IBM Rational

Model-Driven Development for Real-Time and Embedded Systems

Bran Selic
Malina Software Corp. and IBM Canada
selic@acm.org

ON DEMAND BUSINESS

Outline

- **Introduction: Model-Driven Development and UML**
- **On Software Physics and Software Platforms**
- **MARTE: Combining Software Physics with MDD**
- **Summary and Conclusions**

QoS-Constrained Systems

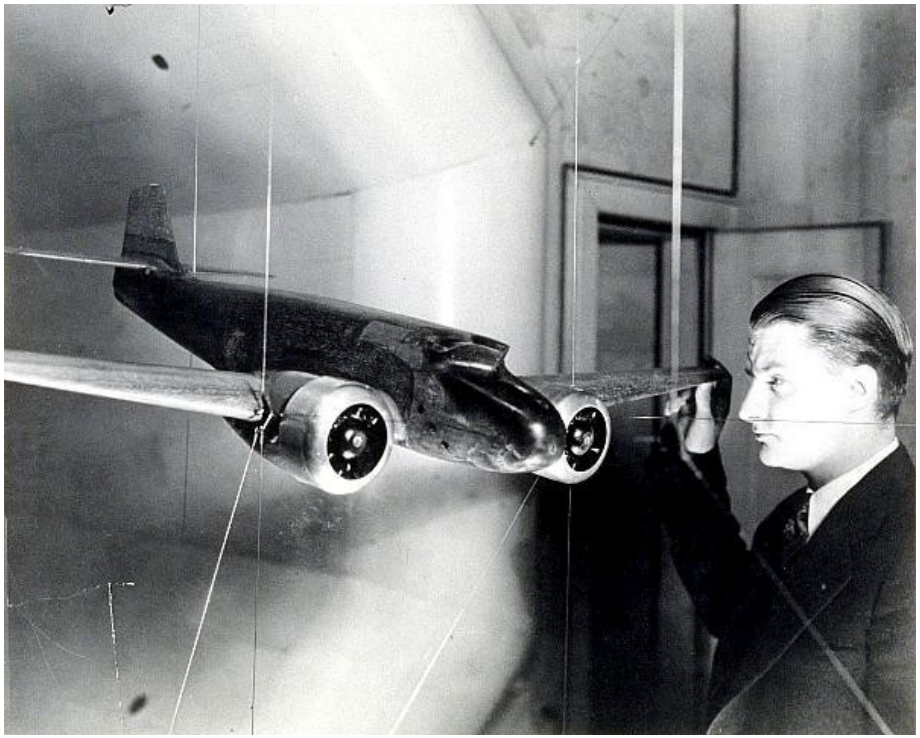
- **Software systems whose functional correctness *critically* depends on meeting one or more *quality of service* (QoS) requirements**
  - Performance, throughput, energy consumption, reliability, etc.
  - Usually real-time embedded systems but not exclusively (e.g., financial trading systems, gaming/simulation systems)
- **Some key design and development challenges of QoS-constrained systems:**
  - Coping with the *complexity* of the physical world
  - Meeting QoS requirements cost effectively in the face of technological and resource limitations

Fred Brooks on Complexity

- **[From: F. Brooks, "The Mythical Man-Month", Addison Wesley, 1995]**
- ***Essential*** **complexity**
  - inherent to the problem
  - cannot be eliminated by technology or technique
  - e.g., the algorithmic complexity of the "traveling salesman" problem
- ***Accidental*** **complexity**
  - due to use of inappropriate technologies or methods
  - e.g., building a skyscraper without modern power tools

Model-Based Engineering: Coping with Complexity

- ***Engineering model*: A reduced representation of some system that highlights its properties of interest from a given viewpoint**



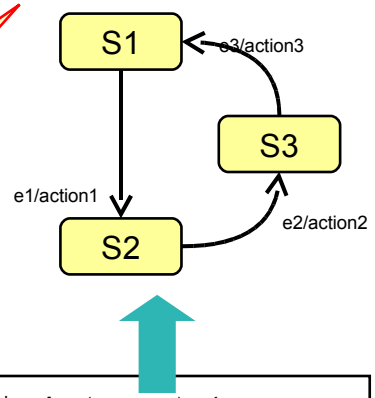
- We don't see everything at once – only the important stuff (abstraction)
- Useful models present key information in a manner that supports understanding and reasoning about a system

Model-Driven Development (MDD) of Software

- **An approach to software development in which models play an *indispensable* role**
- **Based on two time-proven engineering methods:**

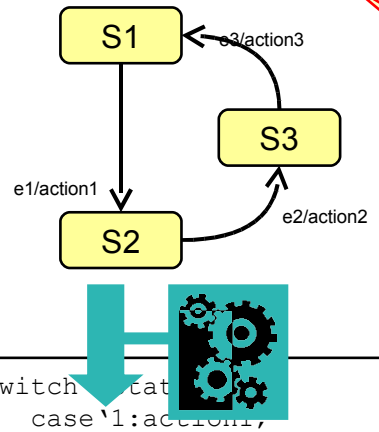
(1) ABSTRACTION

Realm of modeling languages (e.g., UML)




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

(2) AUTOMATION

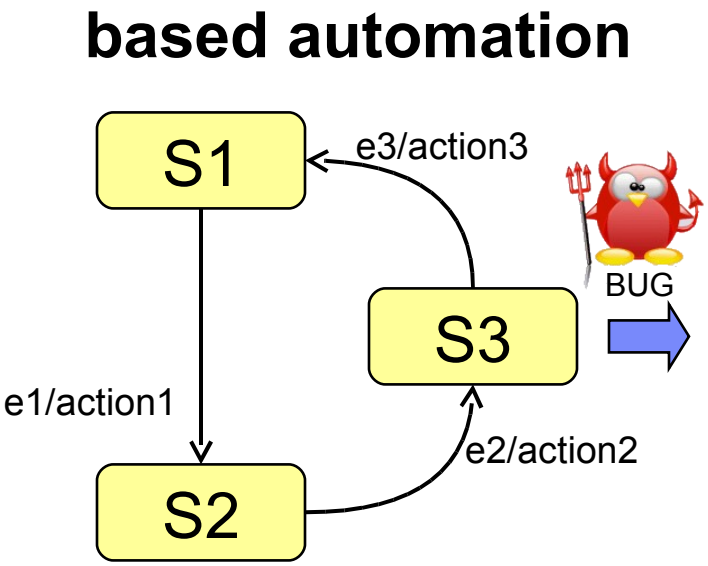



```
switch (state) {
   case‘1:action1;
          newState(‘2’);
          break;
   case‘2:action2;
          newState(‘3’);
          break;
   case’3:action3;
          newState(‘1’);
          break;}
```

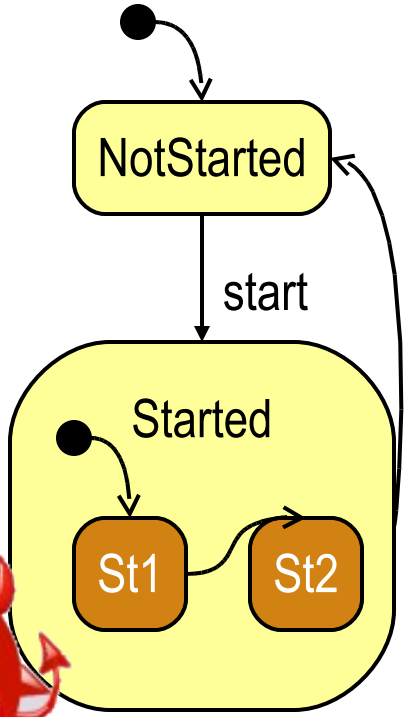
Realm of tools

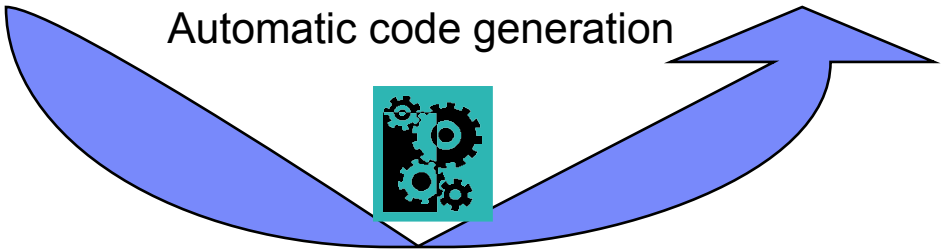
MDD: The Need for Automation

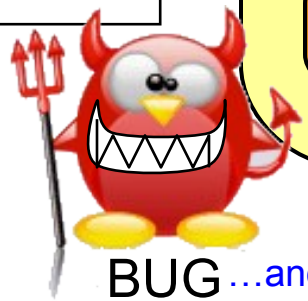
- **The accidental complexity of QoS-aware systems can be greatly mitigated by the appropriate use of computer-based automation**




