On a Specific Model-Based Architecture Description Language: An Approach Based on Standards

Sébastien Gérard, Bran Selic[†]

CEA-LIST

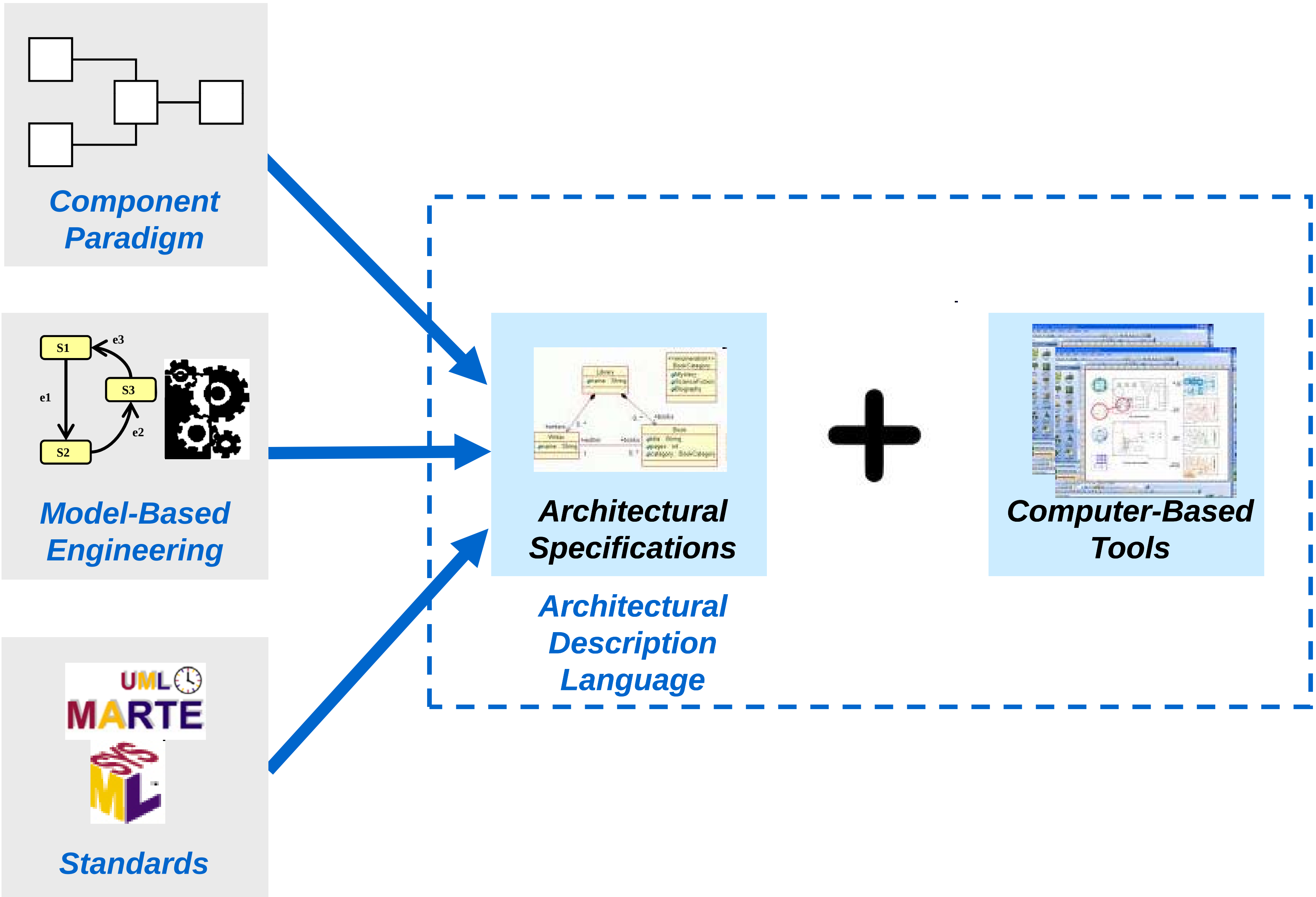
Laboratory of model driven engineering for embedded systems

[†] (Malina Software Corp and Zeligsoft Limited)

Sebastien.Gerard@cea.fr
selic@acm.org

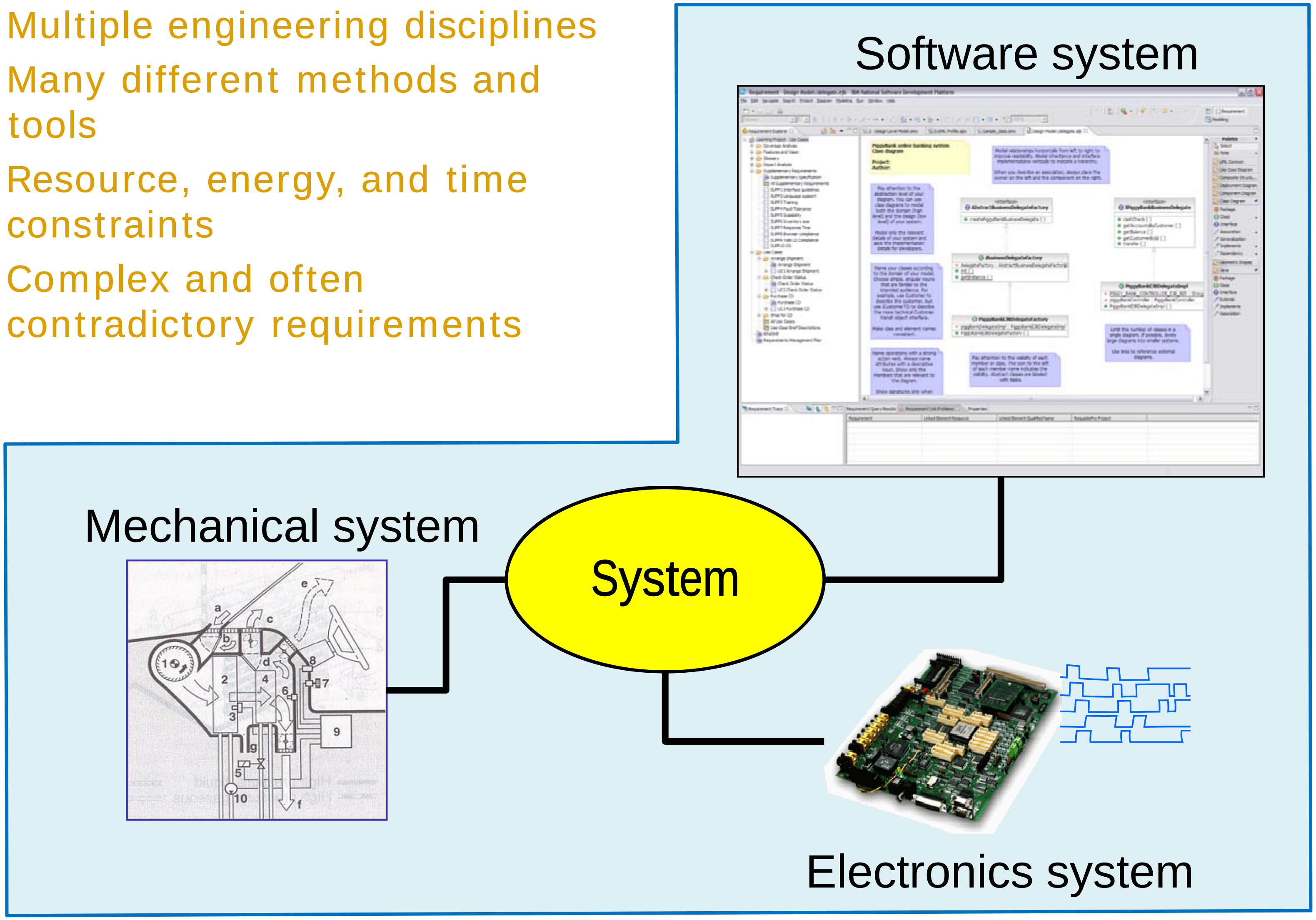
Thesis: A Standards-Based ADL Approach

- **The CEA-LIST approach to complex system of systems design**



The Problem: Real-Time System of Systems Developmen

- **Complex heterogeneous systems responding to real-world events**
  - Multiple engineering disciplines
  - Many different methods and tools
  - Resource, energy, and time constraints
  - Complex and often contradictory requirements



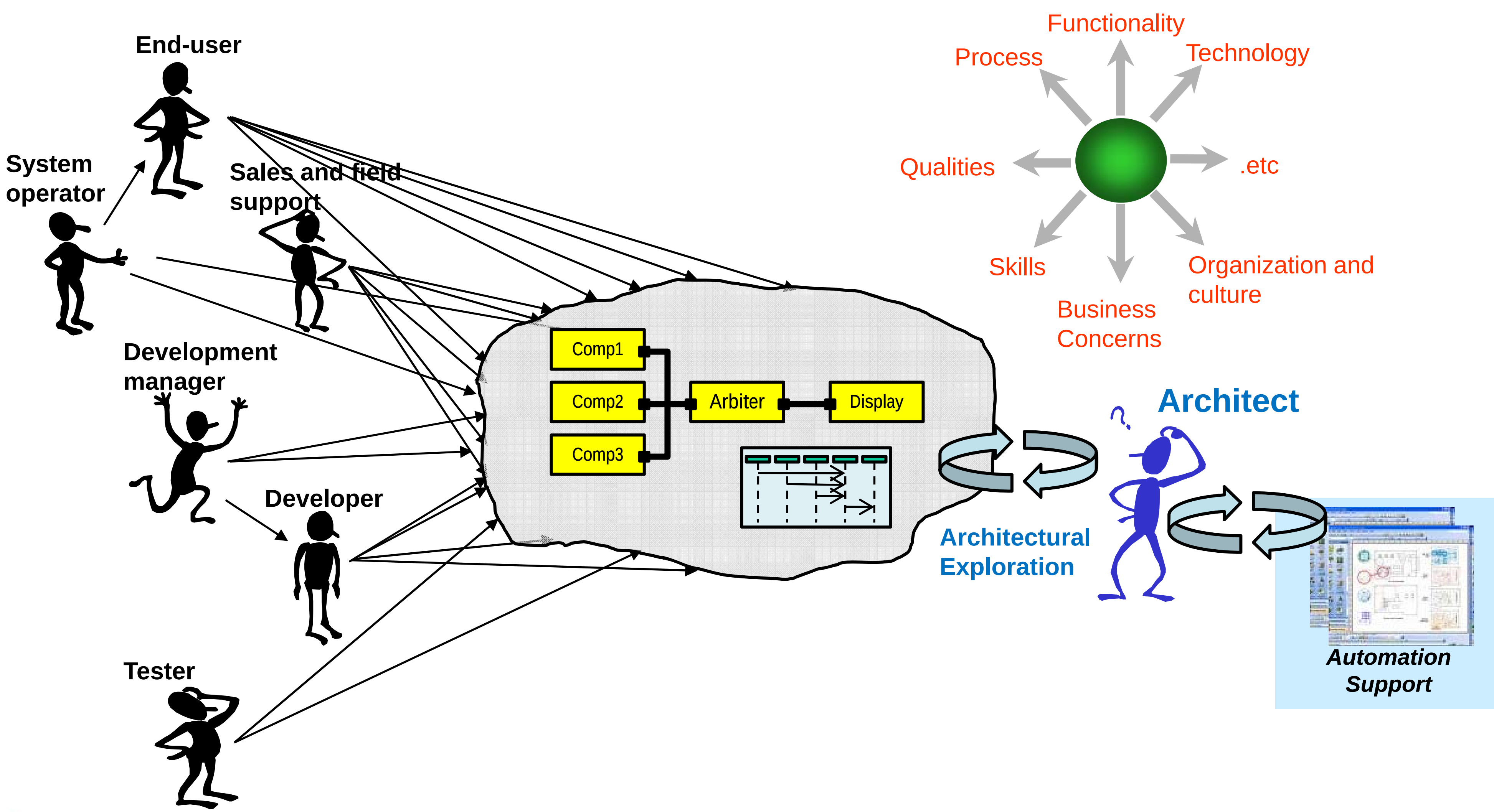
Requirement: System-Level Approach

- **Design the system as a whole rather than as an aggregate of separately designed sub-systems**
  - Ensures system integrity
  - Requires a "big picture" approach; i.e., an ***architecture***
- **ARCHITECTURE [IEEE Standard 1471] :**

