Why Size Matters: Real-Time Systems Engineering with UML

- The standard real-time UML profile -

Bran Selic

Rational Software Inc.

bselic@rational.com

Overview

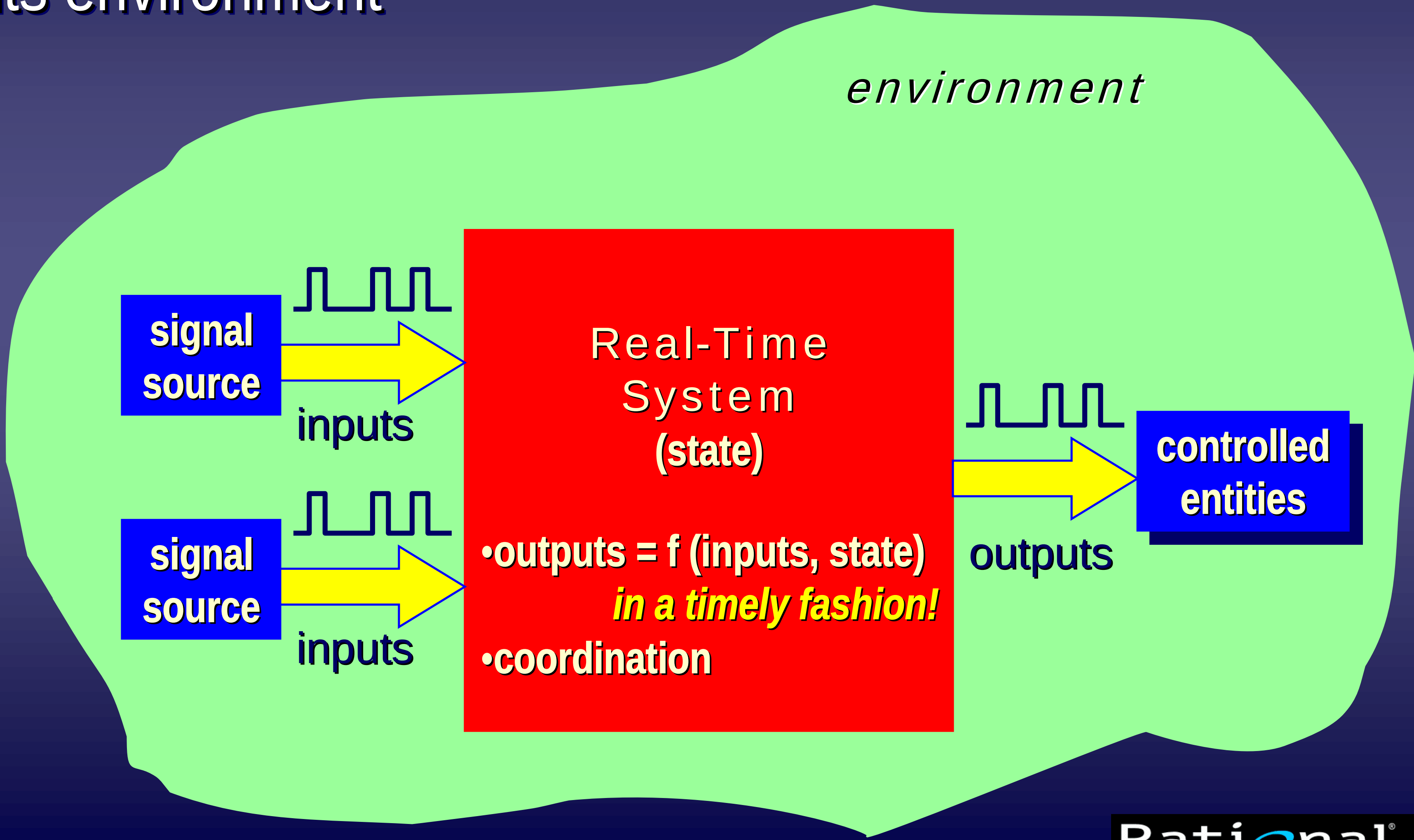
- Real-Time System Characteristics
- The Logical and the Engineering Viewpoints
- The Standard Real-Time UML Profile
  - UML Extensibility
  - Foundation
  - Modeling Time
  - Modeling Concurrency

Common Wisdom...

- When designing software, we are instructed to ignore details of the technology and similar “implementation” issues until we have a sound logical solution to the problem
  - simplifies the design problem (separation of concerns)
  - software is portable to new/different technologies
- But, what about real-time systems?

Real-Time System

- Systems that maintain an *ongoing timely* interaction with its environment



Rational
the e-development company™

Common Aspects of All RT Design

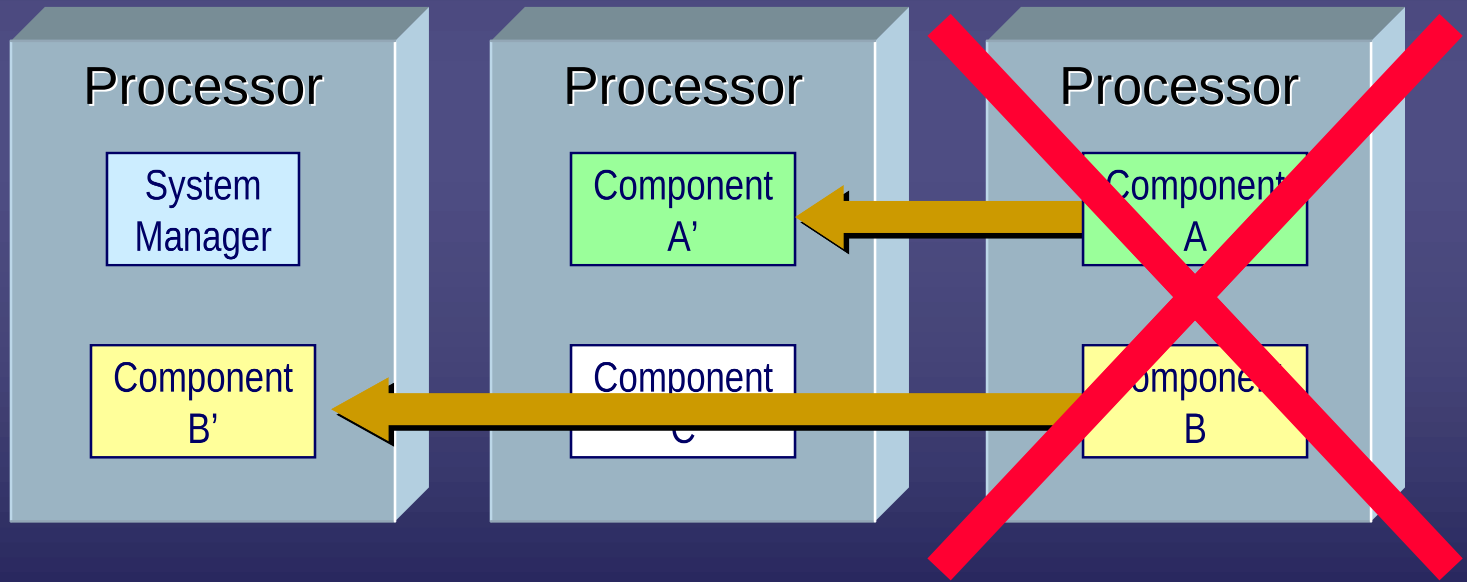
- *The quantitative aspect is significant*
  - Time sensitive (metric view of time is needed)
  - Resource sensitive (memory size, channel capacity)
- *Concurrency is an inherent part of the problem*
  - The real world is concurrent
  - Concurrent activities need to be coordinated
  - Concurrent activities may be asynchronous (non-deterministic)

Complex RT Systems

- Real-time systems with requirements for:
  - supporting complex functionality
  - high dependability (availability, reliability, safety)
  - distribution
  - heterogeneity
  - evolvability
  - scalability
- *While we now have pre-packaged solutions for many categories of small-scale real-time systems, we are only starting to learn how to construct complex real-time systems*

Fault-Tolerance

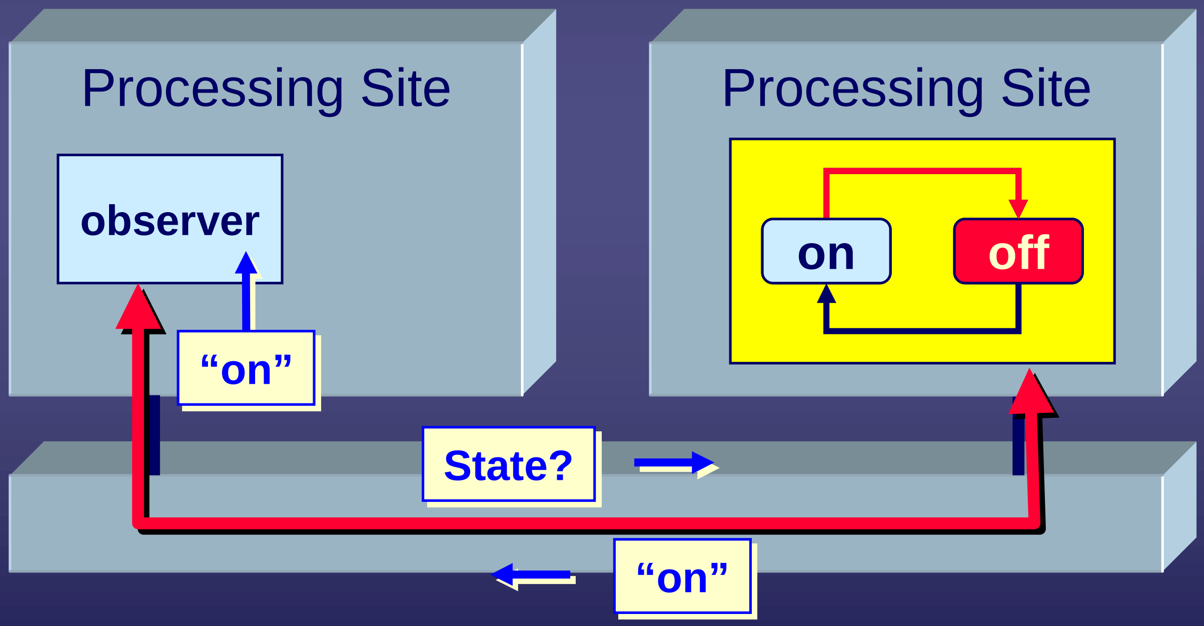
- Example: using spare processor capacity
  - failure detection
  - fault diagnosis
  - failure recovery



- *Optimal recovery strategies may be based on current needs and state of the application!*

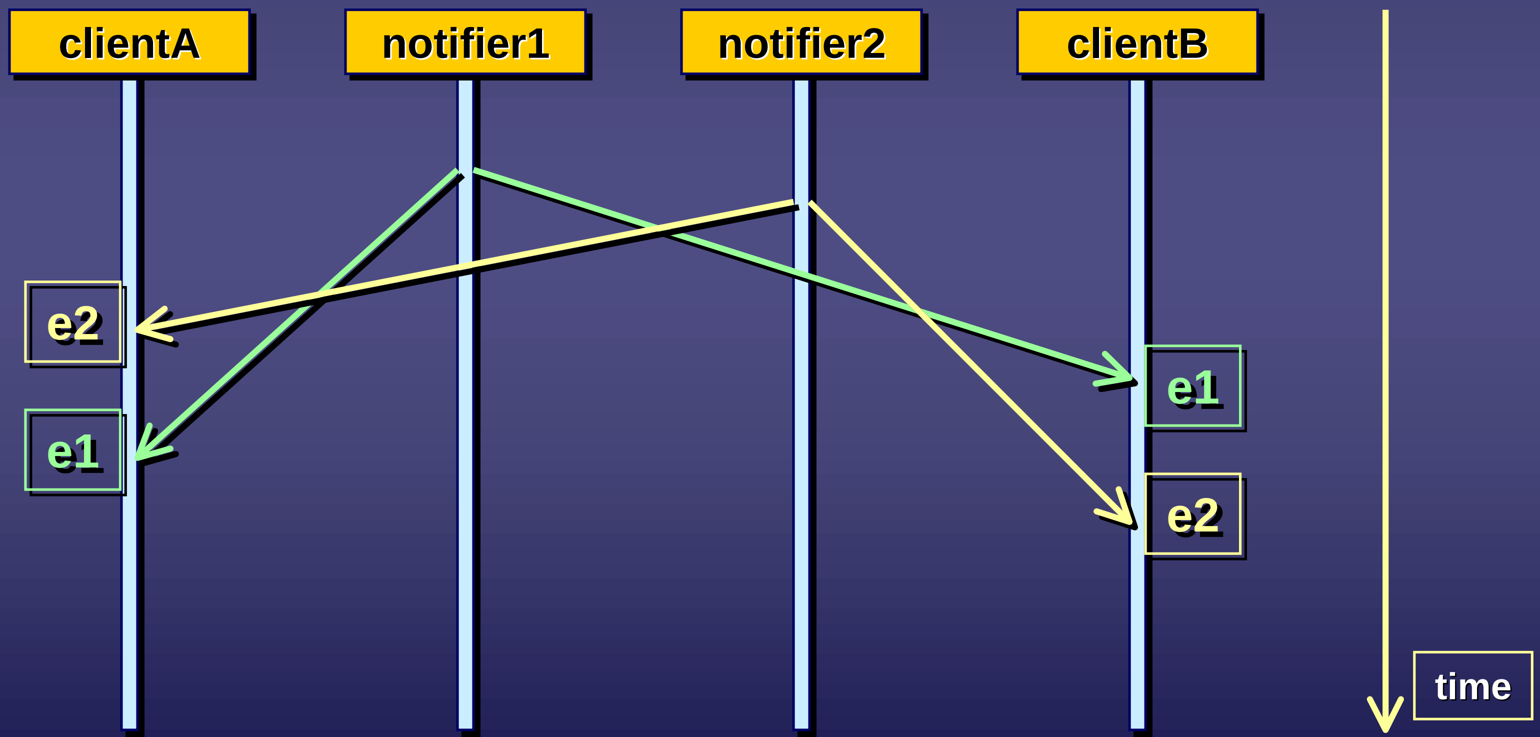
Distributed System Problems (1)

- Possibility of out-of-date state information due to lengthy (and variable) transmission delays



Distributed System Problems (2)

- Inconsistent views of system state:
  - different observers see different event orderings



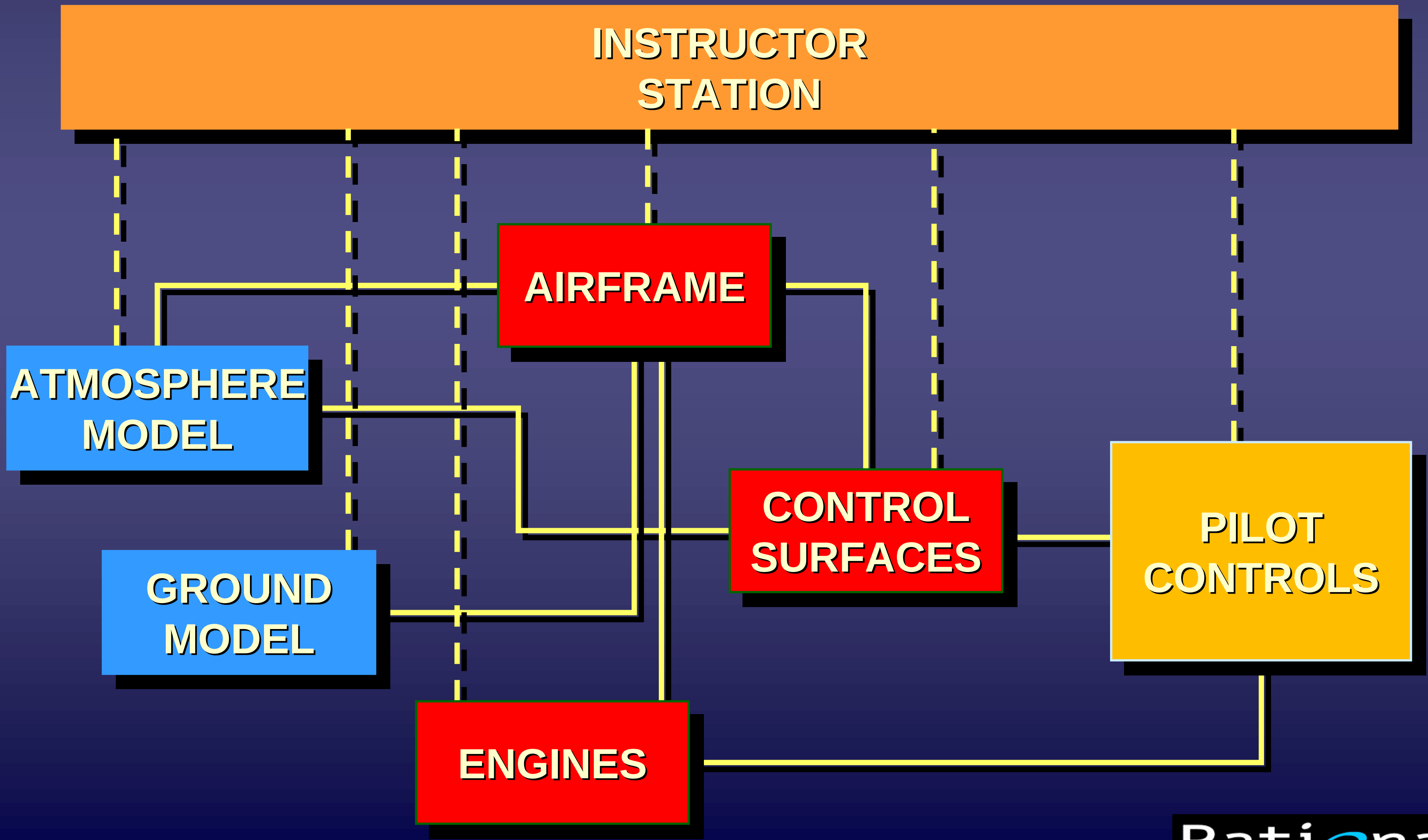
Complex Real-Time Design Issues

- Much of the complexity associated with these systems is the result of the “intrusion” of the inherently complex physical world into the logical world of software
- *The real-time design dilemma:*
  - if the physical world intrudes on the logical world, how can we separate the “logical” world of design from the “physical” world of implementation?

