Simplifying Complex Systems Behavior Modeling for MDA

Janusz Dobrowolski

StateSoft Inc,

JanuszDobrowolski@StateSoft.org

Prof. Bogdan Korel

Illinois Institute of Technology

korel@iit.edu

Presentation outline

- Behavior problem recognition
- Proposed Solution: VeUML
- VeUML and UML comparison
- Conclusions

UML Most Recognized Problem

"The weak area of UML is in the behavioral or dynamic part."

From the 2003 book "MDA Explained"
by A. Kleppe, J. Warmer, W. Bast

The term "weak", as understood by a prospective, user may relate to:

- Varying degree of automated code generated by existing tools
- Limited automated analysis (validation)
- Partially executable models

VeUML proposed solution: 100% executable model generation and validation is possible for complex behavior modeling.

The Big Goal

"The goal over the next 20 years will not be speed, cost or performance; it will be a question of complexity."

Bill Raduchel, Chief Strategy Officer, Sun Microsystems

#1 Software Industry Misconception

“ Though state models offer very good support in some domains (including the telcom, some embedded systems, certain styles of UI), others have only little use of state models (e.g. medical picture processing, accounting, insurance systems).”

Popular perception

VeUML: For the same complexity of behavior problem, VeUML state models are equally usable regardless of the domain including IT, telecom, embedded systems, medical picture processing, accounting and insurance systems.

#2 Software Industry Misconception

“FSM’s are by far not complete.”

Popular perception

VeUML: delivers complete FSM based solution for the complex systems behavior modeling.

#3 Software Industry Misconception

"State transition diagrams aren't created for every class; they are used only for very complex classes".

Source: Book by Wendy Boggs, Michael Boggs
"Mastering UML withRational Rose®"

Reality: In the above approach by the time the state transition diagrams (STD) complexity is fully understood it becomes overwhelming.

Coupling Data and Behavior creates substantial validation scalability problems for Complex Systems

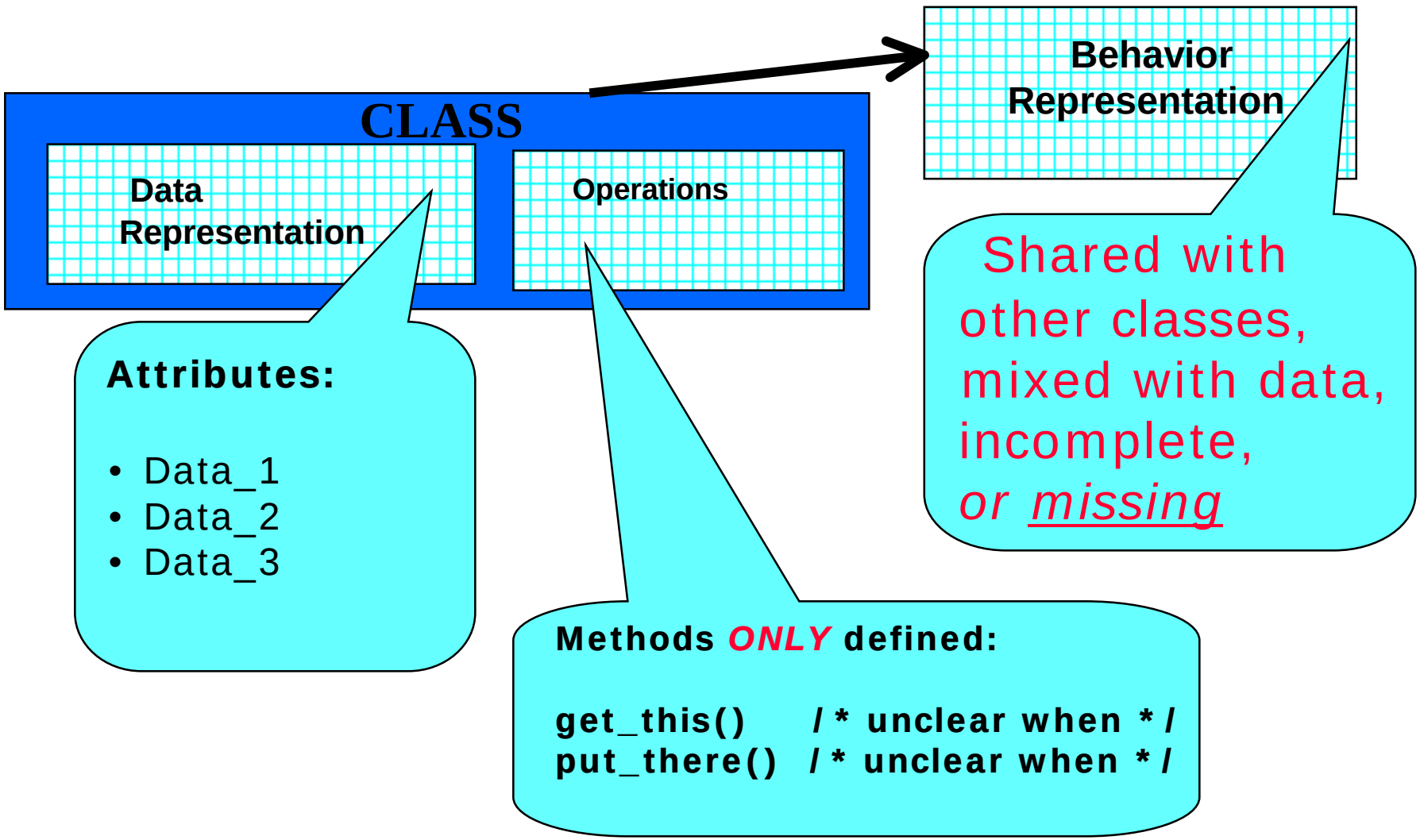
- *State explosion*
- *Inefficient validator execution time*
- *Excessive validator memory requirements*
- *Validator complexity*