  "The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution"
- **How architecture helps:**
  - Facilitates communications between stakeholders and resolution of conflicting requirements
  - Facilitates prediction of key system qualities (e.g., performance, robustness)
  - Identifies technological domains
  - Provides basis for work partitioning
  - Guides finer-grained decision making and implementation (architecture as a pervasive artifact)
  - Guides evolutionary development

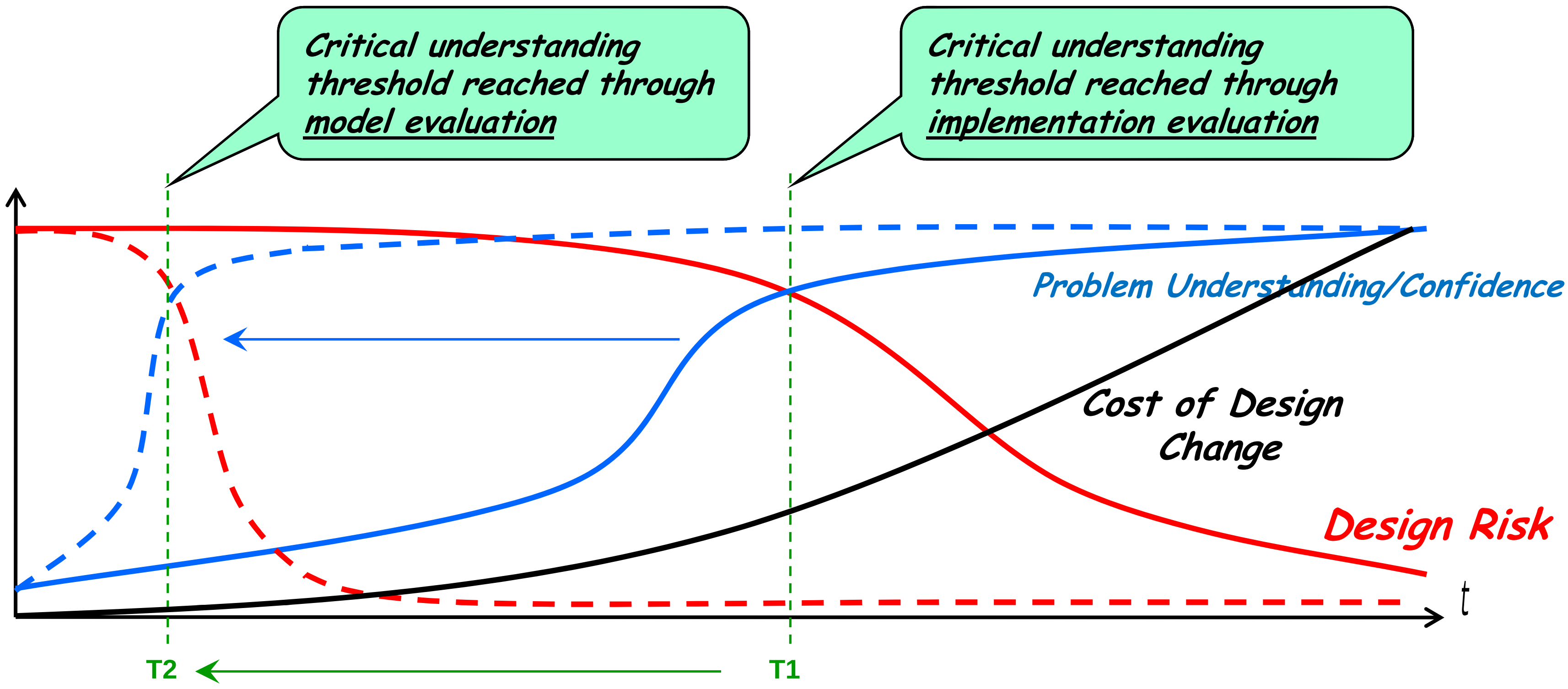
Requirement: Supporting an Iterative Process

- **Complex systems cannot be adequately in one pass**
  - Too many unknowns, too many conflicting design choices
  - Requires *architectural exploration* to gain understanding and intuition



How Architectural Exploration Reduces Risk

- **Repeated evaluation of architectural <u>models</u> (using simulation, formal and informal analyses)**
  - Early experience with the design
  - Early detection of potential design flaws ⇒ less expensive to fix

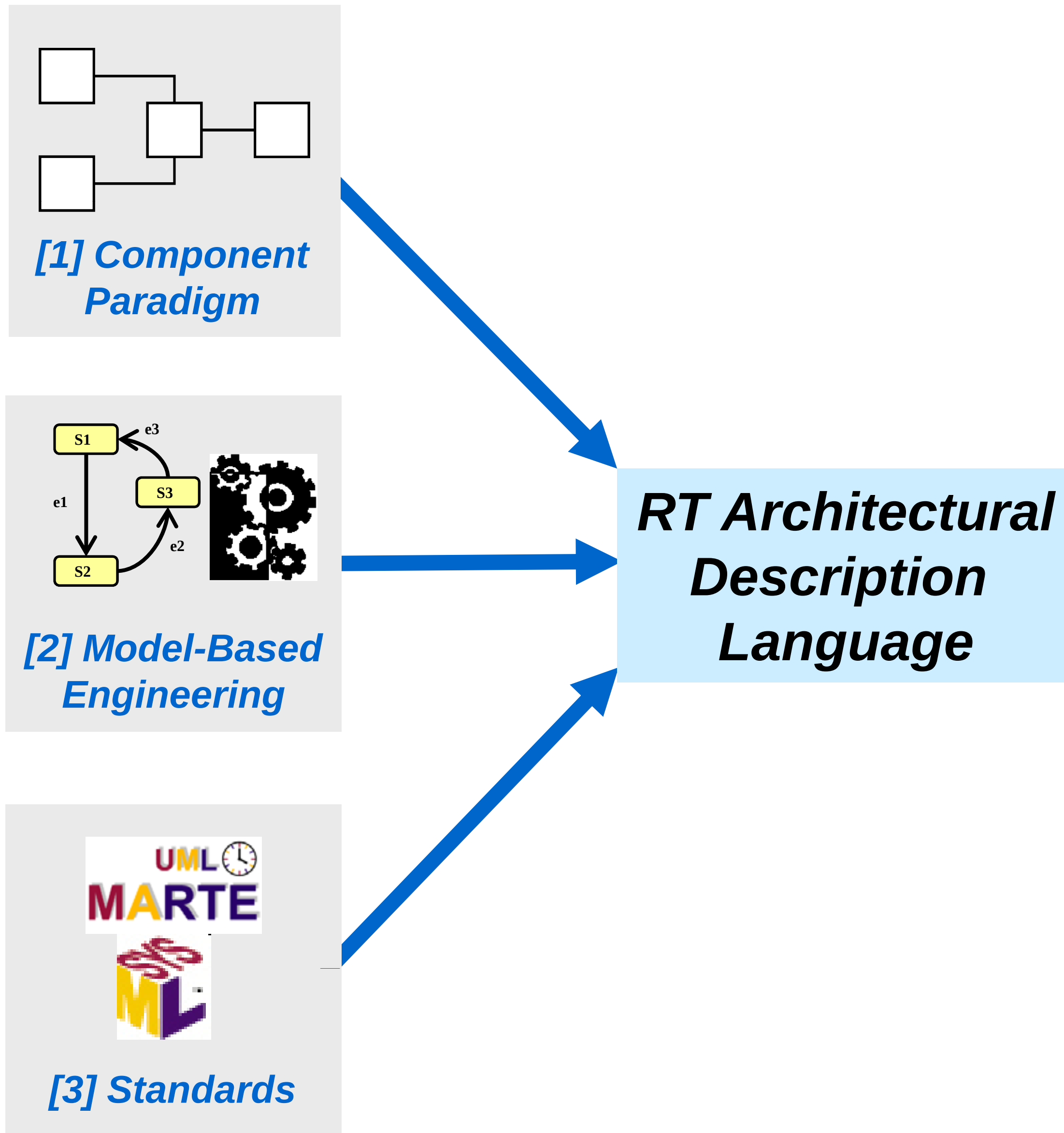


- **ARCHITECTURE [IEEE 1471]: "The <u>fundamental</u> organization…"**
  ⇒ architectural specifications abstract out non-fundamental detail
  ⇒ i.e., they are <u>models</u>
  - "To architect is to model"

- **Characteristics of useful engineering models**
  - Purposeful (i.e., intended for specific purposes/viewpoints/domains/audiences)
  - Abstract (i.e., they leave out inessential detail)
  - ***<u>Understandable</u>*** (easy to comprehend for intended audience)
  - ***<u>Accurate</u>*** (i.e., faithfully represent elements of interest)
  - Predictive (i.e., can be used to predict key system characteristics)
  - Significantly easier and cheaper to construct than the system they represent

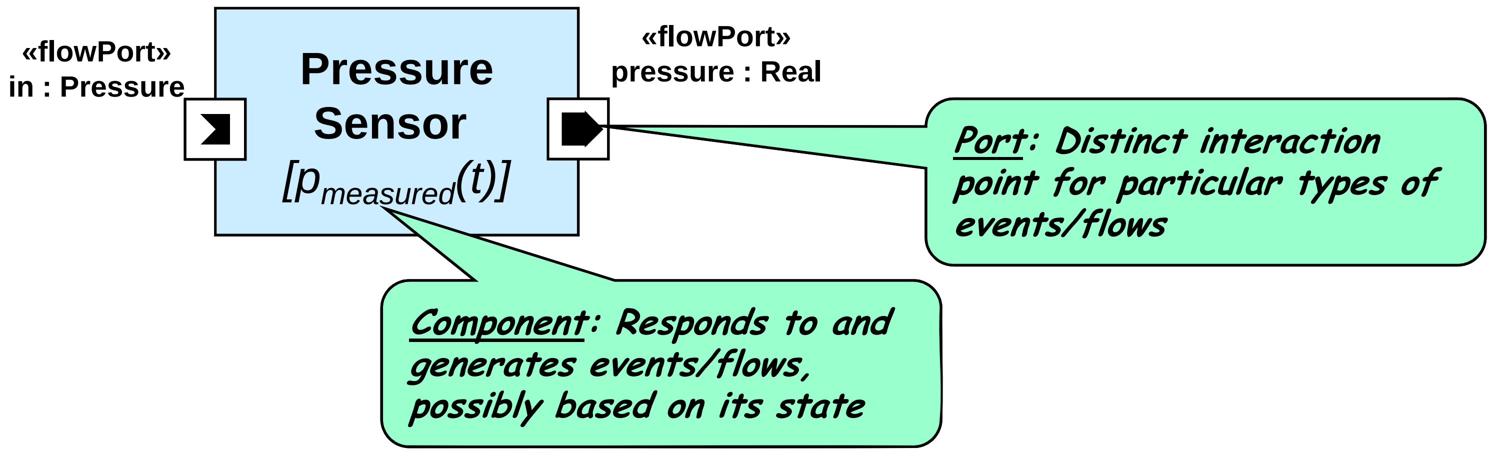
- **Accuracy and understandability, in particular, impose important requirements on modeling languages for describing architectures (*<u>architectural description languages</u>* (*<u>ADLs</u>*))**
  - What should an ADL for real-time systems of systems look like?