- Real-Time System Characteristics
- **The Logical and the Engineering Viewpoints**
- The Standard Real-Time UML Profile
  - UML Extensibility
  - Foundation
  - Modeling Time
  - Modeling Concurrency

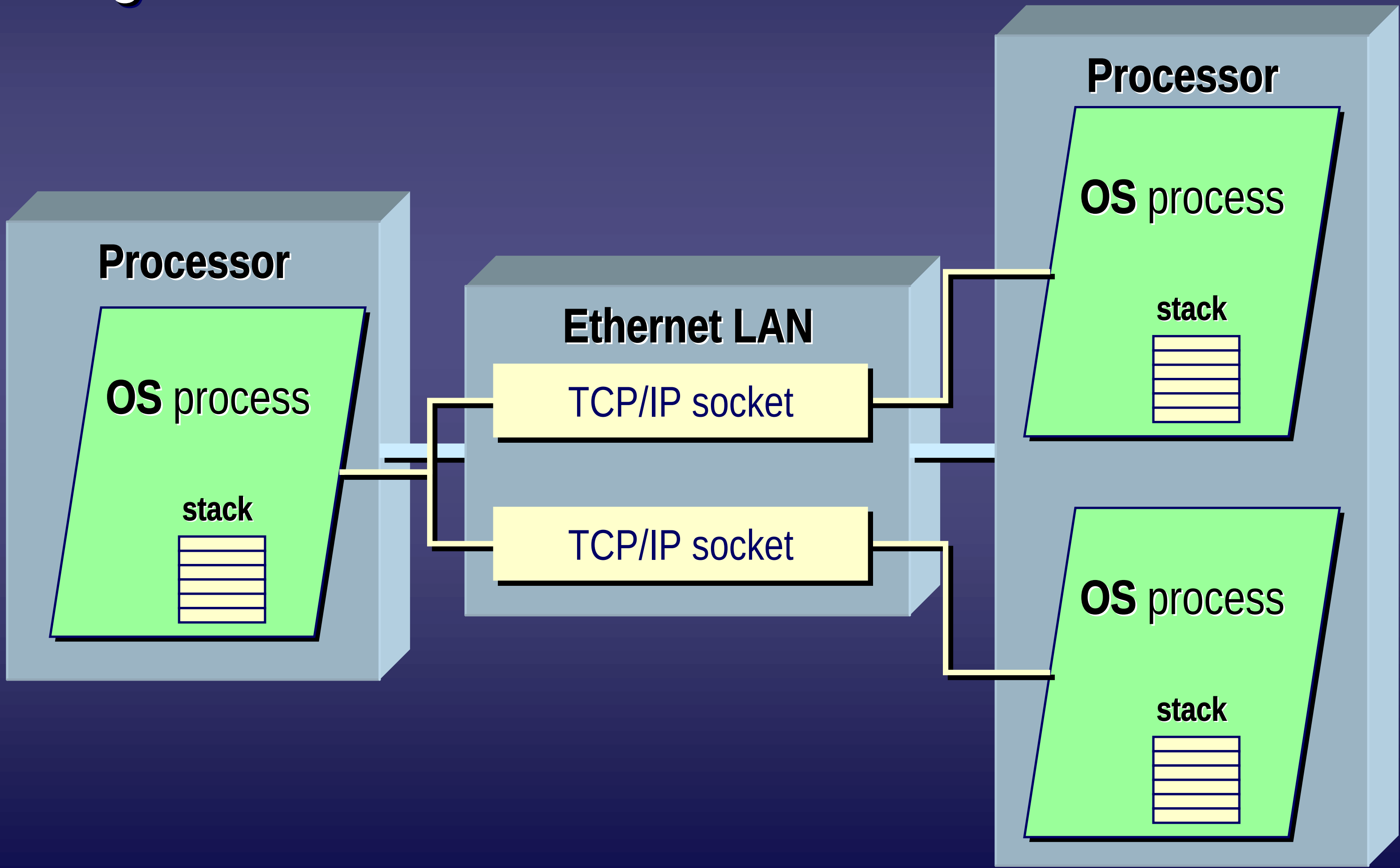
Logical (Functional) Viewpoint

- Network of collaborating "logical devices"
  - e.g., an aircraft simulator

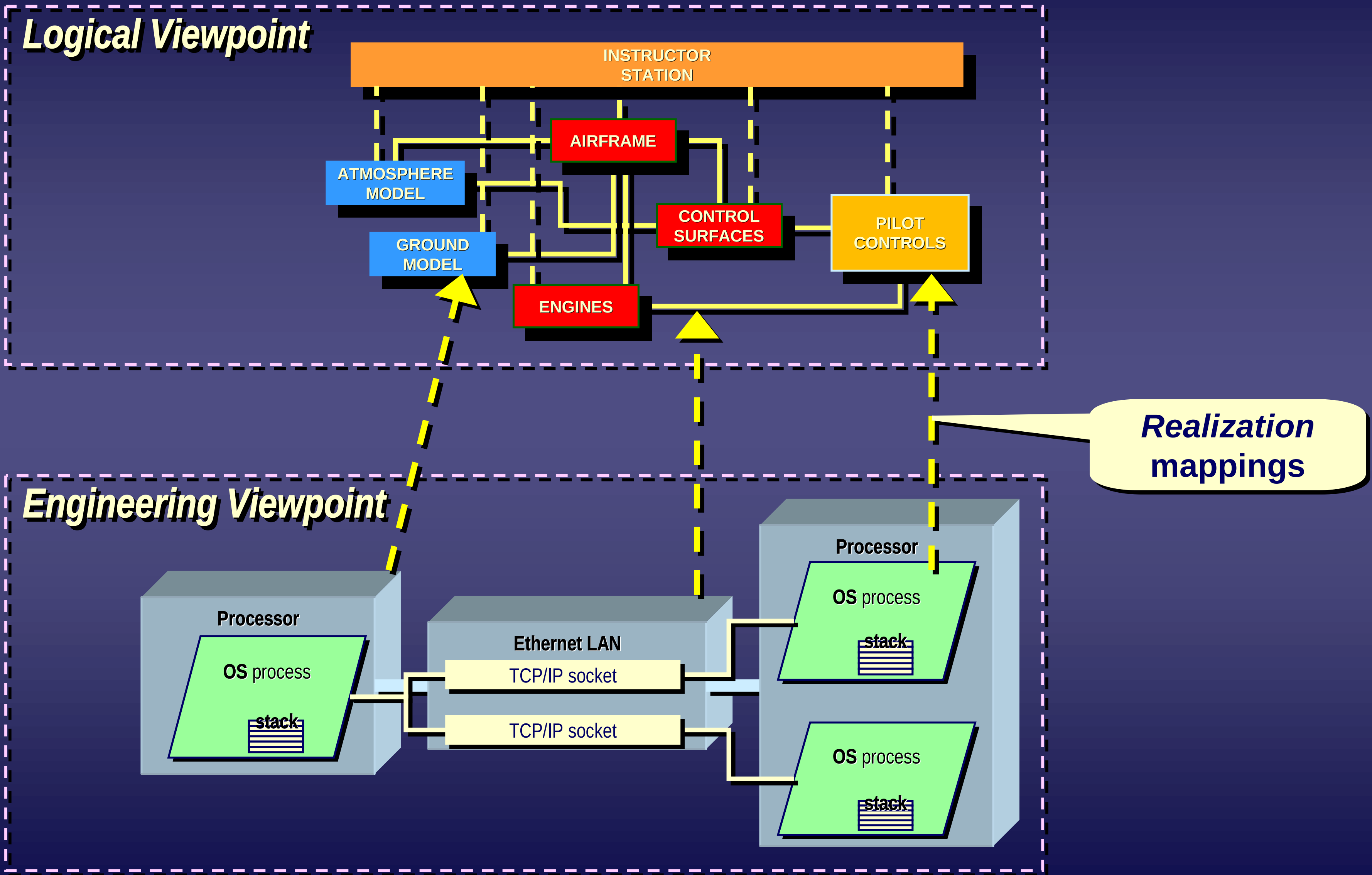


Engineering (Realization) Viewpoint

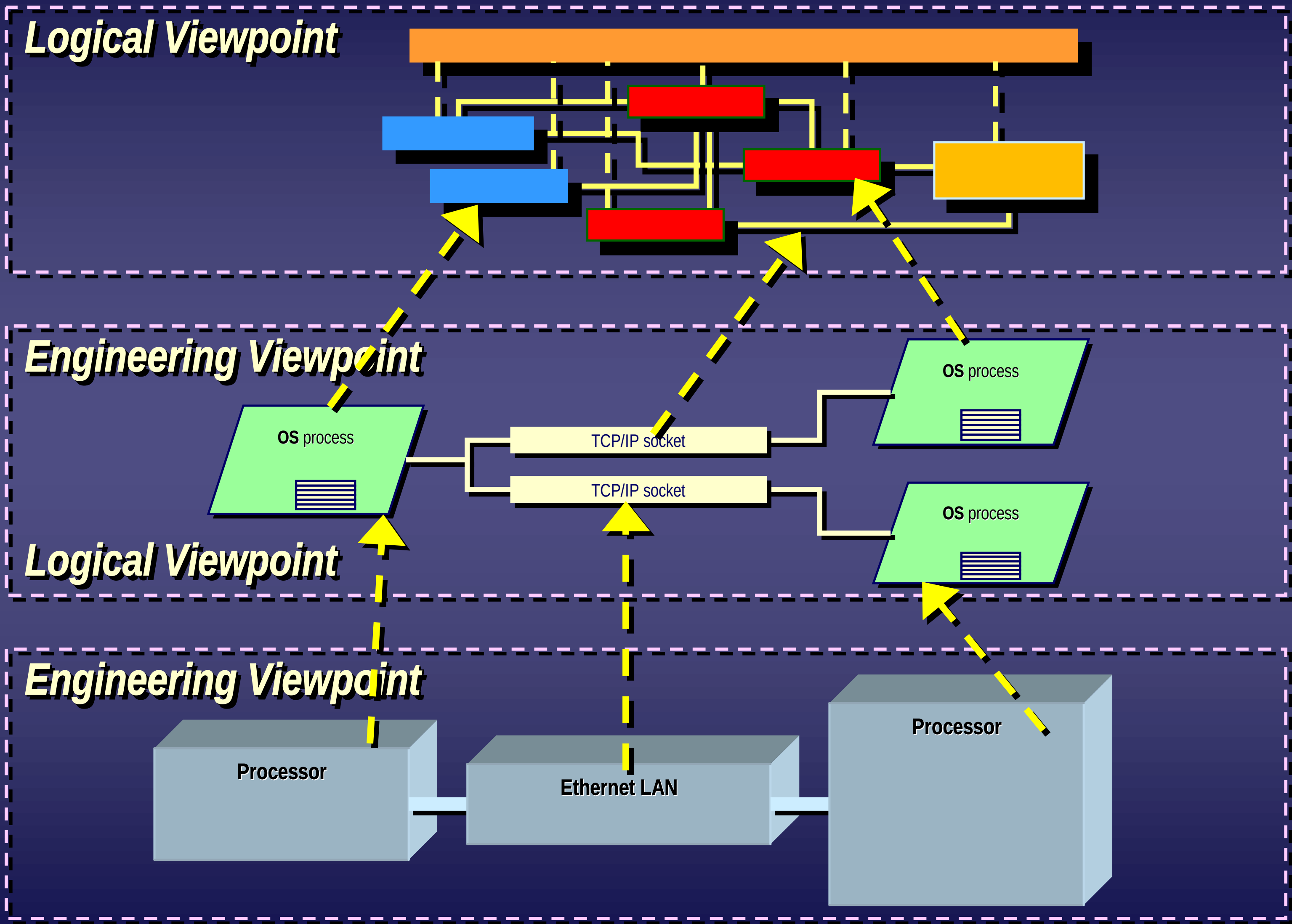
- The realization of a specific set of logical components using facilities of the run-time environment



Viewpoints and Mappings



Multi-Tier Realization Mappings



The Engineering Viewpoint in RT Systems

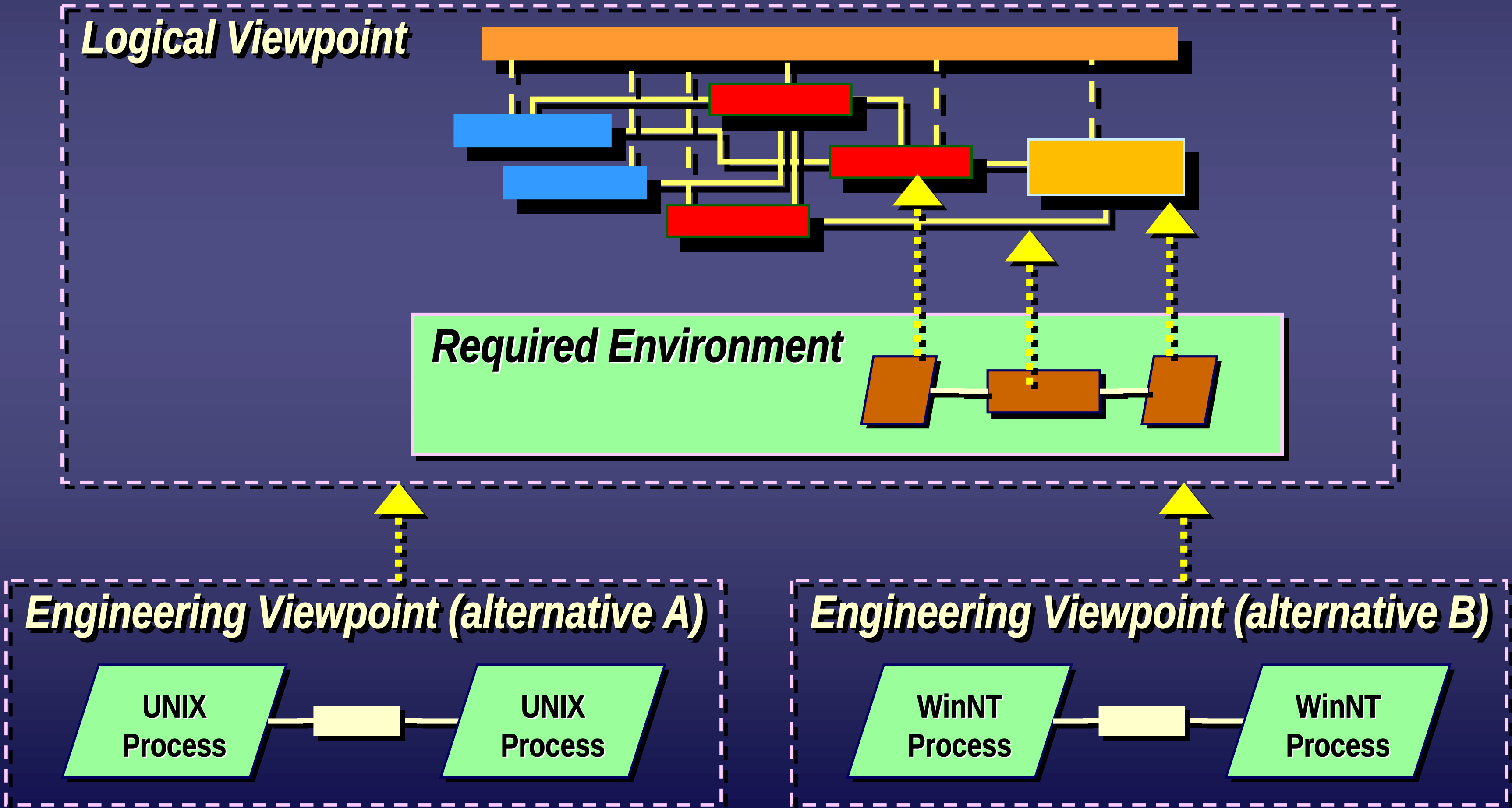
- The engineering viewpoint represents the “raw material” out of which we construct the logical viewpoint
  - the quality of the outcome is only as good as the quality of the ingredients that are put in
  - as in all true engineering, the quantitative aspects are often crucial (How <u>long</u> will it take? How <u>much</u> will be required?…)

