Linda vs. ReSpecT Examples

Andrea Omicini

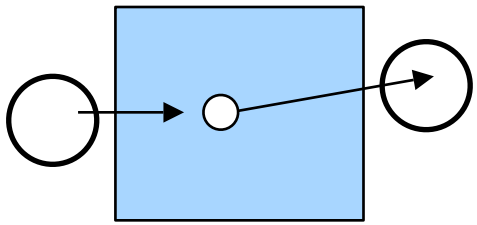
andrea.omicini@unibo.it

Alma Mater Studiorum–Università di Bologna a Cesena

aliCE

Enabling Communication (I)

- Message Passing



SENDER

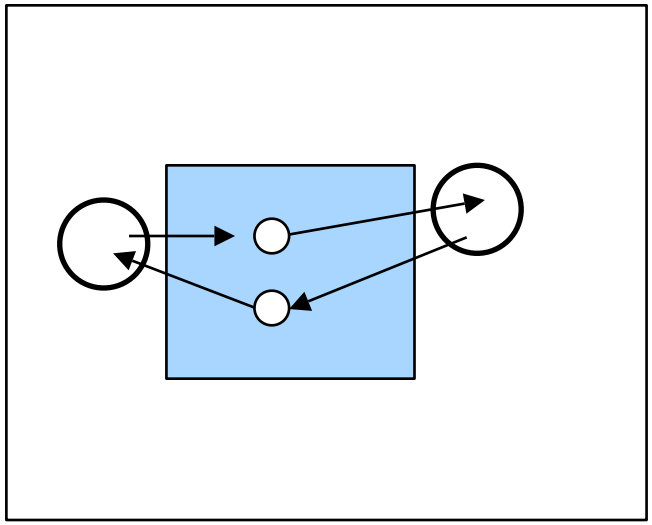
```
out(msg(agentB,content('test',13)))
```

RECEIVER (called agentB)

```
in(msg(agentB,Info))
```

Enabling Communication (II)

- RPC style



SERVICE USER

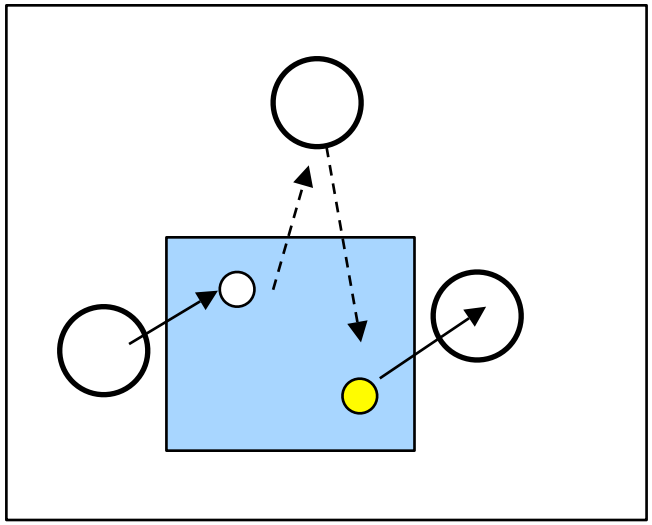
```
…
out(compute_sum(5,8,me))
in(compute_sum_result(me,Value)
…
```

SERVICE PROVIDER

```
in(compute_sum(X,Y,Who))
Sum ← X + Y
out(compute_sum_result(Who,Sum))
```

Enabling Interoperability

- Mediating between different ontologies



Good, but
- the mediation as a coordination activity is charged upon an entity (the service mediator), not upon the medium

Conceptual mismatch

➡ *engineering drawbacks*

SERVICE USER

```
…
out(compute_sum(5,8,me))
in(compute_sum_result(me,Value)
…
```

SERVICE PROVIDER

```
in(make_sum(term(X,Y)))
Sum ← X + Y
out(sum_result(X,Y,Sum))
```

SERVICE MEDIATOR

```
in(compute_sum(X,Y,Who))
out(service_requested(sum(X,Y),Who))
out(make_sum(term(X,Y)))
in(sum_result(X,Y,Sum))
in(service_requested(sum(X,Y),Who))
out(compute_sum_result(Who,Sum))
```