Key Ingredients for a Successful Real-Time ADL

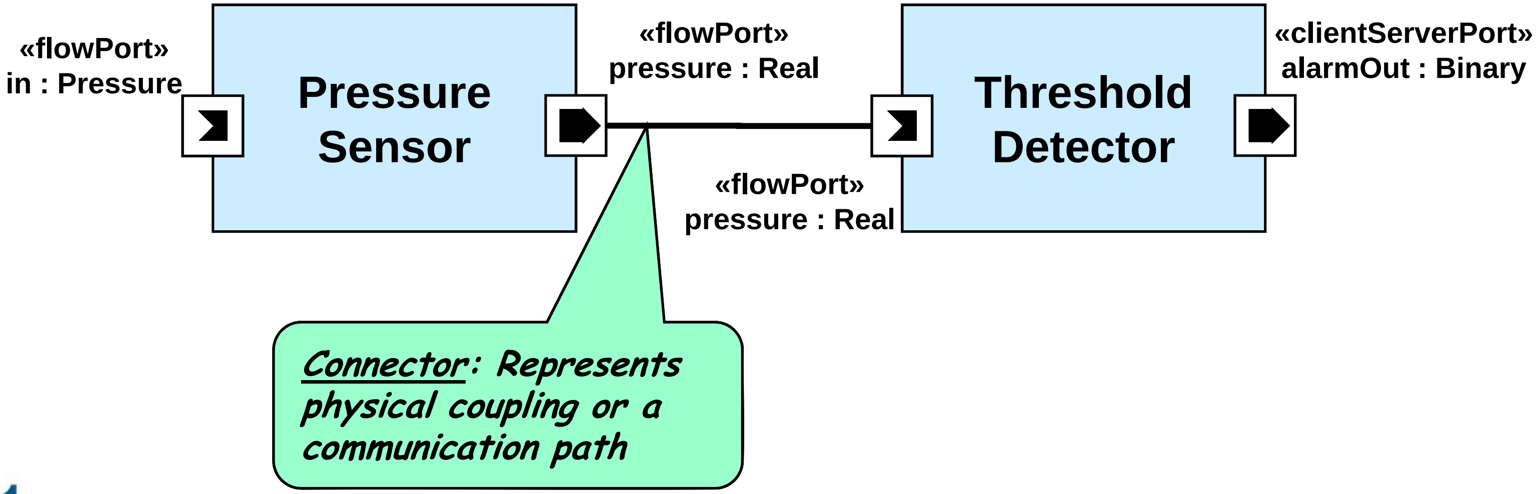


[1] The Component Paradigm

- **Specifically: Object-oriented components**
  - Motivation: Direct representation of both physical and logical elements, which generally have identity and may have state

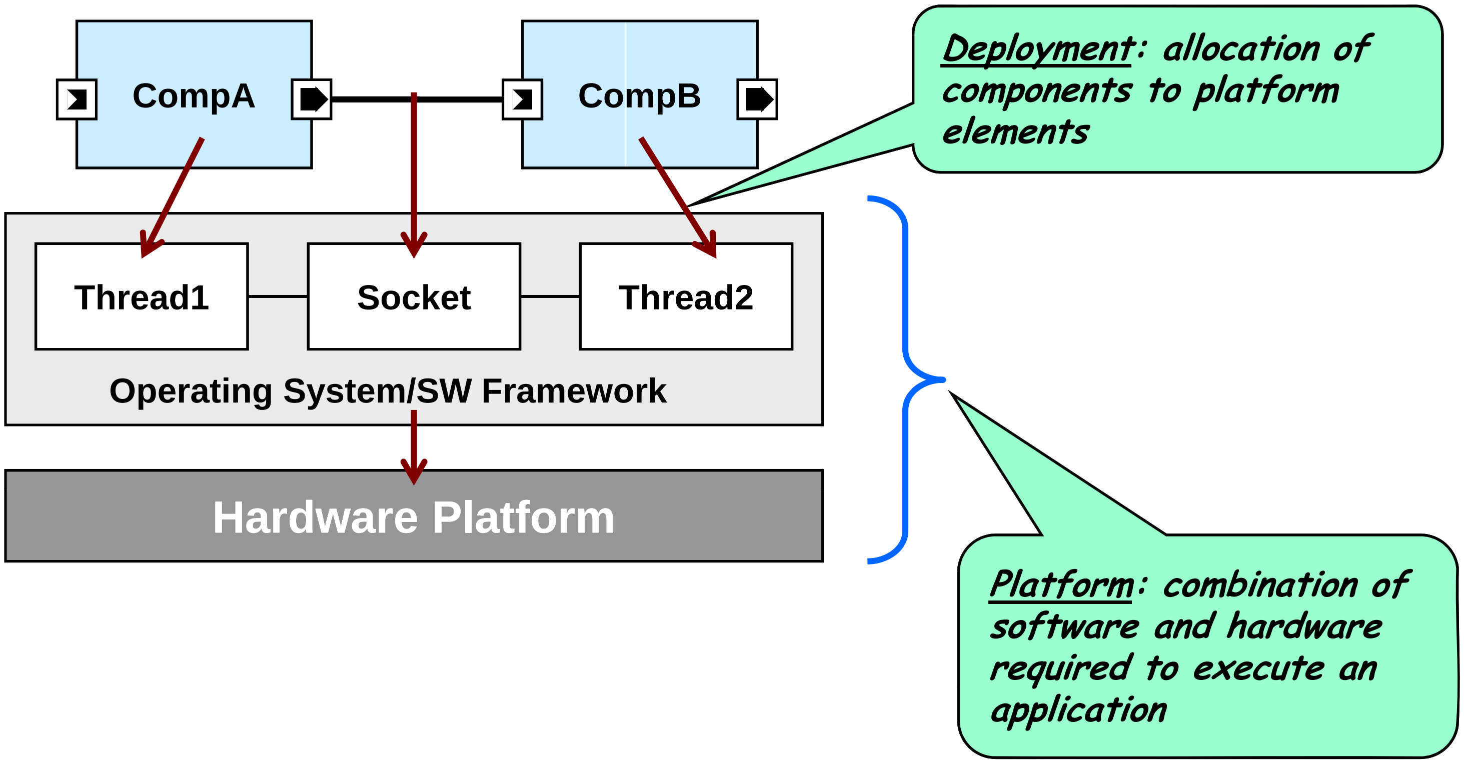


- **Component assemblies (configurations)**



Extending the Component Paradigm for Real-Tim

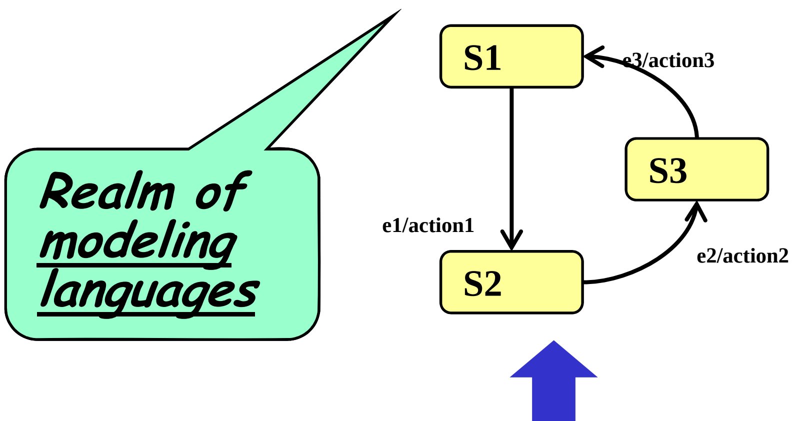
- **The functionality and quality of software components are a function of**
  - the application program logic,
  - the properties of the underlying platform stack (e.g., performance, reliability) and,
  - the deployment of the components to elements of the platform
- **Architects should care about the quality of service of our systems ⇒ these factors must be accounted for during design**



[2] Model-Based Engineering (MBE)

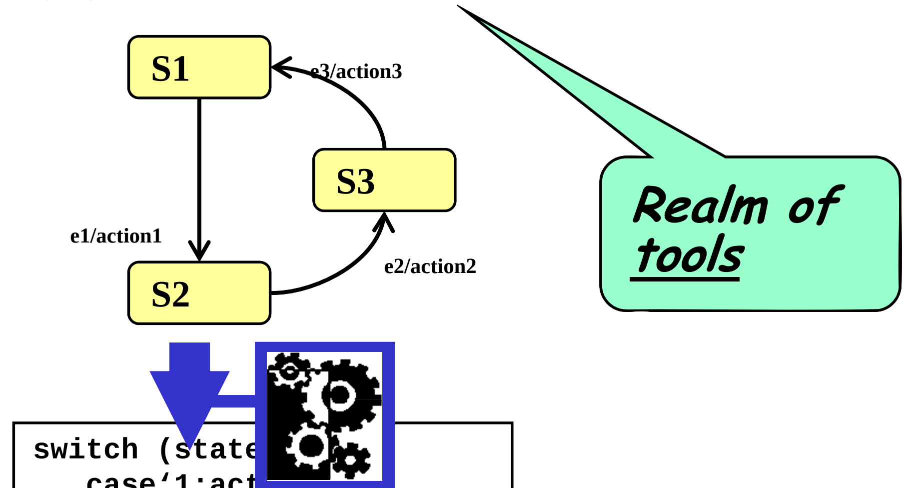
- An approach to system and software development in which software models play an indispensable role
- Based on two time-proven ideas:

(1) ABSTRACTION




```
switch (state) {
   case'1:action1;
          newState('2');
          break;
   case'2:action2;
          newState('3');
          break;
   case'3:action3;
          newState('1');
          break;}
```