Proposed solution: VeUML

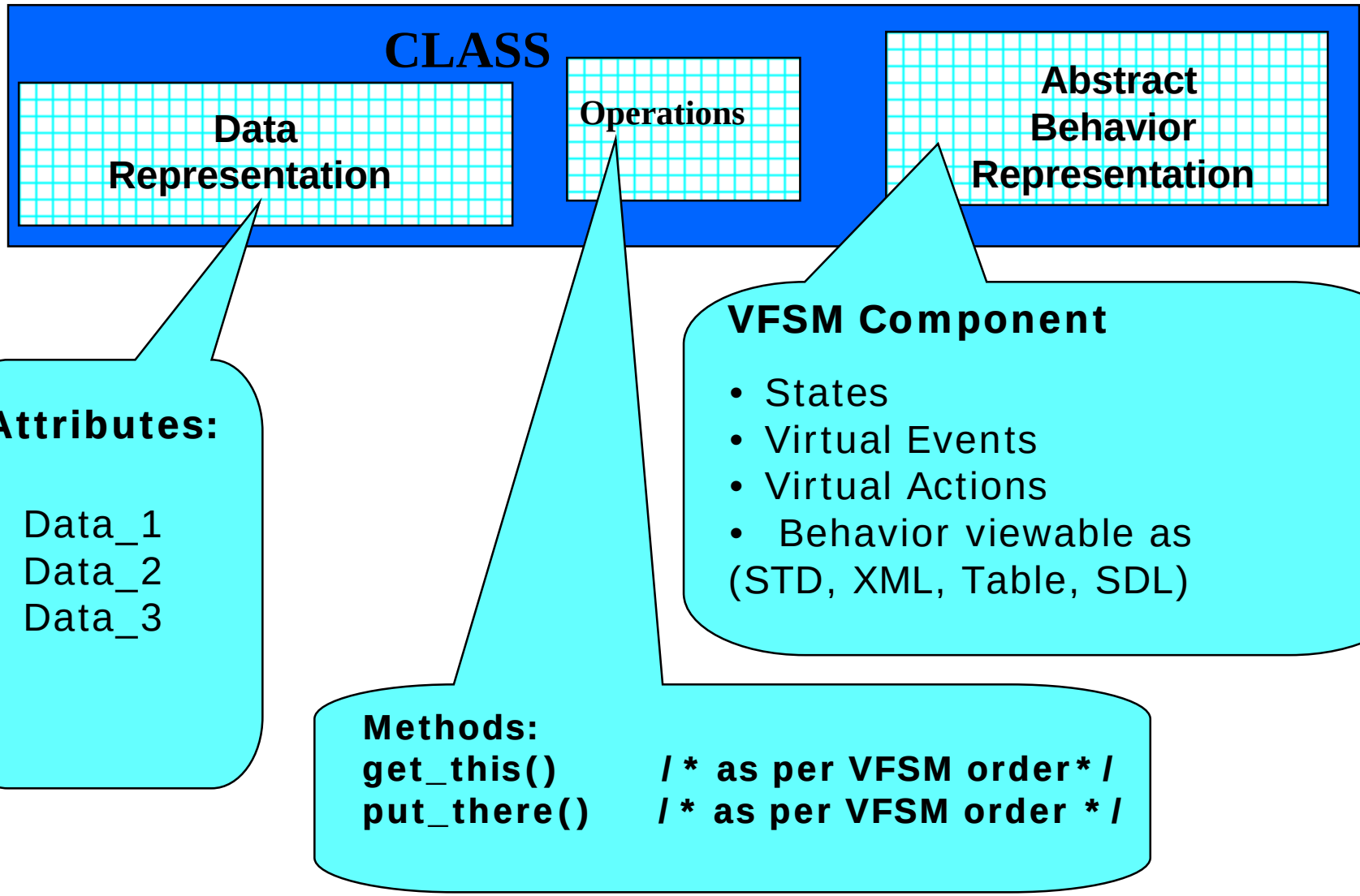
Separation of Class Data
from a Class Behavior:

A viable and proven approach

Components of a Class Model in UML



Components of every PIM Class in VeUML



VeUML – Value Added Proposition

VeUML = MDA (UML) + VFSM

VFSM *Domain Independent Behavior Modeling and Validation*

VFSM (***Virtual Finite State Machine***)

a key element of a ***System Complexity Problem Solution***

VeUML Key Approach to Conquer Behavior Complexity: *Separation of Domains*

Separation of:

- Business Logic from Platform
- Class Data *from* Class Behavior
- Behavior *from* Concurrency -
  (Generalized Objects Factory Pattern)
- Events *from* Data
- Actions *from* Data
- Class behavior *from* the PSM Programming Language *(e.g., C++, Java)*

Key Elements of VeUML behavior notation:

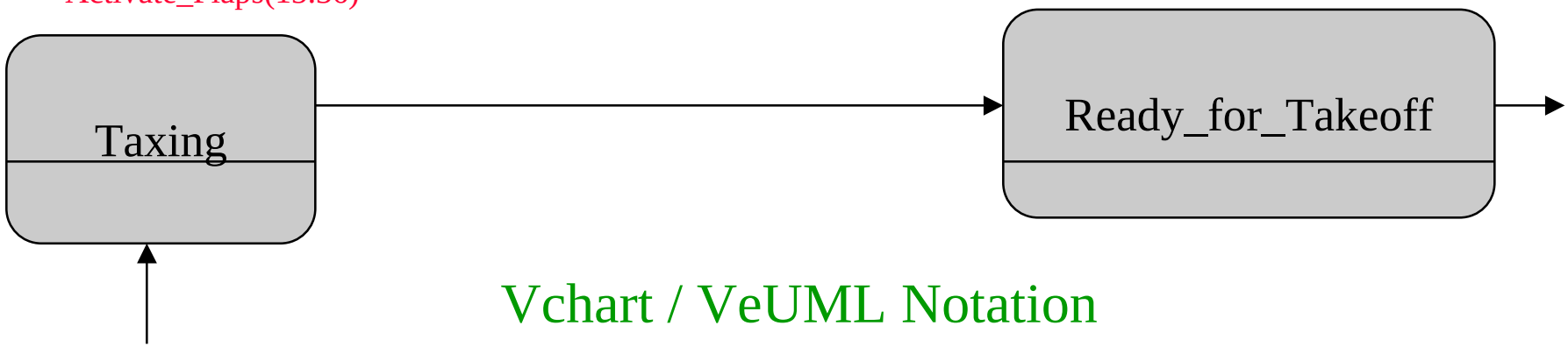
Virtual Environment:

- Virtual Events *(independent from data)*
- Virtual Actions *(independent from data)*
- States (*contain no data*)

