Managing and Using Context Information within the PerLa Language

Fabio A. Schreiber - Letizia Tanca – Romolo Camplani – Diego Viganò

Politecnico di Milano
Dipartimento di Elettronica ed Informazione
Outline

• Introduction
• The CDT context model
• Context Management in PerLa
  – Language support
  – Contextual-block composition
• Examples
• Conclusions
Pervasive systems are widely adopted to monitor many kinds of physical phenomena.

Context-awareness plays a fundamental role since it allows, through the perception of the environment, to make the system autonomic w.r.t. environmental situations and changes.

Context must be managed both at design and run time.
Context management at *design time*

- Context modelling
- Application domain modelling (data, functions)
- Design of the relationship between the context model and the application domain.
Context management at design time

CONTEXT MODEL
The environment is modelled in terms of **dimension nodes**, concept (or value) nodes, attributes, and possibly constraints.

A **context element** is defined as $Dimension = Value$ and a context is a conjunction of context elements.

A context can be represented as a particular subtree of the CDT.
CDT model

Context in PerLa
It is important to separate between

- **NUMERIC OBSERVABLES**
- **SYMBOLIC OBSERVABLES**

**Example:**

**temperatures**

-20°C – 10°C COLD

11°C – 25°C MILD

26°C – 50°C WARM

Context in PerLa
Context management at runtime

- **Context-aware behavior**
- **Symbolic variables (context dim. values)**
- **Observable, numerical variables gathered from sensors**

Other sources of context elements

Context activation (through the association between contexts and “relevant system parts”)

Context in PerLa
Context management at \textit{runtime}

Apply the sensor query only to the sensors in context:

\texttt{phase = 'growth' AND risk='overheat' AND orientation='westward'}
Context management at runtime

- Context sensing (numeric observables)
- Context recognition (symbolic observables)
- Context activation
- Context-aware behaviour to be merged into a middleware and a language to manage pervasive systems hiding the complexity of handling different technologies

SELECT temperature, humidity
WHERE temp > 20
SAMPLING EVERY 1h
EXECUTE IF device_id > 2
Context-aware Sw Behaviour

Design/compile time

Run time

Context in PerLa
Contextual block structure

- ACTIVATION COMPONENT
- ENABLE COMPONENT
- DISABLE COMPONENT
- REFRESH COMPONENT
The PerLa Context Language

CREATE DIMENSION <Dimension Name>
[CHILD OF <Parent Node>] [CREATE ATTRIBUTE $<Attribute Name>] |
{CREATE CONCEPT <Concept Name> WHEN <Condition>
[EXCLUDES <Dimension Name>.<Concept Name>]
[CREATE ATTRIBUTE $<Attribute Name>] | [EVALUATED ON <Low Level Query>]}

PerLa Definition of contexts and action(s) to be performed

CREATE CONTEXT <Context Name>
ACTIVE IF <Dimension>= <Value> [AND <Dimension>= <Value>]*
ON ENABLE (<Context>): <PerLa Query>
ON DISABLE (<Context>): <PerLa Query>
REFRESH EVERY <Period>

Contextual Block

Enable component
Disable component
Example: Given the previous CDT

CREATE DIMENSION Role
CREATE CONCEPT Farmer
WHEN get_user_role()='farmer'
CREATE CONCEPT Oenologist
WHEN get_user_role()='oenologist'
CREATE CONCEPT Driver
WHEN get_user_role()='driver'

CREATE DIMENSION Risk
CREATE CONCEPT Disease
WHEN get_interest_topic()='disease'
CREATE CONCEPT Overheat
WHEN temperature > 30 AND brightness > 0.75;

.....
CREATE CONTEXT Growth_Monitoring
ACTIVE IF phase = 'growth' AND role='farmer' AND Risk='overheat'
ON ENABLE:
  SELECT temperature, humidity
  SAMPLING EVERY 120 s
  EXECUTE IF location = 'vineyard'
  SET PARAMETER 'alarm' = TRUE;
ON DISABLE:
  SET PARAMETER 'alarm' = FALSE;
REFRESH EVERY 24 h;
PerLa language and middleware

