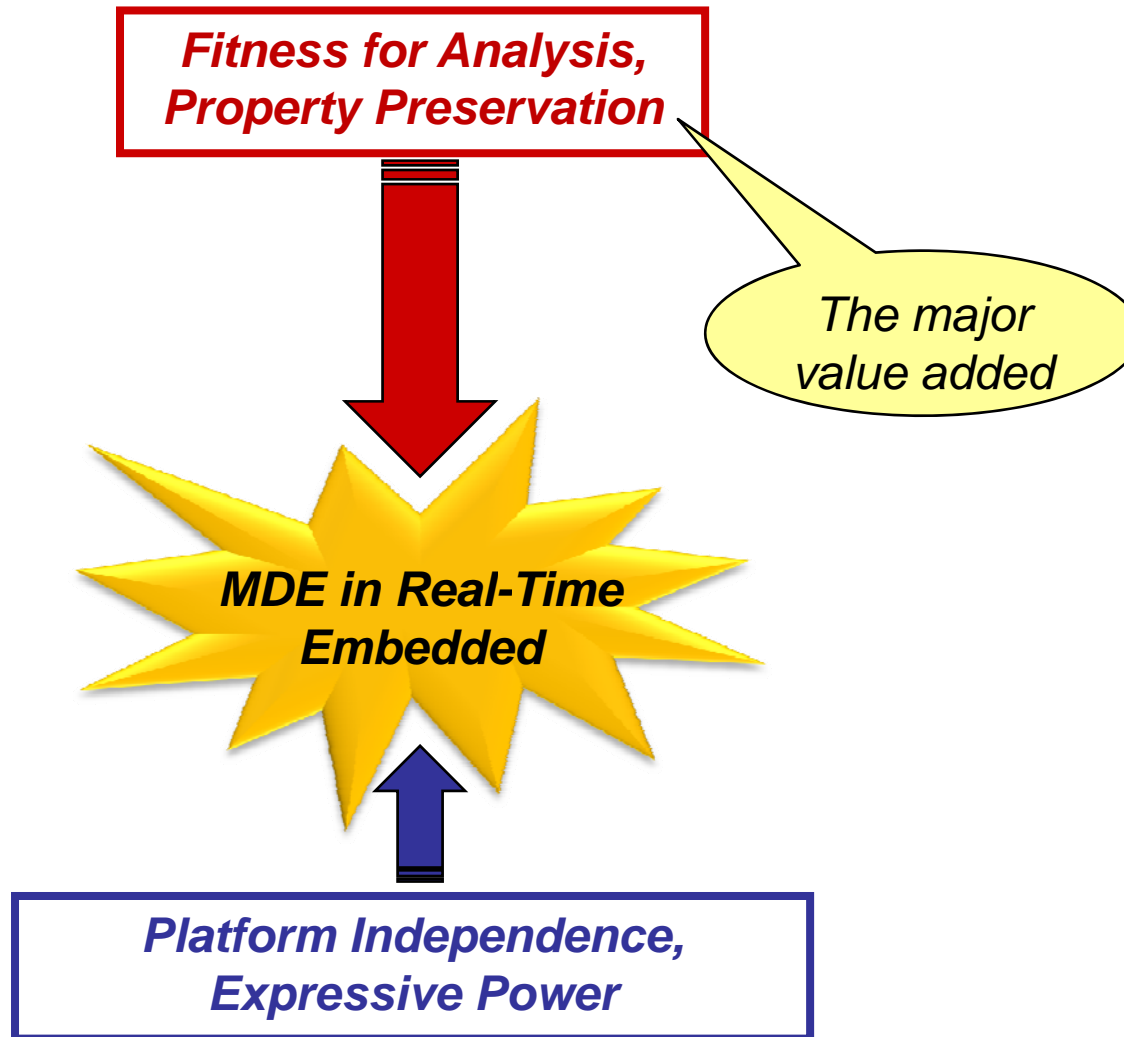
---

- 1. Evaluation of Model-driven Methodologies**
- 2. Current limitations**
- 3. How to improve?**