Engineering-Oriented Design

- Dilemma: *How can we account for the engineering aspects of the system without prematurely and possibly unnecessarily committing to a particular technology?*
- Approach: *Provide an abstract technology-independent but quantified representation of the essential characteristics of the engineering viewpoint as part of the logical viewpoint*
  - use of quantitative methods and predictive models

Viewpoint Separation

- *Required Environment:* a technology-neutral environment specification required by the logical elements of a model



Rational
the e-development company

Required Environment

- Specifies a domain in which certain engineering properties apply:
  - failure characteristics (failure modes, availability, reliability)
  - CPU speeds
  - communications characteristics (delay, throughput, capacity)
  - etc.
- We need
  - A standardized means to specify these properties
  - A means to compare these properties against those of a concrete engineering environment

Quality of Service Concepts

- An abstract, technology-independent representation of the engineering model can be specified using the general concept of *Quality of Service (QoS):*

  a quantitative specification of how some service is performed

  - e.g. throughput, capacity, response time, service policy
- Two sides to QoS specifications:
  - *offered QoS:* the QoS that a server provides to its clients
  - *required QoS:* the QoS that a client requires from a server

Quantitative Methods for RT Systems

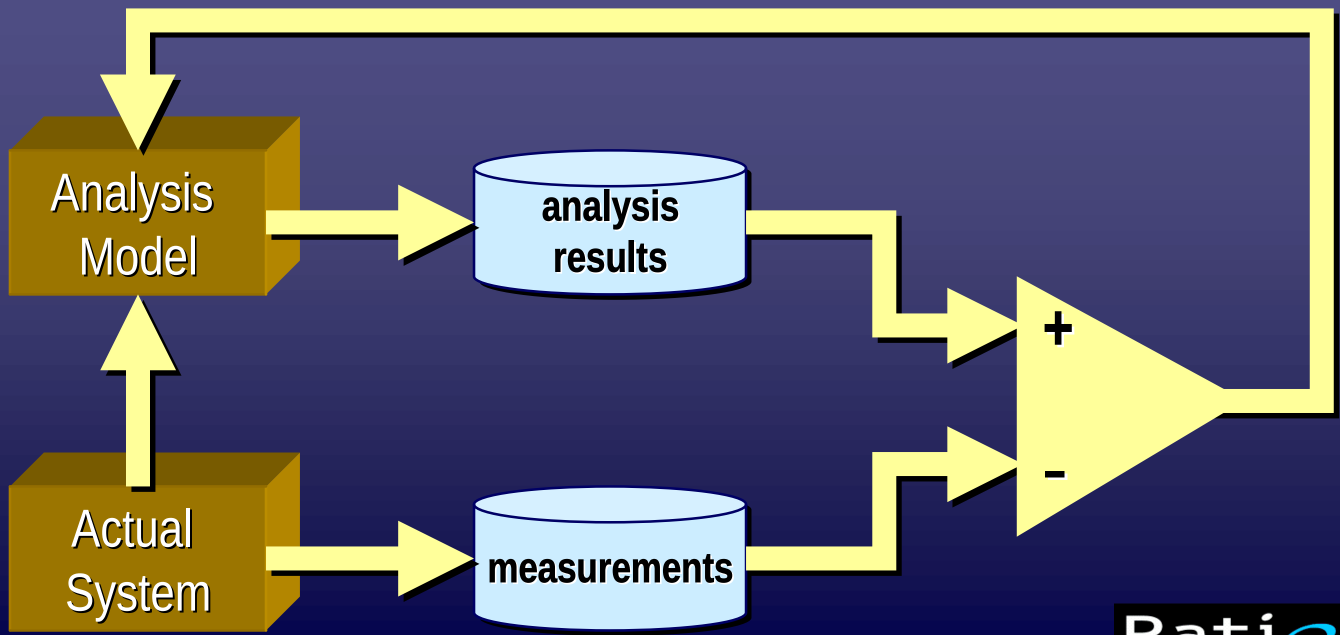
- Once we have included QoS information in our models, we can use *quantitative methods* to formally:
  - predict system characteristics (detect problems early)
  - analyze existing system
  - synthesize the desired system
- Current real-time quantitative method types:
  - Schedulability analysis

    will the system meet all of its deadlines?
  - Performance analysis based on queueing theory

    what kind of response will the system have under load?

Issues with Quantitative Methods

- Require uncommon and highly-specialized skills
- Software is notoriously difficult to model mathematically
  - highly non-linear (detail often matters)
  - models are frequently severely inaccurate
  - typical modeling process is highly manual:



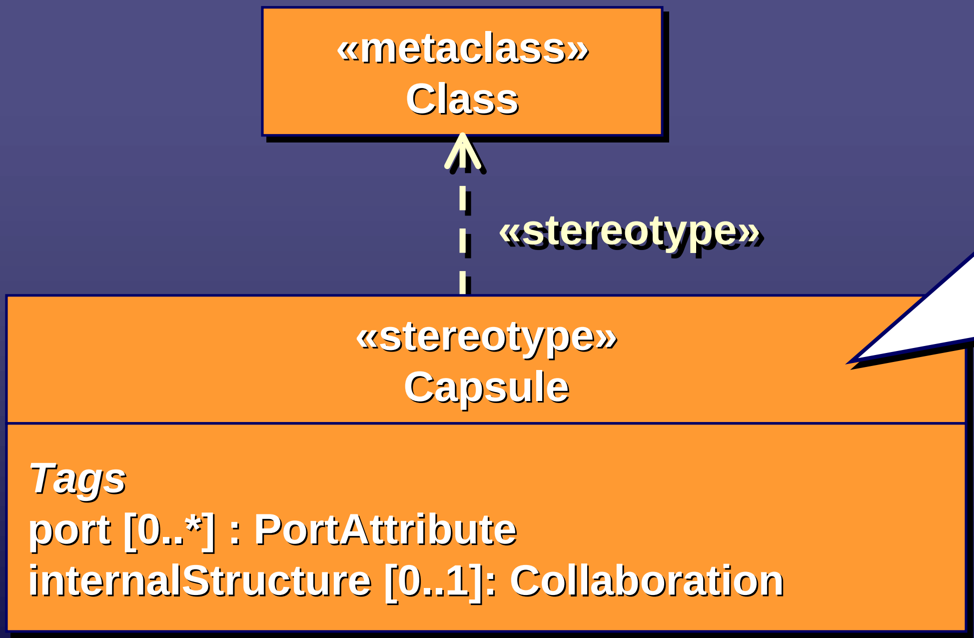
- Real-Time System Characteristics
- The Logical and the Engineering Viewpoints
- The Standard Real-Time UML Profile
  - UML Extensibility
  - Foundation
  - Modeling Time
  - Modeling Concurrency

Semantic Variation in UML

- Semantic aspects that are:
  - undefined (e.g., scheduling discipline), or
  - intentionally ambiguous (multiple interpretations)
- Why?
  - Different domains require different specializations
  - The applicability and usefulness of UML would have been severely constrained if it could not support such diversity
- Standard UML can be used as the common conceptual base for a family of languages
- Mechanisms for specifying semantic refinements: stereotypes, tagged values, constraints

UML Stereotypes

- Specializations of “base” UML modeling concepts (Class, Association, Attribute, etc.)
- Example: a specialization of the UML Class concept
  - Graphical definition style (UML 1.4)



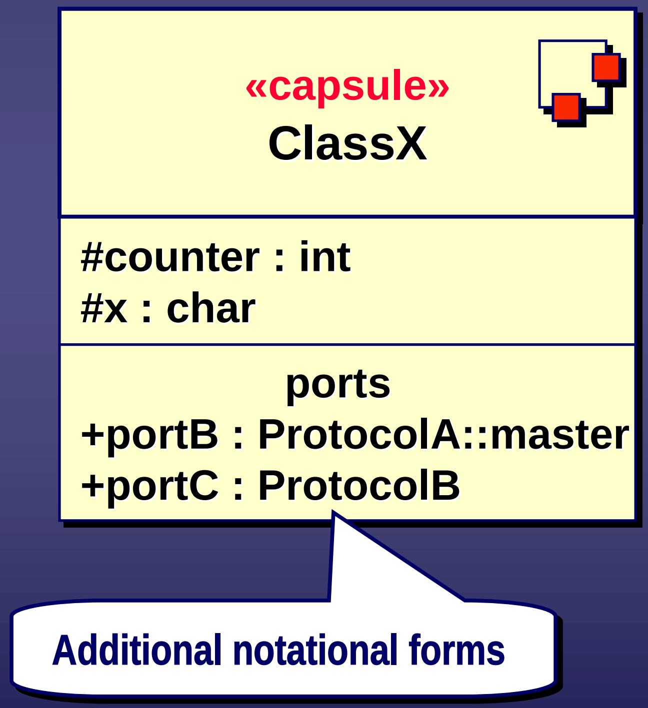
Additional semantic constraints

All instances of this class:

- **are concurrent (isActive = true)**
- **only attributes that are stereotyped as «ports» can be public**
- **cannot have public operations**
- **all attributes that are stereotyped as «capsules» and «connectors» are instantiated automatically with the instantiation of the object**

UML Stereotypes: Example

- An instance of the “capsule” stereotype in a class diagram:



UML Profiles

- A package of related specializations of general UML concepts that capture domain-specific variations and usage patterns
  - *A domain-specific interpretation of UML*
- Fully conformant with the UML standard
  - additional semantic constraints cannot contradict the general UML semantics
  - within the "semantic envelope" defined by the standard

- Real-Time System Characteristics
- The Logical and the Engineering Viewpoints
- **The Standard Real-Time UML Profile**
  - UML Extensibility
  - **Foundation**
  - Modeling Time
  - Modeling Concurrency

The Real-Time A&D Group in OMG

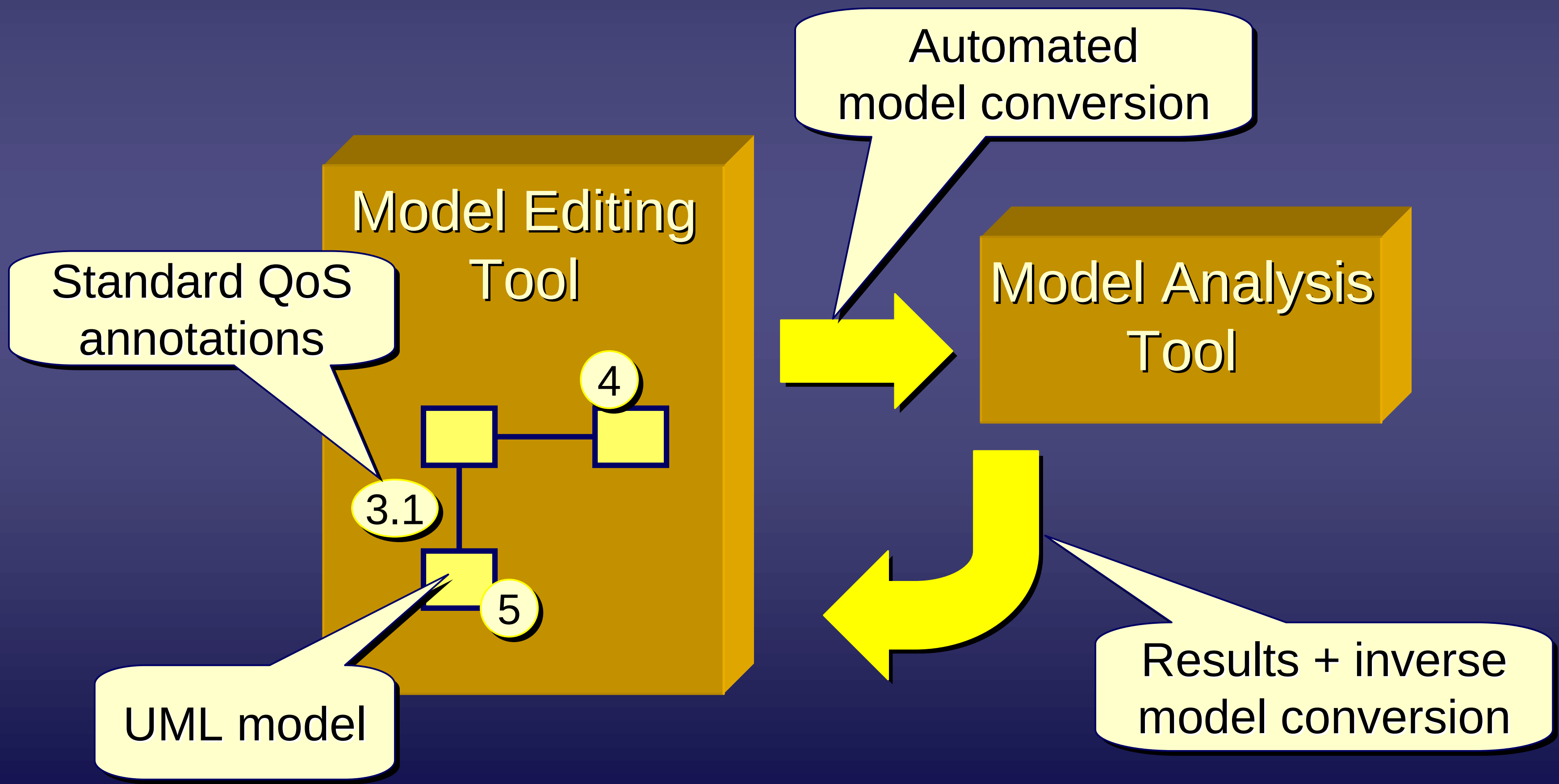
- An OMG working group (www.omg.org)
  - mission: to investigate and issue requests (RFPs) for standard ways and means to apply UML to real-time problems
- Three principal areas of investigation:
  - Time-related modeling issues (RFP.1: issued)
  - General quality of service modeling issues (RFP.2: pending)
  - Architectural modeling issues (RFP.3 ⇒ UML 2.0)
- RFP.1: "*UML profile for scheduling performance and time" (ad/99-03-13)*
  - initial submission August 2000 (ad/2000-08-04)
  - revised submission due June 2001