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





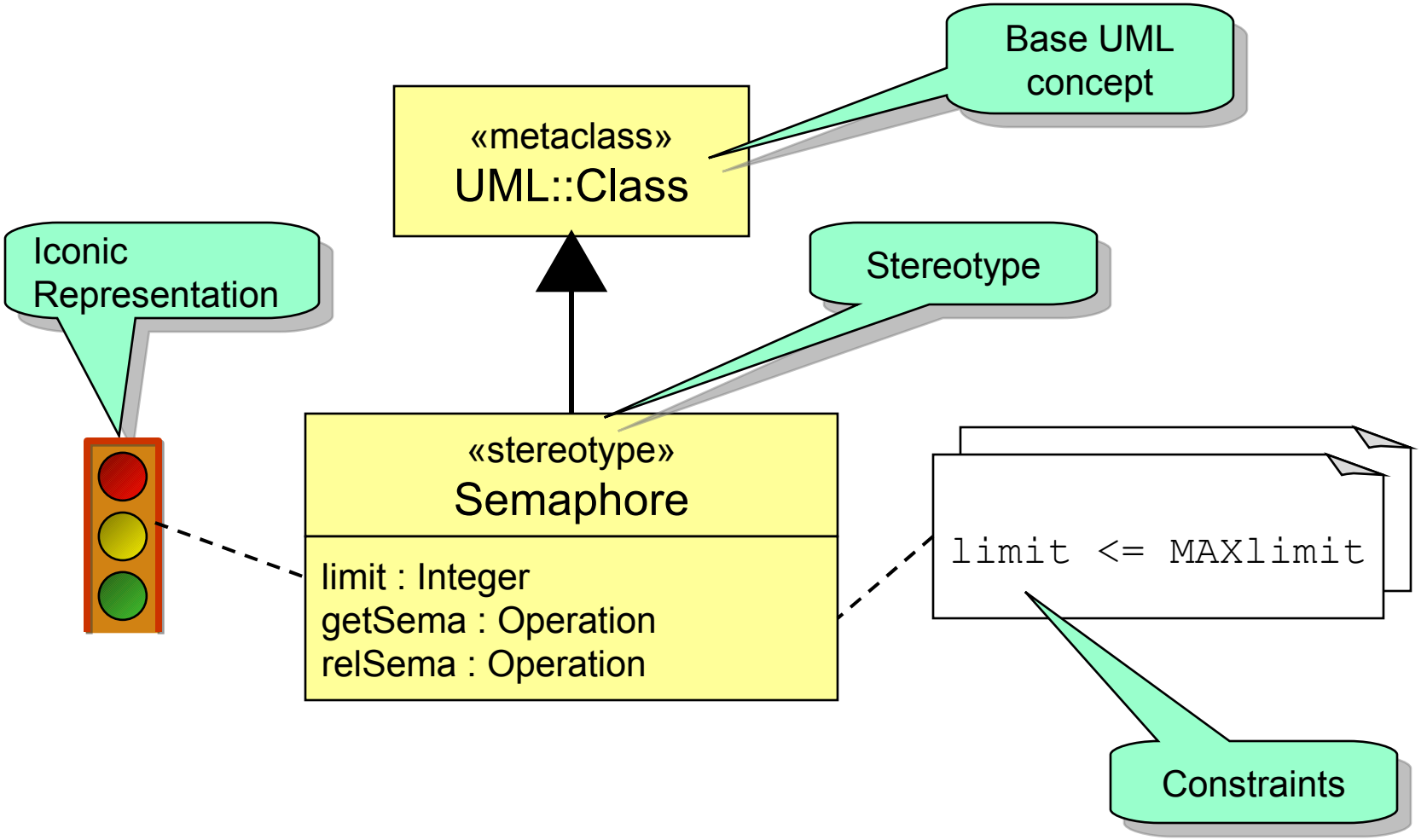


BUG …and what about advanced modeling languages??

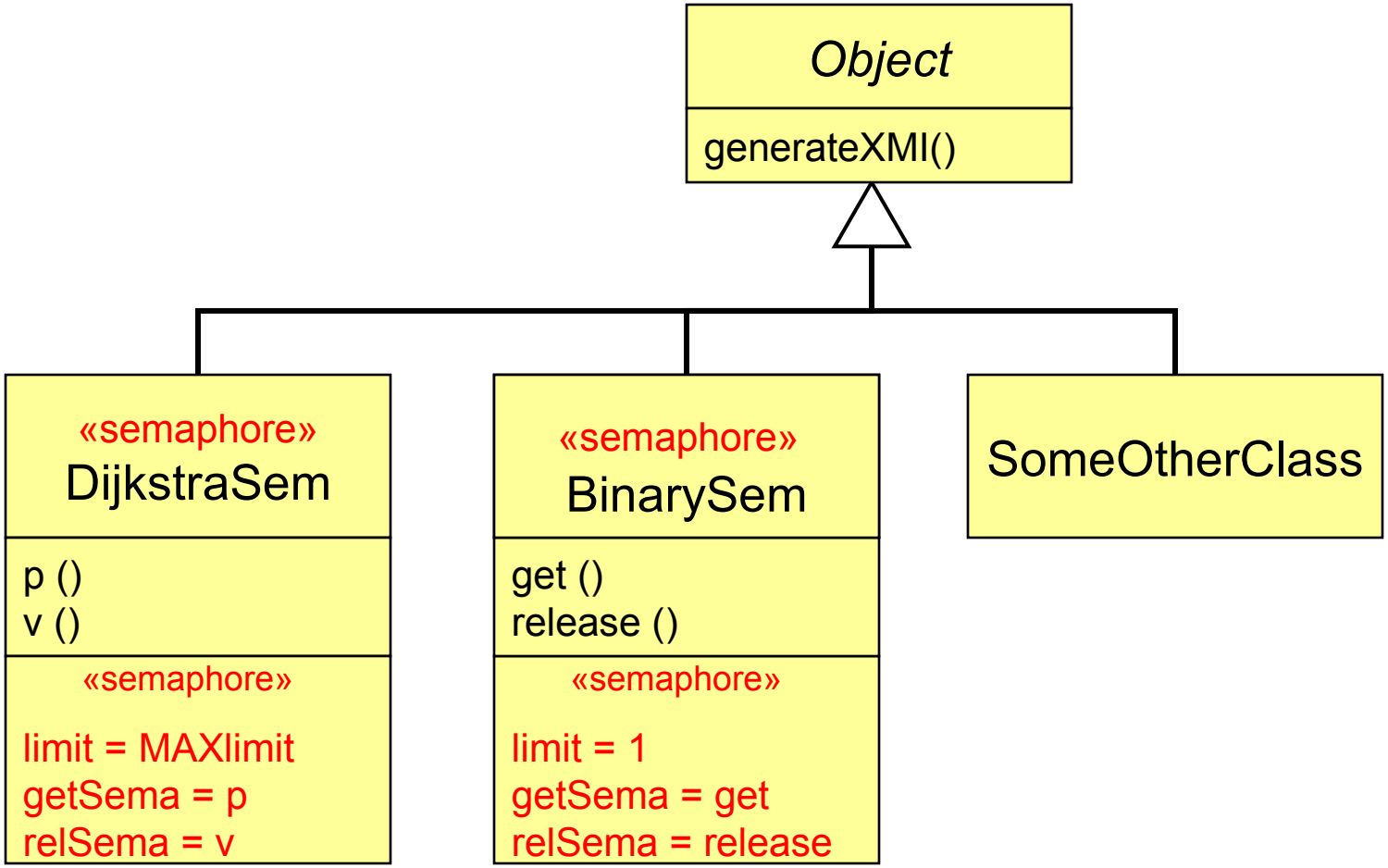
UML 2: A Standardized Language for MDD

- **UML 2 was motivated by the idea of MDD**
  - Needed a semantically more precise form of UML
- **Designed as a "family" of modeling languages**
  - Controlled ambiguity: *semantic variation points*
    - e.g., scheduling policy, type consistency, polymorphism rules
  - Language specialization achieved by providing domain-specific constraints for semantic variation = *profile*
  - Advantages:
    - Reuse of base language tooling infrastructure
    - Reuse of base language knowledge and experience
    - Reuse of base language design
  - Example: ITU-T SDL language for telecom (Z.100)
    - Specified as a UML profile (Z.109)

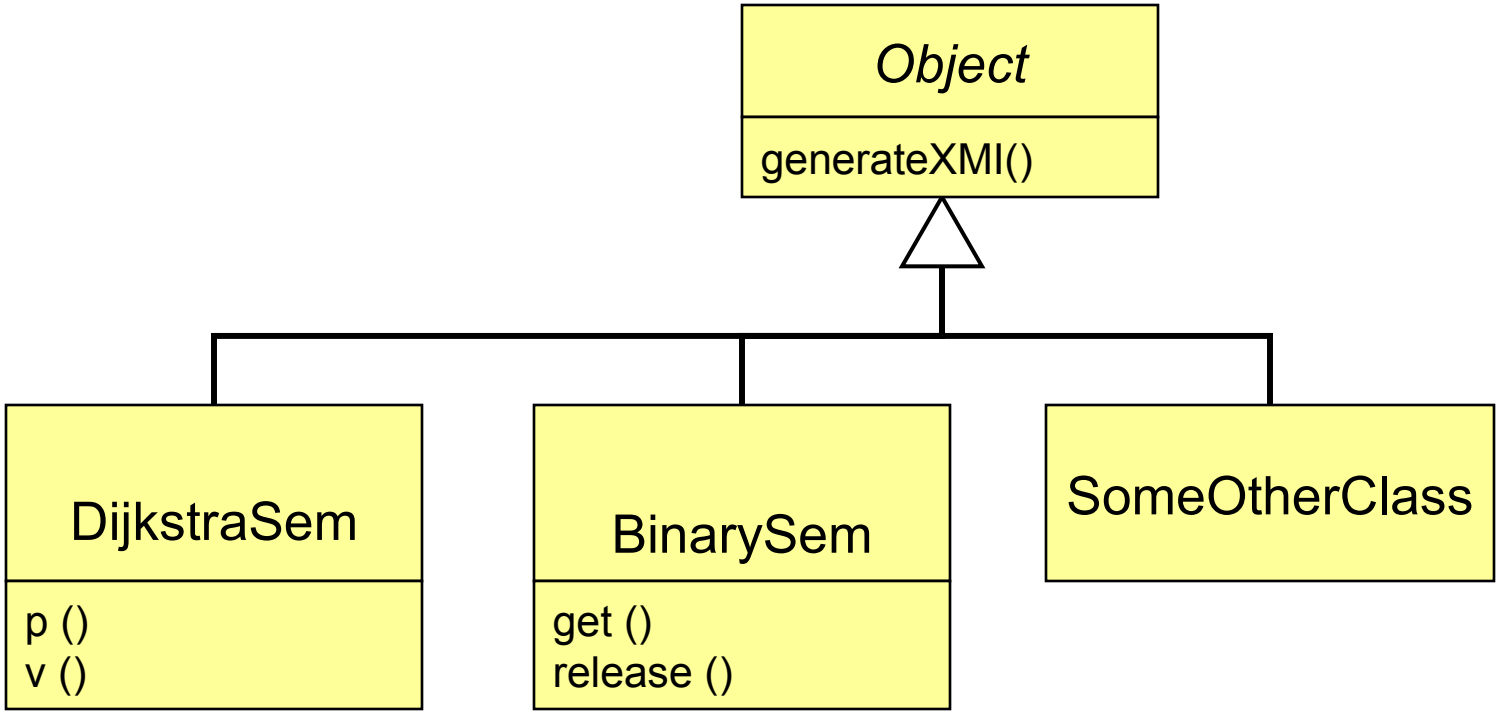
Example: Adding a Semaphore Concept to UML