## Evaluation of Model-driven Engineering Methodologies



## Model-driven engineering in real-time embedded systems



## PSM-to-PIM semantic dependence: examples

---

- **Task ontology: restrictions**
  - **Single suspension point**
  - **Asynchronous communication only**
  - Assumed by several analysis theories: perfectly reasonable
  - The functional specification of task entry point **cannot**
    - Issue suspending calls
    - Allow return parameters with **out** mode
- **Design abstraction: msg-based communication**
  - **Design** a protected msg queue (with size, ceiling priority, etc.)
  - **Design** a server to fetch messages from the queue
  - Clients posts messages on the queue



## State of the art

---

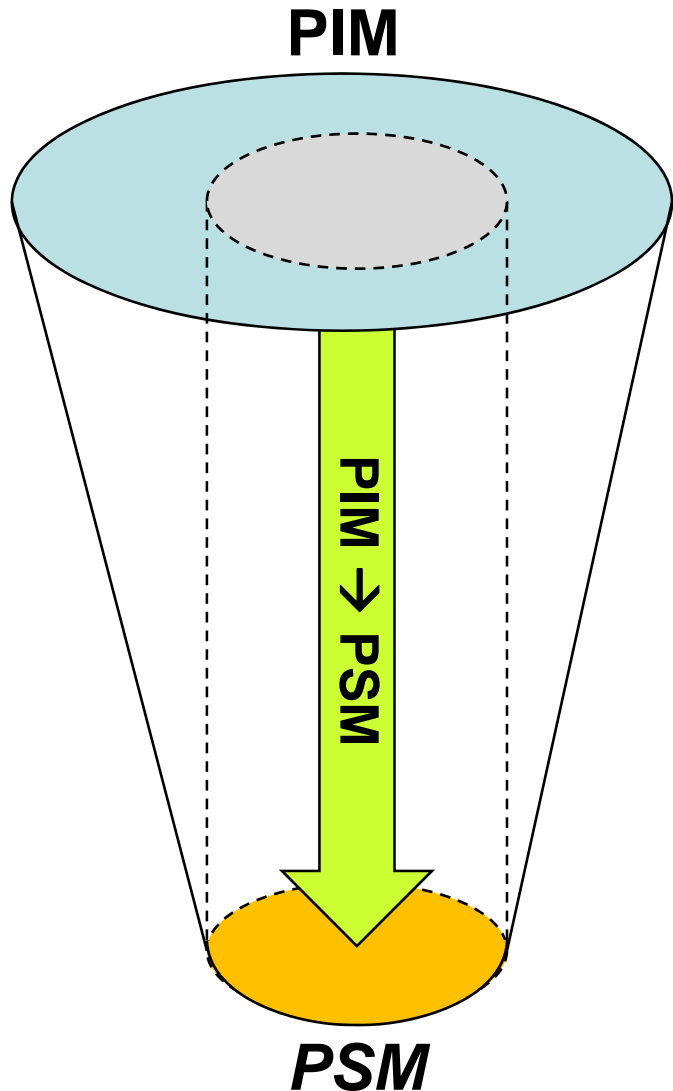
- **AADL and MARTE**
  - **PIM space**
    - PSM economy of expression (threads, shared resources, queues)
    - PSM-induced semantic constraints (to facilitate static analysis)
  - **The PIM is no other than a direct projection of the PSM**
  
- **Goals**
  - **Improve expressive power**
    - Economy of expression
  - **Improve platform independence**
  - **Guarantee property preservation and analyzability**

## Revisiting the Notion of Design Pattern

---

- **Design patterns – *historically***
  - **Design-level solution** to a **recurrent problem** in a given domain
  - Up to the designer: **acknowledge & instantiate**
  
- **Design patterns - *in MDE***
  - **Automated pattern application** via **model transformation**
  - From a **declarative specification** at PIM level
  - By **automatically** acknowledging the **“right”** architecture

## Determined, Declarative and Executive Patterns



### Determined patterns (GoF)

- Explicit instantiation at application level
- Live in PSM projection on the PIM