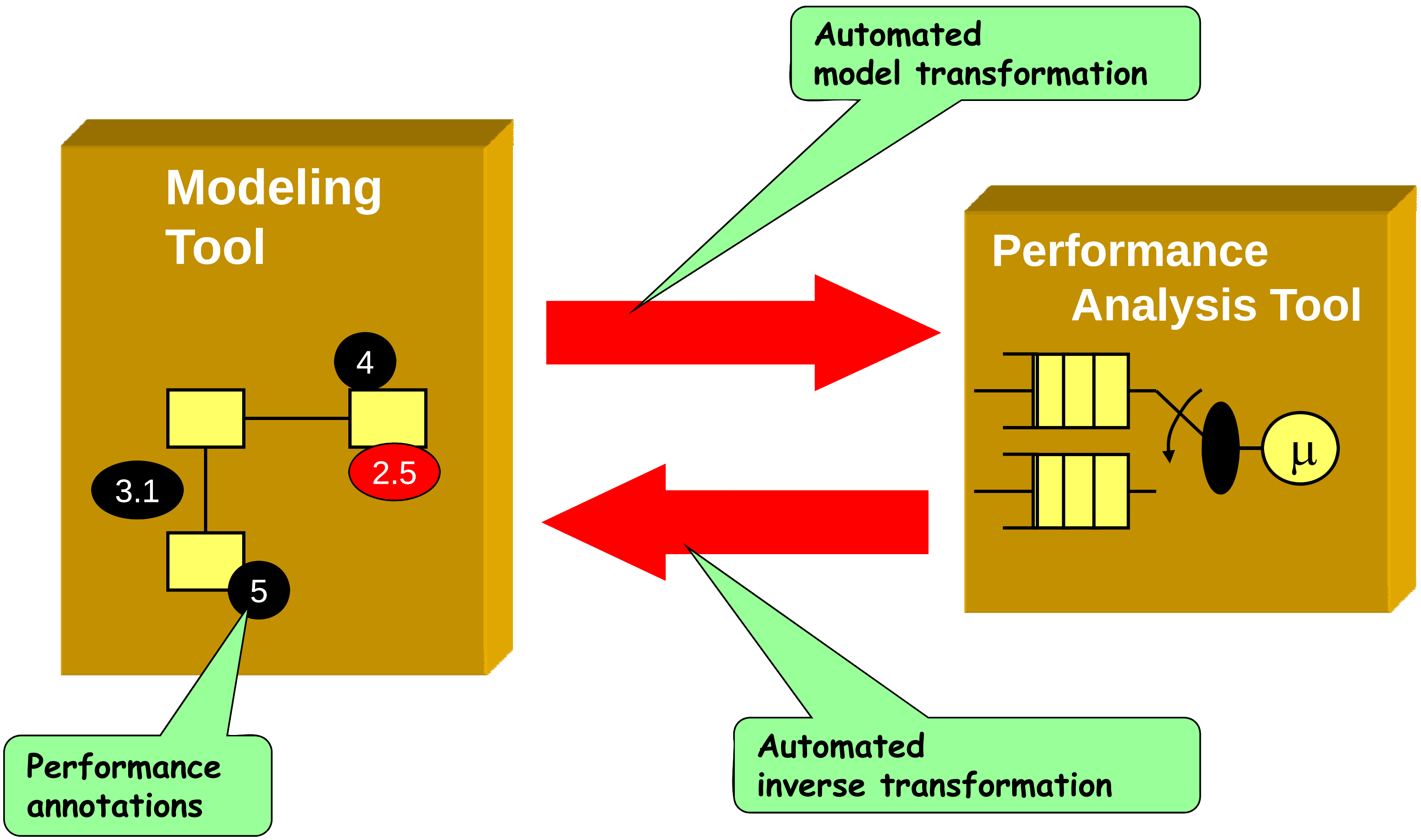
(2) AUTOMATION




```
switch (state
   case'1:act
          newState('2');
          break;
   case'2:action2;
          newState('3');
          break;
   case'3:action3;
          newState('1');
          break;}
```

Automation in MBE

- **Automated analyses, transformations, code generation**
  - Example: performance analysis of UML models using queueing theory



[3] Standards

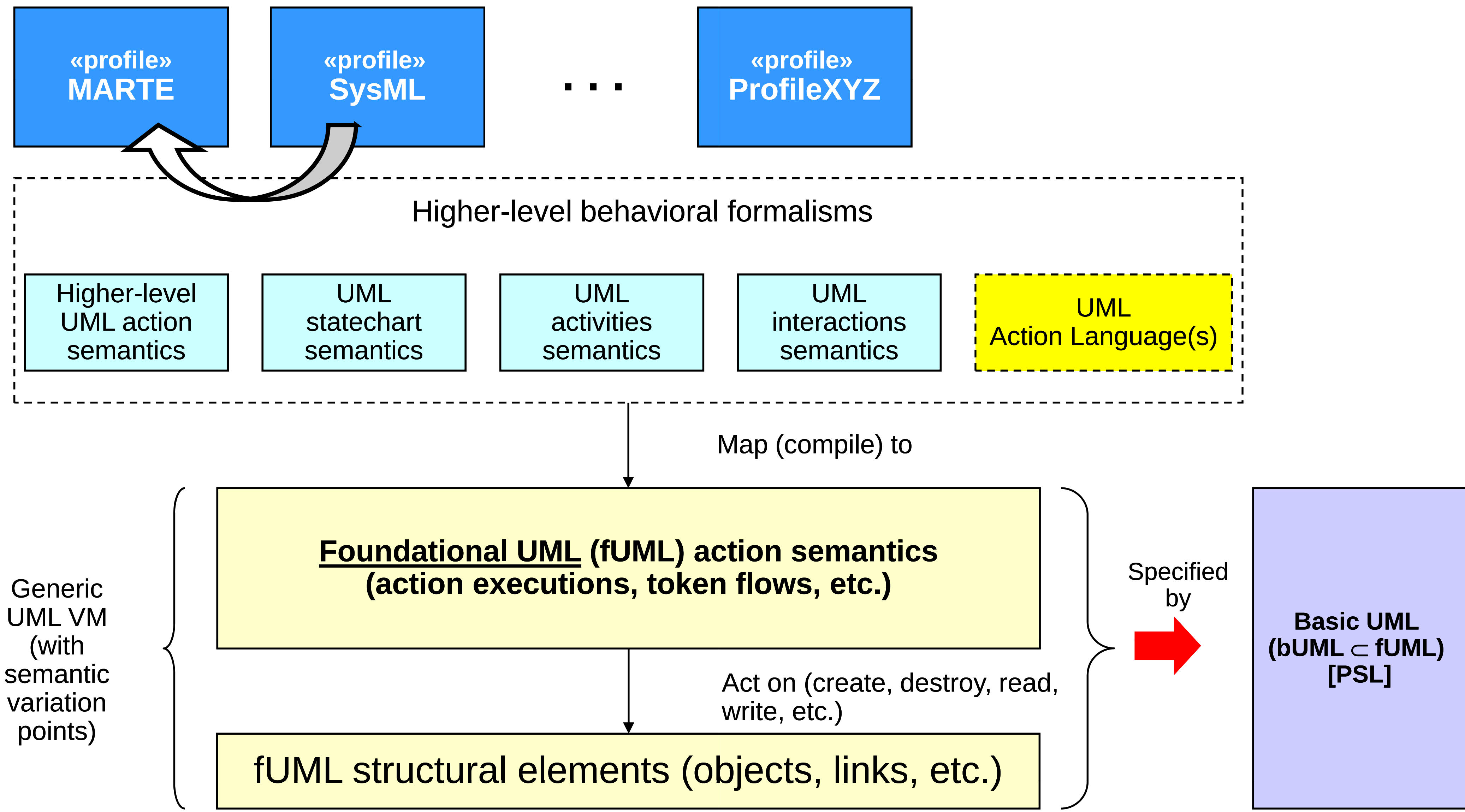
- **Standards have traditionally provided the most dramatic boosts to technological progress**
- **Standards are useful because:**
  - They support specialization
  - Standards are interfaces between different specialization domains
  - Specialization is beneficial since it allows complex topics to receive due attention (no need to worry too much about other specialties)
  - E.g., a vendor specializing in analyzing UML models need not worry about providing a UML editing tool
- **Standards enable vendor independence**
  - Users have a choice of different vendors (no vendor "tie-in")
  - Forces vendors into competing and improving their products

Which Standards for the ADL?

- **The Object Management Group (OMG) has created the Model-Driven Architecture initiative**
  - A comprehensive set of standards in support of MBE including standard modeling languages:
    - UML 2
    - UML profile for Modeling and Analysis of Real-Time and Embedded Systems (MARTE)
    - UML profile for Systems Engineering (SysML)
- **Why UML 2?**
  - Widely used and taught – familiar to many (understandability)
  - Supported by many proprietary and open-source tools (tool support)
  - Supports domain-specific specializations – which can reuse standard UML 2 tools) (accuracy)
- **Why MARTE?**
  - Supports a component model specialized for real-time and systems (accuracy)
    - Including key domain concepts of platform and allocation
  - Can take advantage of UML 2 tools (tool support)
  - Supports engineering analyses (predictiveness)
- **Why SysML?**
  - Intended for modeling systems of systems (accuracy)
  - Compatible with UML 2 and MARTE and their respective tools (tool support)

UML 2: The Shared Semantic Foundation

- **UML 2 has semantics?**



- **UML Profile**
  - A special kind of package containing stereotypes, modeling rules and model libraries that, in conjunction with the UML metamodel, define a group of domain-specific concepts and relationships
- **Profiles can be used for two different purposes:**
  - To define a domain-specific modeling language (*DSML profile*)
  - To define a domain-specific viewpoint *(annotation profile)*
- **Benefits of profile usage**
  - Correctly defined profiles allow direct and effective reuse of the extensive support structure provided for UML (e.g., Tools, methods, experience, training…)
  - DSMLs based on UML profiles share a common semantic foundation which can greatly reduce the language fragmentation problem.

