Traditional Approaches to Modeling

Timeliness, Performance and How They Relate to Modeling, Architecture and Design

Mark S. Gerhardt
Chief Architect
TimeSys Corporation
Pittsburgh, PA 15213

Levels of Real Time Performance

- Measured
  - (quantitative indications) - May or may not be repeatable
- Repeatable Measured
  - Measuring the temperature 3 days DOES NOT help tomorrow
- Statistically Predictable Architecture
  - Statistical - average response and related standard deviation
- Analytically Guaranteed bounded latency
  - Maximum duration between an event and its associated response
- Deterministic

Knowledge of *every* state of a system over time

The above Goals Utilize different architecture, tools, and infrastructure approaches! They also require different modeling techniques!

Analysis and Modeling Concerns

- Measured - ad hoc
- Repeatable
  - control over “hidden” daemons
  - e.g. spoolers, garbage collectors, low - level message handlers
- Statistically Predictable
  - often used architectures : Internal queuing, asynchronous messaging
  - analysis and modeling:
    - Discrete Event Simulation via use cases
    - Queuing Theory
- Analytically Guaranteed Latency
  - often used Architectures - Shared Resources, Fixed Priority Scheduling, “Real-Time” O/S Kernels, Arbitration and Contention
  - analysis and modeling: Rate Monotonic Analysis
- Deterministic
  - cyclic executives, SPARC approach
  - State Machine/Sequence Diagram oriented

Fundamental Relevant Concepts Which Affect Timing Performance

- Resources are finite in nature and bounded in their ability to respond.
- Contention arises for:
  - Allocation and Locking
  - Priority and Preemption Issues
- Understood Examples:
  - Safe Deposit Box Master Keys
  - A One Bathroom House with several children

An Example From an Obvious Domain

- Additional concepts and parameters required:
  - Need expression in order to specify intended performance/timing requirements
  - Need examination and tradeoff of implementation effects during synthesis
  - Quantitative performance indications are possible if additional performance/timing information is captured in the design and implementation via:
    - analysis
    - simulation
    - testing and data reduction
- An obvious domain example illustrates this point

What's Wrong with Traditional Design? - An Example

- **Functional Approach to Family Needs:**
  - **Get Groceries**
  - **Take Kids to Soccer**
  - **Buy parts and fix leaky WC**
  - **Mom also has to prepare dinner for Dad's boss by 6PM**
    (Requirement: Deadline)

- **The "Socially Correct" Implementation Binding**
  - **Mom will shop for the groceries and take the kids to soccer**
  - **Dad will get the hardware and do the repair**

Resources Affect Performance by adding Sequencing or Blocking

- Shared Resource: The Family Car
  - It can only go one place at a time
  - Actions using the car are *SEQUENCED* because they *share* the car
    - This affects performance
  - **Concurrency** is used to improve performance:
    - Families have multiple cars - resource allocation is based upon performance concerns.
    - This family has 2 cars (used by two drivers)
    - Within the concurrent use environment, preemption and blocking becomes concerns in addition to traditional sequencing concerns

What I forgot to tell you

- The Hardware store is adjacent to the market, both 5 minutes away
- The soccer field is 45 minutes away through traffic!

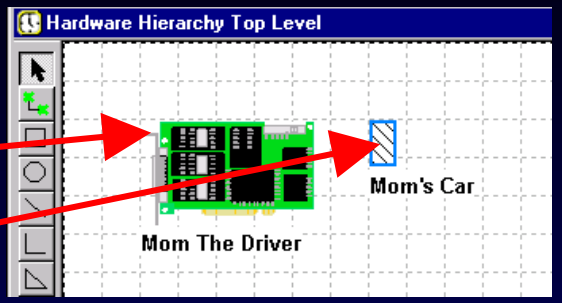
From a performance driven analysis, Mom should not do both the soccer task and the grocery task using the same car.

Giving Dad the job of the WC and the marketing will improve the performance of the system and allow Dad to keep his boss happy and keep his job too!