RT Profile: Guiding Principles

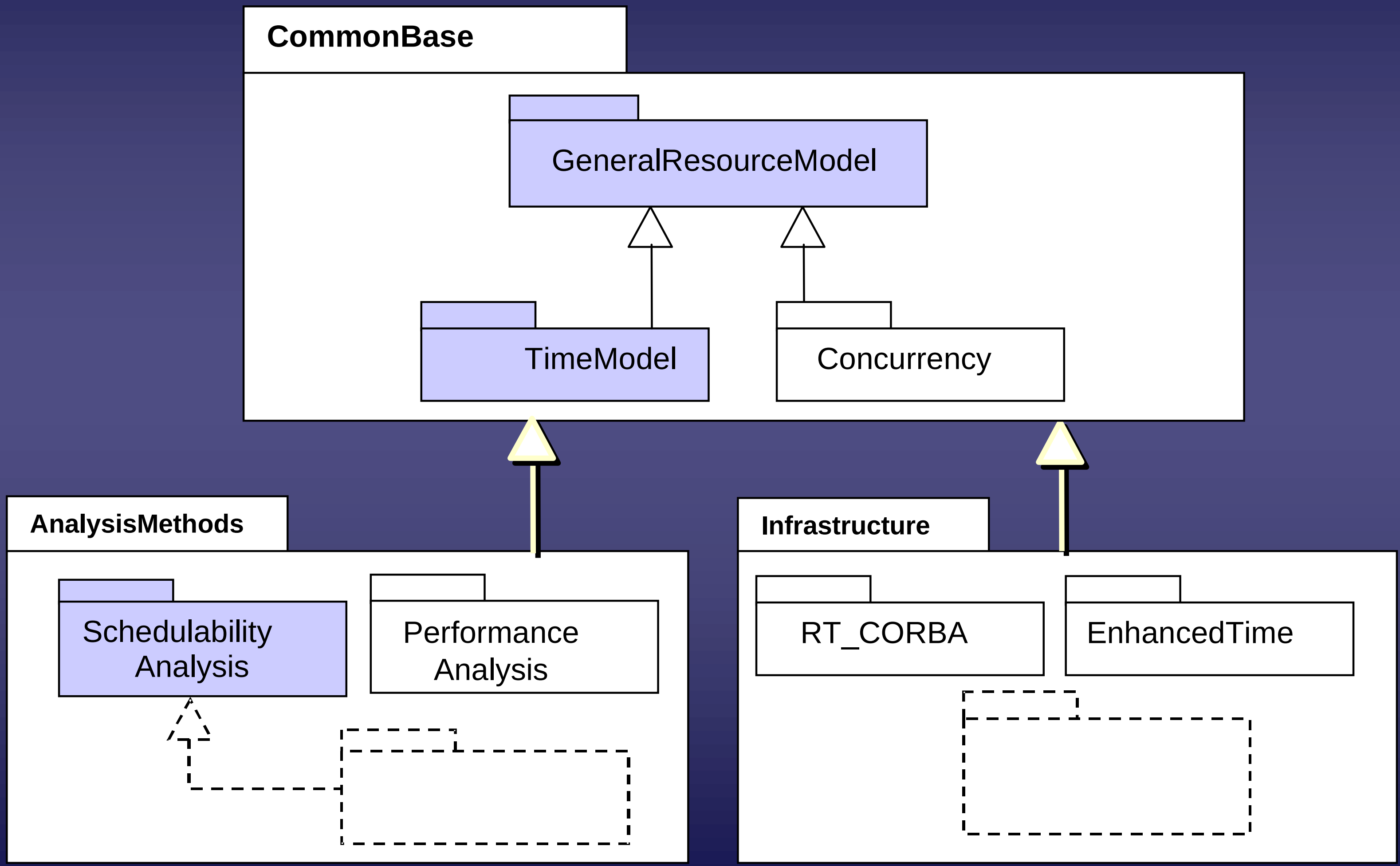
- Ability to specify quantitative information directly in UML models
  - key to quantitative analysis and predictive modeling
- Flexibility:
  - users can model their RT systems using modeling approaches and styles of their own choosing
  - open to existing and new analysis techniques
- Facilitate the use of analysis methods
  - eliminate the need for a deep understanding of analysis methods
  - as much as possible, automate the generation of analysis models and the analysis process itself

Desired Development Model

- Seamless integration of technologies and tools based on standards for real-time modeling

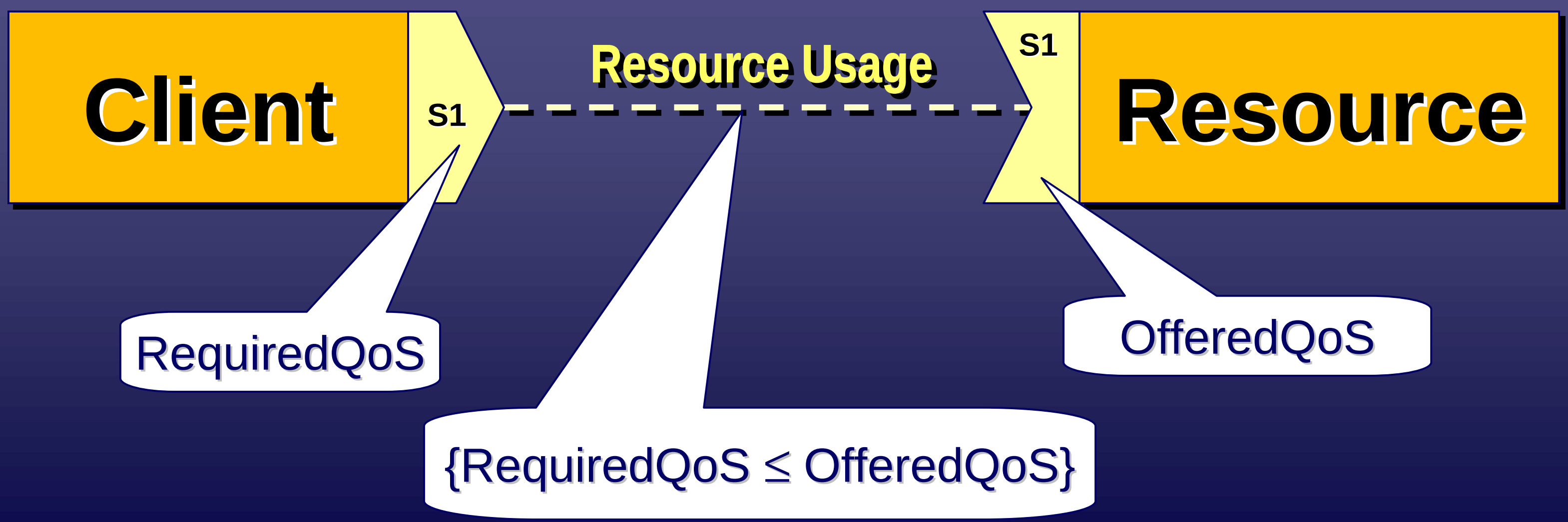


Profile Structure



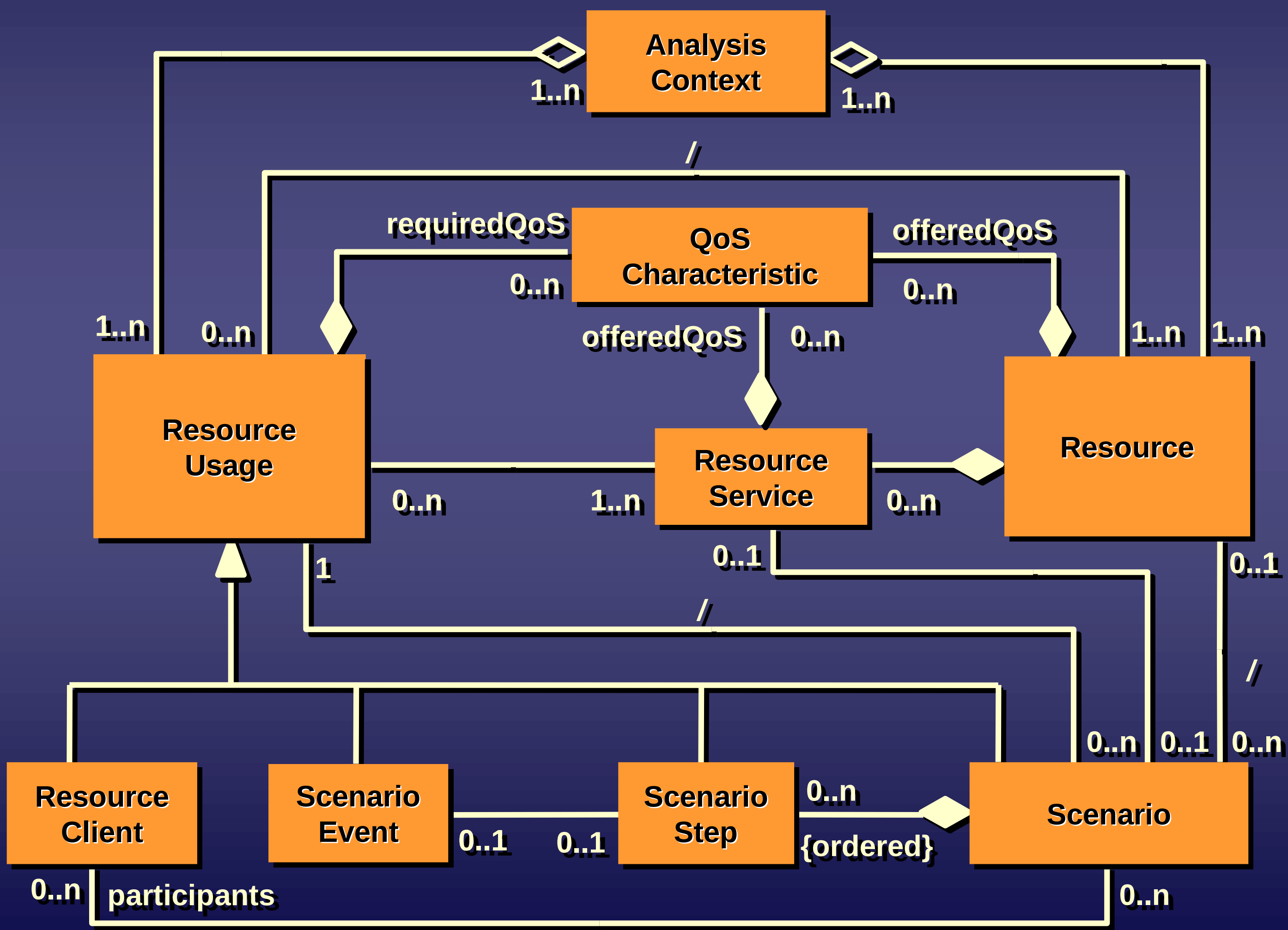
Resources and Quality of Service

- The characteristic of a finite (physical or logical) quantity is captured through the notion of a *resource*
- A resource provides a service characterized by one or more *quality of service (QoS)* attributes
  - capacity, reliability, availability, response time, etc.



The General Resource Modeling Framework

- A domain model (not a metamodel)



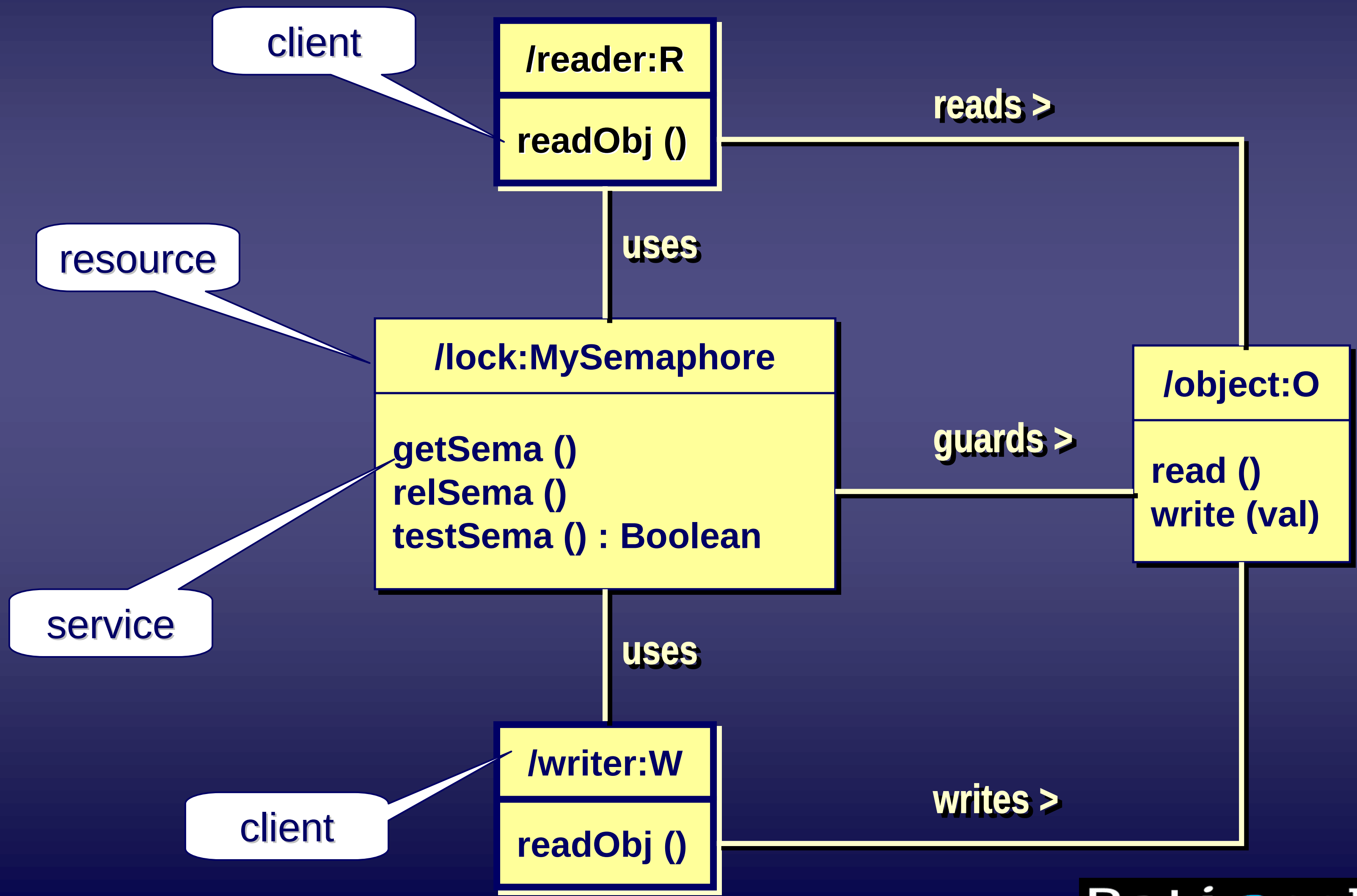
Rational the e-development company

Mapping to UML Models

- The general resource model is just a conceptual model that *unifies the basic abstractions from a variety of different time-oriented analysis methods*
- In a concrete UML model it can appear in many different application-specific forms
  - To perform analysis, analysis tools must be able to automatically detect these general forms
- This can be done using stereotypes that represent the abstractions of the general resource model