Example: Using the Stereotype



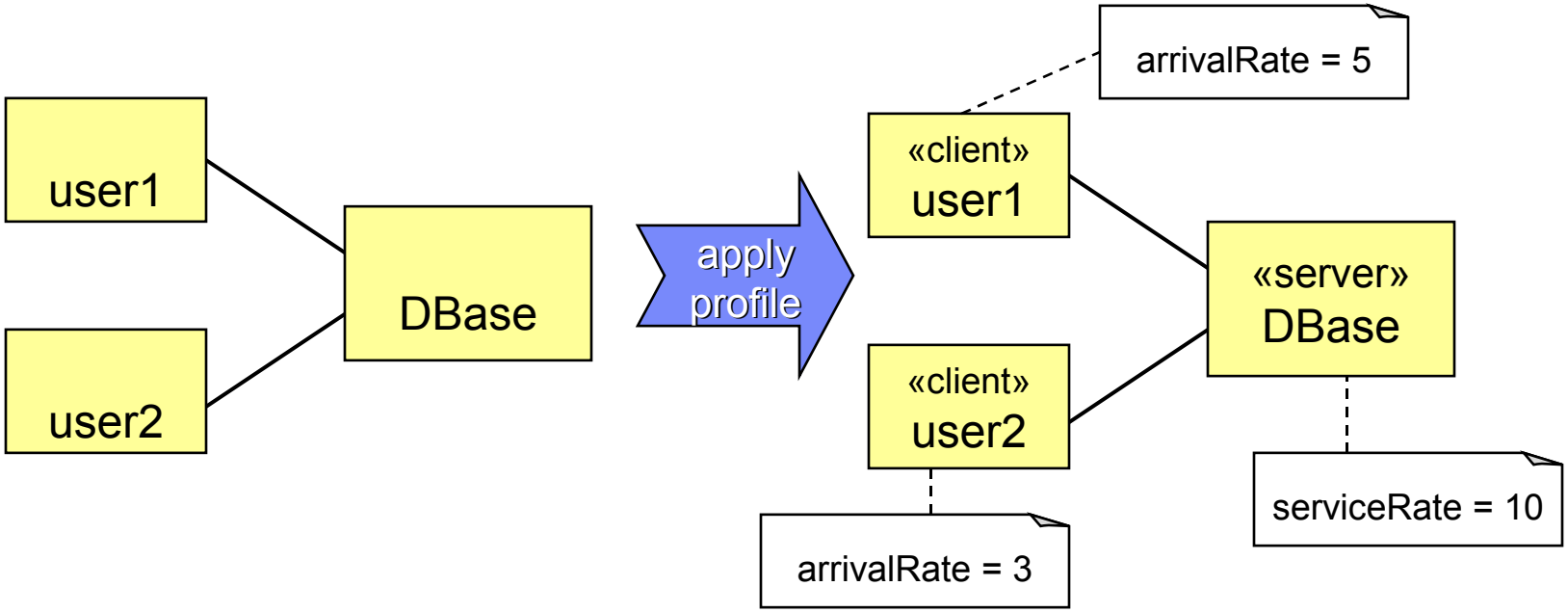
Example: Using the Stereotype



- **It is possible to "unapply" a profile, which leaves just the underlying base classes**

UML Profiles as a Viewpoint Mechanism

- A profile can be used as an overlay mechanism that can be dynamically applied or "unapplied" to provide a domain-specific interpretation of a UML model
- Example: viewing a UML model fragment as a queueing network (for performance analysis)



UML and QoS

- **Standard UML does not provide any standardized means for specifying QoS characteristics**
  - *A symptom of a general shortcoming in the software engineering culture*
- **Several profiles related to UML 1.x to deal with this:**
  - UML Profile for Scheduling, Performance, and Time (SPT)
  - UML Profile for Fault Tolerance and Quality of Service
- **Most recently: UML Profile for Modeling and Analysis of Real-Time and Embedded Systems (MARTE)**
  - An evolution of SPT for UML 2

Outline

- Introduction: Model-Driven Development and UML
- On Software Physics and Software Platforms
- MARTE: Combining Software Physics with MDD
- Summary and Conclusions

Two Divergent Views

A Very Ancient View

"All machinery is derived from nature, and is founded on the teaching and instruction of the revolution of the firmament."

- Vitruvius
On Architecture, Book X
1st Century BC

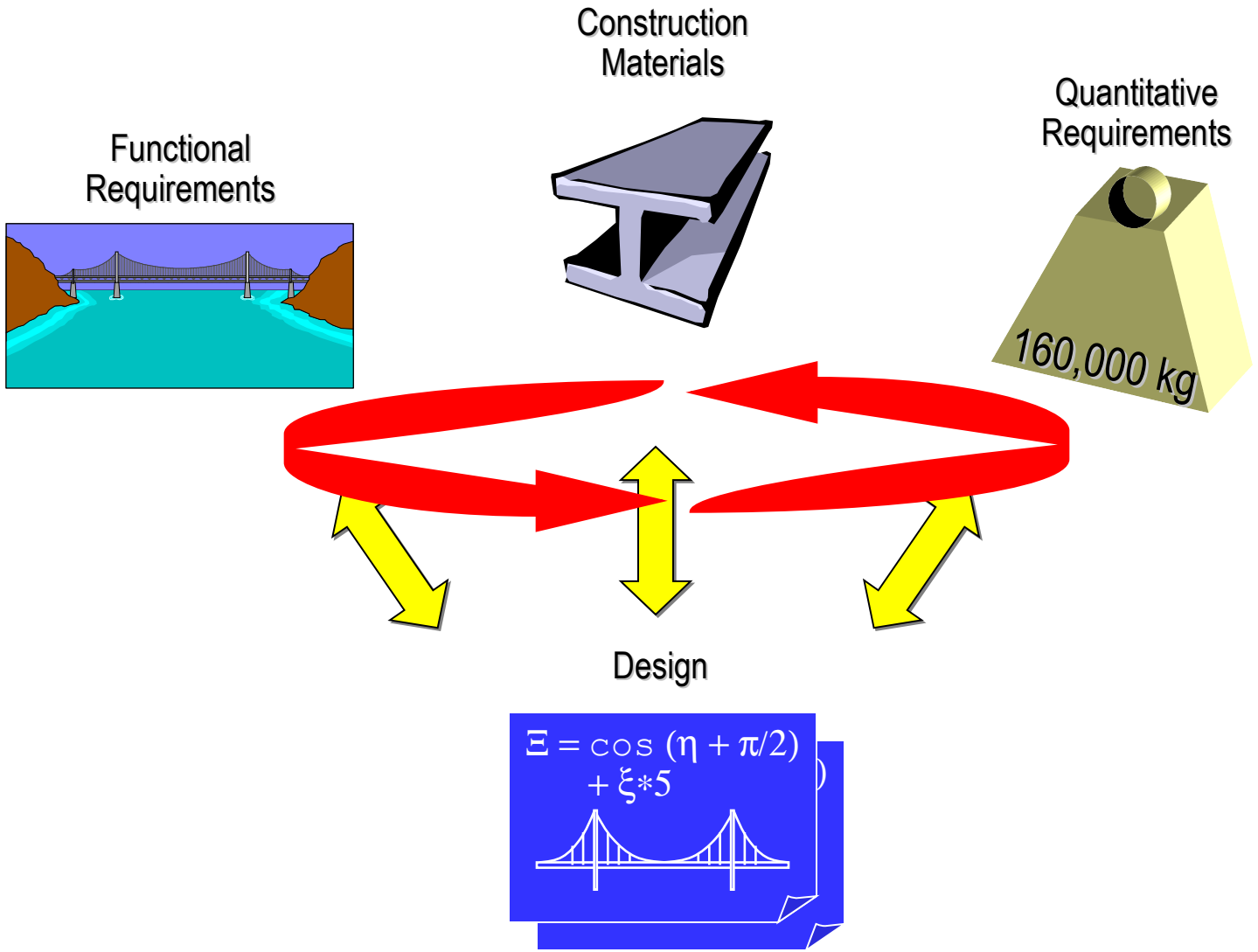
…and a Very Modern One:

"Because [programs] are put together in the context of a set of information requirements, they observe no natural limits other than those imposed by those requirements. Unlike the world of engineering, there are no immutable laws to violate."