Sample Analysis and Notation

Schedulable Resource

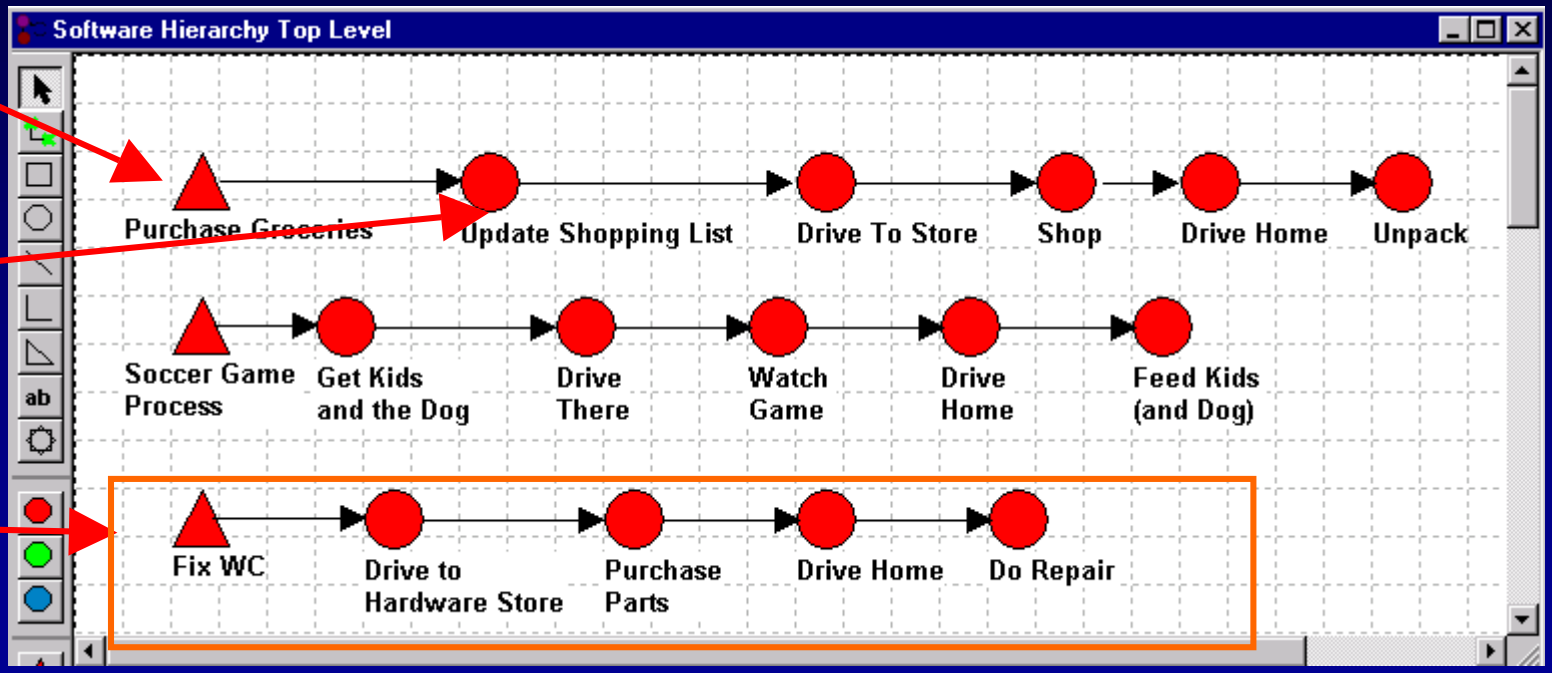
Logical Resource



Trigger

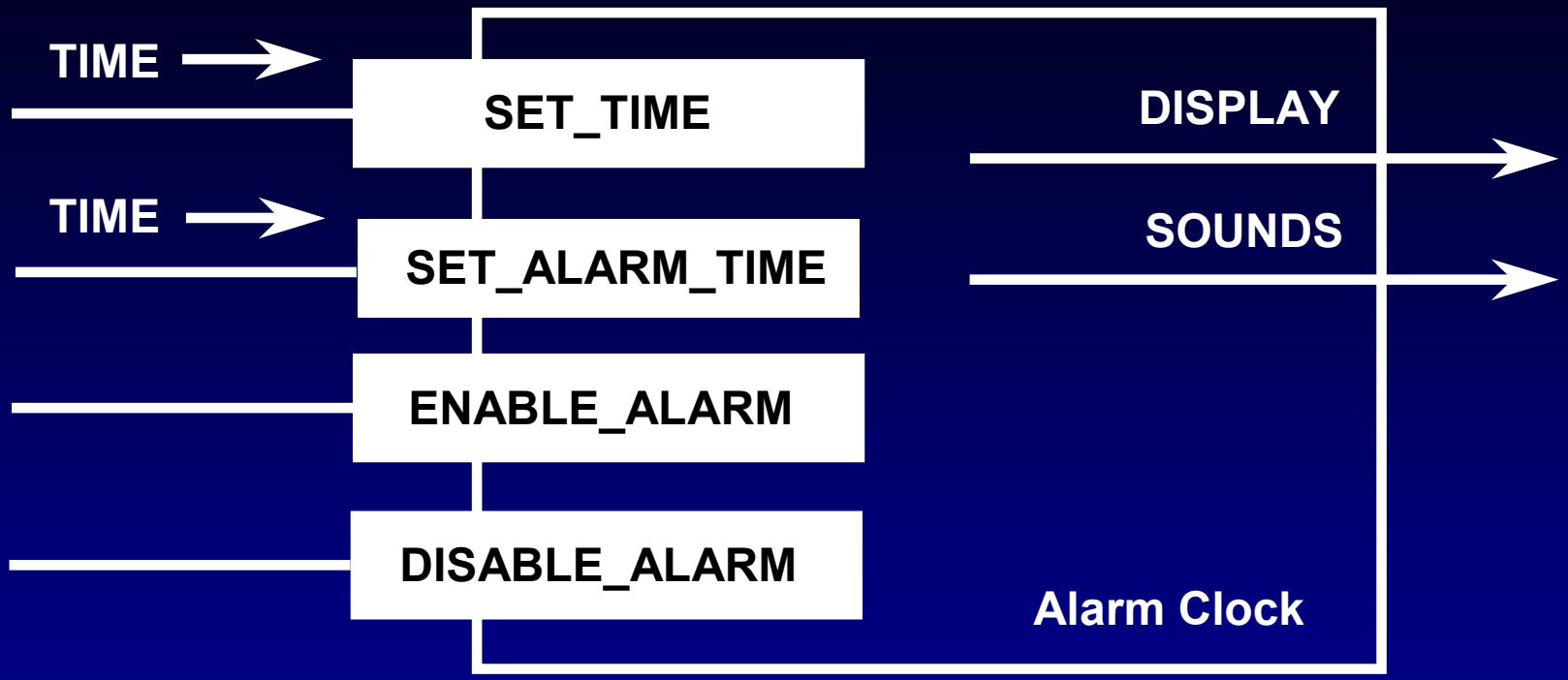