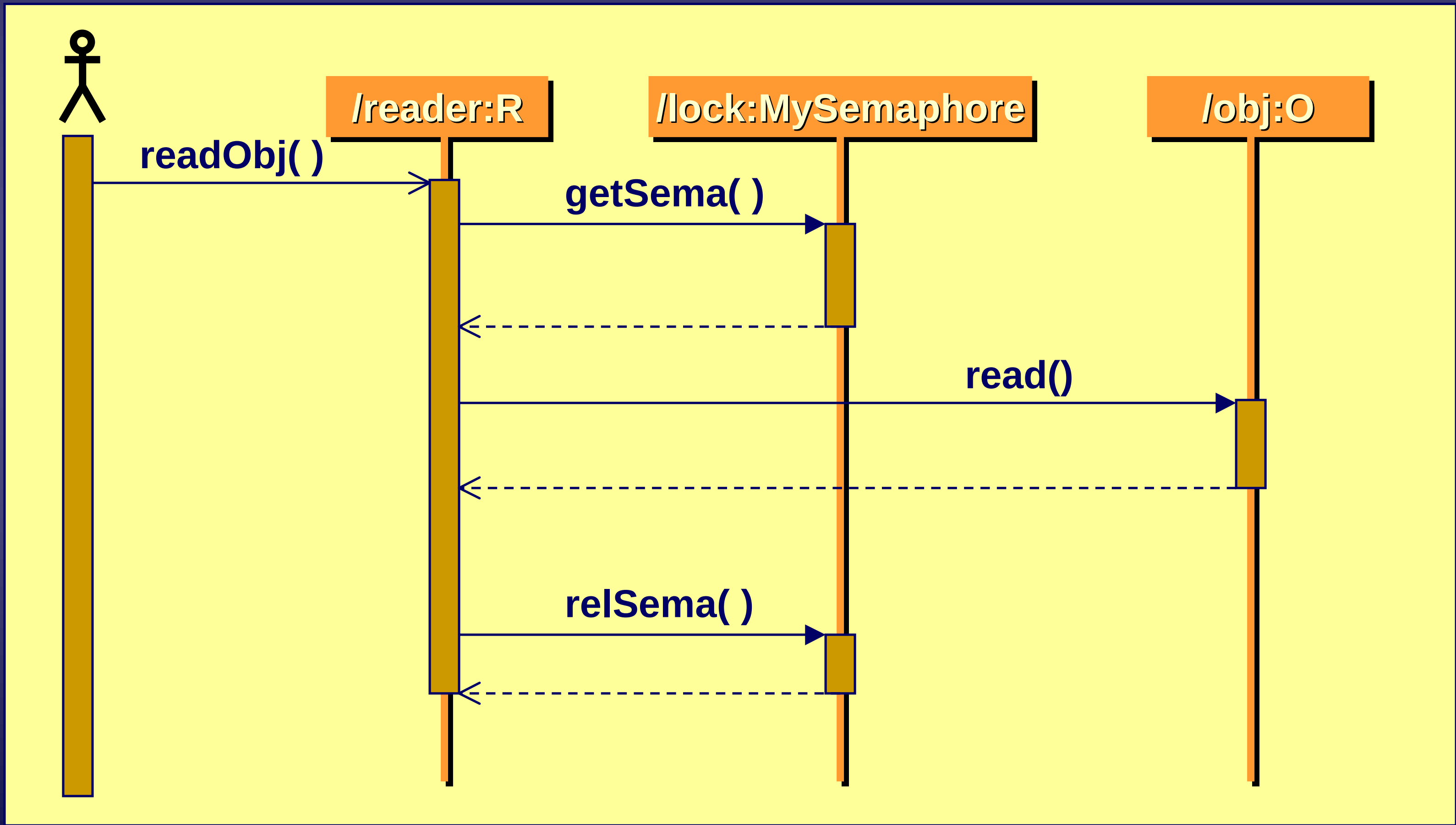
Typical Example (1 of 2)

- An object protected by a semaphore (collaboration spec)



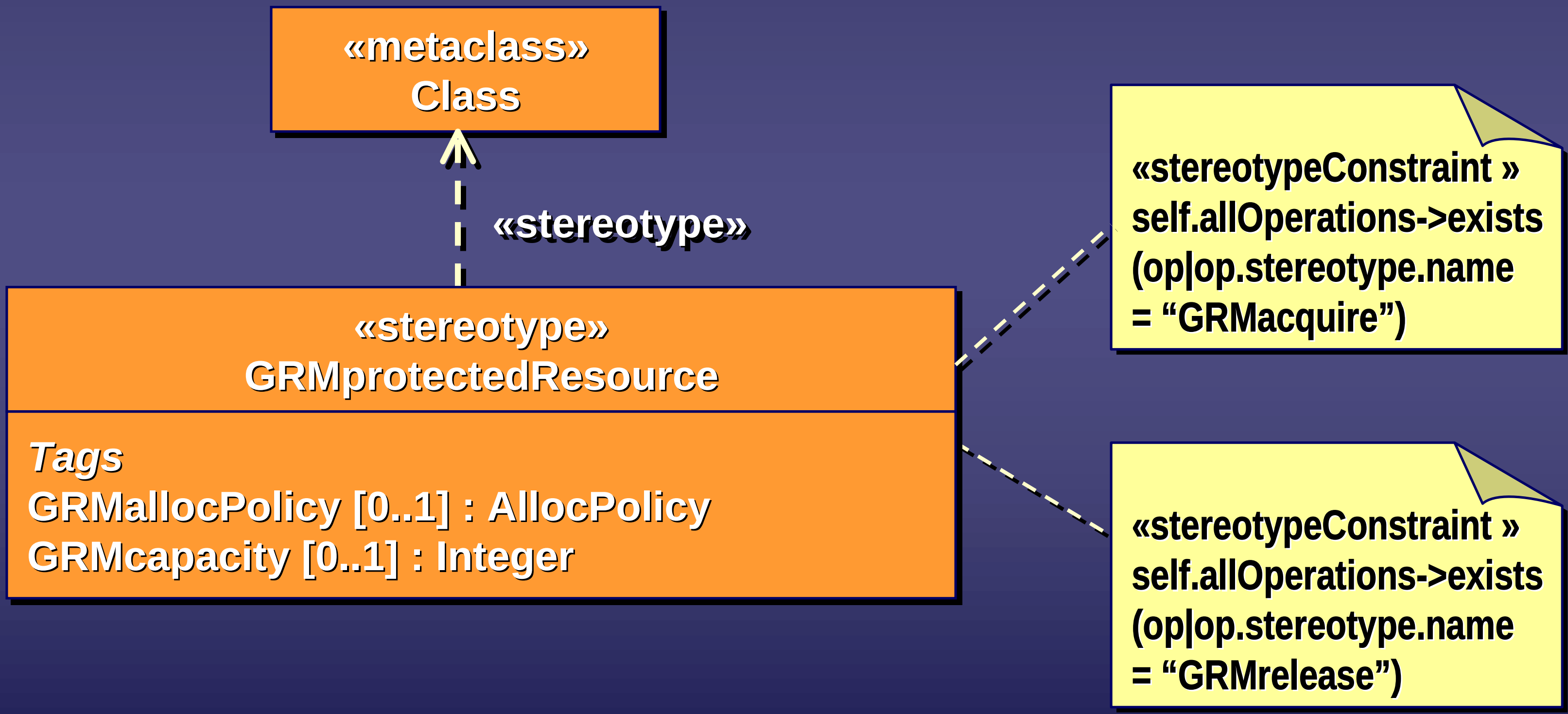
Typical Example (2 of 2)

- Interaction spec (sequence diagram)



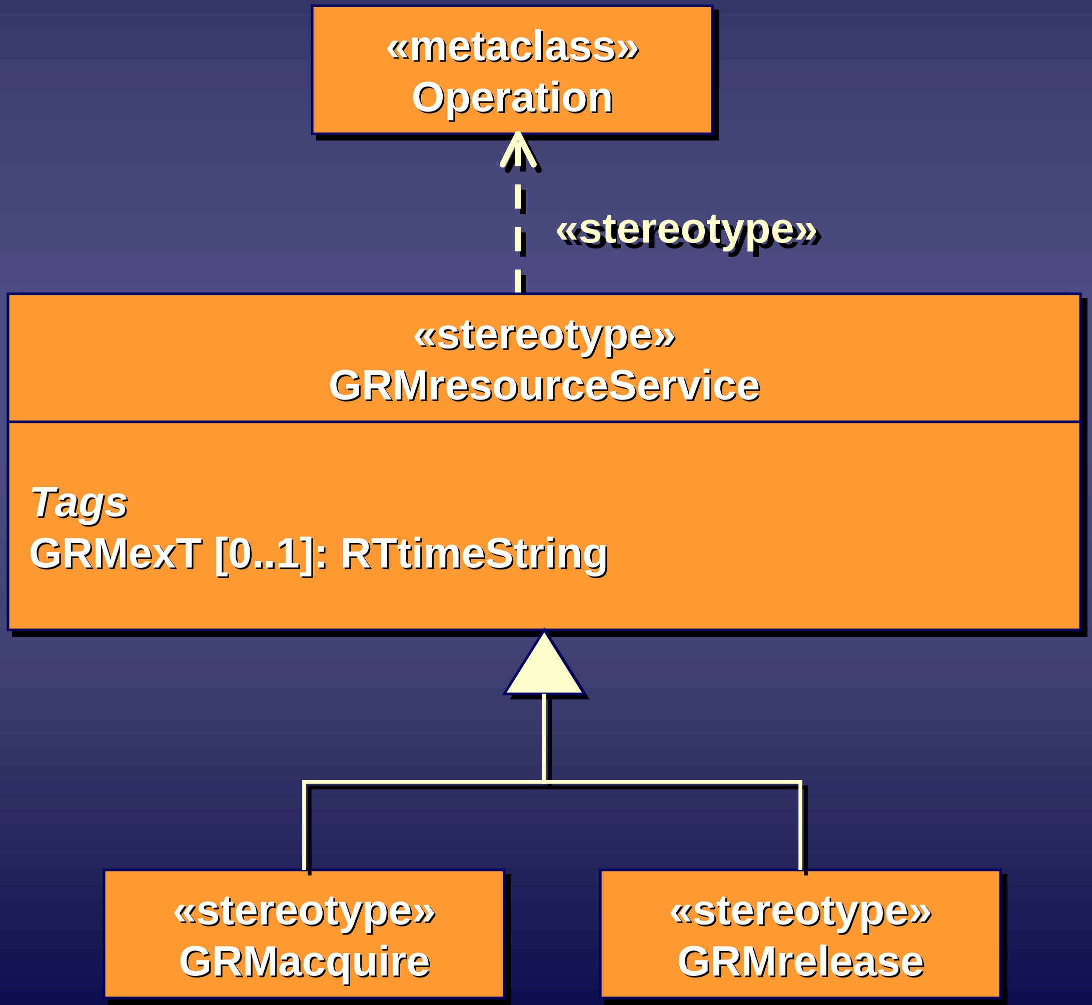
A Stereotype from the Real-Time Profile

- The real-time profile defines the concept of a “protected” resource derived from the generic UML concept of a Class (object)



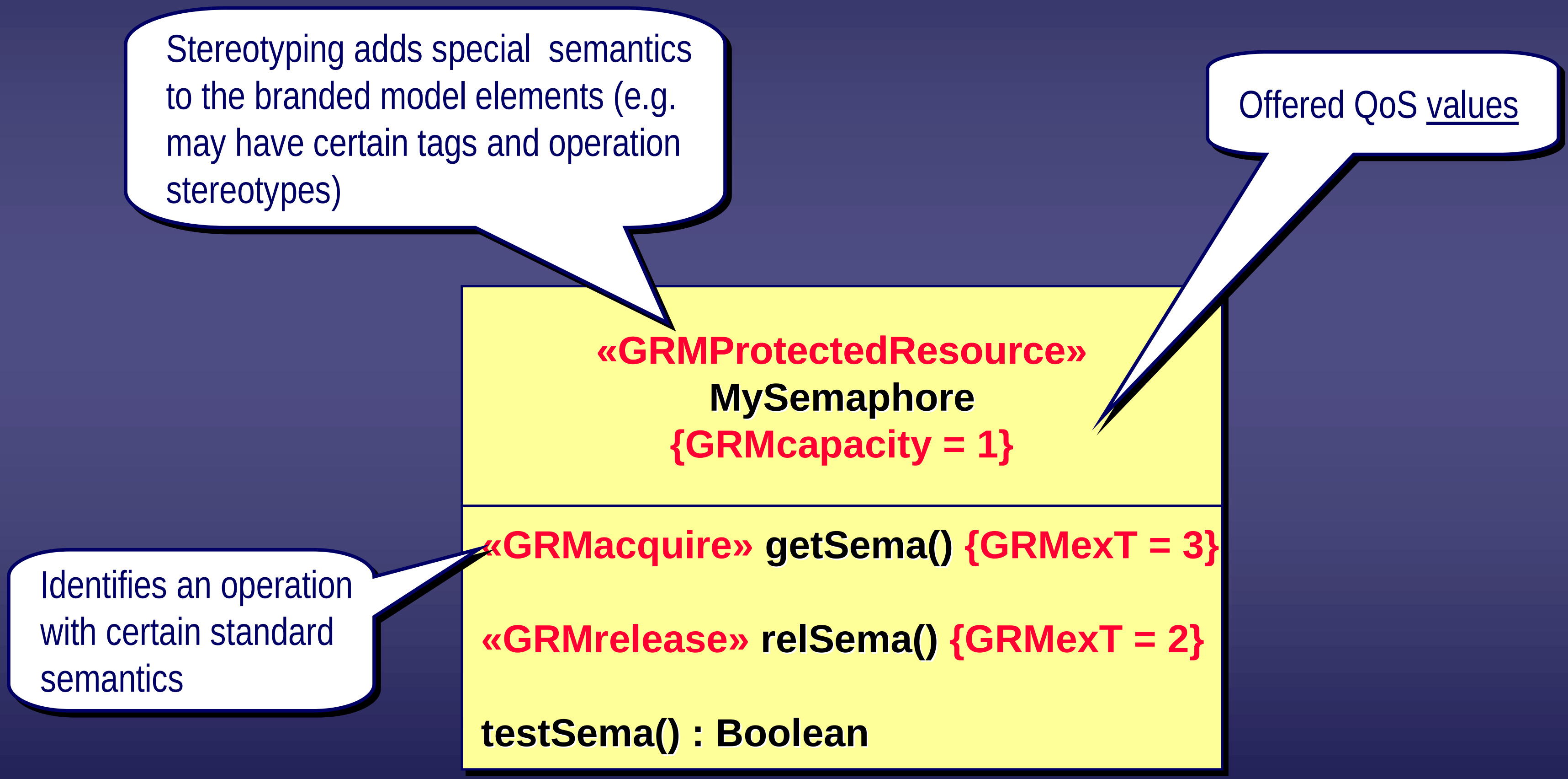
More Real-Time Stereotypes

- Concept of a timed operation (for QoS specs)