Example of behavior modeling notations

Statechart / UML Notation

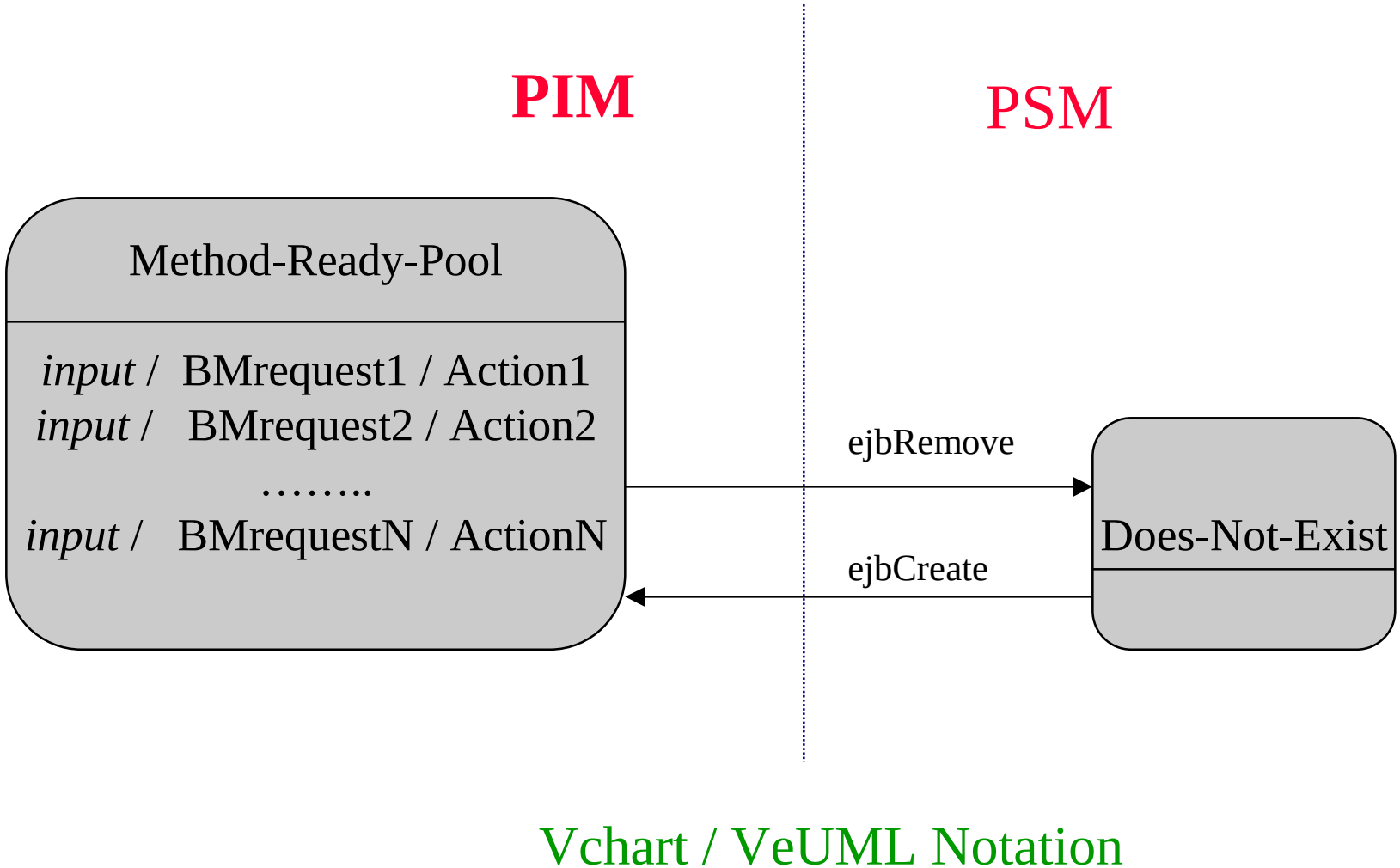
Start_request ()[((RPM > 3000) & (RPM < 5000)) & ((Temp > 90) & (Temp < 120)) & NOT_Alarm] / Activate_Flaps(15.36)



Vchart / VeUML Notation

Start_request & RPM_inRange & Temp_inRange & NOT_Alarm / Activate_Flaps_low

EJB Stateless Session Bean



Vchart / VeUML Notation

VeUML Approach to Conquer Behavior Complexity

- **VFSM Behavior Model for Every PIM Class**
  Advantage: simplifies the behavior modeling

- **Automatic Executable Generation for behavior model**
  Advantage: increased reliability, performance and degree of automatically generated system