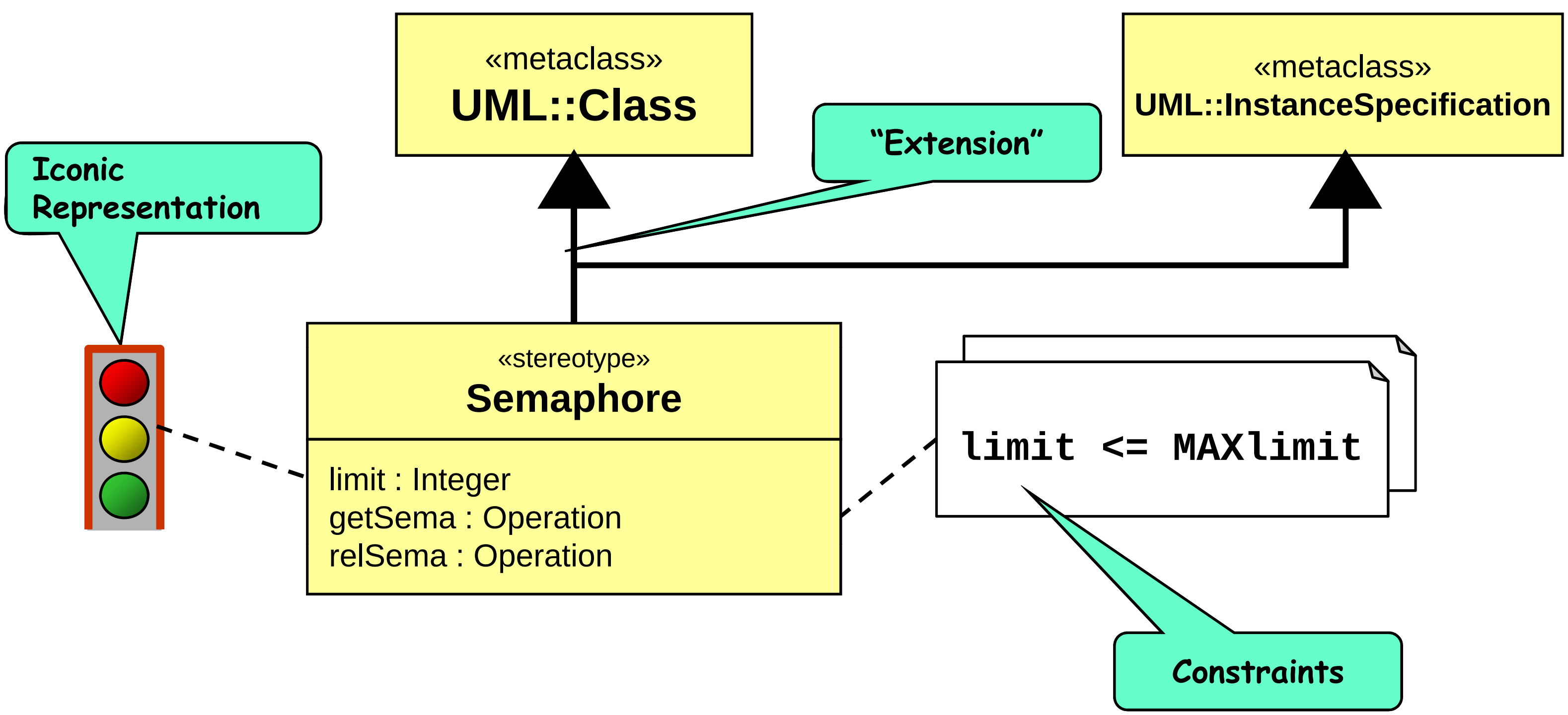
Example: Adding a Semaphore Concept

- **Semaphore semantics:**

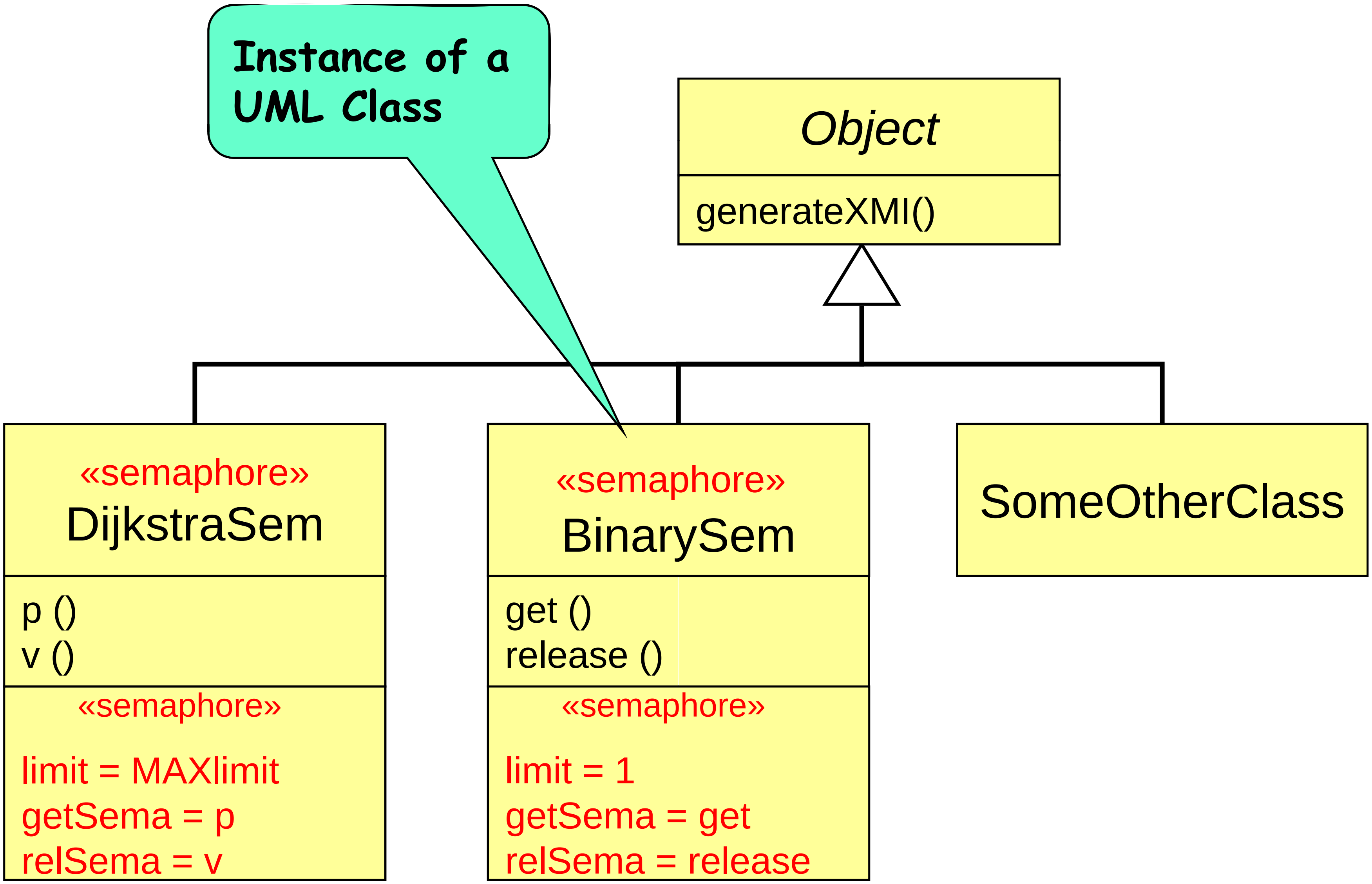
  "A specialized <u>object</u> that limits the number of concurrent accesses in a multithreaded environment. When that limit is reached, subsequent accesses are suspended until one of the accesses threads releases the semaphore, at which point the earliest suspended access is given access."

- **What is required is a special kind of <u>object</u>**
  - ...that has all the general characteristics of UML objects
  - ...but adds refinements

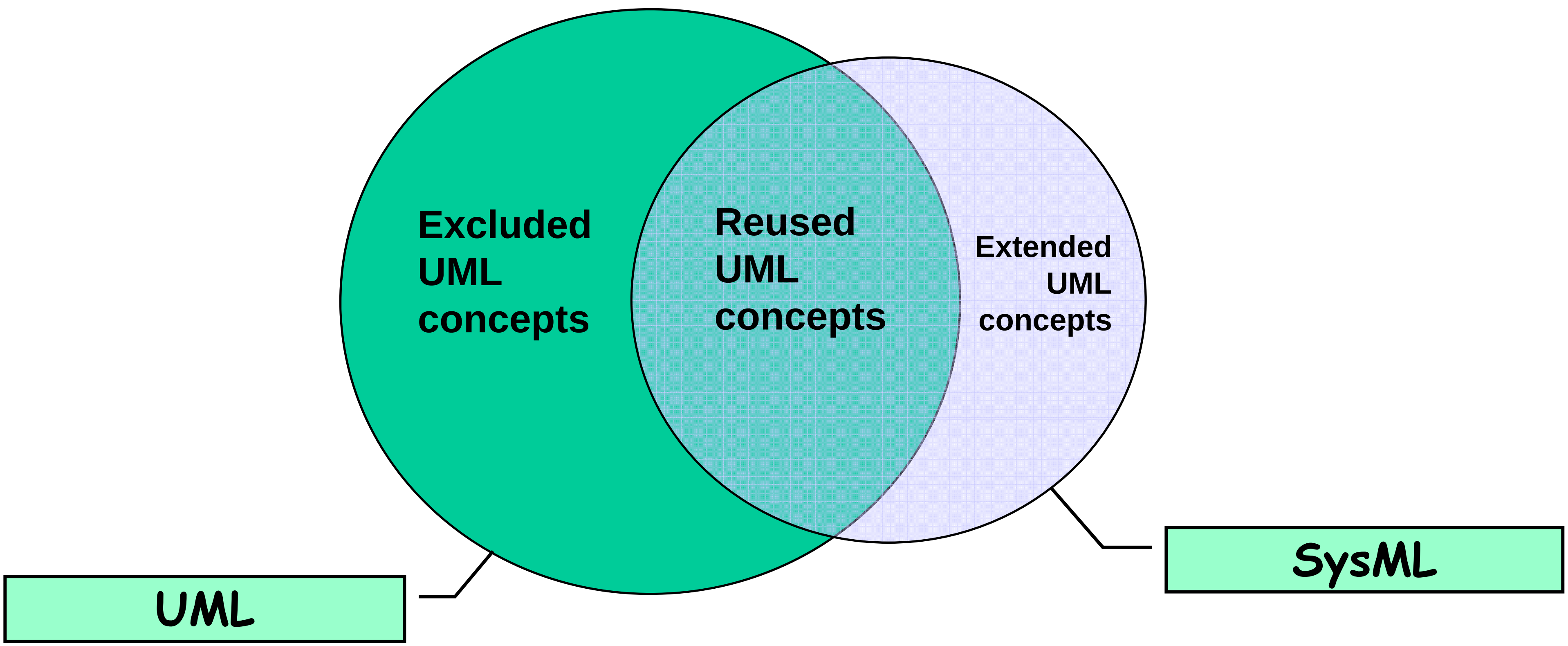
  ⇒ Extend those UML concepts that represent objects: e.g., Class and InstanceSpecification