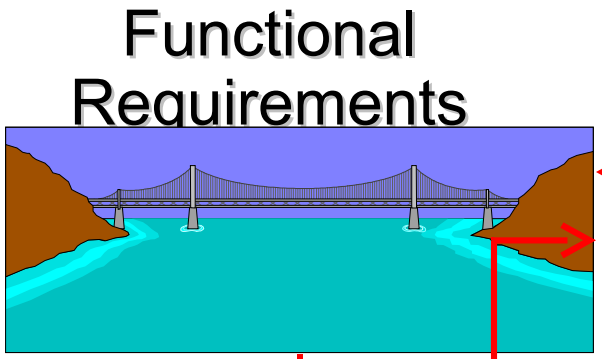
- **Wei-Lung Wang**
Comm. of the ACM (45, 5)

May 2002

Traditional Engineering Design



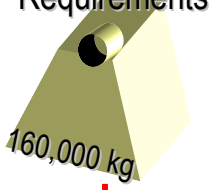
How Things are Typically Done in Software



¬OK

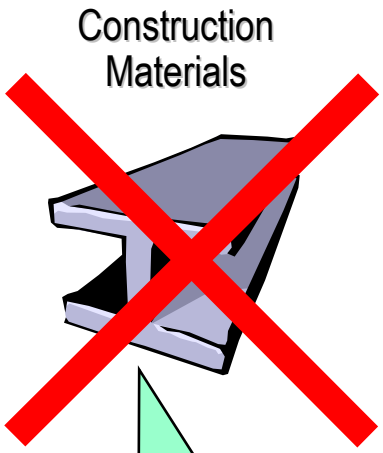
OK

Quantitative Requirements



¬OK

The concerns are serialized with the functional ones given *significantly* greater priority

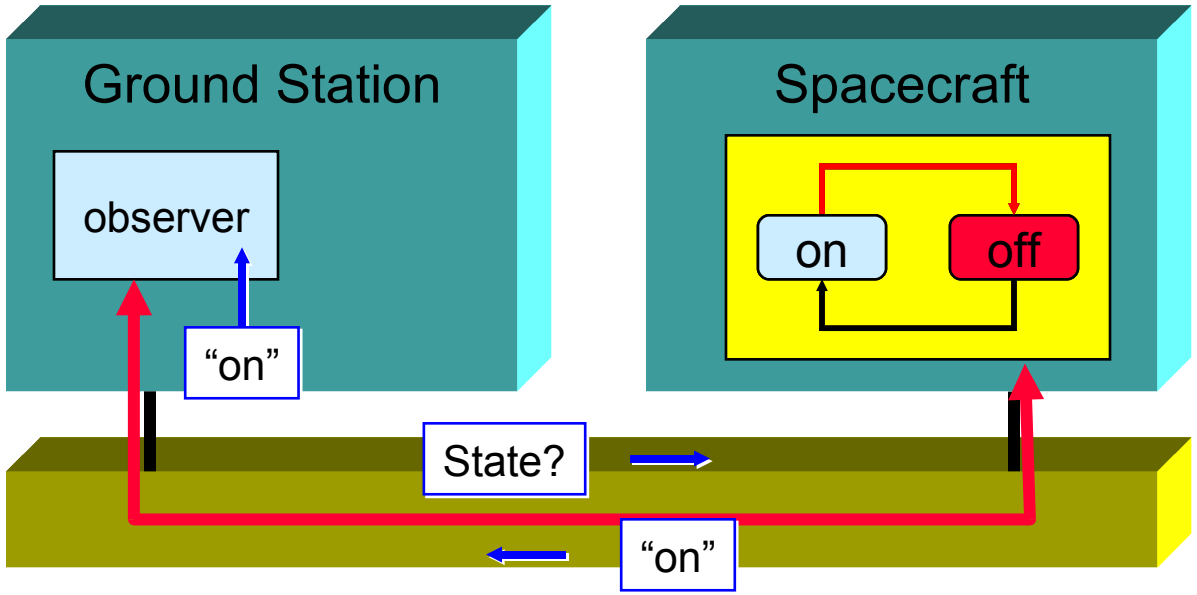


The potential impact of technological characteristics on design is often ignored

What is Software Made of?

Case Study: Effect of Transmission Delays

- **The issue of out-of-date status information**



Software Physics: The Great Impossibility Result

It is not possible to guarantee that agreement can be reached in finite time over an asynchronous communication medium, if the medium is lossy or one of the distributed sites can fail

- Fischer, M., N. Lynch, and M. Paterson, "Impossibility of Distributed Consensus with One Faulty Process" *Journal of the ACM*, (32, 2) April 1985.

- *In many practical systems, the physical platform is a primary design constraint that cannot be overcome by layers of software*

  Computer system = software + hardware

- *Yet, students are still being taught that "platform concerns" are second order issues*

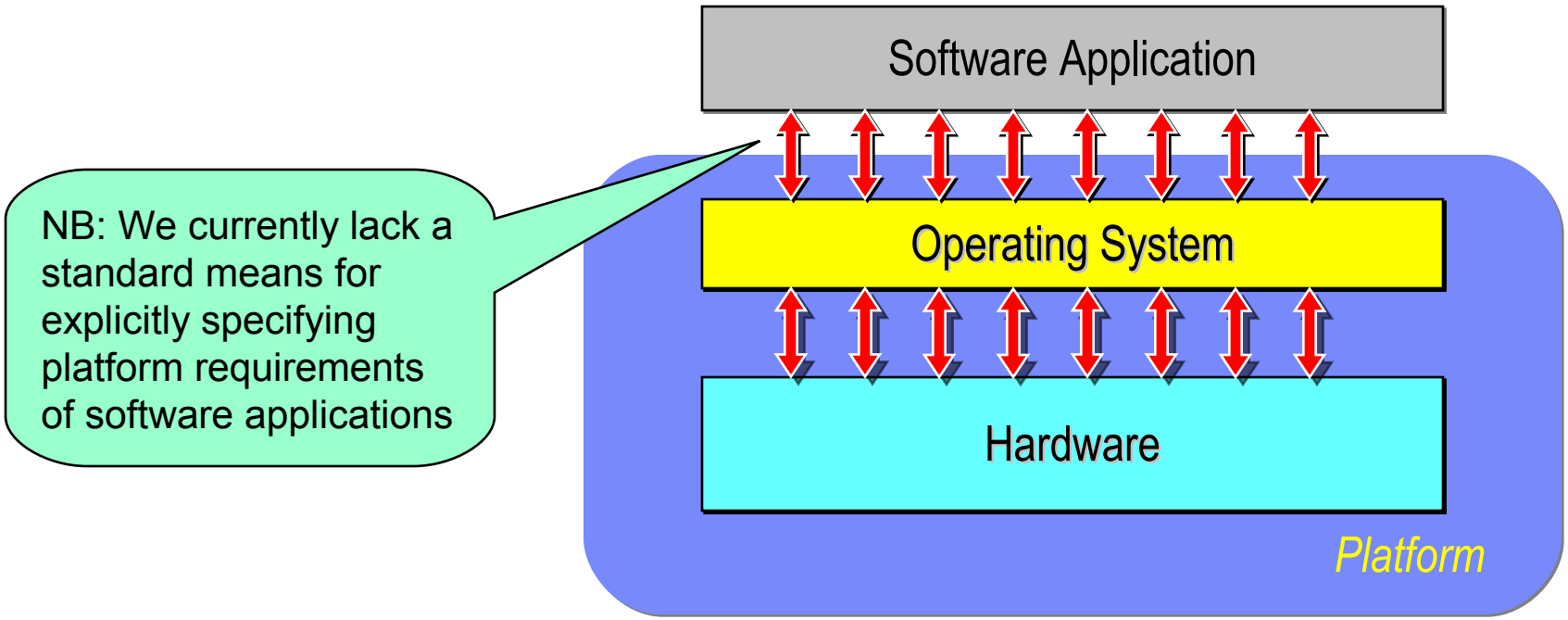
The Dangers of Disregarding Physics

"Time has been systematically removed from theories of computation, since it has been viewed as representing the annoying property that computations take time."