### Executive patterns

- Features/constraints specific of the PSM
- Implemented in the OS/Middleware
- Instantiated via model transformation
- Goal: trustful representation of run-time platform rather than code generation



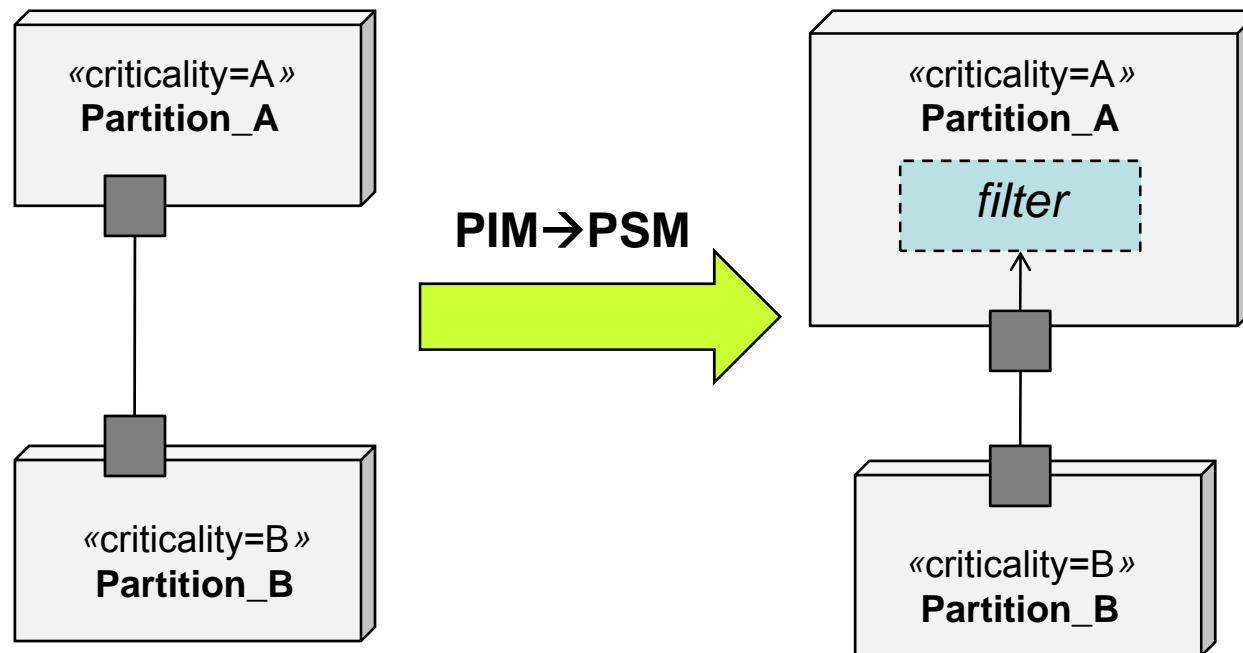
### Declarative patterns

- Offered by modeling infrastructure
- Visible to the user
- Instantiated via model transformations
- May be implemented at Application, Middleware or OS level