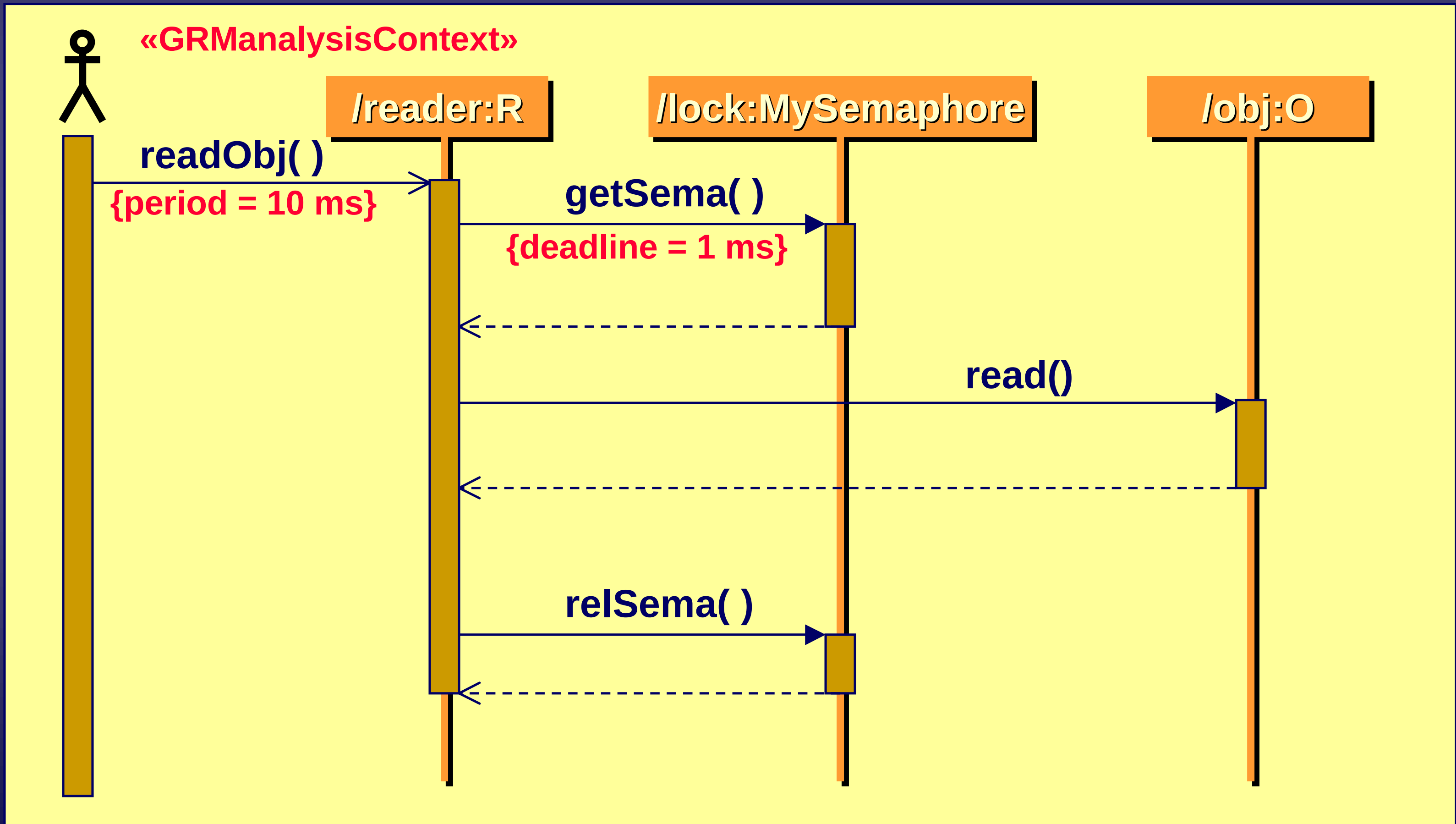
Using the RT Stereotypes

- Annotate the UML model



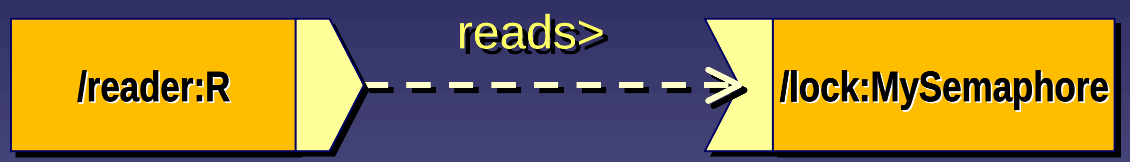
Usage Scenarios and Required QoS

- Usage scenario expressed as a sequence diagram

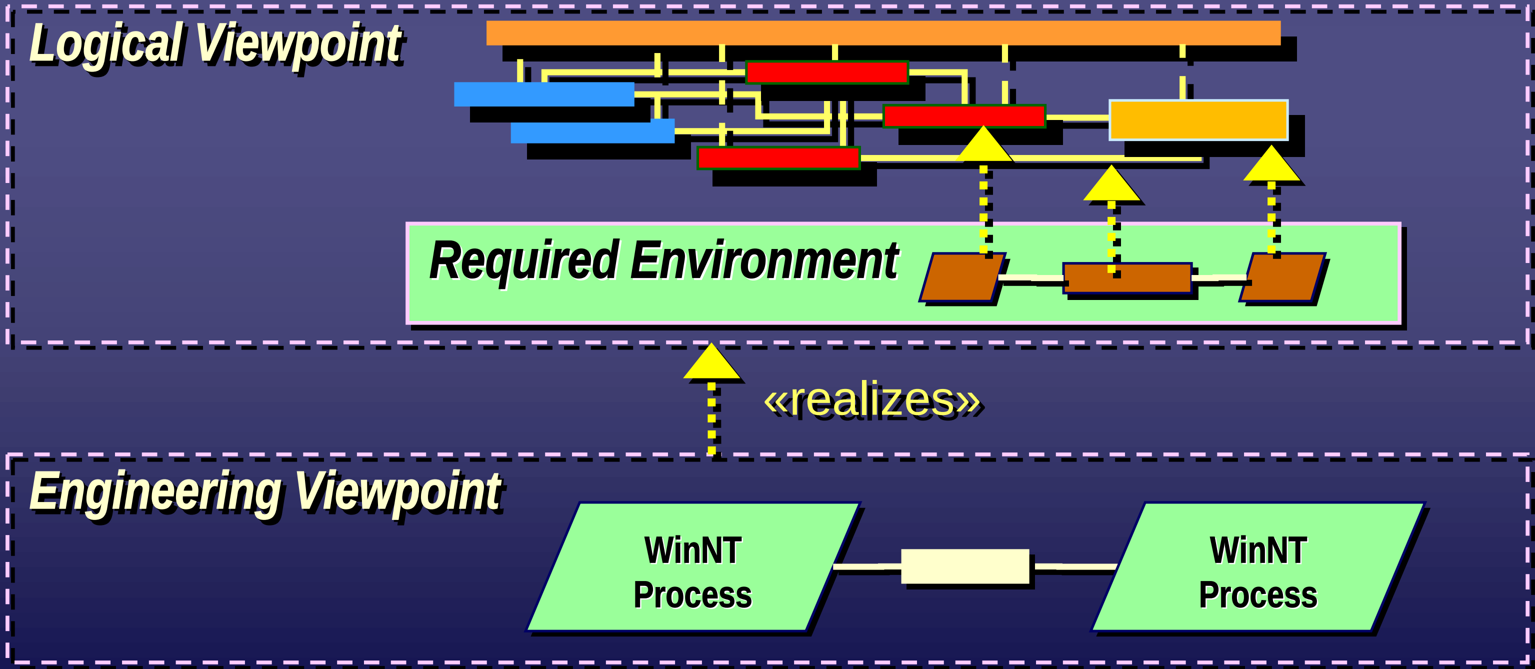


Two Interpretations of Resource Model

- The peer interpretation

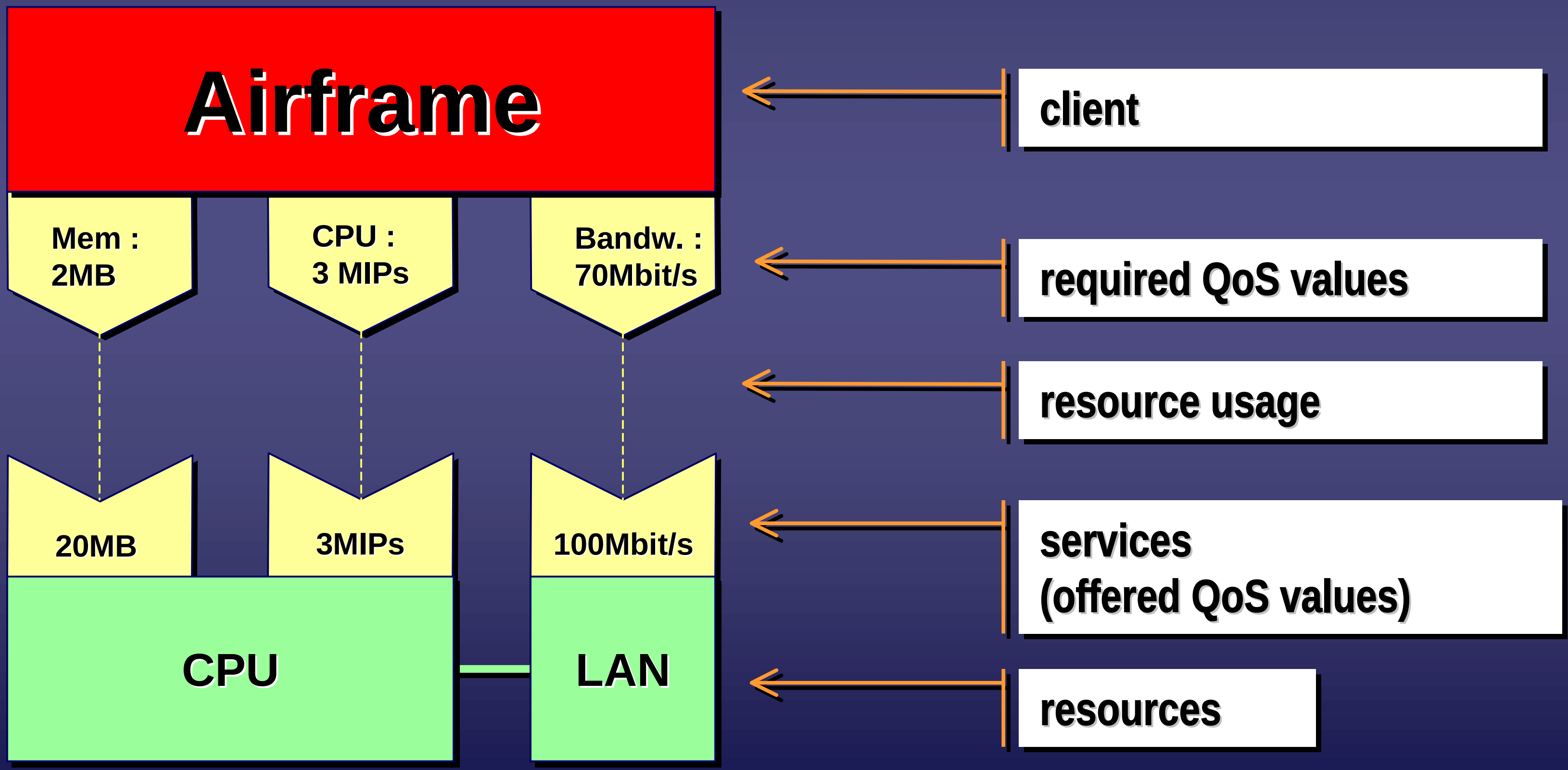


- The layered interpretation (the 2-viewpoint model)



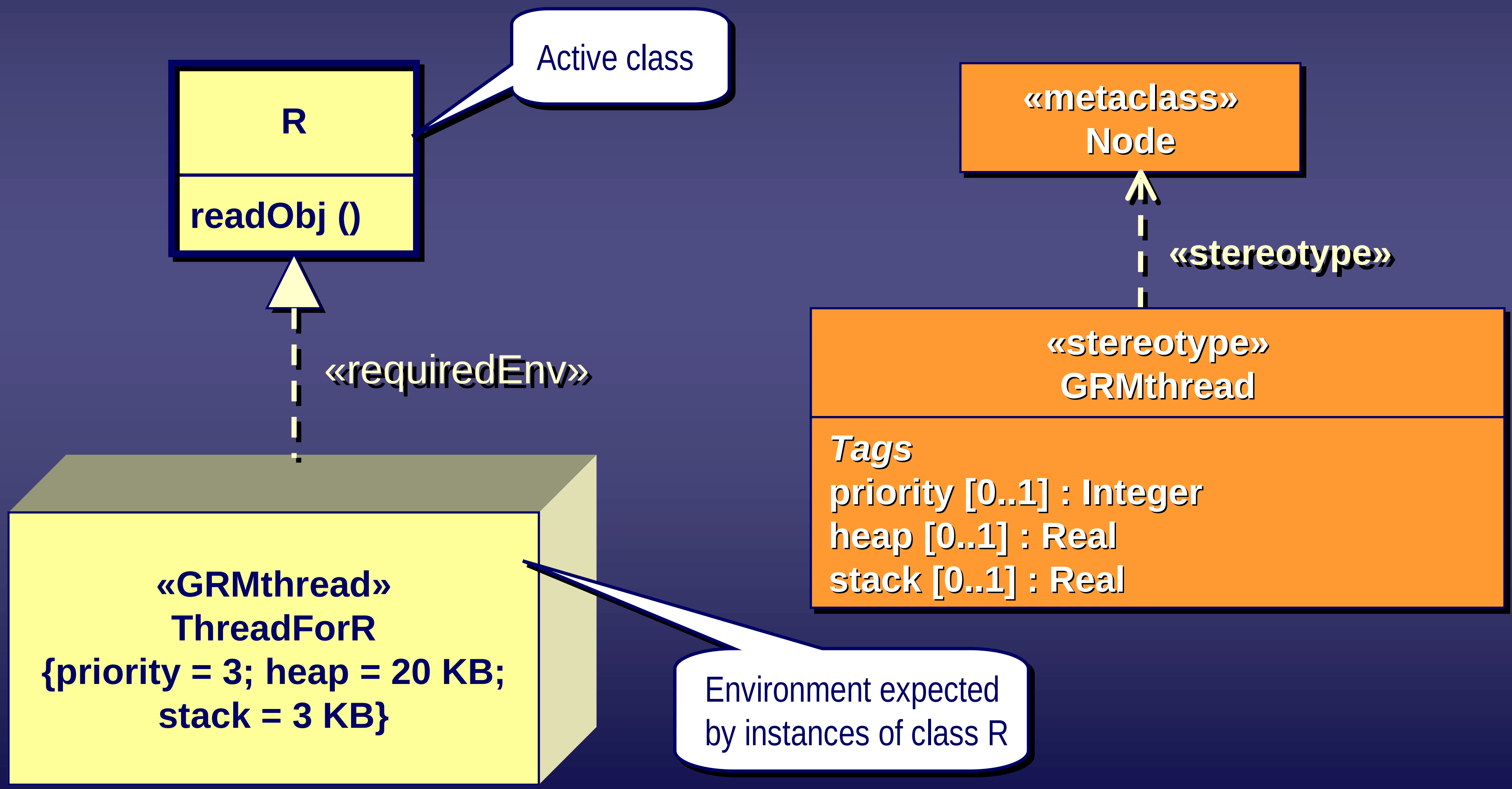
The Layered Interpretation

- A software task that requires a specific minimal operating environment



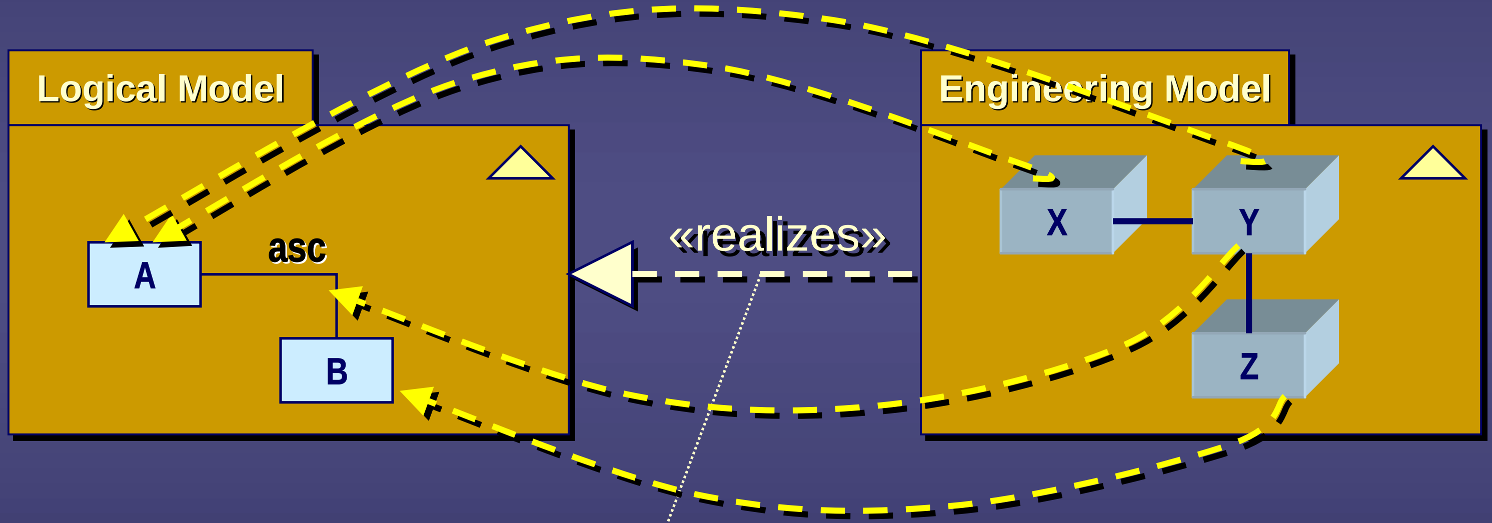
Specifying Required Environments in UML

- Using specializations of the UML Node concept



Modeling Realization in UML

- An association between models with explicit deployment mappings between model elements

