Situated Coordination

Distributed Systems L-A
Sistemi Distribuiti L-A

Andrea Omicini & Matteo Casadei
andrea.omicini@unibo.it, m.casadei@unibo.it

Ingegneria Due
Alma Mater Studiorum—Università di Bologna a Cesena

Academic Year 2008/2009
1. Extending ReSpecT
2. A Case Study
Outline

1. Extending ReSpecT

2. A Case Study
Situatedness & Coordination

Situatedness...

- essentially, strict coupling with the environment
- technically, the ability to properly perceive and react to changes in the environment
- one of the most critical issues in distributed systems
  - conceptual clash between pro-activeness in process behaviour and reactivity w.r.t. environment change
- still one of the most critical issues for artificial intelligence & robotics

... & coordination

- essentially, situatedness concerns interaction between processes and the environment
- technically, situatedness can be conceived as a coordination problem
  - how to handle and govern interaction between pro-active processes and an ever-changing environment
Goals

Overall goal of the research

- putting coordination models to test in the challenging context of situatedness

- understanding how classical coordination languages need to be extended to support the coordination of situated processes & distributed systems
ReSpecT tuple centres for environment engineering

- Distributed systems are immersed into an environment, and should be reactive to events of any sort
- Also, coordination media should mediate any activity toward the environment, allowing for a fruitful interaction

⇒ ReSpecT tuple centres should be able to *capture general environment events*, and to generally *mediate process-environment interaction*

Situating ReSpecT: extensions

- In [Casadei and Omicini, 2009], the ReSpecT language has been revised and extended so as to *capture environment events*, and *express general MAS-environment interactions*

⇒ ReSpecT captures, reacts to, and observes general environment events
⇒ ReSpecT can explicitly interact with the environment
ReSpecT tuple centres are extended to capture two classes of environmental events

- the interaction with sensors perceiving environmental properties, through environment predicate `get(<Key>,<Value>)`
- the interaction with actuators affecting environmental properties, through environment predicate `set(<Key>,<Value>)`

Source and target of a tuple centre event can be any external resource

- a suitable identification scheme – both at the syntax and at the infrastructure level – is introduced for environmental resources

Properties of an environmental event can be observed through the observation predicate `env(<Key>,<Value>)`
Environment communication

- The ReSpecT language is extended to express explicit communication with environmental resources.
- The body of a ReSpecT reaction can contain a *tuple centre predicate* of the form:
  - \( \langle \text{EnvResIdentifier} \rangle \ ? \ \text{get}(\langle \text{Key} \rangle, \langle \text{Value} \rangle) \)
    enabling a tuple centre to get properties of environmental resources.
  - \( \langle \text{EnvResIdentifier} \rangle \ ? \ \text{set}(\langle \text{Key} \rangle, \langle \text{Value} \rangle) \)
    enabling a tuple centre to set properties of environmental resources.
Transducers

- Specific environment events have to be translated into well-formed ReSpecT tuple centre events.
- This should be done at the infrastructure level, through a general-purpose schema that could be specialised according to the nature of any specific resource.
- A ReSpecT transducer is a component able to bring environment-generated events to a ReSpecT tuple centre (and back), suitably translated according to the general ReSpecT event model.
- Each transducer is specialised according to the specific portion of the environment it is in charge of handling—typically, the specific resource it is aimed at handling, like a temperature sensor, or a heater.
Outline

1. Extending ReSpecT

2. A Case Study
A set of real sensors are used to measure some environmental property (for instance, temperature) within an area where they are located. Such information is then exploited to govern suitably placed actuators (say, heaters) that can affect the value of the observed property in the environment. Sensors are supposed to be cheap and non-smart, but provided with some kind of communication interface – either wireless or wired – that makes it possible to send streams of sampled values of the environmental property under observation.

Accordingly, sensors are active devices, that is, devices pro-actively sending sensed values at a certain rate with no need of being asked for such data—this is what typically occurs in pervasive computing scenarios.

Altogether, actuators and sensors are part of a distributed system aimed at controlling environmental properties (in the case study, temperature), which are affected by actuators based on the values measured by sensors and the designed control policies as well.

