MANAGEMENT OF Pervasive Data and Context Awareness

Proff. Fabio A. Schreiber – Letizia Tanca

Context management elements

• **Design time:**
  – Context modeling
  – Application domain modeling (data, functions)
  – Design of the relationship between the context model and the application domain
Design time

CONTEXT-AWARE DATA TAILORING
Context management elements

Run time:
- Context sensing (numeric observables)
- Context recognition (symbolic observables)
- Context feeding (binding)
- Context aware behaviour
Collecting context data

- Sensor data are used to collect “object” information
- They can also be used also to gather the current context
- Typical sensor-derived context data: time and location
- More examples:
  - light, latitude and time-of-the-day used to derive the season,
  - wind speed and orientation to derive vineyard situation, etc.

Context-aware sensor query

Suppose we want to monitor the temperature of some containers inside a container ship

BUT

We are only interested in those containers exposed at direct sunlight

Every container has a light sensor on the outside, and a temperature sensor in the inside, not directly connected to each other.

We only have to query those sensors which are in the light, and we are not interested in those sensors which are in the shade → context-aware reduction of the query space
More context-aware examples

Monitor the tampering sensor of the accessible containers when the ship is docked (GPS information): if tampering is detected monitor nearby containers at a higher frequency.

Photo by Patrick Boury (CC)
PerLa and Context management

- PerLa defines a language to manage, configure and maintain a pervasive system. In its original conception, PerLa does not provide context definition statements.
- Up to now, we are working towards providing PerLa with context definition statements.

**Key Idea**: extend the language to manage context

Managing Context: quick look

- During the analysis of the application domain, the designer identifies the “realistic set” of contexts.
- The designer must provide:
  - A context validity condition (a condition on numerical observable variables)
  - A set of actions to be performed when the validity condition becomes true
  - (possibly) a set of actions to be performed when the validity condition become false
  - The evaluation frequency of the validity condition
PerLa Context Management Functions

- **CDT Declaration**
  Allows to specify all the application-relevant dimensions and the values they can assume

- **Context Creation**
  Allows to declare a context on a defined CDT and control its activation

- **Context Detection**
  Allows to discover whether one of the defined context has become active or has been deactivated

- **Performing Context Actions**
  Executes the PerLa HLQ or LLQ specified for the active context

---

The updated PerLa Middleware

![Diagram](image-url)
Example: putting all together

Validity expr. "grapes_harvest"  VALID WHEN
SELECT GPS_VALUE, TIMESTAMP, ID_NODE
WHERE GPS_VALUE = ($LAT_X, $LONG_Y) AND
TIMESTAMP > "15 Sept 2009" AND
TIMESTAMP < "16 Oct 2009" AND
ID_NODE = $AN_ID OR
ANOTHER_ID...

ON ENABLE
CREATE OUTPUT STREAM
DataFromWineyard (SensorID ID, avgTemp FLOAT, avgHum FLOAT, varTemp FLOAT, varHum FLOAT)
AS HIGH: EVERY 1 h
SELECT AVG (temperature, 10 m),
AVG (humidity, 10 m),
VARIANCE (temperature, 10 m),
VARIANCE (temperature, 10 m)
FROM RawDataFromWineyard
GROUP BY SensorID

CREATE STREAM
RawDataFromWineyard
(SensorID ID,
temperature FLOAT,
humidity FLOAT)
AS LOW: EVERY 10 m
SELECT humidity, temperature
SAMPLING EVERY 60 s

ON DISABLE
DROP RawDataFromWineyard
REFRESH EVERY 10 m

Context in PerLa 14