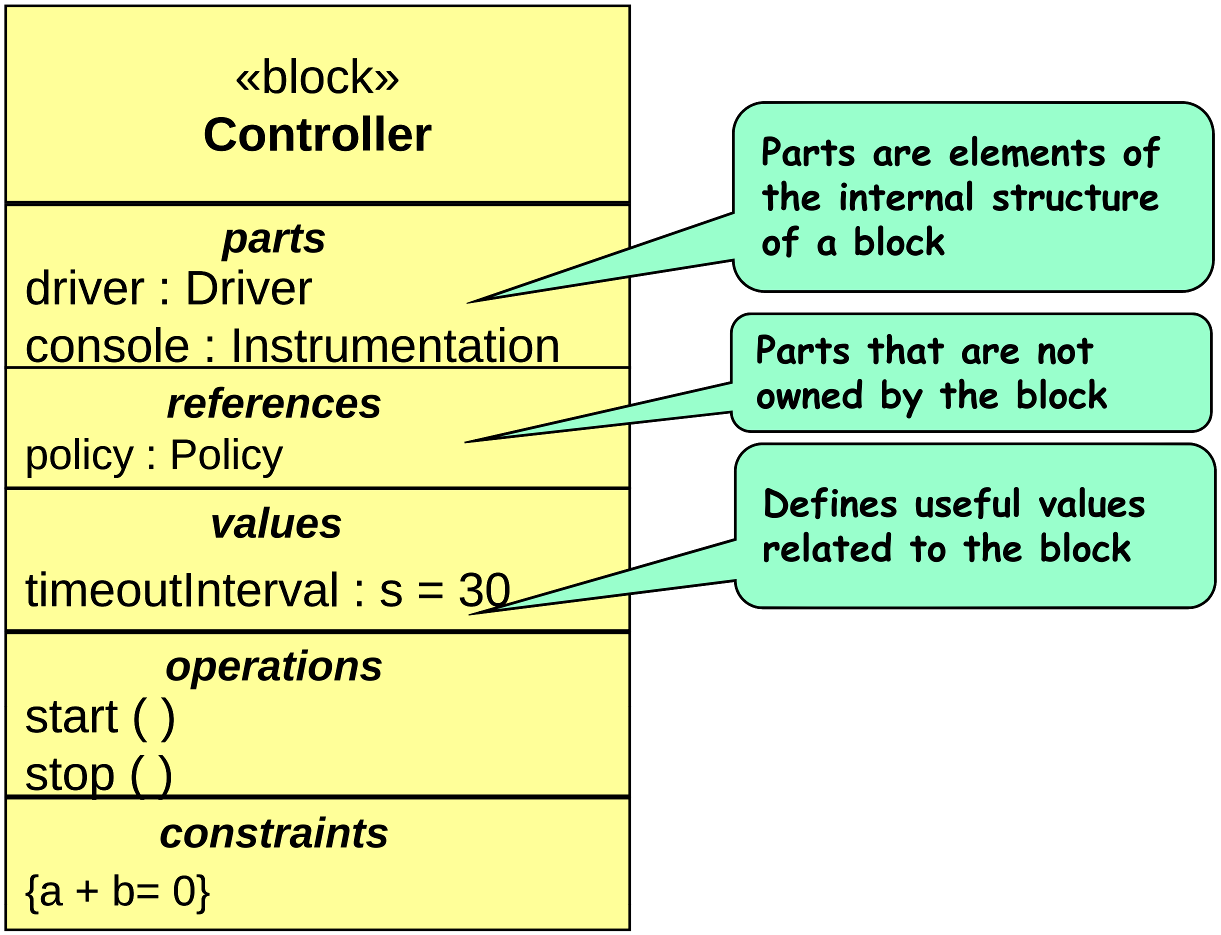
Example: Applying the Stereotype



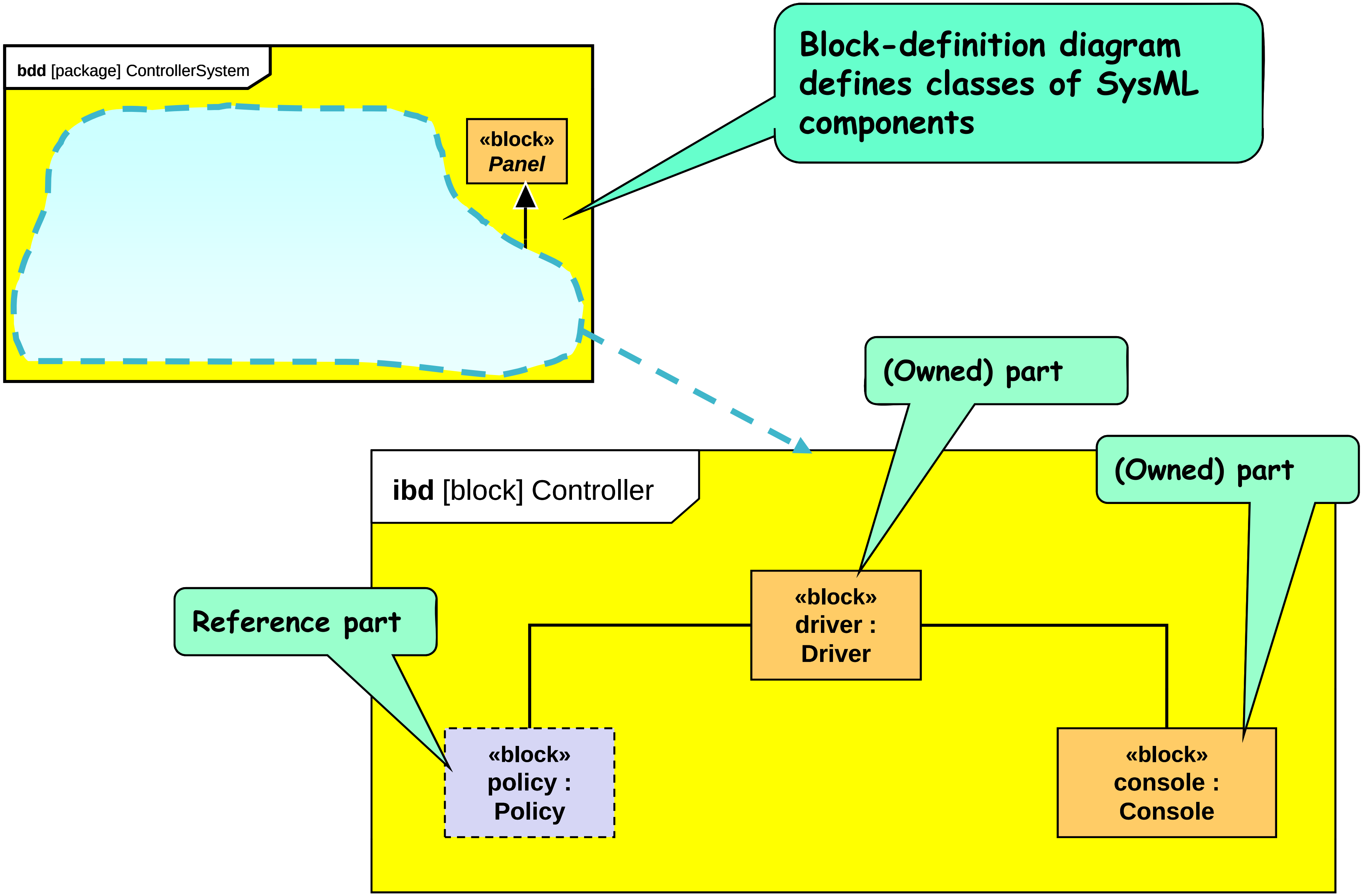
- **Language for systems engineers**
  - For modeling hybrid component-based systems
  - Reuses a subset of UML (state machines, use cases, activities, interactions)
  - Refines structure modeling concepts
    - Class diagrams = block definition diagrams
    - Structure diagrams = internal block diagrams
  - Adds new capabilities (requirements diagrams, parametric diagrams)
  - Refines concept of “flow” to include modeling of continuous quantities



- **Systems engineering specific extension of the UML Class concept**
- **Can represent diverse things including**
  - Hardware elements, software elements, data, physical devices, logical devices, etc.