Coordination policies can be suitably automated and encapsulated within environment artifacts controlling sensors and actuators.
Case Study: ReSpecT-based Architecture

- **Artifact**: Represented by a block with a <<sensor>>, <<actuator>>, and <<aggregator>>.
- **Agent**: Connected to the Artifact via a link.
- **Sensor/Tuple Centre**: Represents the ReSpecT system.
- **Physical Area**: Where the sensors and actuators are located.

Key Symbols:
- Red circles = group of sensors
- Black block = actuator
- White shape = transducer

External event stream:
- Connects to the Artifact.
Case Study: Artifact Structure

Artifacts are internally defined in terms of ReSpecT tuple centres:

- **<<sensor>> artifacts** wrapping real temperature sensors which perceive temperature of different areas of the room
- **<<actuator>> artifacts** wrapping actuators, which act as heating devices so as to control temperature
- **<<aggregator>> artifact** provides an aggregated view of the temperature values perceived by sensors spread in the room since it is linked to **<<sensor>> artifacts**:
  - **<<sensor>> artifacts** update tuples on **<<aggregator>> artifact** through *linkability*
Case Study: Sensor Artifacts

%(1)
reaction( get(temperature, Temp), from_env, (  
    event_time(Time), event_source(sensor(Id)),  
    out(sensed_temperature(Id,Temp,Time)),  
    tc_aggr@node_aggr ? out(sensed_temperature(Id,Temp)) )  
).

%(2)
reaction( out(sensed_temperature(_,Temp,_)), from_tc, (  
    in(current_temperature(_)),  
    out(current_temperature(Temp)) ) 
).

**Behaviour**

- Reaction (1) is triggered by external events generated by a temperature sensor
- Reaction (2) updates current temperature
Case Study: Aggregator Artifacts

%(4)

reaction( out(sensed_temperature(Id,Temp)), from_tc, ( 
in(total_temperature(OldTotalTemp),
in(sensed_temperature(Id,OldTemp)),
TotalTemp is OldTotalTemp - OldTemp + Temp,
out(total_temperature(TotalTemp),
rd(number_of_sensors(SensorNo),
AvgTemp is TotalTemp / SensorNo,
in(average_temp(_)), out(average_temp(AvgTemp)) )
).

Behaviour

- Reaction (4) keeps track of the current state of the average temperature
Case Study: Agents

Observable behaviour

Agents are goal-oriented and proactive entities that control temperature of the room

1. **get local information from sensor**
   
   tc_sens@node_i ? rd(current_temperature(Temp_i))

2. **get global information from aggregator**
   
   tc_aggr@node_aggr ? rd(average_temp(AvgTemp))

3. **deliberate action** by determining TempVar based on Temp_i and AvgTemp

4. **act upon actuators** (if TempVar ≠ 0)
   
   tc-heat_i@node_i ? out(change_temperature(TempVar))
Case Study: Actuator Artifacts

\%(3)\n\nreaction( out(change_temperature(TempVar)), from_agent, actuator_i ? set(temp_inc,TempVar) ).

Behaviour

When the controller agent deliberate an increment in the temperature

- a \texttt{tc-heat\textunderscore i@node\textunderscore i ? out(change_temperature(TempVar))} reaches the actuator artifact

- by reaction (3), a suitable signal is sent to the actuator, through the suitably-installed transducer
Conclusion

In this lesson we
- discussed the role of coordination languages for situatedness
- extended the ReSpecT coordination language to support situatedness
- showed a simple case study of a ReSpecT-based pervasive system for controlling environmental properties in physical areas—the example was actually built with physical sensors

To Do

- Generalising the abstraction of transducer in ReSpecT
- Experimenting situated ReSpecT to the modelling and design of other pervasive scenarios
1. Extending ReSpecT

2. A Case Study
Situated Coordination

Distributed Systems L-A
Sistemi Distribuiti L-A

Andrea Omicini & Matteo Casadei
andrea.omicini@unibo.it, m.casadei@unibo.it

Ingegneria Due
Alma Mater Studiorum—Università di Bologna a Cesena

Academic Year 2008/2009