E. Lee, UC Berkeley

- **Failure to distinguish between computational theory and software engineering practice**
- **Lack of fundamental quantitative analysis skills**
  - Basic "back of the envelope" calculation
- **Consequences**
  - 7-second dialtone delay
  - 8 GB PC application

What Software is Made Of



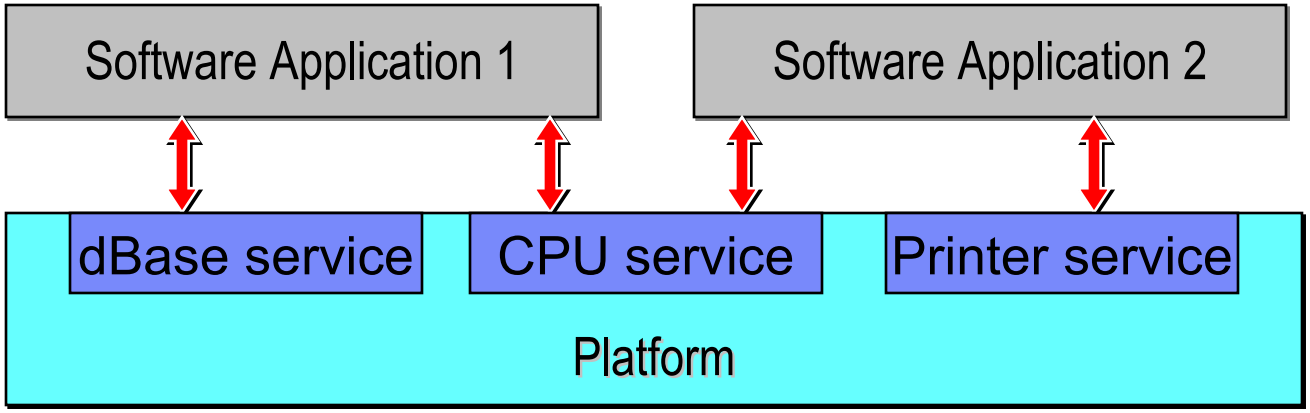
- ***Platform:***

  the full complement of software and hardware required for an application program to execute correctly

But What About Platform Independence?

- **An important and useful notion**
  - Helps abstract away irrelevant technological detail
  - Allows software portability
- **But…**
  - "Things should be made as simple as possible but no simpler" (A. Einstein)
  - Not all aspects of a platform are necessarily irrelevant
- ***Platform independence does not imply platform ignorance***
  - There is a way of achieving platform independence that accounts for relevant platform characteristics

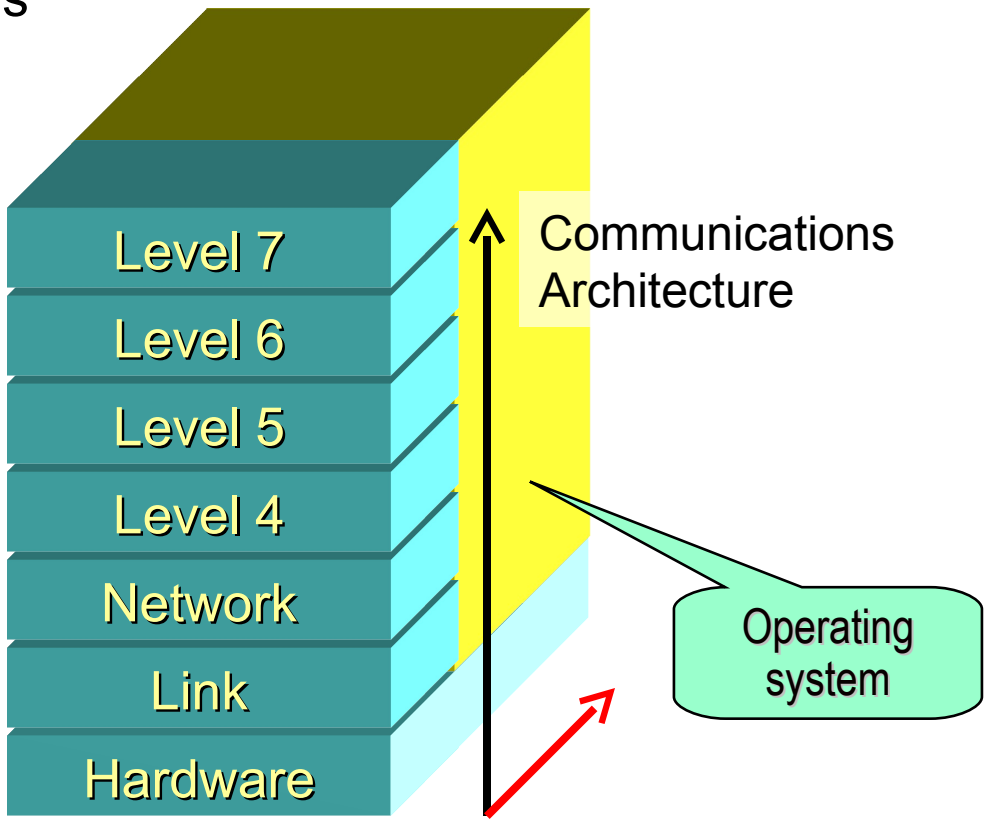
Platforms as Service Providers



- **The relationship between applications and platforms can be represented as an instance of the client-server pattern**
  - NB: Most platforms can support multiple independent applications
  - Services are often shared by multiple applications
- To deal with hardware platforms, we must generalize the concept of service to include more than just software API-type services:
  - CPU (processing) service
  - Special device services (e.g., sensors and actuators)
  - Storage service, etc.

Layered Platforms

- **In real systems, layering is a complex multi-level and multi-dimensional relationship**
  - Cannot be accurately described by simple two-dimensional representations



Resources, Services, and QoS

- **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
- **The extent to which a particular service is limited in its capacity to provide its service (due to the limitations of underlying resources) is expressed by its offered *quality of service* (QoS)**

Quality of Service

- ***Quality of Service:***

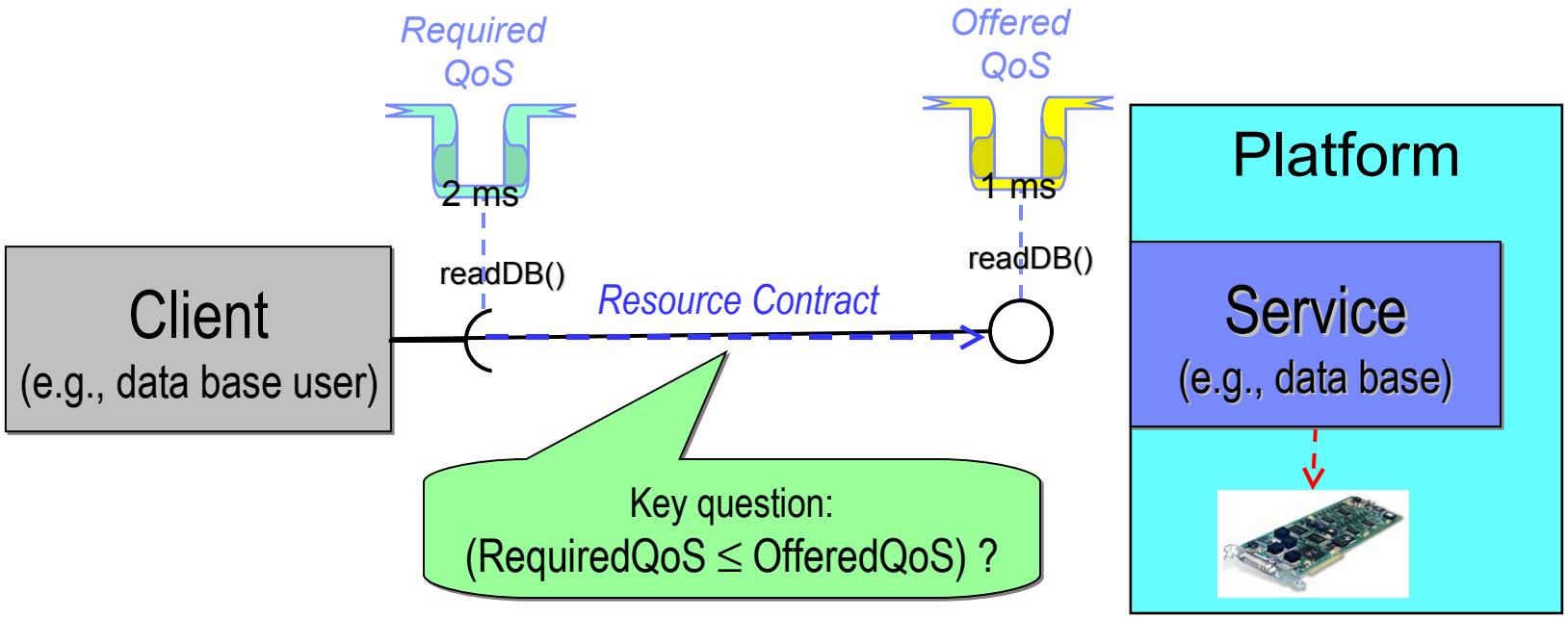
  the degree of effectiveness in the provision of a service

  - e.g. throughput, capacity, response time

- **The two perspectives of QoS:**
  - *offered QoS:* the QoS that is available (supply side)
  - *required QoS:* the QoS that is required (demand side)