- **Eliminating shortcomings of Round Trip Engineering for the behavior models**
  Advantage: Executable always in synch with the models

VeUML Approach to Conquer Behavior Complexity

- **Automatic Model Validation**

  Advantage: Increased reliability and early defect detection

- **Reuse and Portability** of Behavior Models

  Advantage: True productivity gains

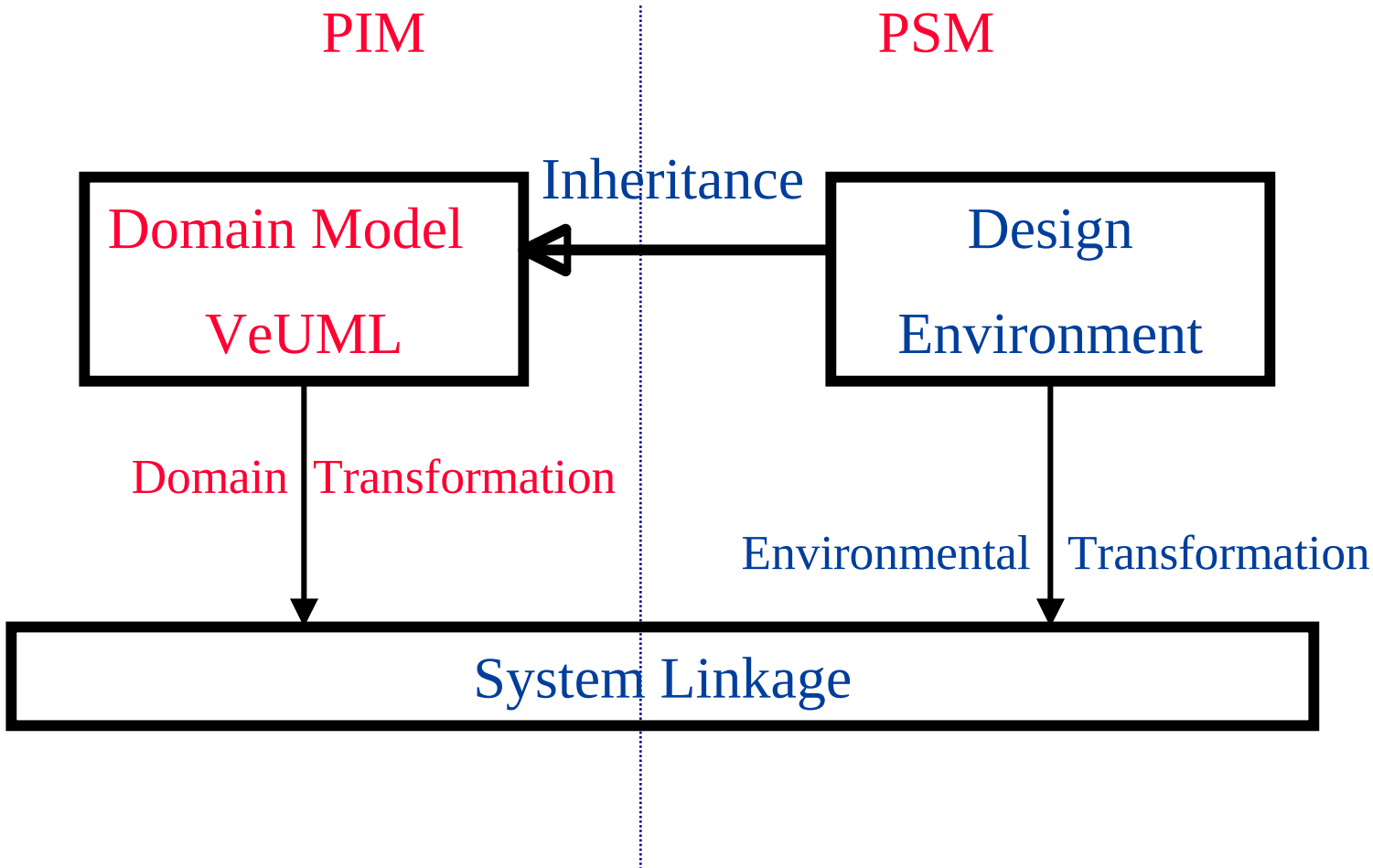
- **Behavior Models Tied to Formal Models**