Action

Thread



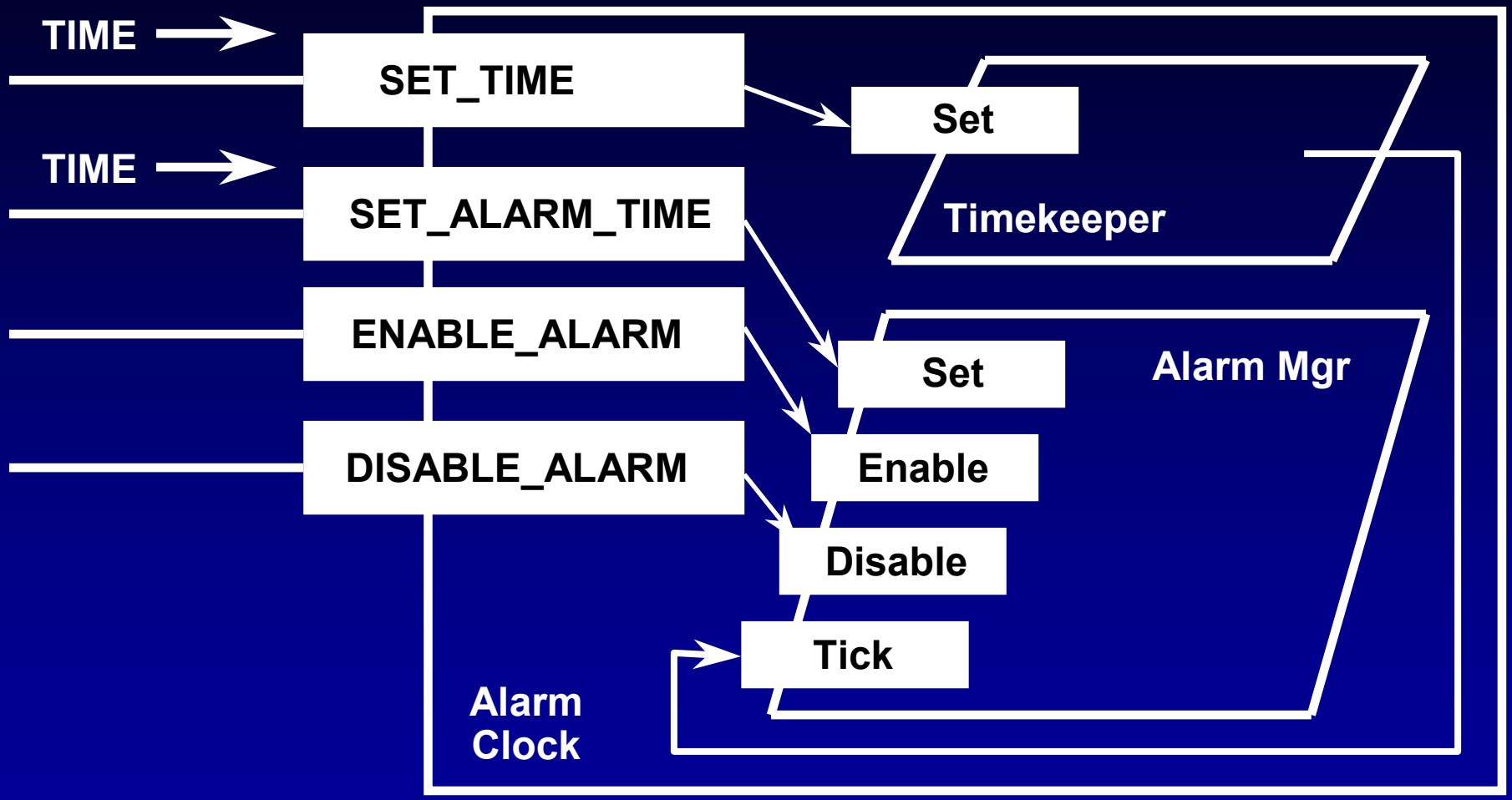
What if the Purchase thread had a 20 minute deadline ?

Second Example- Alarm Clock



- The MOST IMPORTANT function of the alarm clock is NOT in the functional interface description!
- It happens when nothing interfaces with the alarm clock!