- CDT creation and maintenance
- Context detection
- Perform context actions

The CM associates to every dimension of the CDT a table that contains the values of every numeric observable sampled from the environment and that is used in relation with the symbolic observables which describe that dimension.
PerLa language and middleware

In the previous example we declared a context that includes the observable “overheat” (declared using the numeric temperature and brightness):

<table>
<thead>
<tr>
<th>ID</th>
<th>Temperature</th>
<th>Brightness</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>28</td>
<td>0.60</td>
</tr>
<tr>
<td>3</td>
<td>31</td>
<td>0.71</td>
</tr>
<tr>
<td>4</td>
<td>33</td>
<td>0.80</td>
</tr>
</tbody>
</table>

The context can be considered as active for all the sensors for which the rule is true, and the context-aware actions will be performed only on them.
Contextual block automatic composition

- **Problem**: given a CDT, the number of possible contexts exponentially grows with the dimensions number 😞
- E.g.: 5 dimensions, 3 concepts/dim (average) ➔ >500 contexts!

- **Solution**: automatic composition of the contextual block, based on partial components:

  CONTEXTUAL BLOCK COMPOSITION

  DESIGN TIME  ⊕_D

  RUNTIME  ⊕_R
Contextual block automatic composition

Activation component
Enable component
Disable component
Refresh component

Context in PerLa
The designer’s tasks

- Components association (components library)
- Generation of all the possible contextual blocks
- Possible manual adaptation

...{CREATE CONCEPT <Concept Name> WHEN <Condition>
   WITH ENABLE COMPONENT <PerLa_Query>
   WITH DISABLE COMPONENT <PerLa_Query>
   WITH REFRESH COMPONENT <Period>
   ...

- Building of a composite contextual blocks library
- Verification of the composite block correctness (QueryAnalyzer/Optimizer)
- If required for peculiar situations

Design time
Design time vs. run time composition

- **Design time:**
  - Fully controlled by the designer
  - Static vision

- **Run time:**
  - Autonomic behaviour of the system
    - Contextual blocks are composed only for the active context
  - No further changes allowed
  - Performance issues (more contexts can be simultaneously active causing frequent context switching!)

A suitable trade-off is a designer’s choice based on the system requirements
An example

- WITH ENABLE COMPONENT:
  - SELECT MAX(temperature)
  - SET PARAMETER 'alarm' = TRUE;
- WITH DISABLE COMPONENT:
  - SET PARAMETER 'alarm' = FALSE;
- WITH REFRESH COMPONENT:
  - 5s

ON ENABLE:
- SELECT MAX(temperature), equipment_id
  - SAMPLING EVERY 5s
  - SET PARAMETER 'alarm' = TRUE;
ON DISABLE:
- SET PARAMETER 'alarm' = FALSE;
  - REFRESH EVERY: 1s

- Clause optimization (e.g: SELECT)
- Highest refresh frequency selection (lowest time constant)
Example 1: office risk management

• Suppose to have a pervasive system to monitor the potential risks at the DEI Department of Politecnico di Milano
Example 1: office risk management

- Suppose to have a pervasive system to monitor the potential risks at the DEI Department of Politecnico di Milano
Example 1: office risk management

- Suppose to have a pervasive system to monitor the potential risks at the DEI Department of Politecnico di Milano

- We want to define a context-aware behaviour to control:
  - Fire
  - Earthquake
Example 1: office risk management

• Suppose to have a pervasive system to monitor the potential risks at the DEI Department of Politecnico di Milano

\[
\text{Fire Monitoring} = (\text{Role} = \text{Safety responsible}) \land (\text{Risk} = \text{Fire}) \land (\text{Location} = \text{Laboratory})
\]
Example 1: office risk management