  Advantage: Reliable and deterministic executable generation and validation

VeUML remedy for Software Development Pain

Problem	**VeUML Solution**
Complexity	*Separation of concerns* – data from behavior; events from data, behavior from concurrency, actions from data etc..
Cost	*Automatic executable generation*, automatic validation and simplified test. Resulted savings 45% *(AT&T/Lucent Data)*
Portability to different platforms	*Service/Business Logic represents Interlingua* - model and automatically generated executable are platform independent
Quality	*Late defects reduction by* 40% by exhaustive validation and early detection *(AT&T/Lucent Data)*
Performance	*Optimized Predictable* performance of VFSM executor

Fundamental VeUML System Decomposition Paradigm: PSM ***Inherits*** from PIM



VeUML extensions: VeSTD, VeXML, VeMSC, VeSDL:

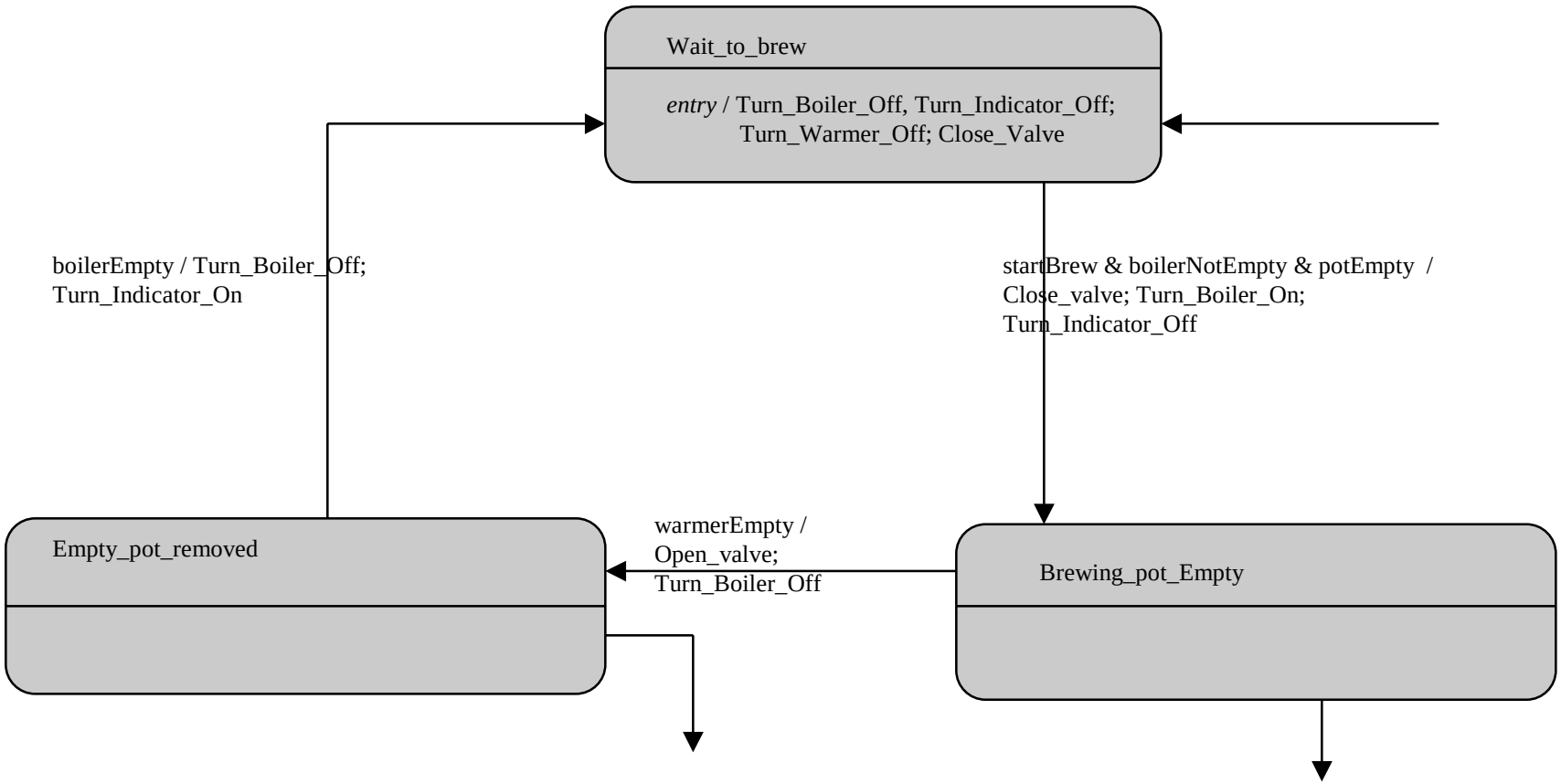
* **Immediate automatic representation conversion always possible:**