Interoperability in TuCSoN

- Ontology mediation charged upon the medium

Linda Style

SERVICE USER

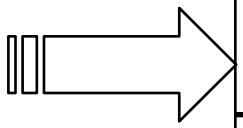
```
…
out(compute_sum(5,8,me))
in(compute_sum_result(me,Value)
…
```

SERVICE PROVIDER

```
in(make_sum(term(X,Y)))
Sum ← X + Y
out(sum_result(X,Y,Sum))
```

SERVICE MEDIATOR

```
in(compute_sum(X,Y,Who))
out(service_requested(sum(X,Y),Who))
out(make_sum(term(X,Y)))
in(sum_result(X,Y,Sum))
in(service_requested(sum(X,Y),Who))
out(compute_sum_result(Who,Sum))
```



TuCSoN Style

SERVICE USER

```
…
out(compute_sum(5,8,me))
in(compute_sum_result(me,Value)
…
```

SERVICE PROVIDER

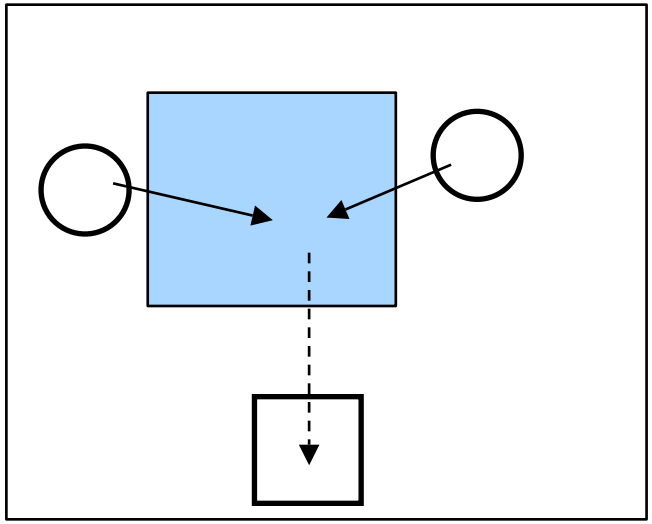
```
in(make_sum(term(X,Y)))
Sum ← X + Y
out(sum_result(X,Y,Sum))
```

MEDIATION POLICY in ReSpecT

```
reaction(out(compute_sum(X,Y,Who)),(
    in_r(compute_sum(X,Y,Who)),
    out_r(service_requested(sum(X,Y),Who)),
    out_r(make_sum(term(X,Y)) )).
reaction(out(sum_result(X,Y,Sum)),(
    in_r(sum_result(X,Y,Sum)),
    in_r(service_requested(sum(X,Y),Who)),
    out_r(compute_sum_result(Who,Sum)) )).
```

Basic Synchronisation (I)

- Synchronisation



Synchronised agent

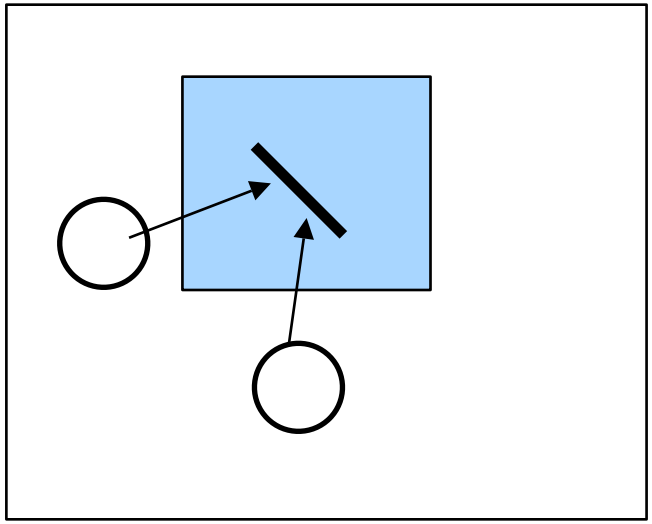
```
<outside sync region>
…
in(token)
<inside sync region>
out(token)
…
<outside sync region>
…
```

HYPOTHESIS
Initial space content with the tuple `token`

To have synchronised region allowing N users inside
→ N tuples `token`

Barrier Synchronisation (II)