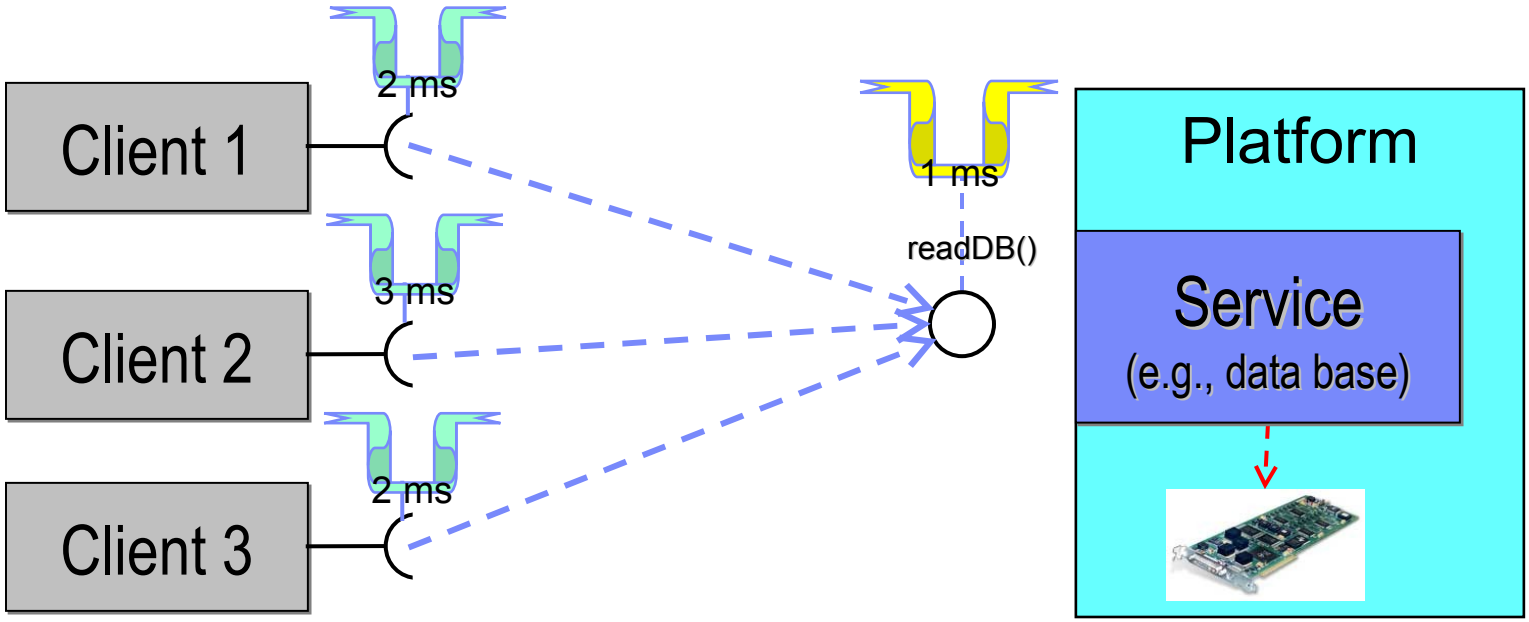
Reconciling Required and Offered QoS

- **Key analysis question: Does the service (platform) have the capacity to support its clients?**
  - i.e., does supply meet demand?



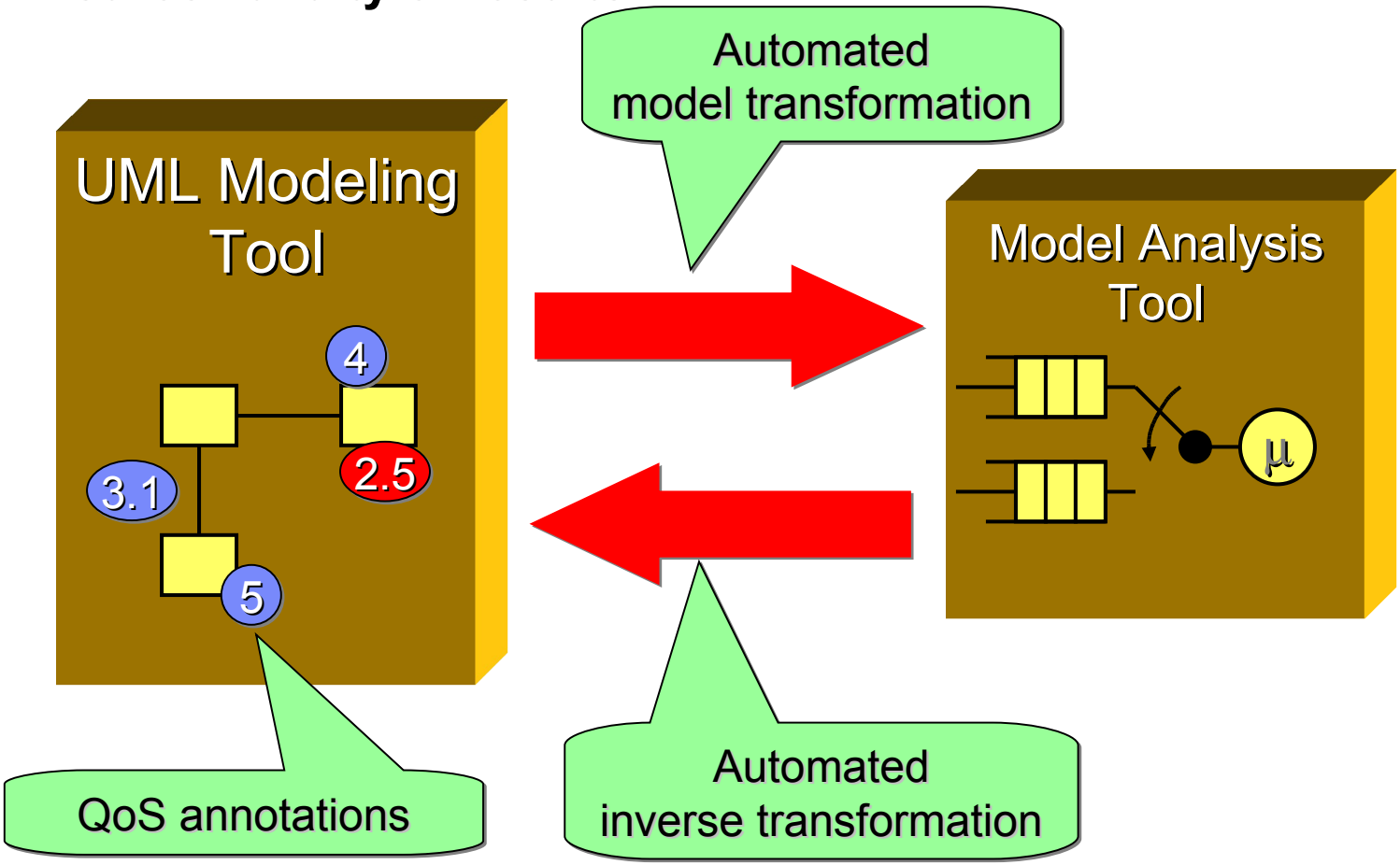
Resource Contention Issues

- **Co-located applications are often designed independently**
  - It can be extremely difficult to predict how they will affect each other
  - Fortunately, a number of methods have emerged for certain key types of QoS (e.g., schedulability analysis, queueing theory)



Objective: Automating Complex KPI Analyses

- **Reduces need for rare and expensive analysis expertise**
- **Ensures validity of results**



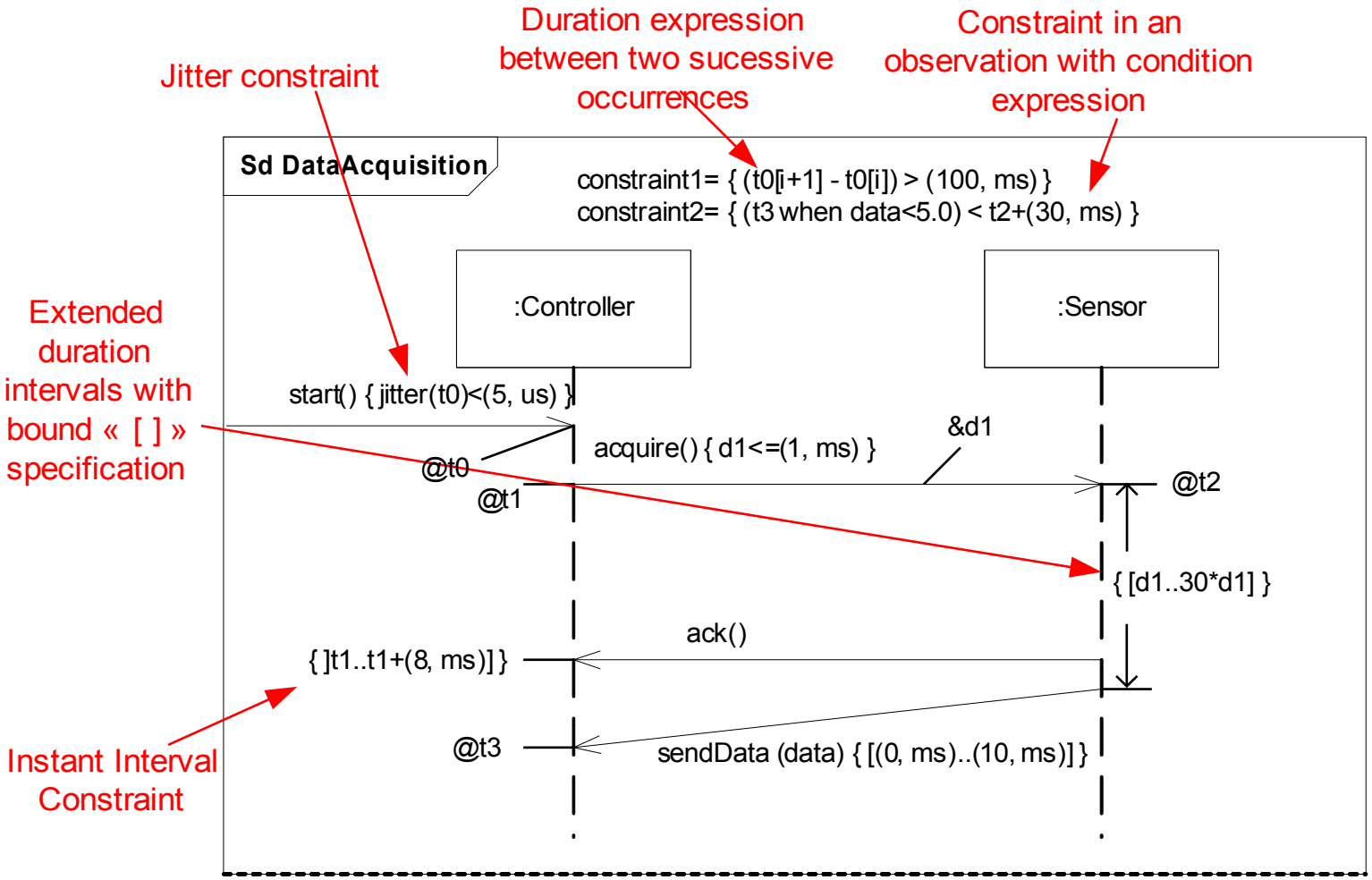
Outline

- Introduction: Model-Driven Development and UML
- On Software Physics and Software Platforms
- MARTE: Combining Software Physics with MDD
- Summary and Conclusions