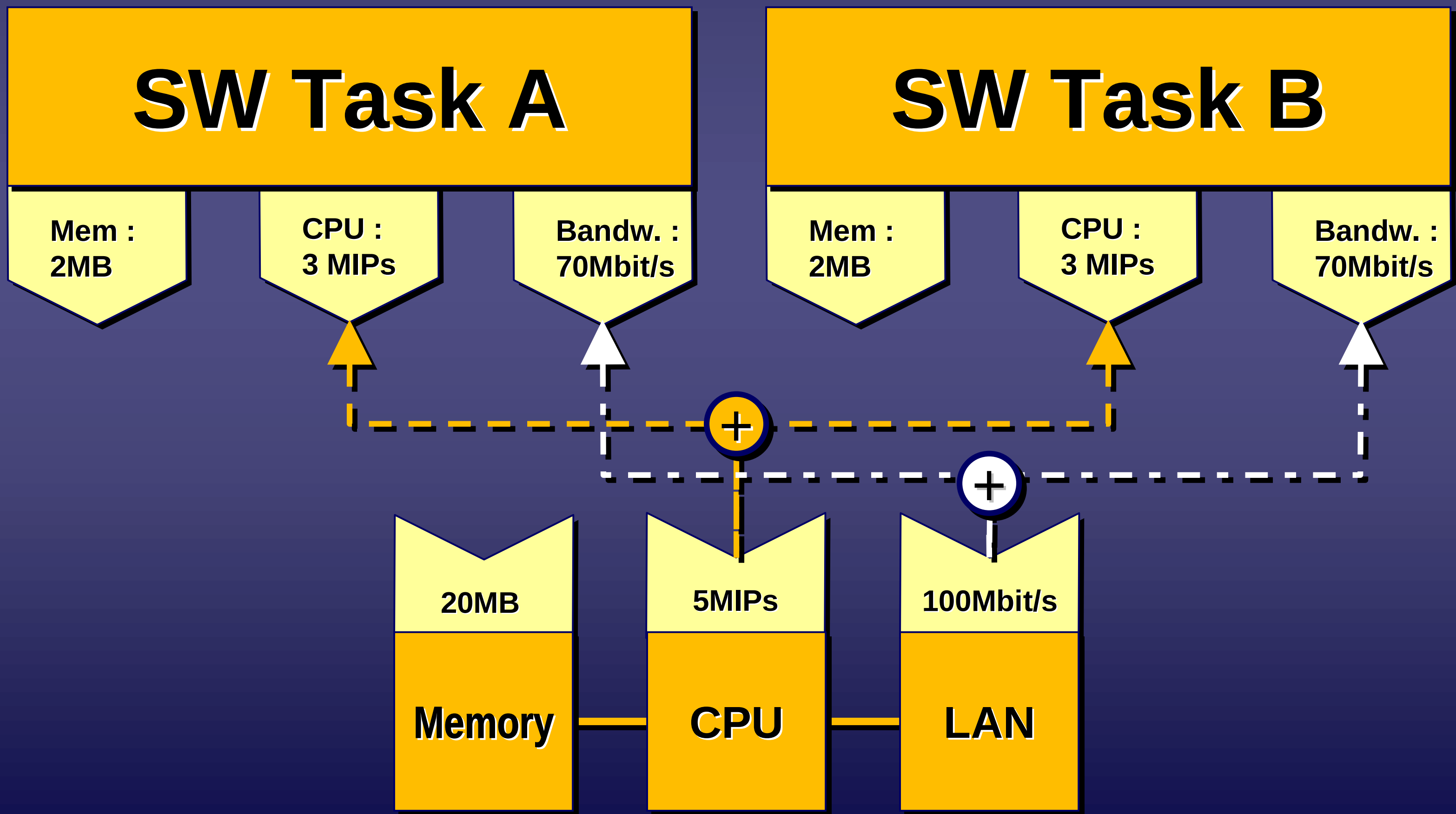


Source element	Dest. elements
A	X, Y
asc	Y
B	Z

deployment table

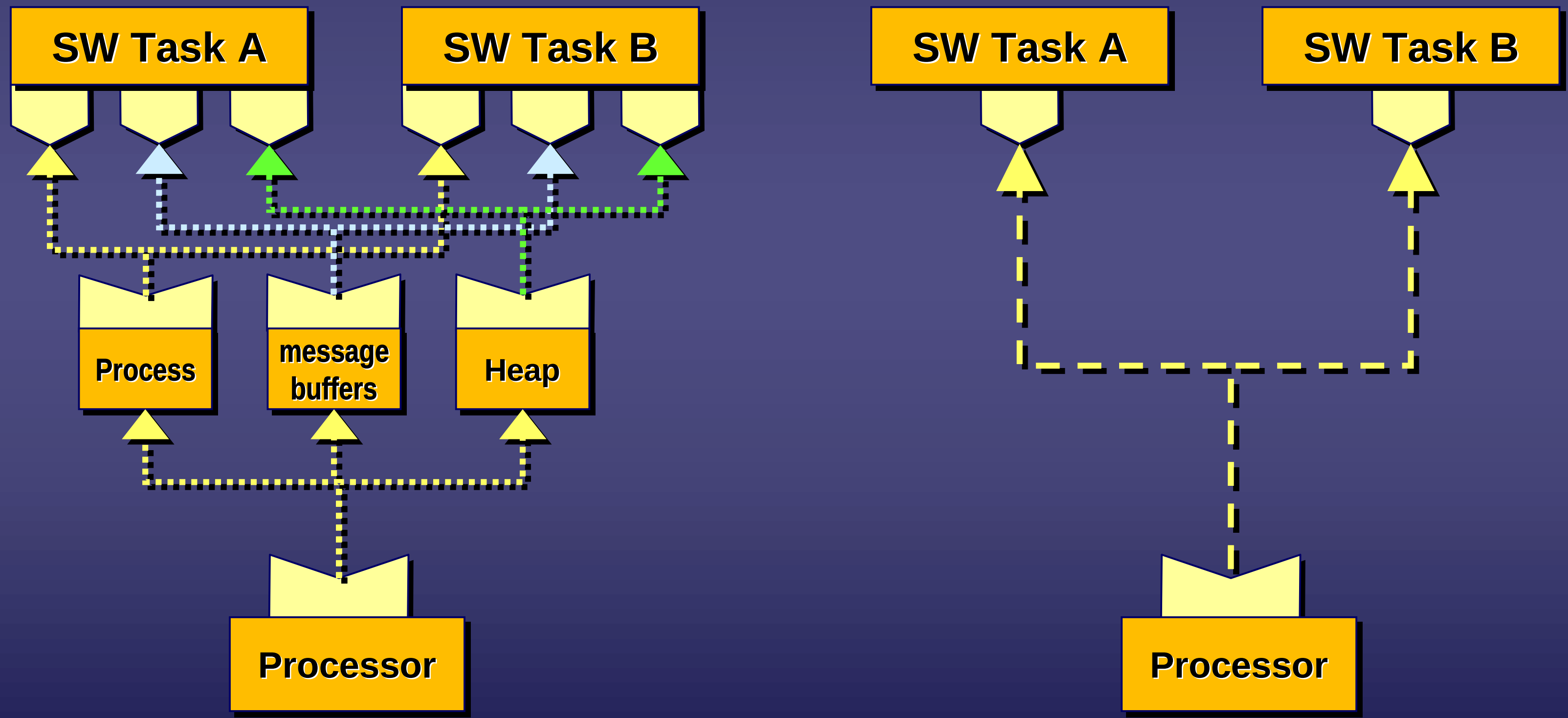
Resource Sharing

- Shared resources complicate QoS contract validation
  - service-specific composition rules



Realization Relationships

- The precise semantics of the relationship depend on the chosen level of abstraction and resource type



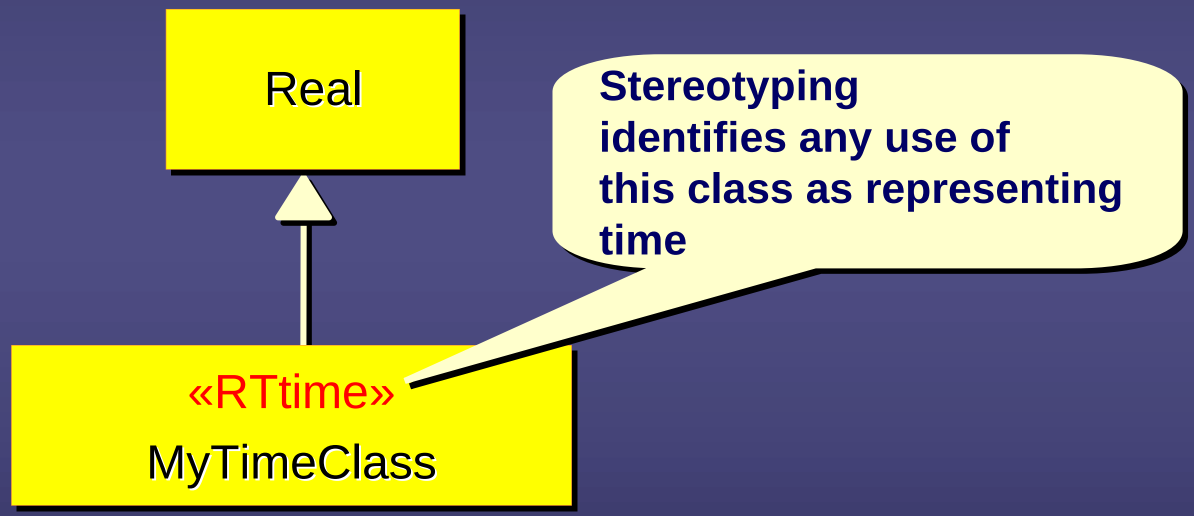
- Real-Time System Characteristics
- The Logical and the Engineering Viewpoints
- The Standard Real-Time UML Profile
  - UML Extensibility
  - Foundation
  - Modeling Time
  - Modeling Concurrency

The Model of Time in UML 1.4

- Unbiased and uncommitted:
  - Time data type declared but not defined (could be either continuous or discrete)
  - No built-in assumptions about global time source (open to modeling distributed systems)
- Related concepts:
  - Time events: generated by the occurrence of a specific instant
  - Assumes some kind of run-time Timing Service

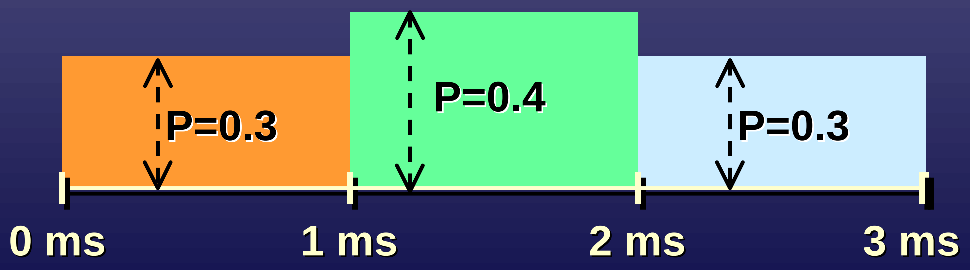
RT Profile: Modeling Time

- «RTtime»: a stereotype of Classifier (and Instance)
  - supports both continuous and discrete time representations



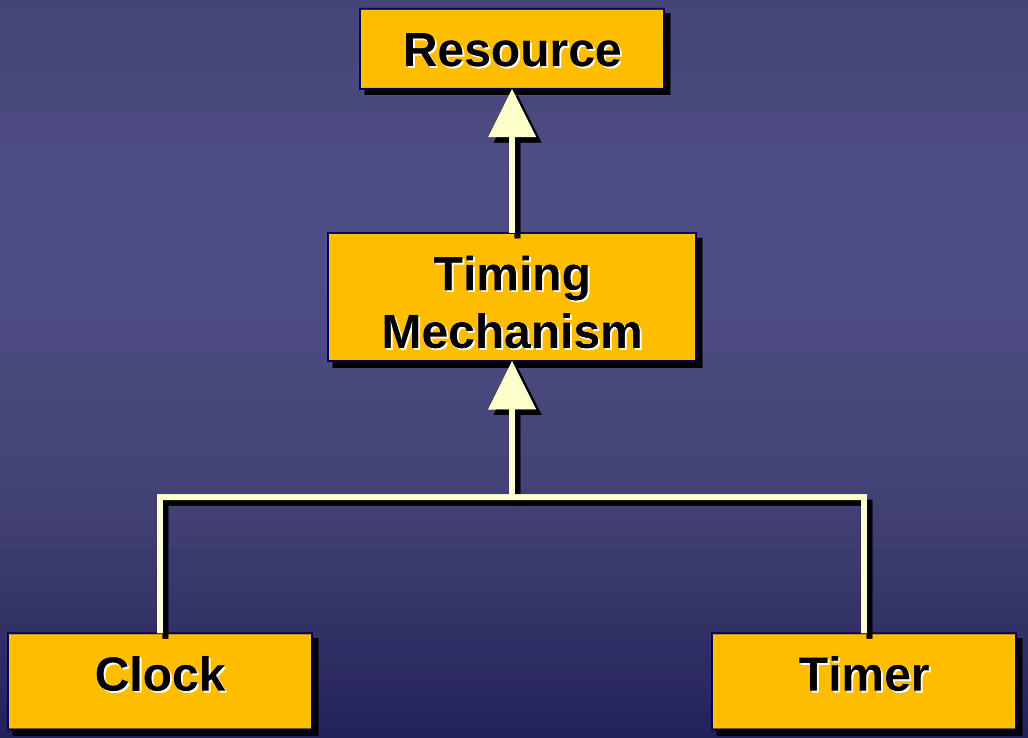
Specifying Time Values

- Time values can be represented by a special stereotype of Value («RTtimeString») in different formats; e.g.
  - "12:04" (time of day)
  - "5.3 ms" (time interval)
  - "2000/10/27" (date)
  - "Wed" (day of week)
  - .$param ms" (parameterized value)
  - "poisson 5.4 sec" (time value with a Poisson distribution)
  - "histogram 0:0.3 1:0.4 2:0.3 3 ms"



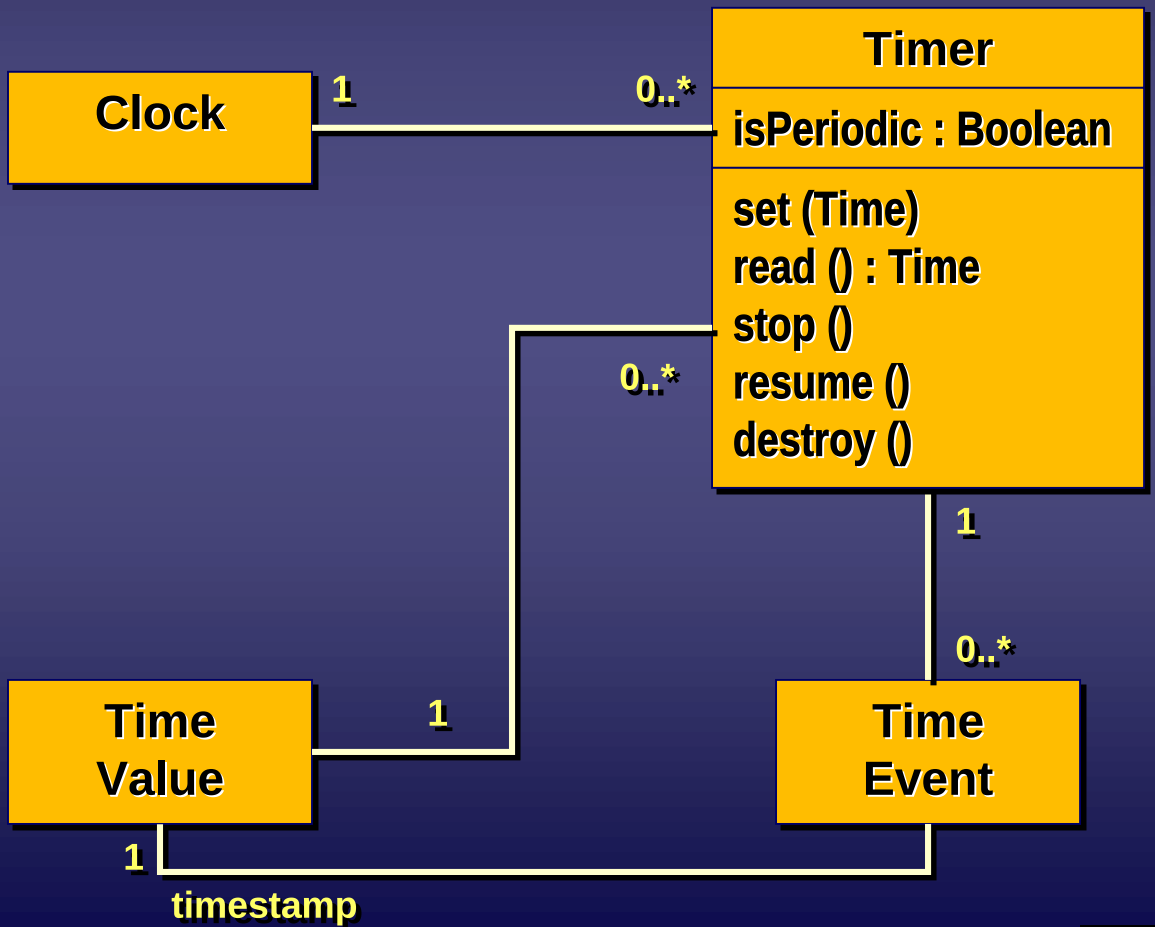
RT Profile: Modeling Timing Mechanisms

- Guidelines for modeling time and timing facilities
  - Based on the general resource model