Adding Concurrency

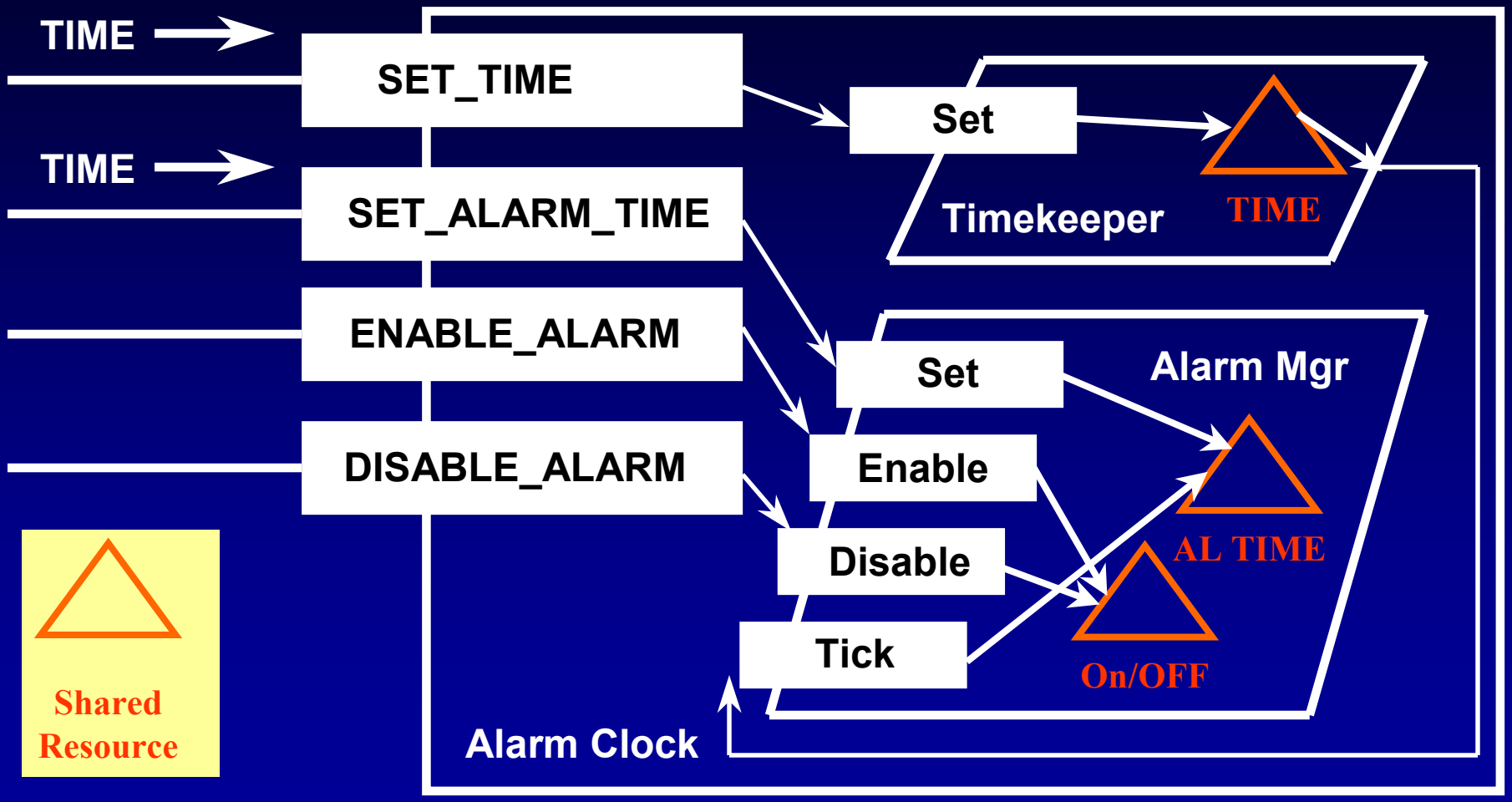


The Consequences of Concurrency at the Bank

- Cooperation between the concurrent threads:
  - Sharing of the value for Time
  - Sharing of alarm time and wakeup enabled state

- The semantics used to make the problem tractable presents **SHARED RESOURCES** as a basic architecture concept
  - Different sharing policies (discussed later) have a **MAJOR** effect on performance timing

Adding Shared Resources and Contention



Refocusing OO Development for Timing and Performance Assessment

Required Behavior Representation in the Model

- Object Abstractions must explicitly convey the following kinds of behavior:
  - Autonomous activity
    - (Periodic, Aperiodic, and Statistical invocation)
  - Reactive Behavior for each Callable Operation
    - Interaction Protocol
    - Computational Path Length
    - Abnormal Completion and Timing
  - Shared Entities
    - Explicit - shared resources
    - Implicit - MUTEX or locking behavior

Modeling: Partitioning and Functionality Simultaneously

- Architecture is partitioning a concept into “parts”
- Encapsulation of the part provides explicit boundaries:
  - Scope and visibility control
    - Locality and local meaning
  - Explicit communication with other parts
    - Cooperation semantics - synchronous, async, polling, etc.
  - Responsibilities
    - Services provided (external)
    - Autonomous Activities - internal services
    - Arbitration for multiple desired actions
  - Persistence
    - Initialize (possibly distributed)
    - Finalize
- The parts must cooperate meaningfully to fulfill expected requirements