- Barrier Synchronisation



Agent A

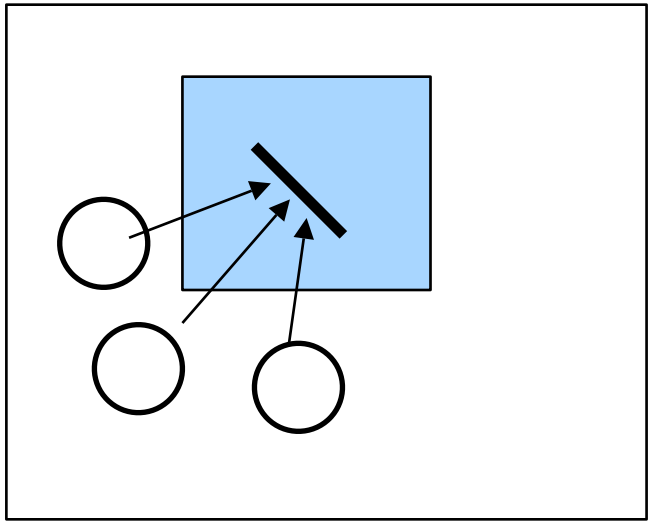
```
…
<before barrier>
…
out(ready(agentA))
rd(ready(agentB))
<agents A and B
are now synchronised>
```

Agent B

```
…
<before barrier>
…
out(ready(agentB))
rd(ready(agentA))
<agents B and A
are now synchronised>
```

Barrier Synchronisation (III)

- Barrier Synchronisation with 3+ entities



Agent A

```
…
out(ready(agentA))
rd(ready(agentB))
rd(ready(agentC))
…
```

Agent B

```
…
out(ready(agentB))
rd(ready(agentA))
rd(ready(agentC))
…
```

Agent C

```
…
out(ready(agentC))
rd(ready(agentA))
rd(ready(agentB))
…
```

Good, but
- Adding an agent → changing the behaviour of all the other agents
- Every agent must be aware of all the other ones

Barrier Synchronisation in TuCSoN

- Encapsulating the barrier synchronisation policy

Linda Style

Agent A

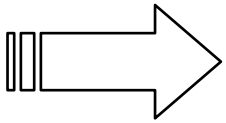
```
...
out(ready(agentA))
rd(ready(agentB))
rd(ready(agentC))
...
```

Agent B

```
...
out(ready(agentB))
rd(ready(agentA))
rd(ready(agentC))
...
```

Agent C

```
...
out(ready(agentC))
rd(ready(agentA))
rd(ready(agentB))
...
```



TuCSoN Style

ANY agent

```
...
out(ready)
rd(ready_all)
...
```

BASIC BARRIER SYNCHRONISATION in ReSpecT

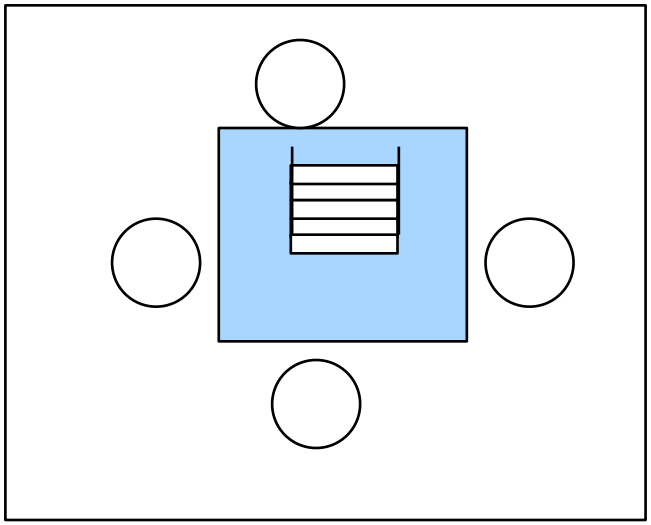
```
reaction(out(ready),( in_r(ready),
    in_r(ready_entities(N)), N1 is N+1,
    out_r(ready_entities(N1)) )).
reaction(out_r(ready_entities(N)),(
    in_r(ready_entities(N)),
    out_r(ready_entities(0)),
    rd_r(barrier_size(N)), out_r(ready_all))).
```

HYPOTHESIS

- Initial space content with `ready_entities(0)`
- `barrier_size(N)` tuple to specify number of coordinables to be synchronised

Resource Sharing / Allocation

- A dynamic/open set of agents accessing the same resource (ex: a printer) according to a coordination policy (ex: First Come First Served)



Each user agent

```
…
in(next_ticket(T))
T1 ← T + 1
out(next_ticket(T1))
in(turn(T))
  <use the resource>
out(turn(T1)
…
```

HYPOTHESIS
Initial space content includes the tuples:

```
next_ticket(0)
turn(0)
```

Good, but
- Changing the coordination policy
→ changing all the other entities
- Malicious/Failing agents?

