Objects vs. Agents

Appropriateness

Limits
Objects vs. Agents

- **OOP Principles**
  - Encapsulation
  - Inheritance
  - Polymorphism

- **Agent Characteristics**
  - Autonomous
  - Intelligent
  - Interact via messages

Encapsulation aids in autonomy.
- Hidden state
- Strict interaction defined by interface
- Message $\approx$ method invocation

Inheritance eases construction.

Polymorphism aids flexibility & openness.
- Subclasses distinguishable as needed

Use Design Patterns

Appropriateness

- Encapsulation aids in autonomy.
  - Hidden state
  - Strict interaction defined by interface
  - Message $\approx$ method invocation

Inheritance eases construction.

Polymorphism aids flexibility & openness.
- Subclasses distinguishable as needed

Use Design Patterns
Sample Agent

```
cycle()
synch1()
sense() : GenericSensorData
updateView(oldView : GenericView, sensorData : GenericSensorData) : GenericView
plan(view : GenericView) : GenericPlan
execute(plan : GenericPlan)
report()
```

Limits

- Objects do not automatically address:
  - Mobility
  - Separate threads of control
  - Bounded resources and autonomy
  - Asynchronous message passing
The Actor Model

Defined by Agha at Univ. of Illinois at Urbana-Champaign.

Actors
- Actor = object with own thread of control.
- Communicate via asynchronous message-passing.
  - Delivery guaranteed
  - Time unbounded
  - Order not guaranteed

Actor Primitives
- send(a,v)
  - a = receiver
  - v = contents
- newactor(e)
  - e = expression to evaluate
  - Returns address
- ready(b)
  - b = behavior (new state)

Can be added to any language.
- Lisp, Java, etc.
Fairness

- Every busy actor eventually makes progress.
- Every ready actor will eventually receive all messages that were sent.
- Unless stuck, every actor will eventually process all messages.

Non-deterministic, but compatible with temporal logic
- Eventually, may, never.

**Time in Actor Model**

**Fig. 1.1.** In response to a message, an actor can (1) modify its local state, or (2) create new actors, or (3) send messages to acquaintances.
Java Limitations

- Passive object model
  - Threads ≠ objects.
  - Synchronized keyword too limited.
    - All or nothing.
  - Message passing not built in.
    - JMS is new.
- No mobility (migration)

SALSA

- Variant of Java
- Supports
  - Token-passing
  - Join continuations
  - Universal naming
  - Remote asynchronous communication
  - Migration
Universal Naming

- Open system naming should be:
  - Unique
  - Independent of location
- Split names into
  - Universal Actor Name (UAN)
  - Universal Actor Location (UAL)

Universal Naming II

- Universal Actor Name (UAN)
  - Based on Universal Resource ID (URI)
  - Use naming server to provide authoritative answer for UAL.
  - uan://wwc.travel.com/reservations/air/agent
- Universal Actor Location (UAL)
  - rmssp://wwc.aa.com/international/reservations/agent
  - RMSP = Remote Message Sending Protocol
Universal Naming III

Fig. 1.2. By providing a persistent name to an actor (UAN), the actor can migrate from a host to another without breaking existing references.

SALSA Example

Create a travel agent, notify naming server.

```java
behavior TravelAgent {
    void printBinary() {...}
    public void set(String[] args) {
        TravelAgent a = new TravelAgent();
        try {
            a<c-bind="urn://wvc.travel.com/reservations/air/agent",
            "http://wvc.sa.com/international/reservations/agent");
        } catch (Exception e) {
            standardOutput<println(e);
        }
    }
}```
**ActorSpaces**

“ActorSpaces is a communication model that compromises the efficiency of point-to-point communication in favor of an abstract pattern-based description of groups of message recipients.”

-- Agha, Jamali, Varela.

---

**ActorSpaces II**

**Features**

- Patterns: Specify groups of message receivers, based on their attributes.
- Actorspaces: Scoping mechanism for pattern matching.
- Capabilities: Control operations.

**Actors publish attributes.**

- Other actors look for them.
Migration in ActorSpaces

- Messages are redirected, not preprocessed.
- RMSP server maintains hashtable.
  - Maps relative UALs to actual SALSA/Java references.
  - When actors migrant, leave a forwarding agent behind temporarily.

References

  - [http://osl.cs.uiuc.edu/Papers/Agents.html](http://osl.cs.uiuc.edu/Papers/Agents.html)
References II

  - [http://citeseer.nj.nec.com/callsen94open.html](http://citeseer.nj.nec.com/callsen94open.html)
- The InfoSleuth Agent system