## Executive patterns: example

- **Communication filters**

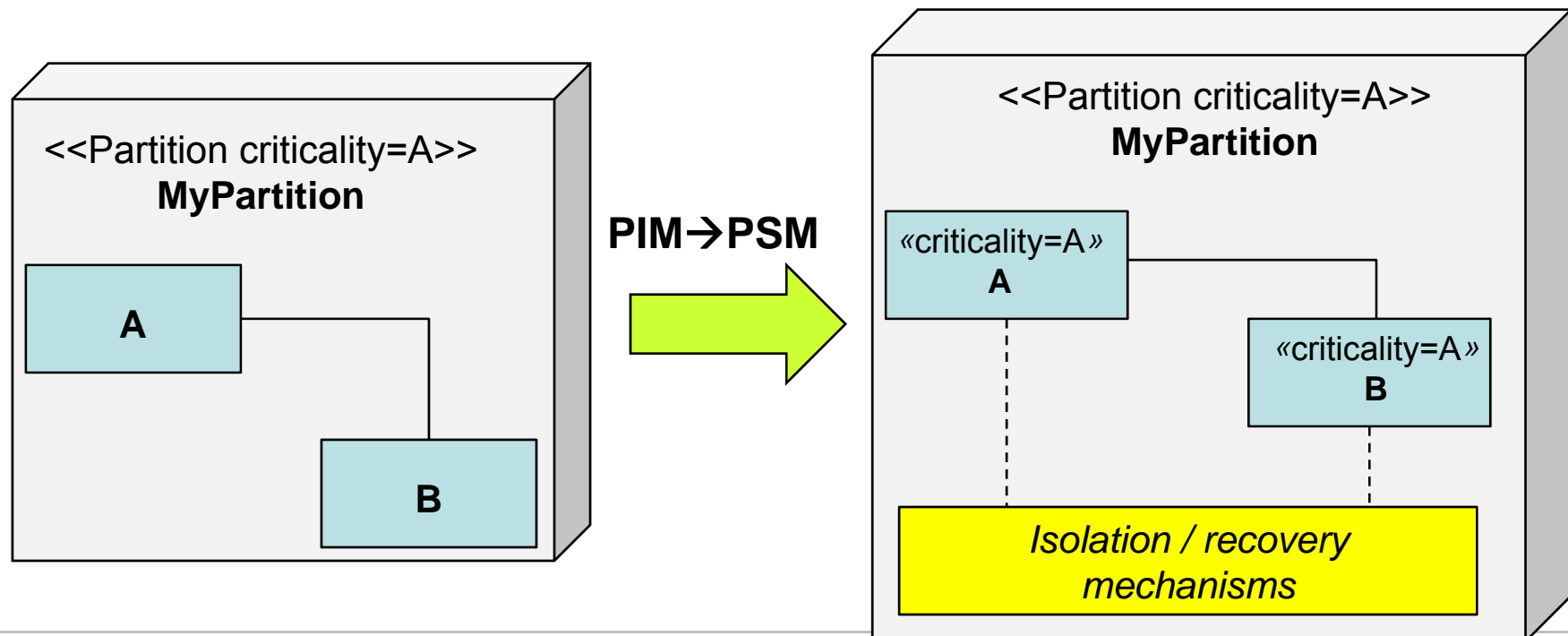
- A lower-criticality partition wishes to communicate to a higher-criticality one
- Check permissions & insure data integrity
- **No user action required**
- **Automatically realized by model transformation**
- **Solution implemented in the OS/Middleware**



## Declarative patterns: example

- **Partition**

- User action: attach criticality level to partitions
  - All entities deployed on a partition inherit its criticality
  - The required isolation mechanisms are automatically attached to partitions
- **Automatically realized by model transformation**
- **Solution implemented in the OS/Middleware**



## Executive/Declarative Patterns: the Callback

---

- **To permit synchronous deferred calls**
  - Prohibited by most analysis theories
- **To extend the PIM modeling space**
  - Beyond the projection of PSM
- **Impact on both functional and architectural specifications**
  - Implemented at application *and* middleware layer
- **See paper for details**

## Discussion and Conclusions (I)

---

- **Evaluation of model-driven methodology**
  - Property preservation
  - Economy of expression
  - Platform independence
- **Current methodologies**
  - Driven by PSM-induced constraints
    - For example: static analysis
  - PIM is just a projection of the PSM

## Discussion and Conclusions (II)

---

- **Patterns in MDE**
  - Beyond explicit instantiation by the user (a-la GoF)
  - Rely on automated model transformations
    - From explicit declarative specification
    - Automatically by acknowledging a need implied in the PIM
- **Results**
  - More expressive (→ more effective) design process
    - PIM space less constrained than direct project of PSM
    - Assurance of property-preserving solution
      - Essential in the high-integrity real-time domain
    - Economy of expression
      - Better fit with the true intent of MDE