The Modeling Process and Its Concepts

- 3 **Simultaneous** Types of Choices:

  Functionality, Concurrency, Synchronization
- Early system decomposition is usually done using *FUNCTIONAL COHERENCE* as the criterion for decomposition
- The domain specific requisite cooperation and collaboration between the allocated "parts" are a consequence of the partitioning.
- Resource and Communications constraints play an important role **IMMEDIATELY**!
- "**Middle Out**" development - taking temporal performance expectations into account via continued analysis - is the only practical way to assure obtaining performance relevant systems.

- Layered and Deferred Implementation- Motivates characterization of **offered** and **supplied** properties

Concepts Required by the Model

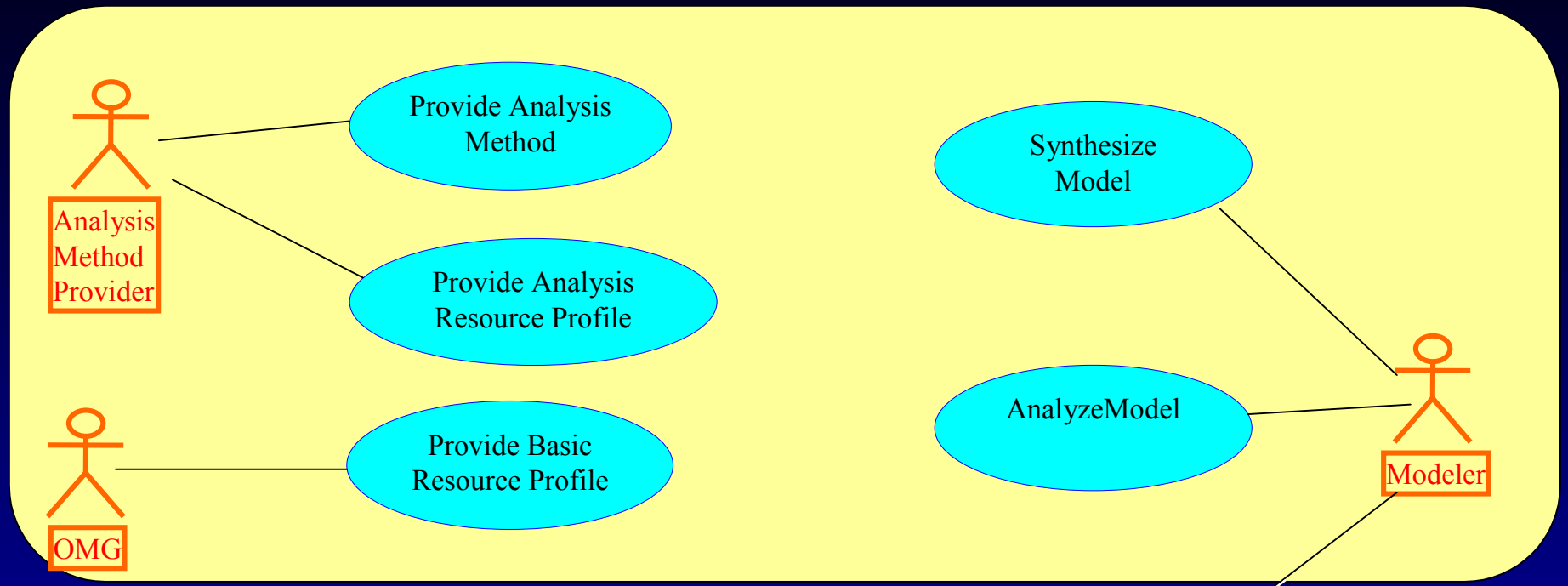
- Reactive Operations must convey:
  - Computational Path Length
  - Algorithm Behavior (Logical Behavior + Path Length)
  - Communication and Synchronization
    - Cooperation + Resources Used
    - Context Sensitive initiation Conditions (Guards, Barriers)
  - Pathological Outcomes (Behavior + Timing)
- Autonomous Operations must convey:
  - Initiation Timing - Periodic, Aperiodic, Statistical
  - Algorithmic Behavior (Logical Behavior + Path Length)
  - Pathological Outcomes (Behavior + Timing)
  - Communication and Synchronization
    - Cooperation + Resources Used
    - Context Sensitive initiation Conditions (Guards, Barriers)