Introducing Physics to MDD: The MARTE Profile

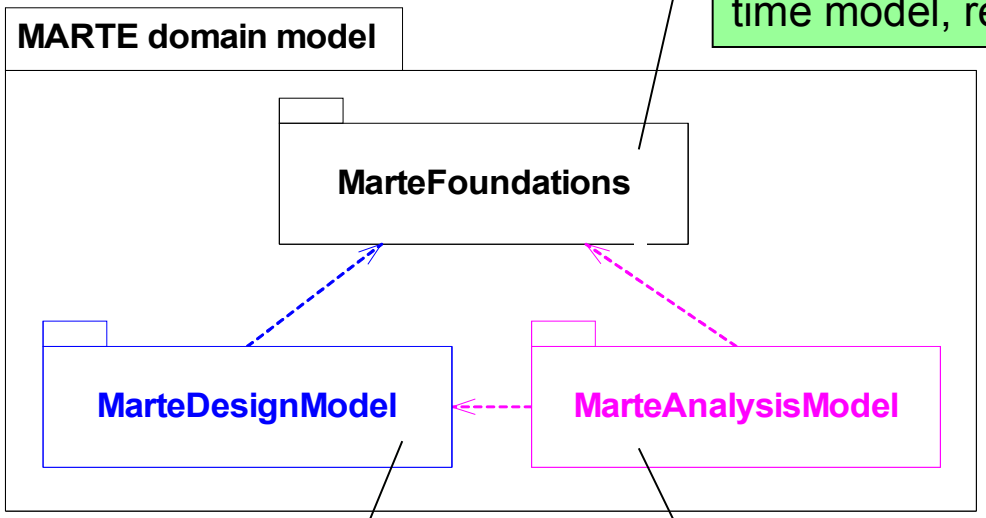
- **UML profile for Modeling and Analysis of Real-Time and Embedded Systems (MARTE)**
  - An OMG standard profile, based on UML 2
- **Support precise modeling of key RTE systems phenomena**
  - Qualitative and quantitative modeling of HW and SW and relationships between them
- **Supports automated analyses of KPIs of RTE systems**
  - Schedulability analyses
  - Performance analyses
- **Also supports specification of complex functional relationships between QoS characteristics**

Example MARTE Annotations



Slide courtesy of Sebastien Gerard, CEA-LETI

Architecture of the MARTE specification



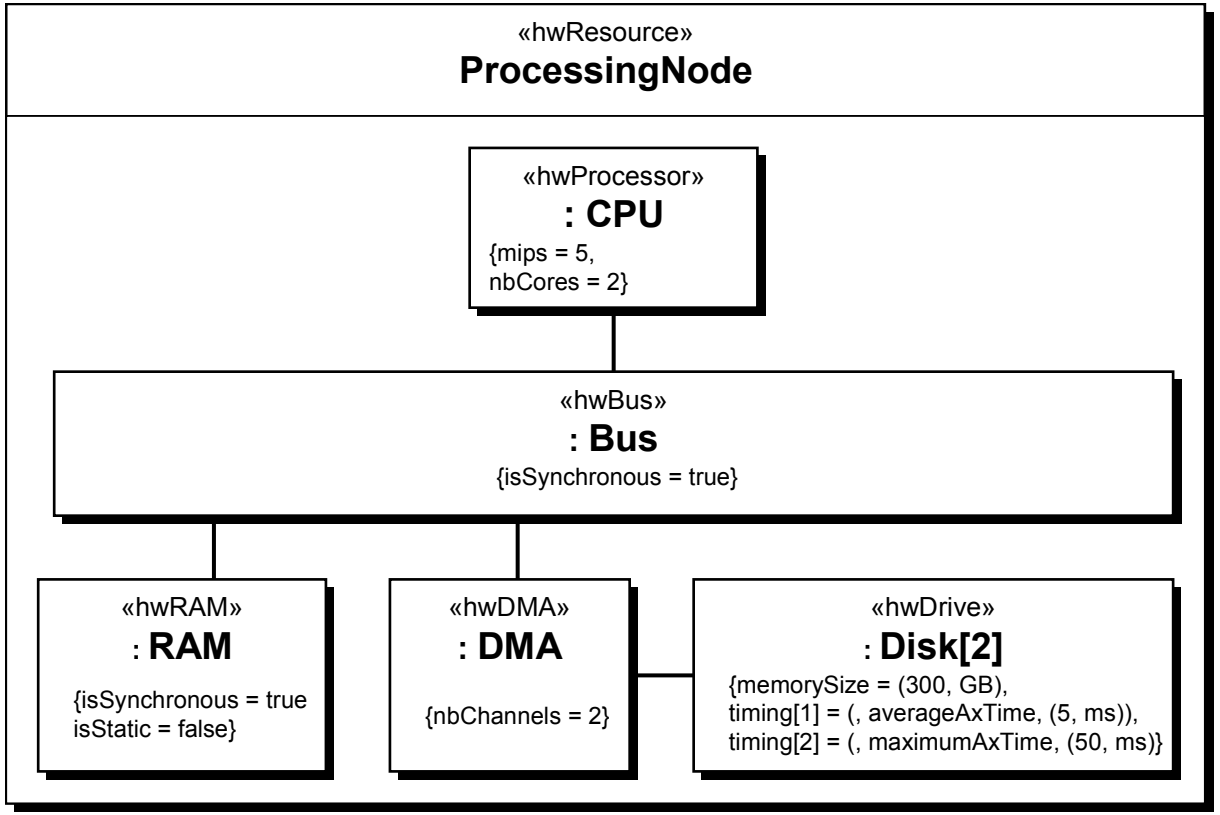
Foundations for RTE systems modeling and analysis (e.g., time model, resource model)

Specialization for precise modeling of RTE systems (e.g., CPUs, concurrency threads)

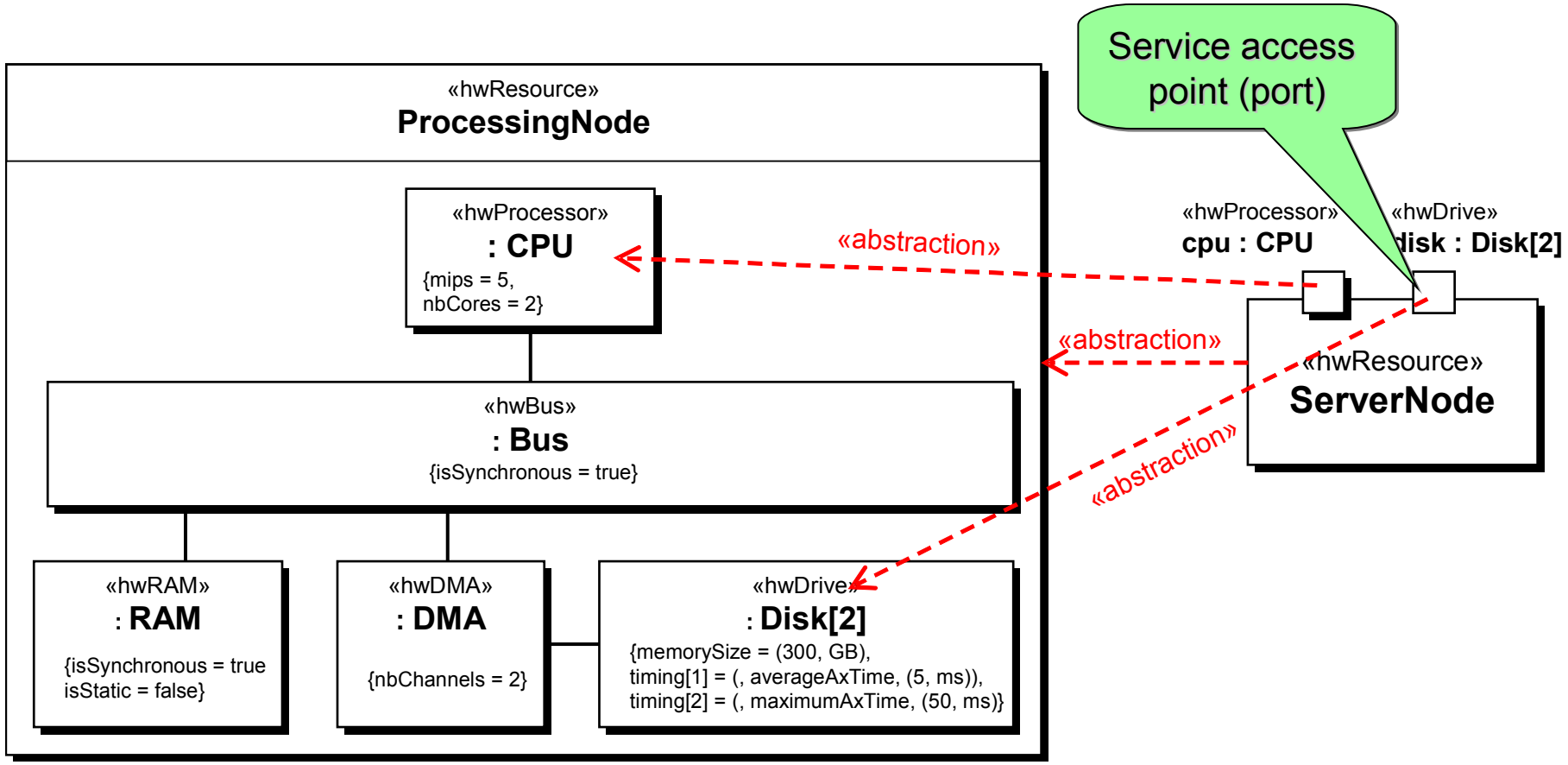
Specialization formal analyses of RTE systems (schedulability, performance)