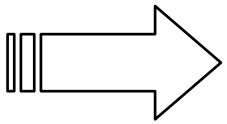
Resource Sharing / Allocation in TuCSoN (I)

- Encapsulating Sharing Policy
  - Scale down complexity to a synchronisation problem

Linda Style

Each user agent

```
…
in(next_ticket(T))
T1 ← T + 1
out(next_ticket(T1))
in(turn(T))
  <use the resource>
out(turn(T1)
…
```



TuCSoN Style

EACH USER

```
…
in(resource_token(<my name>))
<use the resource>
out(resource_token(<my name>))
…
```

SHARING COORDINATION LAWS in ReSpecT

```
reaction(in(resource_token(Who)),( pre,
  in_r(tickets(N)), N1 is N + 1,
  out_r(tickets(N1)),
  out_r(turn(Who,N)) )).
reaction(out_r(turn(Who,N)),(
  rd_r(current_turn(N)),
  out_r(resource_token(Who)) )).
reaction(out(resource_token(Who)),(
  in_r(resource_token(Who)),in_r(turn(Who,N)),
  in_r(current_turn(N)), N1 is N+1,
  out_r(current_turn(N1)) )).
reaction(out_r(current_turn(N)),(
  rd_r(turn(Who,N)),
  out_r(resource_token(Who)) )).
```

Resource Sharing / Allocation in TuCSoN (II)

- Changing / Adapting Sharing Policy
  - From FIFO strategy to LIFO strategy

unchanged behaviour for agents

Each user agent

```
…
in(resource_token(<my name>))
<use the resource>
out(resource_token(<my name>))
…
```

changing only the glue code

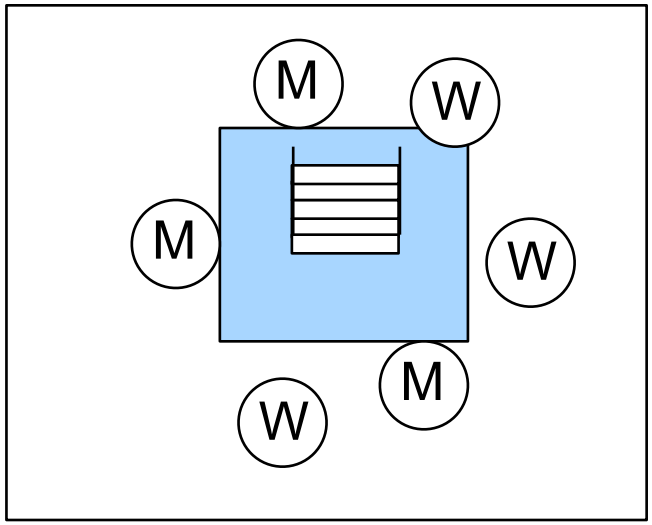
LIFO SHARING POLICY

```
reaction(in(resource_token(Who)),( pre,
  in_r(last(N)), N1 is N + 1,
  out_r(last(N1)),
  out_r(heap(Who,N1)),
  out_r(check) )).

… [OK, you got the idea]
```