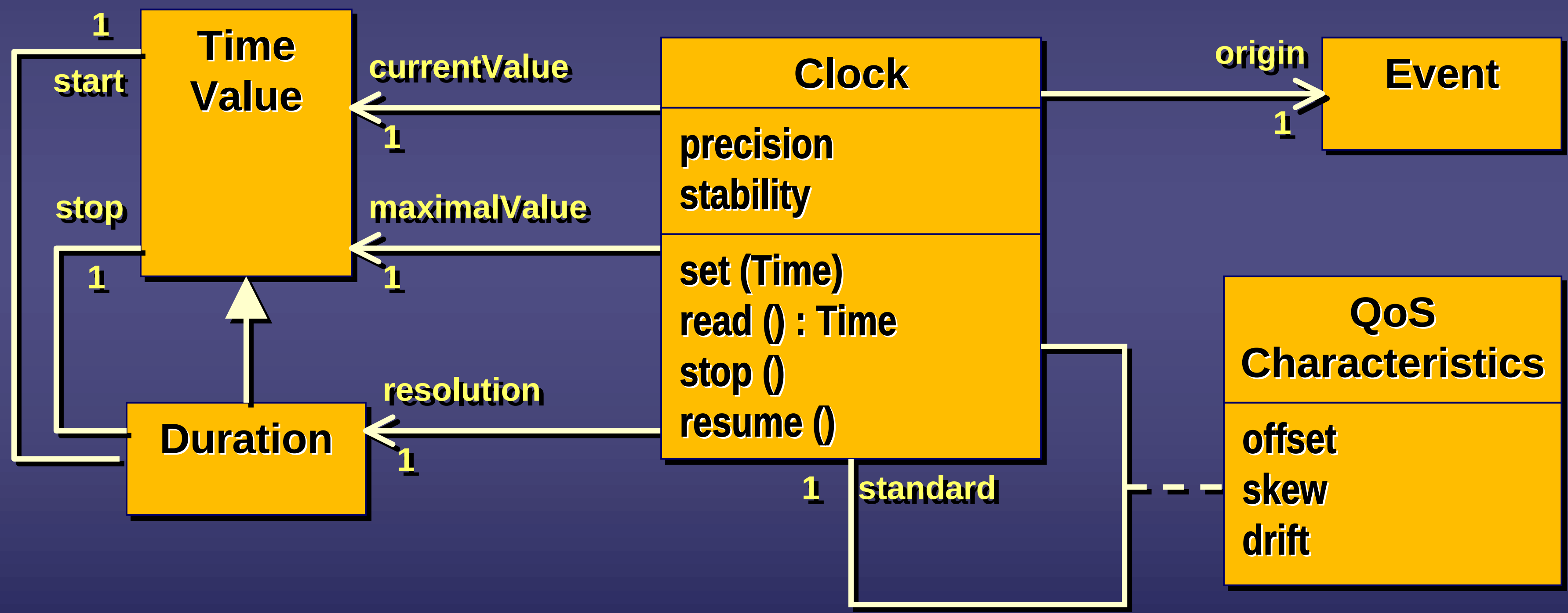
Modeling Timers

- Resource that generates events when a particular instant in time has been reached



Modeling Clocks

- Resource for telling the “time of day”



Notation: Timing Marks and Constraints

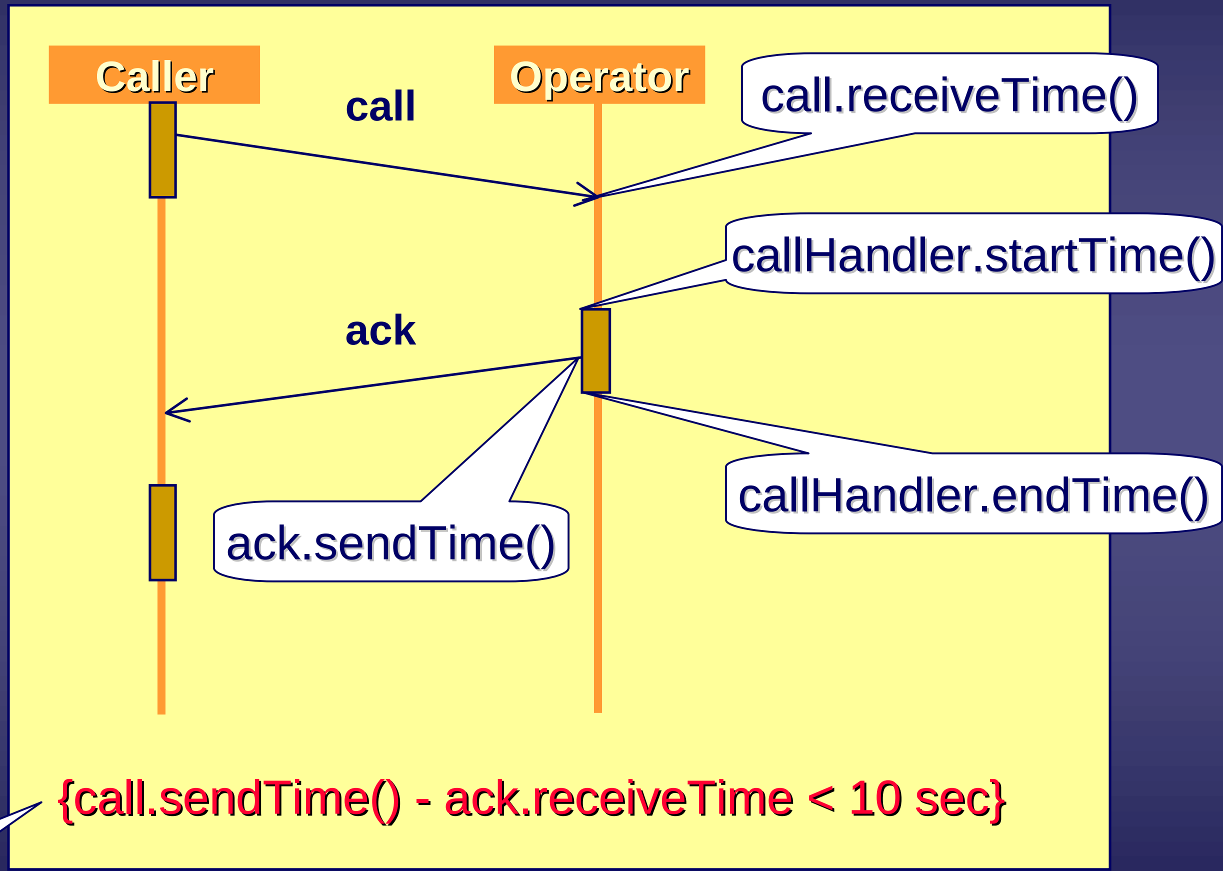
♦ A *timing mark* identifies the time of an event occurrence

- On messages:

  sendTime()
  receiveTime()

- On action blocks (new):

  startTime()
  endTime()

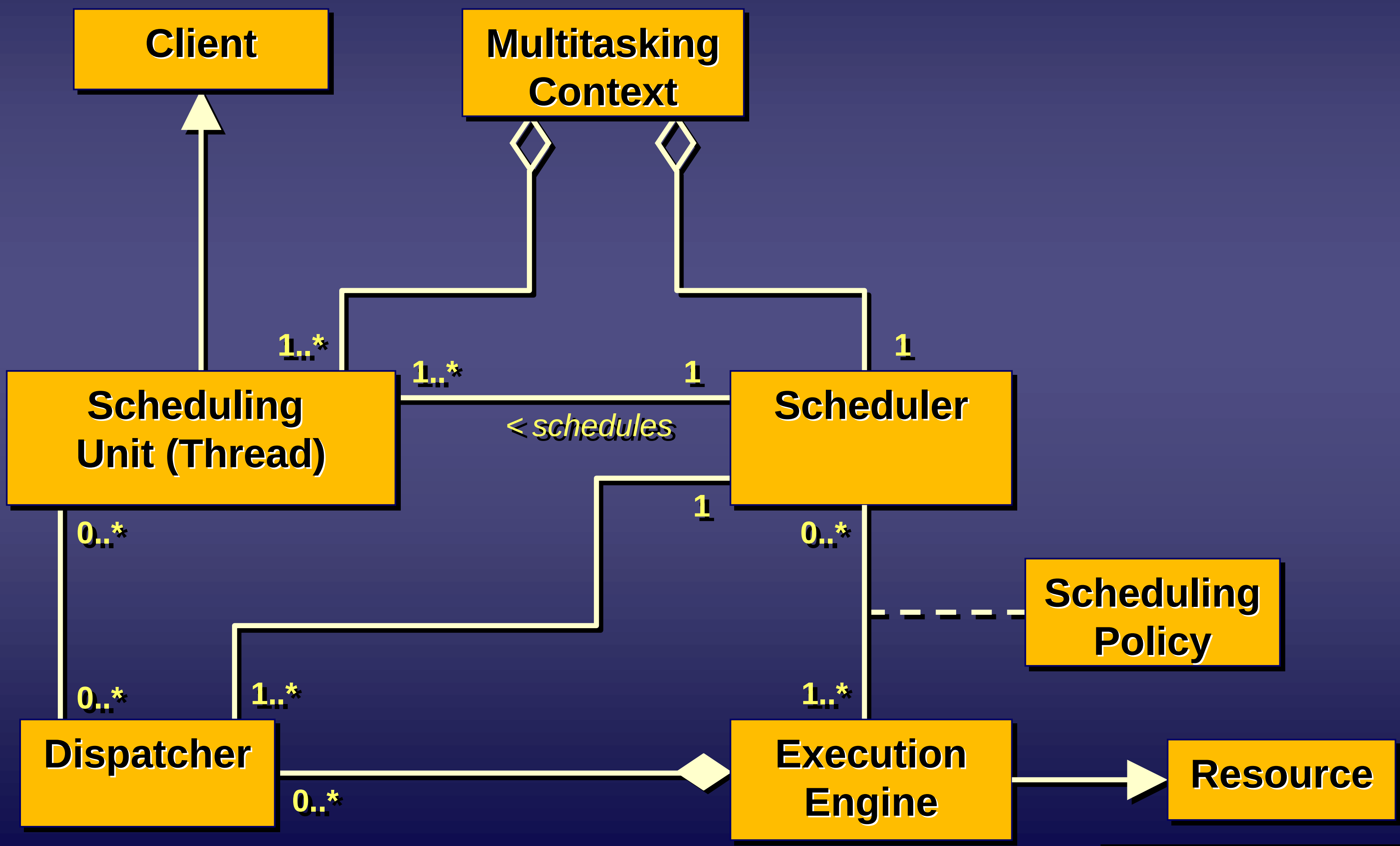


Timing constraint

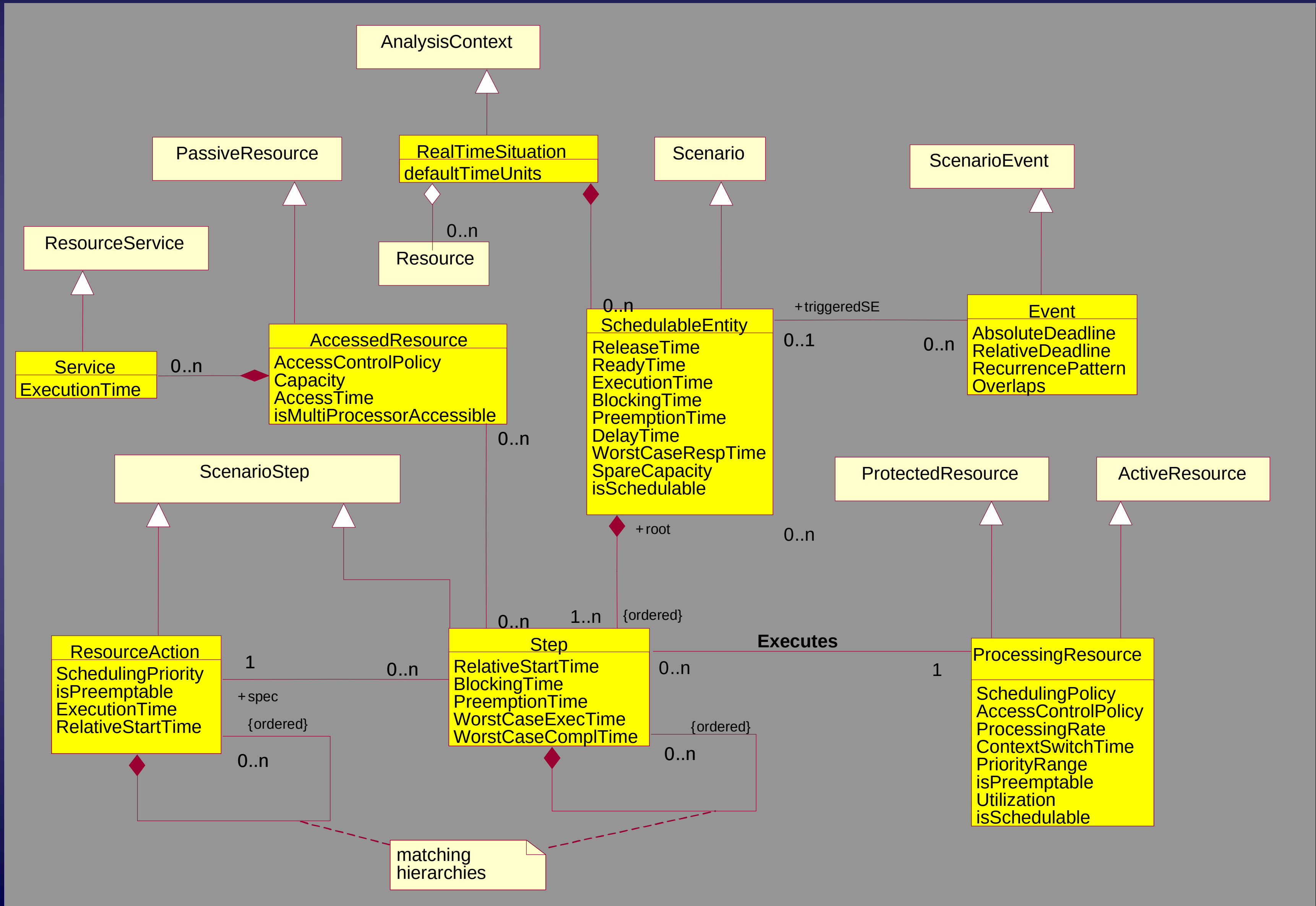
- Real-Time System Characteristics
- The Logical and the Engineering Viewpoints
- The Standard Real-Time UML Profile
  - UML Extensibility
  - Foundation
  - Modeling Time
  - Modeling Concurrency

Concurrency at the Engineering Level

- Generic model of a multi-tasking environment



Schedulability Model



Summary: RT Design and Engineering

- In complex RT systems, the logical design is strongly influenced by the physical characteristics of its engineering environment
- In such systems, it is usually crucial to know if a system will meet its non-functional requirements (throughput, response time, availability, etc.)
- The QoS-based approach described here can serve as a basis for:
  - quantitative analysis of UML-based models
  - a real-time modeling standard that will facilitate automated exchange between design and analysis tools

Bibliography

- Cooper, R., Introduction to Queueing Theory, The Macmillan Company, 1972.
- I. Jacobson, G. Booch, and J. Rumbaugh, “The Unified Software Development Process,”, Addison-Wesley, 1999.
- Klein, M. et al., A Practitioner’s Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems, Kluwer Academic Publishers, 1993.
- OMG, “The Unified Modeling Language” version 1.3, The Object Management Group, August 1999.
- OMG, “UML™ Profile for Scheduling, Performance, and Time - Request for Proposal”, The Object Management Group, March 1999 (doc ad/99-03-13).
- J. Rumbaugh, I. Jacobson, and G. Booch, “The Unified Modeling Language Reference Manual,”, Addison-Wesley, 1999.
- B. Selic, “Turning Clockwise: Using UML in the Real-Time Domain”, *Communications of the ACM*, vol.42, no.10, October 1999 (pp.46-54).
- B. Selic and J. Rumbaugh: “Using UML for Modeling Complex Real-Time Systems,” ObjecTime Limited and Rational Software Corp., March 1998. (http://www.rational.com)