(Slide credit of S. Gerard)

Example: Hardware Platform Model

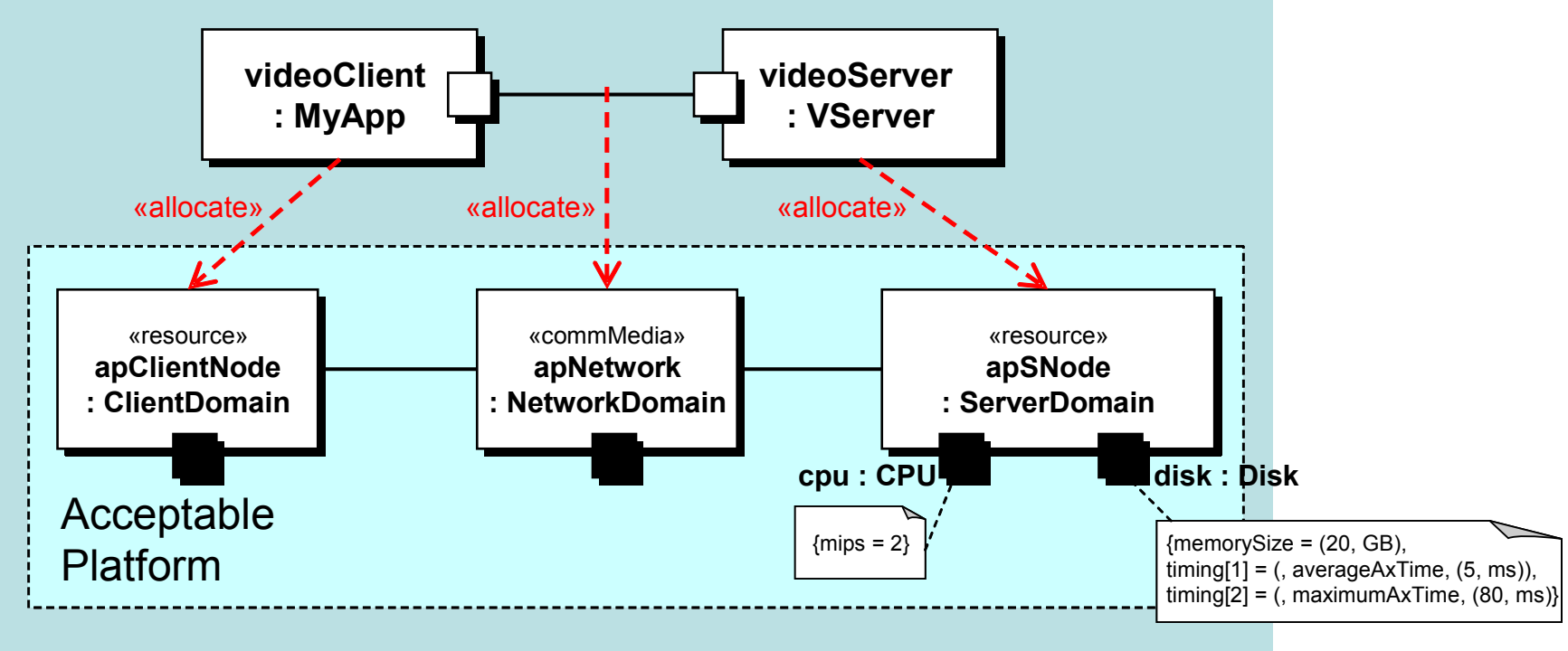


Example: Allocating an Application to a Platform



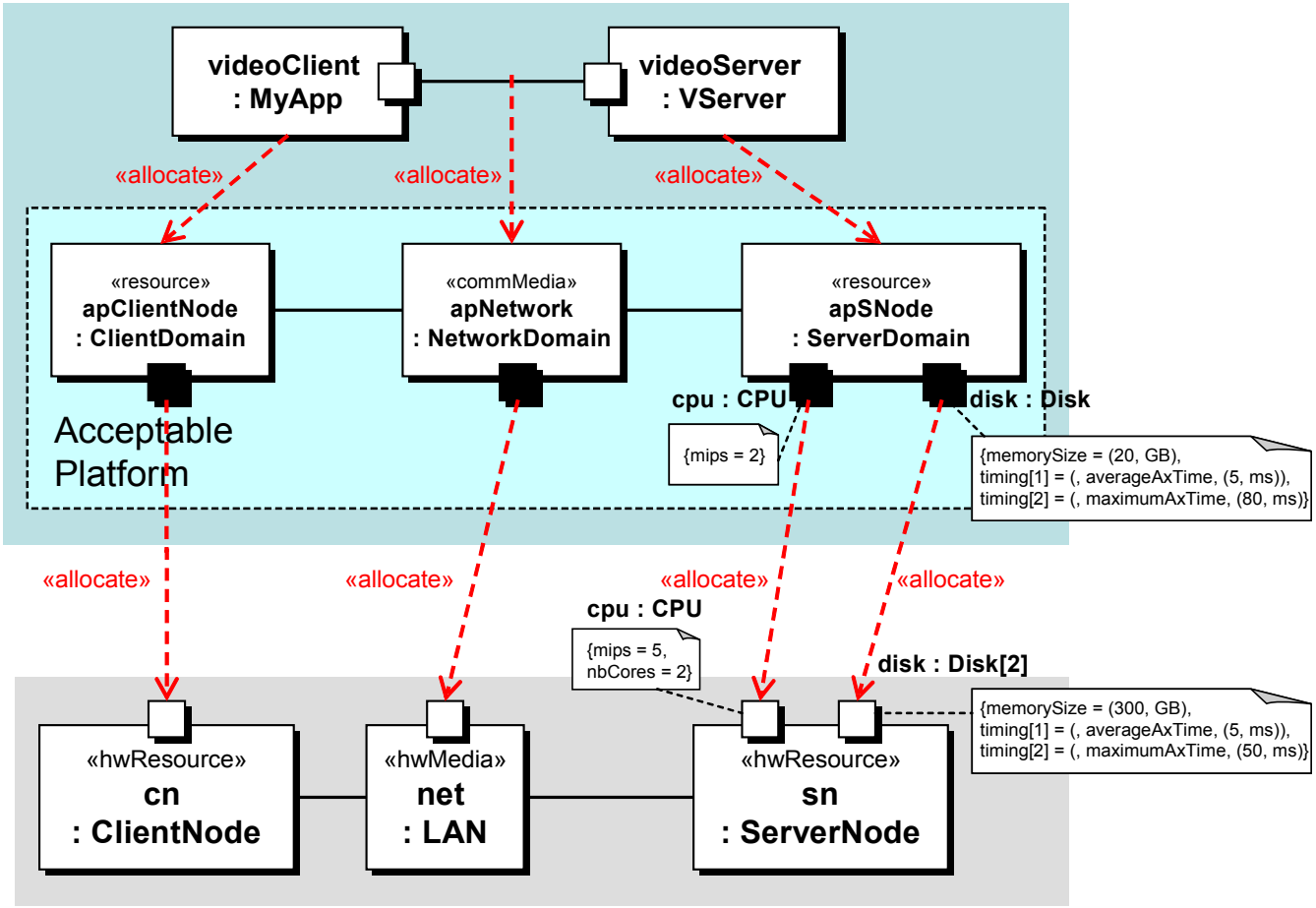
Specifying Required QoS with MARTE

- **An application can include a spec of an "acceptable platform" that defines minimal acceptable QoS values**
  - Provides platform independence with platform awareness



Matching Required and Offered QoS

- This combination of models can be formally analyzed



Outline

- Introduction: Model-Driven Development and UML
- On Software Physics and Software Platforms
- MARTE: Combining Software Physics with MDD
- Summary and Conclusions

Summary and Conclusions

- **MDD offers some significant opportunities to advance the state of the art of software development**
  - Use of abstraction (to deal with complexity) and automation (to bolster reliability and productivity)
- **UML 2 combined with the MARTE profile provides a powerful and standardized combination that fully opens up the application of MDD to real-time and embedded systems**
  - Ability to specify problems and solutions at higher levels
  - Including the ability to be precise about platform-independence

Questions

selic@acm.o