  VeSTD <-> VeMSC <-> VeSDL <-> VeXML

*** Consistency in OOA/OOD view:**

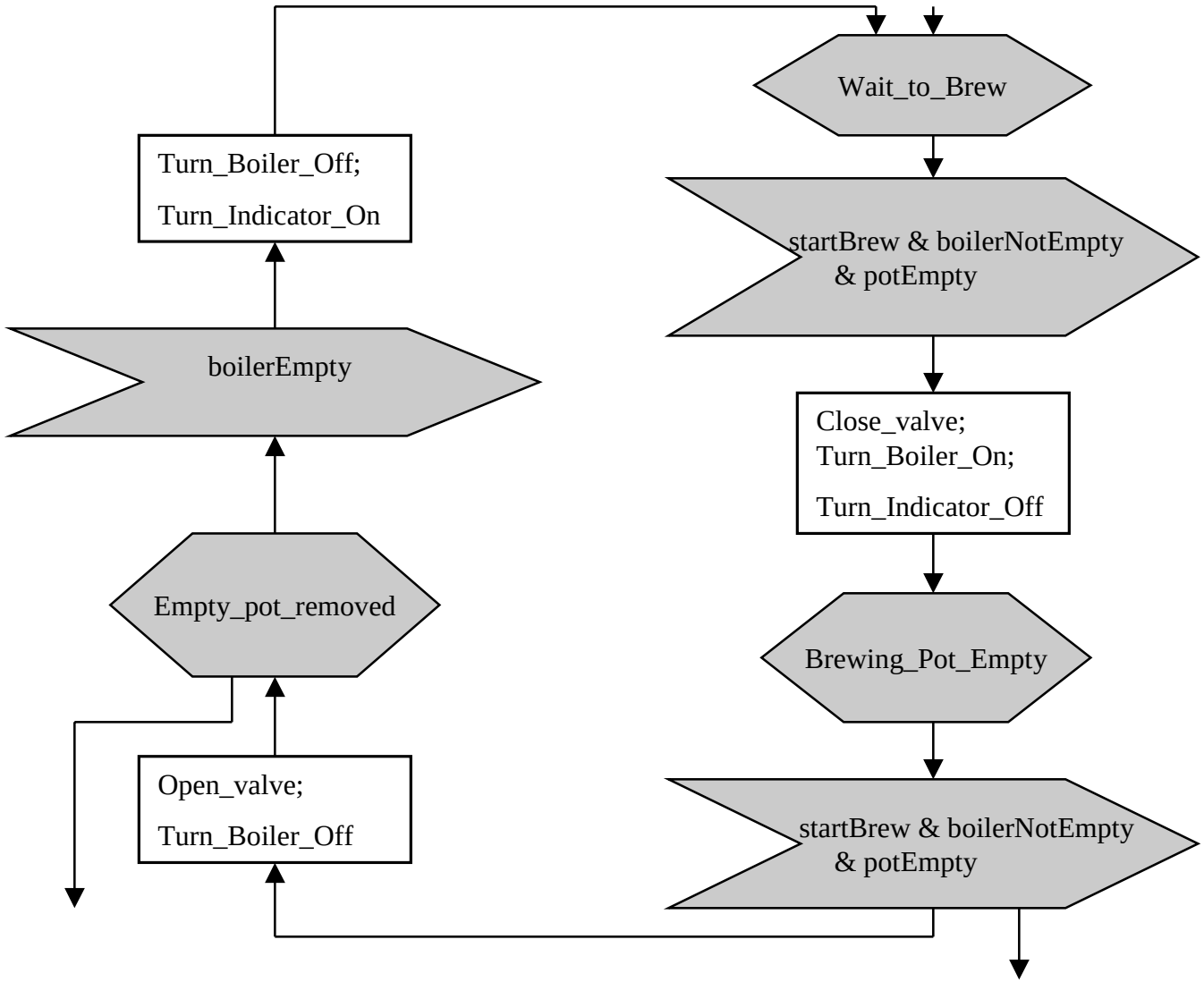
According to the OOA/OOD principle Architect/Designer/Implementer view of a system should be the same. In VeUML the MDA behavioral model is the only target executable representation Architect/Designer/Implementer work with.