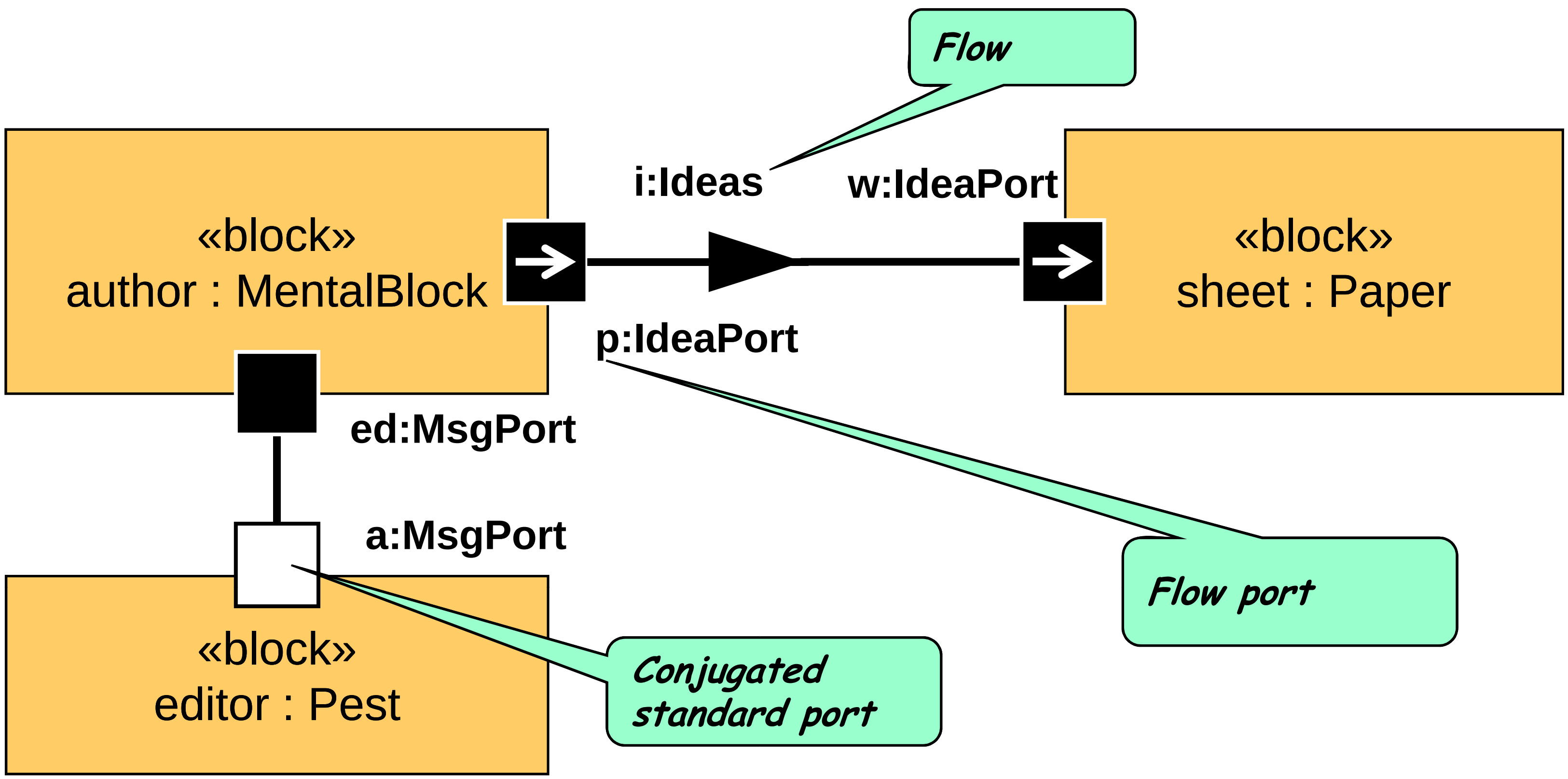


- **Block definitions and block configurations**



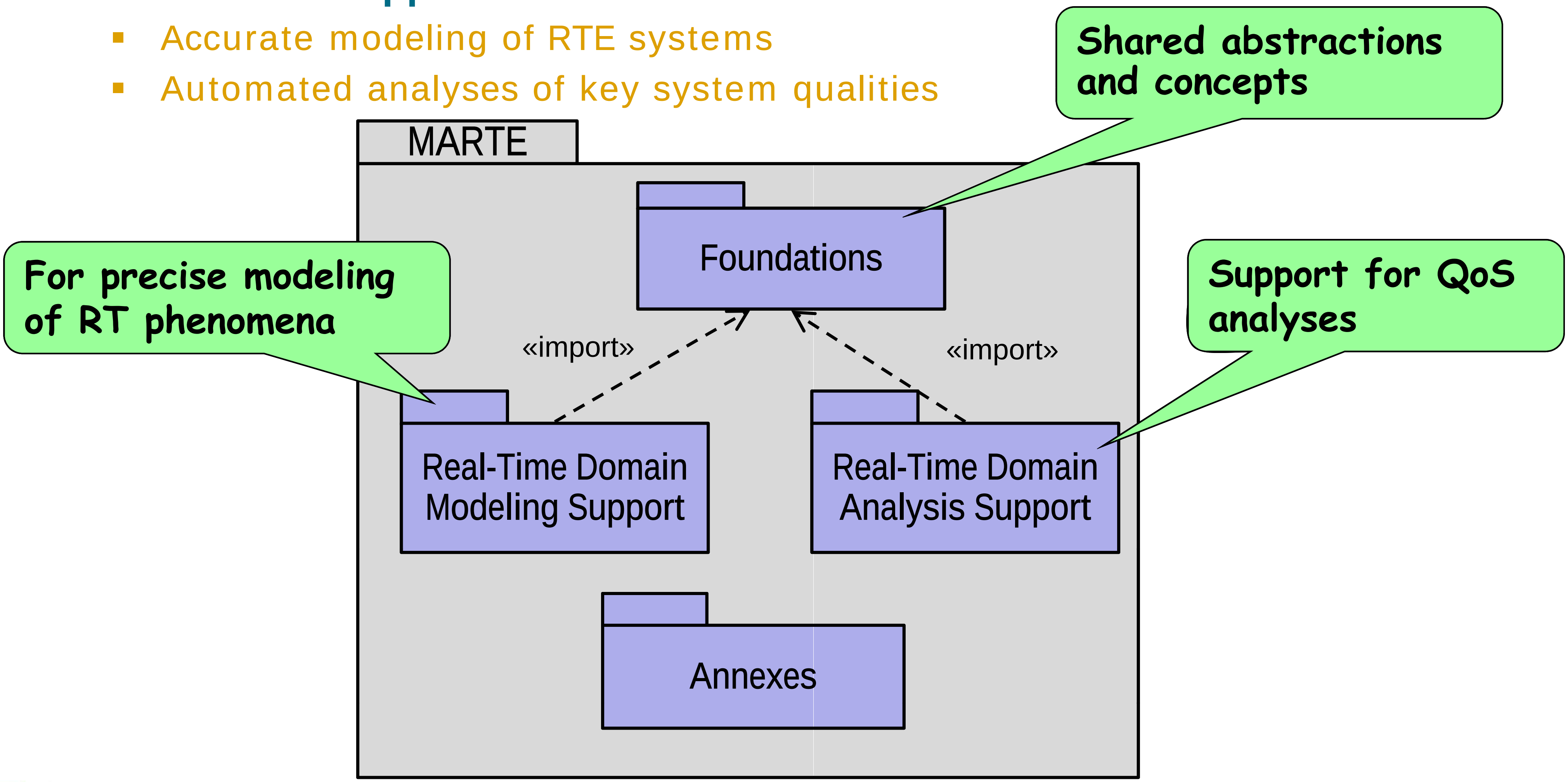
SysML Ports and Flows

- **Supplements standard UML port concept with flow ports**
- **A flow models some streaming phenomenon**
  - Either continuous or discrete
  - Energy, liquids, electrical signals, data packets
- **This concept has been adopted and expanded in MARTE**



MARTE Profile: Structure

- Modeling and Analysis of Real-Time and Embedded Systems (MARTE)
- Includes a general facility for specifying quantitative and physical characteristics of software systems and platforms and their functional relationships
- Intended to support
  - Accurate modeling of RTE systems
  - Automated analyses of key system qualities



A Component-Based Solution

- **Everything is a component**
  - Platform: a set of server components
  - Application: a set of client components
- **Platform components are called *resources***
  - Platform objects that provide services through their interfaces
  - Possess *quality of service (QoS)* characteristics
- ***Resource*:**
  - A facility or mechanism *with limited capacity* required to attain some functional objective (e.g., perform a platform service)
- **The limited nature of resources is due to the finite nature of the underlying hardware platform(s)**
  - Contention for shared resources is the primary source of complexity related to platforms

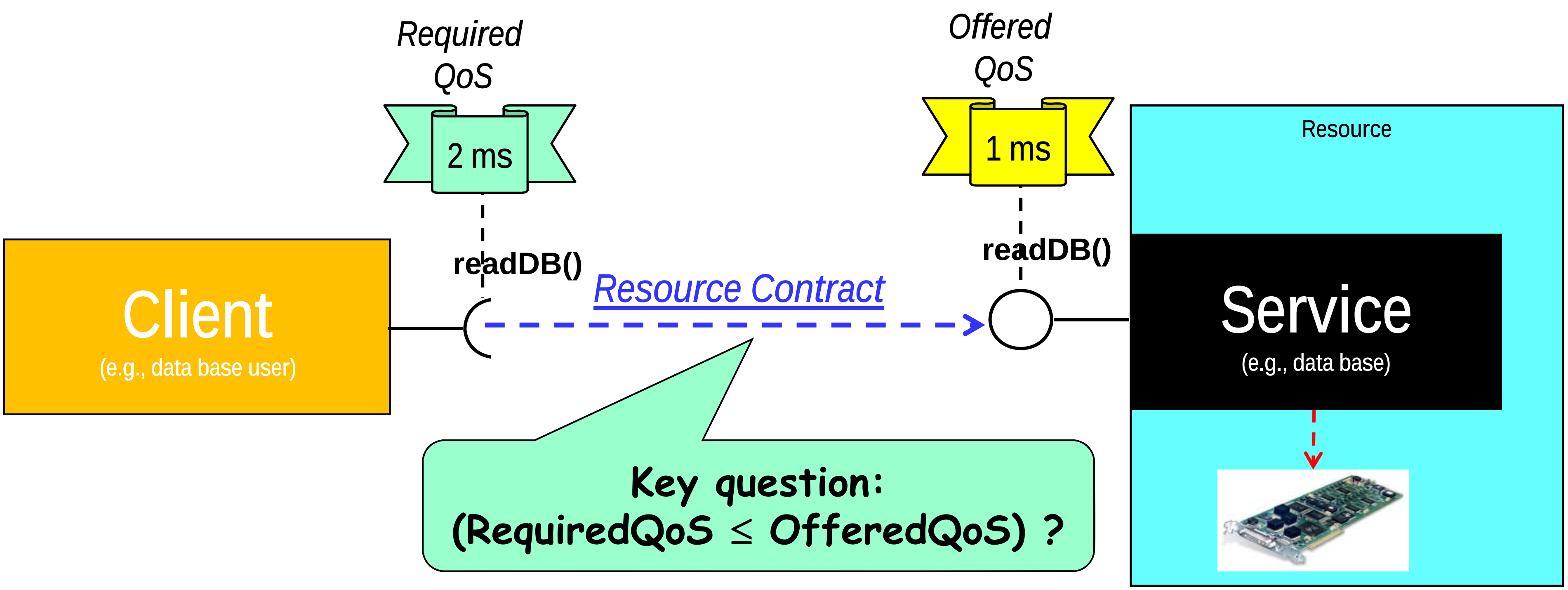
MARTE Core Concept: Quality of Service

- ***Quality of Service*:**

  the degree of effectiveness in the provision of a service

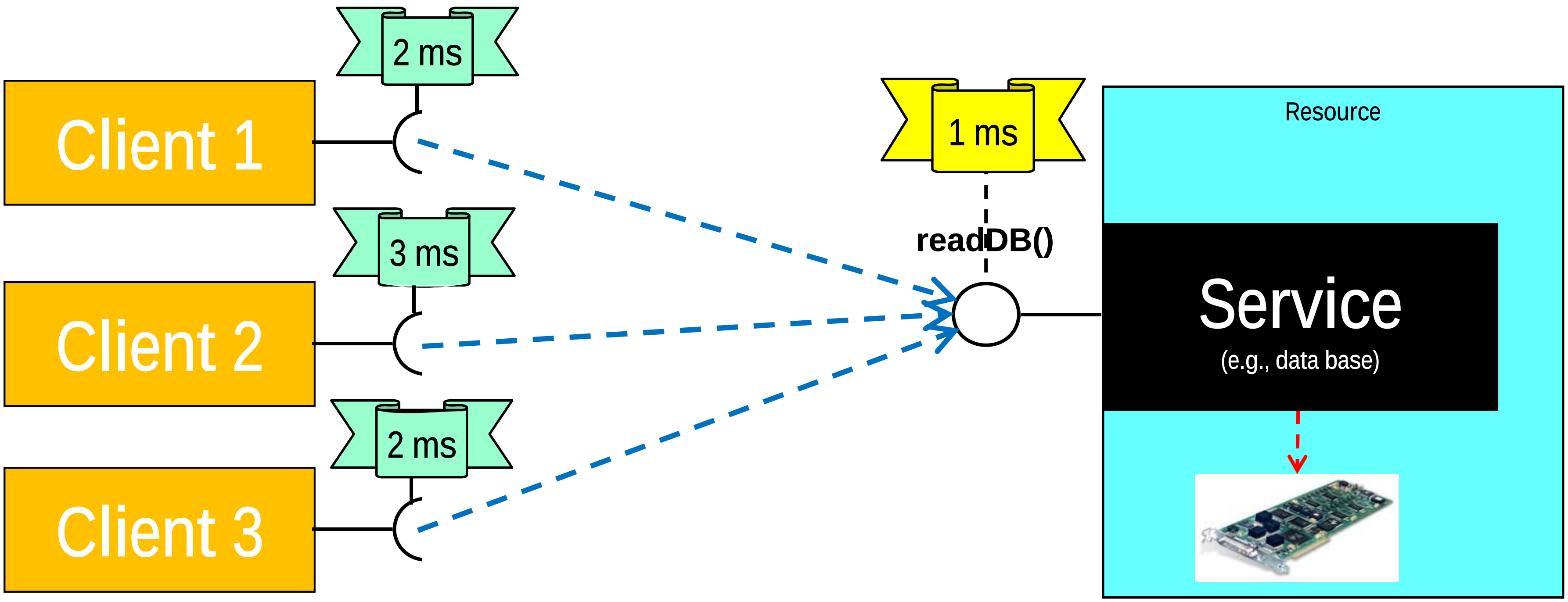
  - e.g. throughput, capacity, response time
- **The two sides of QoS:**
  - *offered QoS*: the QoS that is available (supply side)
  - *required QoS*: the QoS that is required (demand side)
- **Resources characterized by an offered QoS**
- **Application components by a required QoS**
  - NB: Resources may also have their own required QoS in multilayer platform models

- **Key analysis question: Does a resource have the capacity to support its clients?**
  - i.e., does supply meet demand?



