C12 - ARCHITECTURAL STYLES

Bob Nielson
PIPES AND FILTERS

(a) pipes and filters

(b) batch sequential
BUFFERED PIPES AND FILTERS

Buffered Communication in Active Pipes and Filters

pipeline: I | G | V | P | W | O

- Pipes are *information-hiding components*
  - they are active elements (threads or processes)
  - they include public interface operations, private code, and private data
- Filters operate *as asynchronous processes*
  - with *synchronization* at the buffers

Monday, March 15, 2010
Class Diagram for the Buffer Pattern

Input Filter
- pushing to its output buffer
- pulling from its input buffer

Output Filter
- pulling from its input buffer and pushing to its output buffer

Buffer is an active component, as are the filters

Buffer
- store
- head
- tail
- number
- size
  + get();
  + put(x);
  + empty?();
  + full?();

Semaphore
  integer s
  acquire();
  release();

buffer protocol:
- must not insert into a full store
- must not remove from an empty store
- must not allow simultaneous invocation of get() and put(x)
A Semaphore Pattern

- Semaphores can be generalized:

#### Semaphore

- `integer s`
- `int acquire ()`
- `void release ()`

- `release is the same in all subclasses; hence it is a concrete operation`

- `waitSemaphore`
  - `integer s = 1;`
  - `binarySemaphore ();`
  - `int acquire ();`

- `balkSemaphore`
  - `integer s = 1;`
  - `balkSemaphore ();`
  - `int acquire ();`

- `countSemaphore`
  - `integer s = N;`
  - `countSemaphore ();`
  - `int acquire ();`

*waiting and balking semaphores are also called binary semaphores*
A LAYERED EXAMPLE
The OSI 7-Layer Architecture --

clients

7: Application
6: Presentation
5: Session
4: Transport
3: Network
2: Data Link
1: Physical

(Hardware)

NOTATION

denotes a component
denotes a dependency
denotes an interface

Each component has a name, provided features, an interface for the component above it, and a dependency on the next-lower component.
The Publisher-Subscriber Design Pattern in UML
aka Subject-Observer

Publisher
- NotifyList
  - attach(handle);
  - detach(handle);
  - notifyAll();

{abstract}

ConcretePublisher
- PublisherState
  - ConcretePublisher();
  - getState();

ConcreteSubscriber
- SubscriberState
  - ConcreteSubscriber();
  - notifyMe();
  - other operations

Subscriber
- notifyMe();

{abstract}

Diagram shows the relationships between Publisher and Subscriber, with methods and states defined for each.
The Model-Viewer-Controller Pattern (MVC)
-- using UML & Buschmann’s page 129 terminology --

The “Subject”

Observer
update

Model
- core data
- set of Observers
- attach(Observer)
- detach(Observer)
- notify
- getData
- service

View
- myModel
- myController
- initialize(Model)
- makeController
- activate
- display
- update

Controller
- myModel
- myView
- initialize(Model, View)
- handleEvent
- update

user input

service includes processing of user input

update is update notification by Model

user output

a model can have multiple views and multiple controllers
A BLACKBOARD ARCHITECTURAL PATTERN

Blackboard
solution data
table data
inspect();
update();

1..n

Knowledge Sources
sourceState
updateBlackboard();
executeCondition();
executeAction();

Start-up:
a client calls initialize;
Control initializes the
Blackboard and the
Knowledge Sources;
then initializes loop

1..n

1

Control
controlState
initialize();
loop();
nexSource();

1

1

client

Monday, March 15, 2010
REFERENCES

- UML Diagrams – © Dick Fairley