VeSTD Example – Partial diagram



This Coffeemaker example is based in part on the book by
Robert C. Martin, "Object-Oriented C++ Applications Using The Booch Method", Prentice Hall, 1995.

VeSDL Example – Partial diagram

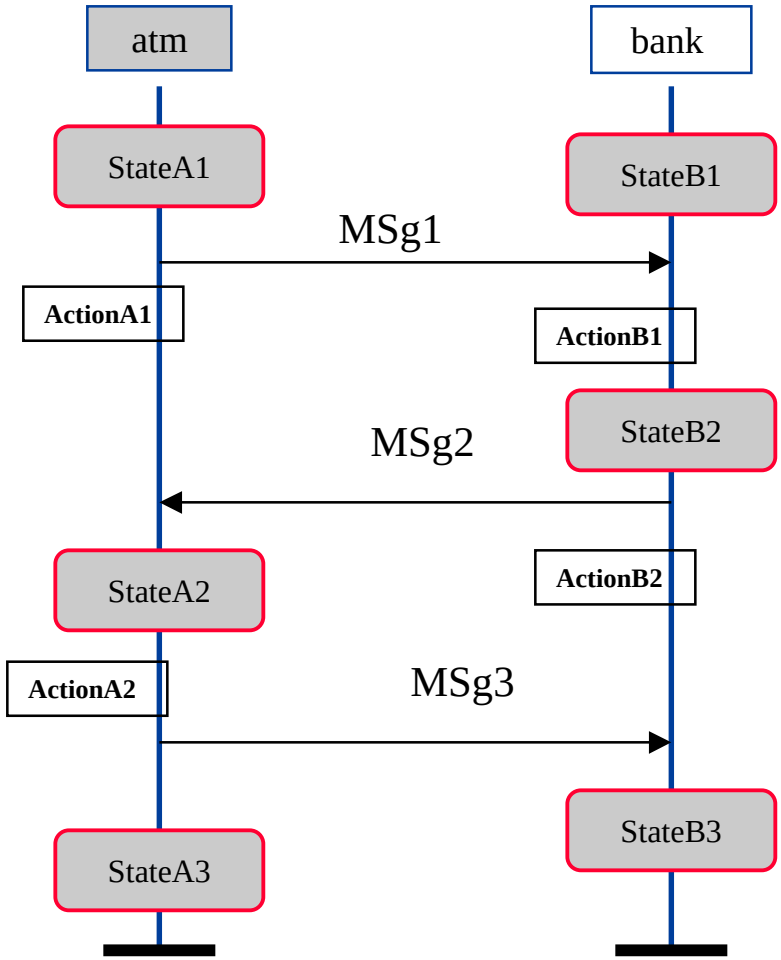


VeXML Example – Partial listing

```
<?xml version="2.0"?>
<!-- (c) 2002 StateSoft, Inc.  All rights reserved. -->
<XML>
  <STATE>
    <TRANSITION>
      <CONDITION>  "potNotEmpty / "
        <ACTION>
          "Close_valve;  Turn_Boiler_On;  Turn_warmer_On"
        </ACTION>
          <NEXTSTATE>
            "Empty_Pot_removed"
          </NEXTSTATE>

      </CONDITION>
    <TRANSITION>
    </STATE>
</XML>
```

VeMSC Classes Behavior Generation from System Model MSC



In VeUML one can automatically synthesize target executable model from elementary representations.

Other approaches often use elementary representation for high level simulation only.

VeUML extension concepts: VeSTD, VeXML, VeMSC, VeSDL. What do they represent as a Modeling Language Category?

It may be:

T E M L – (*new term)*
Target Executable Modeling Language

Interlingua – (IBM® term) where source model and executable are programming language independent.

Conclusions: Benefits of Separation of Domains in VeUML Behavior Modeling

- **Automatic executable generation and validation for systems of substantially higher complexity**

- **Significant degree of reuse due to separation of domains**