Augmenting the UML

The UML RT Submission

- UML Profile
  - Extensions to UML to capture such things as Quality of Service and Time
  - Framework for supporting actual analysis methods and actual infrastructures
  - Completed (and useable) examples of support for some analysis methods, and at least the RT-CORBA infrastructure

The Various Modeling Interests



Additional Concepts for Augmented UML - Resources

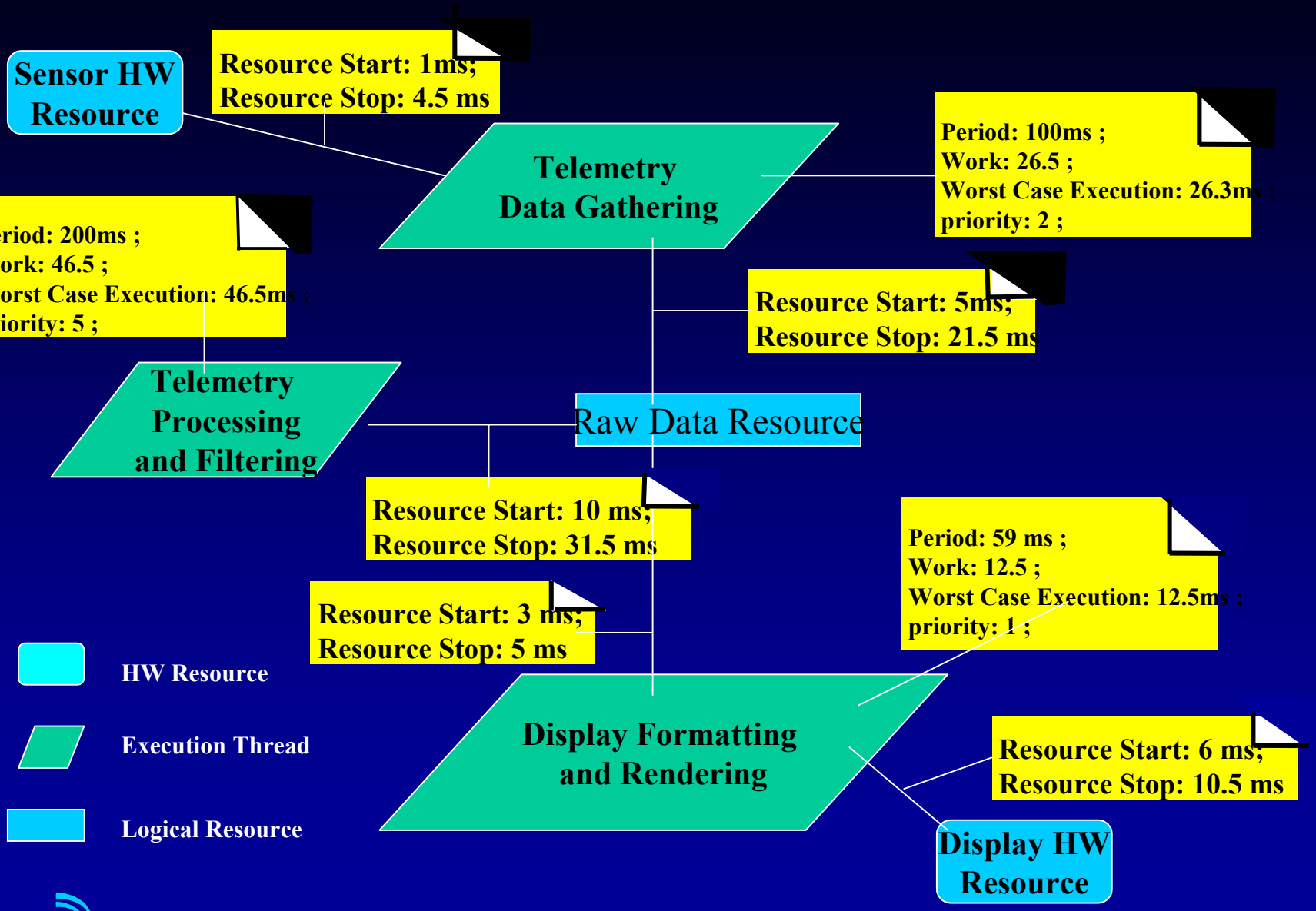
- Bounded implementation artifacts - RESOURCES
  - Finite server capacity
  - Exclusivity of use
    - Protected or locked
    - may be preemptible
  - May have associated capture, setup, shutdown, and release times
- Clients need to indicate which resources they use and how long they use them

