AOP: From Revolution to Evolution

Alex Muscar

University of Craiova, Romania

Context (I)

Developing complex distributed systems using Multi-Agent Systems

Why agents?

Writing distributed systems is hard;
we need a higher level of abstraction

Context (II)

We’re going to look at this from a programming language perspective

Quite a few agent languages

Jason, GOAL, MetateM, 2APL, 3APL, CLAIM, ...

Common traits?

Highly domain specific, focus on single agents and… they are all written in Java

Common traits?

Highly domain specific, focus on single agents and… they are all written in Java

Common traits?

Highly domain specific, focus on single agents and… they are all written in Java

Common traits?

Highly domain specific, focus on single agents and… they are all written in Java

Important aspects

- Agents
- Organizations
- Environments

Important aspects

- Agents
- Organizations
- Environments

Agents

Concurrently executing *entities* with *asynchronous*, *reactive* behavior

Single agent abstractions

Objects, Actors, Reactive Objects

Single agent abstractions

Objects, Actors, Reactive Objects

Important aspects

- Granularity: object
- Execution model: sequential
- Communication: sync
- Message ordering: strict

Single agent abstractions

Objects, Actors, Reactive Objects

Important aspects

- Granularity: actor
- Execution model: concurrent
- Communication: async
- Message ordering: no

Single agent abstractions

Objects, Actors, Reactive Objects

Important aspects

- Granularity: object
- Execution model: concurrent
- Communication: sync & async
- Message ordering: strict

Close, but no cigar

We’re still at the object level

Plans, not programs

Communicating event loops from the E language

Communicating event loops

- Pending deliveries
- Turns

Communicating event loops

- Pending deliveries
- Turns

Pending deliveries (I)

Asynchronous message sends are queued to be sent later and they return *promises*

Pending deliveries (II)

Eliminate the risk of deadlocks because plans don’t get interrupted to wait for another plan

Communicating event loops

- Pending deliveries
- Turns

Turns (I)

Send a pending message, process incoming messages *serially* and execute synchronous calls

Turns (II)

By providing serializability the risk of race conditions is eliminated

Important aspects

- Agents
- Organizations
- Environments

Prototypes

Simple and flexible

Prototypes

Simple and flexible

Prototypical delegation (I)

Objects pass messages to other messages

Prototypical delegation (II)

Objects can „point” to other objects (their prototypes)

Prototypical delegation (III)

When an object does not „understand” a message it will delegate it to its prototype

Prototypes

Simple and flexible

Flexibility

Changing an object’s prototype changes its „capabilities”

Organization

Group common functionality in traits

Organization

Group common functionality in traits

Trait

Abstract and control common behavior

Object capabilities

Traits can be seen as capabilities

Important aspects

- Agents
- Organizations
- Environments

Agents and Artifacts

Artifacts are generic bundles of behavior that provide a usage interface

Open issues (I)

Prototypes + object capabilities

Open issues (III)

(Partial) static typing?

Open issues (IV)

Actually write the language ^_^

Thanks