Beyond Functionality

What to capture during architecture derivation

Mark S. Gerhardt

(650)208-3994

gerhmark@pacbell.net
The following PREVIEW has been approved for ALL AUDIENCES
By the Object Oriented Association of America

The CLASS Advertised has been rated

REstricted

Any Structured Design Requires An Accompanying Object Oriented Guardian
Early System and Software Architecture Development Concerns Fidelity Issues

• Functional Expectations
  – Domain and Algorithm Concerns
  – Show up as “Computational Path Length”

• Conceptual Partitioning
  – Preliminary choices of subsystems
  – Allocation of functionality
  – **IMPLICIT** communication, synchronization, and sharing

• Component Abstraction - Objects
  – Functional Abstraction through **CLASSES** and **INSTANCES**
  – Dynamic **BEHAVIOR** of cooperation, waiting, blocking
    between objects **HIDDEN** in traditional Architecture descriptions
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.
Refocusing OO Methods for Real-Time
Partitioning and Functionality Go Together

- 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) – Includes creation dependencies
    - Finalize - Includes termination dependencies
- The parts must cooperate meaningfully and in a timely way to fulfill expected requirements. The requirements affect the partitioning.
The MOST IMPORTANT function of the alarm clock is NOT in the functional interface description!

It happens when nothing interfaces with the alarm clock!
Partitioning, Functionality and Real_time

• 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.
The Enhanced Alarm Clock #1 - Concurrency – One Thread for each Time Constraint
Consequences of Concurrency

• 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 \textit{SHARED RESOURCES} as a basic architecture concept
  – Different sharing policies (discussed later) have a \textbf{MAJOR} effect on performance timing
The Enhanced Alarm Clock #2 - Concurrency and Sharing

**Alarm Clock**
- SET_TIME
- SET_ALARM_TIME
- ENABLE_ALARM
- DISABLE_ALARM

**Timekeeper**
- Set
- Enable
- Disable

**Alarm Mgr**
- On/Off
- Set

**Shared Resource**

**Diagram Notes**
- TIME
- AL_TIME
- AL TIME
- On/Off

**Flowchart**
- Alarm Clock
- Timekeeper
- Alarm Mgr
- Shared Resource
Layers of Performance

- 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
Characterizing Objects to Support Analyzability

• 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)
Causality-

Invocation Dependency, Timing, and Scheduling
X Windows and Callbacks

- Client provides procedures to be “called back” by the X Windows Server when specific actions (MMI buttons, clicks, etc.) occur
- Information about these procedures is “registered” with the X windows server - use a furnished C procedure for this
- X windows Server executes and uses the user-furnished callback as if it were an X-written library procedure
The Callback Problem

- The X Server blocks EVERY callback until the current one is completed:
Interactions, Context, and Timing
Killing a Ghost

**Dynamic Behavior**

- Pacman moves in response to joystick
- Pacman Overlaps Power Dot and is Hungry!!
- Power Dot Makes Ghost Vulnerable
- Pacman kills the vulnerable Ghost
Example: Cooperating Processes and Meeting Deadlines

- **Writing A CD-R:**
- **Original Design:** Continuous periodic writing regardless of buffer state
- **Major constraint is the writer which must have sufficient data in the buffer to “stream” onto the media.** (Two periodic deadlines to both write data and to furnish data)
- **Avoid BUFFER UNDERRUN…….**

---

Data Provider

Non-contiguous files take longer to navigate…

Device Writer (periodic)

Late Data causes **FAILURE**
Improved CD-R Architecture

- Preventing buffer under-run:
  - Revised design: Reallocate critical timing requirement to ONLY the writer and ASYNCHRONOUSLY start it.
  - DO NOT attempt local write until enough data is available - the only periodic time requirement is now WITHIN the writing process - not between two processes.

![](diagram.png)

Only writes when queue is not empty

Non-contiguous files take longer to navigate...

Late Data causes DELAY but not Failure
UML RT Efforts – Rhapsody and RapidRMA
Where to go next

• Rhapsody +UML RT intentions
• Concurrency semantics:
  – Event causality using sporadic server vs “periodic”
  – Annotations for run to completion vs preemption vs locking
• “Bounded Regions”
  – Assert performance properties across region – e.g. guaranteed latency
  – look for appropriate consistent communication and scheduling semantics
• Abstraction of this detailed behavior to higher levels- e.g the upward encapsulation of a blocking call must preserve the blocking property and blocking parameters