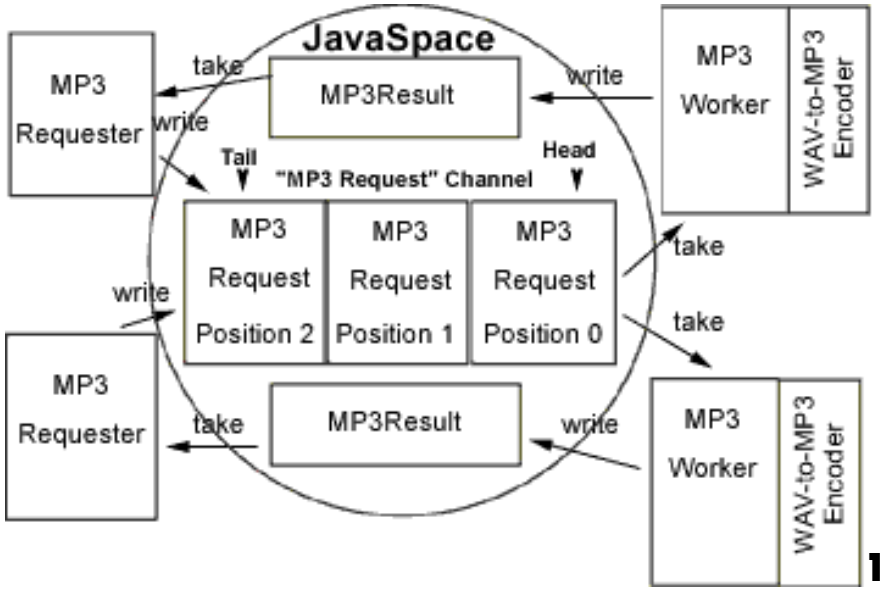
Task Allocation

- Task allocation to an open set of *workers*, with task request provided by an open set of *masters*, according to some policy
  - MP3 Service Case Study: building a distributed Internet-based MP3 encoding service
    - masters request WAV → MP3 conversion
    - workers provide the conversion
    - service provision policy: FIFO
      - possibly dynamically/adaptable



from the articles "Make Room for JavaSpaces" by Susan Hupfer – Java World electronic magazine, Jiniology Serie

Also in the book:
"Java Spaces: Principle and Patterns" AW.



Task Allocation: The Linda Approach

MP3 REQUESTER (master)

```
while (true) {
     acquireFromGUI(FileName)
     readRawData(FileName,RawData)
     in(tail(T))
     T1 ← T + 1
     out(tail(T1))
     out(mp3request(T1,FileName,
                    RawData,myId))
     in(mp3result(FileName,
                  ResultData,myId))
}
```

MP3 CONVERTER (worker)

```
while (true) {
     rd(tail(T))
     in(head(H))
     if (T<H){
          out(head(H))
     } else {
           H1 ← H + 1
        out(head(H1))
        in(mp3request(H,FileName,Data,
                      FromWho))
        MP3Data ← from_raw_to_data(Data)
        out(mp3result(FileName,
                      MP3Data,FromWho))
     }
}
```

good, but the coordination burden is almost upon the coordinables

- changing policy ➡ changing coordinables

-...

Task Allocation: The TuCSoN Approach

MP3 REQUESTER (master)

```
while (true) {
    acquireFromGUI(FileName)
    readRawData(FileName,RawData)
    out(mp3request(FileName,RawData,myId))
    in(mp3result(FileName,ResultData,myId))
}
```

MP3 REQUESTER (worker)

```
while (true) {
    in(mp3request(FileName,Data,FromWho))
    MP3Data ← from_raw_to_data(Data)
    out(mp3result(FileName,MP3Data,FromWho))
}
```

FIFO TASK ALLOCATION POLICY in ReSpecT

```
reaction(out(request(_,_,_)),(
  rd_r(workers_available(N)),
  N>0)).
reaction(out(request(Name,Data,From)),(
  rd_r(workers_available(N)),
  N == 0,
  in_r(tail(T)), T1 is T + 1, out_r(tail(T1)),
  in_r(request(Name,Data,From)),
  out_r(req_queue(T1,Name,Data,From)))).
reaction(in(request(Name,Data,From)),( pre,
  rd_r(head(H)),rd_r(tail(T)),
  T < H)).
```

```
reaction(in(request(Name,Data,From)),( pre,
  in_r(head(H)), rd_r(tail(T)),
  T >= H,
  H1 is H + 1, out_r(head(H1)),
  in_r(req_queue(H,N1,D1,F1)),
  out_r(request(N1,D1,F1)))).
reaction(in(request(_,_,_)),( pre,
  in_r(workers_available(N)),
  N1 is N + 1,out_r(workers_available(N1)))).
reaction(in(request(_,_,_)),( post,
  in_r(workers_available(N)),
  N1 is N - 1, out_r(workers_available(N1)))).
```