The Back-Door Coupling Problem

Architecturally independent components (applications) can become implicitly coupled if they share a platform resource



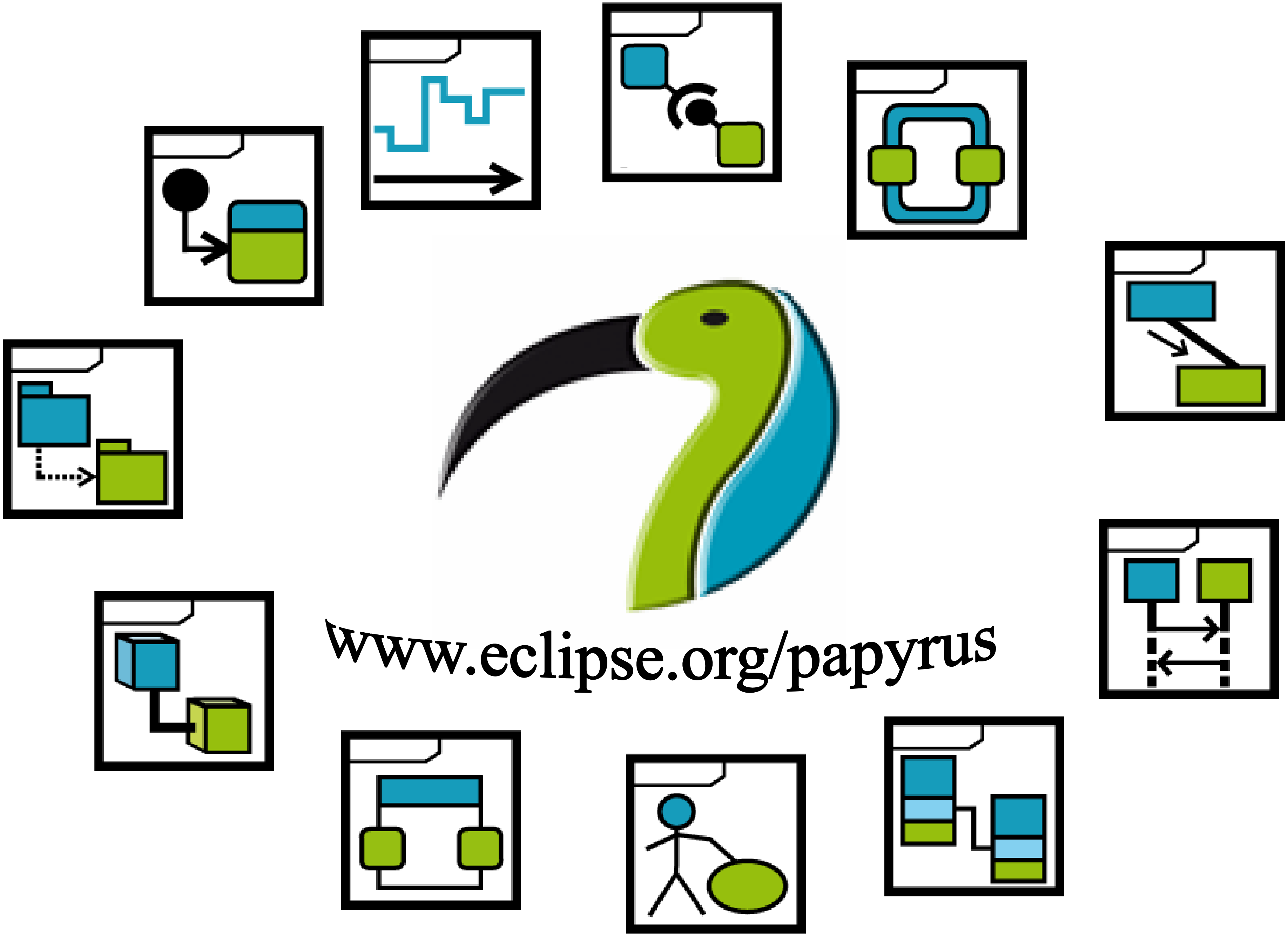
The four pillars of MARTE

- **Pillar1: QoS-aware Modeling**
  - **HLAM**: for modeling high-level RT QoS, including qualitative and quantitative concerns.
  - **NFP**: for declaring, qualifying, and applying semantically well-formed non-functional concerns.
  - **Time**: for defining time and manipulating its representations.
  - **VSL**: the Value Specification Language is a textual language for specifying algebraic expressions.
- **Pillar 2: Architecture Modeling**
  - **GCM**: for architecture modeling based on components interacting by either messages or data.
  - **Alloc**: for specifying allocation of functionalities to entities realizing them.
- **Pillar3: Platform-based Modelling**
  - **GRM**: for modeling of common platform resources at system-level and for specifying their usage.
  - **SRM**: for modeling multitask-based design
  - **HRM**: for modeling hardware platform
- **Pillar4: Model-based QoS Analysis**
  - **GQAM**: for annotating models subject to quantitative analysis.
  - **SAM**: for annotating models subject of scheduling analysis.
  - **PAM**: for annotating models subject of performance analysis.

A Partial List of Projects Using MARTE

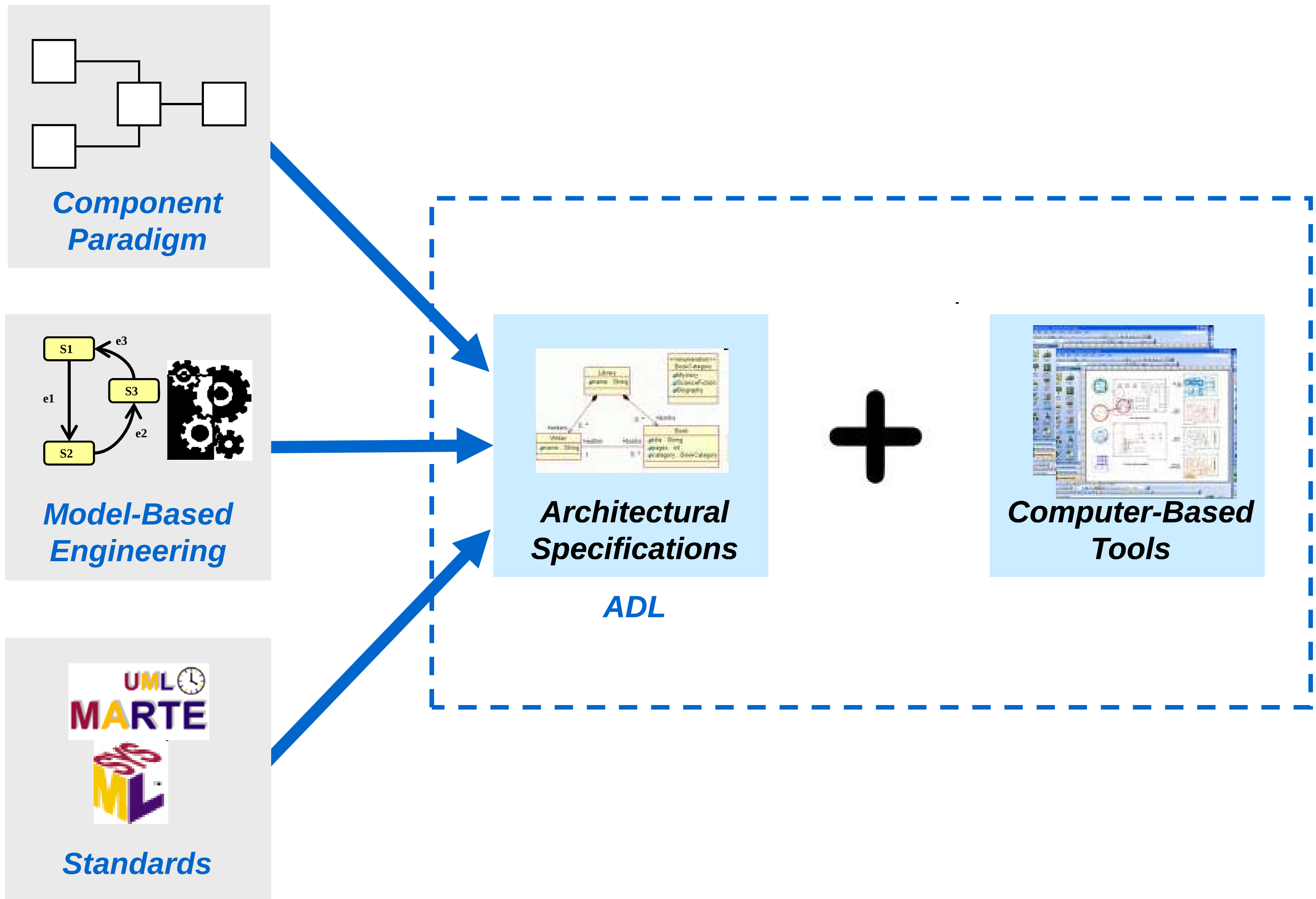
Project name	Project type	Budget. (EUR)	Extract of the partners list
VERDE	EC* ITEA	26 M	CEA, Thales, Alstom, ABB, EADS
ADAMS	EC* FP7	300 K	CEA, Thales, Volvo Tech.
INTERESTED	EC* FP7	6.7 M	CEA, Airbus, Thales, Siemens, Magneti Marelli
LAMBDA	System@tic	5.3 M	CEA, Thales, ST, INRIA
EDONA	System@tic	16 M	CEA, Continental, PSA, Renault
TIMMO	EC* ITEA	8.6 M	CEA, Continental, VW, Siemens
ATESST v1 & v2	EC* STREP	4 M + 3 M	CEA, Volvo Tech, Continental, Siemens
SATURN	EC* FP7	193.2 K	Thales, Artisan Sw., Univ. Paderborn
IMOFIS	System@tic	4 M	CEA, Alstom, Renault
Usine Logicielle	System@tic	15.2 M	CEA, INRIA, Dassault, EADS, EDF, Thales,
OpenEmbDD	System@tic , Minalogic, AerospaceValley	9 M	CEA, Airbus, France Telecom, Thales, INRIA, Verimag
MeMVaTEx	System@tic	2.1 M	CEA, Continental, INRIA
CESAR	EC* ARTEMIS	62.5 M	CEA, Airbus, Delphi, ABB, Dassault, INRIA
PROTEUS	French Project, ANR		Dassault, LIP6, CEA, Thales

Open-Source Tool Support: Papyrus



Summary: A Standards Based ADL

- The combination of SysML, MARTE, and UML 2, provides a foundation for a practical ADL, supported by both commercial and open-source tools



TACK!

QUESTIONS, COMMENTS, ARGUMENTS...