Additional Concepts for Augmented UML - QoS

- Timing indications need to be captured and recorded in the design and implementation

- For clients or partially implemented composite container elements for each specified operation we need to characterize acceptable worst case response time
- For implementations, we need to characterize:
  - computation time
  - blocking time
  - connection/synchronization time

A Sample of the Thinking

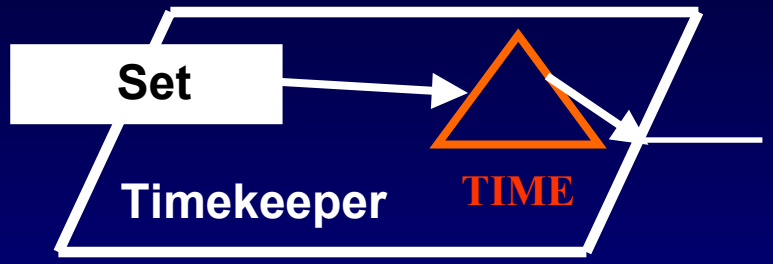


Layers, Deferred Implementation, and QoS

- Each architecture layer must COLLECT AND MAINTAIN performance information

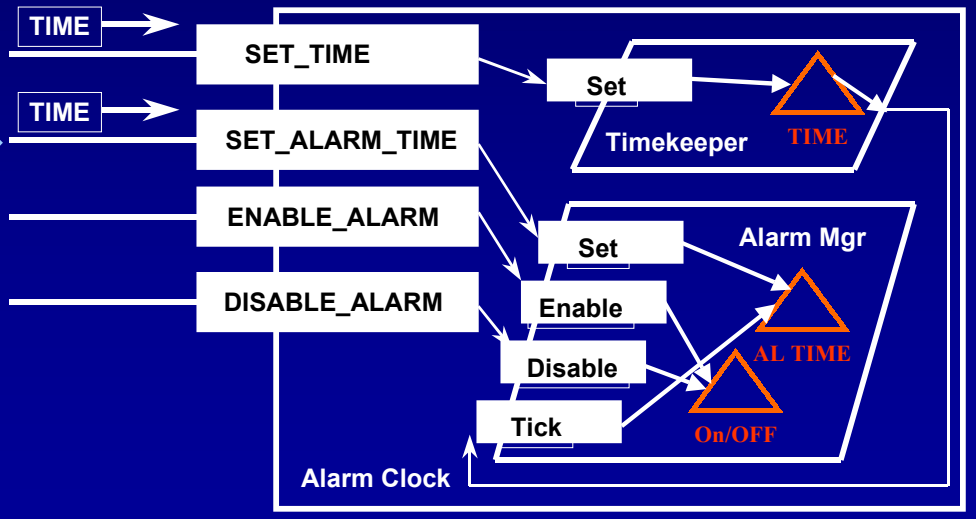
Path Length: Timekeeper.Set

Blocking: Time Resource



Path Length: Alarm Clock.Set Al Time mapped to Alarm Mgr.Set Al Time

Blocking: 0.5 sec mapped to Alarm Mgr.AL Time resource



Status of UML RT

- **Milestones:**

  RFP Draft Submission -Nov 2000

  Final Submission - June 2001

Contact Information

Mark S. Gerhardt
TimeSys Corporation
4516 Henry Street
Pittsburgh, PA 15213
mark@timesys,com
(650)208-3994 (Pacific Time)