- Suppose to have a pervasive system to monitor the potential risks at the DEI Department of Politecnico di Milano

\[ \text{Earthquake Monitoring} = (\text{Role} = \text{Professor}) \land (\text{Location} = \text{Office}) \land (\text{Risk} = \text{Earthquake}) \]
Fire Risk concept

CREATE CONCEPT Fire
WHEN temperature > 40
WITH ENABLE COMPONENT:
SELECT MAX(temperature)
SET PARAMETER 'alarm' = TRUE;
WITH DISABLE COMPONENT:
SET PARAMETER 'alarm' = FALSE;
WITH REFRESH COMPONENT: 5s

Earthquake Risk concept

CREATE CONCEPT Earthquake
WHEN delta_x > 2 AND delta_y > 3
WITH ENABLE COMPONENT:
SELECT delta_x, delta_y;
WITH REFRESH COMPONENT: 1s
Example 1: office risk management

**Location**

- **Office**
  - CREATE CONCEPT: Office
  - WHEN `get_current_location()` = 'Office'
  - WITH ENABLE COMPONENT:
    - SELECT `office_floor`
  - SAMPLING EVERY 2m

- **Laboratory**
  - CREATE CONCEPT: Laboratory
  - WHEN `get_current_location()` = 'Laboratory'
  - WITH ENABLE COMPONENT:
    - SELECT `equipment_id`
  - SAMPLING EVERY 5s
  - WITH REFRESH COMPONENT: 1s

- **Conference Room**
  - CREATE CONCEPT: Conference Room
  - WHEN `get_current_location()` = 'Conf. room'
  - WITH ENABLE COMPONENT:
    - SELECT `room_name`
  - SAMPLING EVERY 20s
  - WITH REFRESH COMPONENT: 1s

**Role**

- **Professor**
  - CREATE CONCEPT: Professor
  - CREATE ATTRIBUTE $Professor_Id
  - WHEN `get_current_role()` = 'Professor'
  - WITH ENABLE COMPONENT:
    - SELECT `professor_name`, `professor_surname`
    - WHERE 'professor_id' = CDT.Role.Professor_ID

- **Safety_Respnsible**
  - CREATE CONCEPT: Safety_Respnsible
  - WHEN `get_current_role()` = 'Safety Responsible'

- **Janitor**
  - CREATE CONCEPT: Janitor
  - WHEN `get_current_role()` = 'Janitor'

Context in PerLa
Example 1: office risk management

\[
\text{Fire Monitoring} \equiv A \oplus G \oplus D
\]

Context in PerLa
Example 1: office risk management

Fire_Monitoring (Role = Safety responsible) ∧ (Risk = Fire) ∧ (Location = Laboratory)

Fire_Monitoring ≡ A ⊕ G ⊕ D

CREATE CONTEXT Fire_Monitoring
ACTIVE IF (temperature > 40 AND ...)
ON ENABLE:
   SELECT MAX(temperature), equipment_id
   SET PARAMETER 'alarm' = TRUE;
   SAMPLING EVERY 5s
ON DISABLE:
   SET PARAMETER 'alarm' = FALSE;
   REFRESH EVERY 1s;
Example 2: vineyard monitoring

$Growth\_Monitoring \equiv (\text{Role} = \text{Farmer}) \land (\text{Phase} = \text{Growth}) \land (\text{Disease}.\text{Type} = 3) \land (\text{Disease}.\text{AffectedHectares} = 200)$

$Transport\_Monitoring \equiv (\text{Role} = \text{Driver}) \land (\text{Phase} = \text{Transport}) \land (\text{Risk} = \text{Overheat})$
Example 2: vineyard monitoring

\[\text{Growth Monitoring} \equiv (\text{Role} = \text{Farmer}) \land (\text{Phase} = \text{Growth}) \land (\text{Disease.Type} = 3) \land (\text{Disease.AffectedHectares} = 200)\]

\[\text{Transport Monitoring} \equiv (\text{Role} = \text{Driver}) \land (\text{Phase} = \text{Transport}) \land (\text{Risk} = \text{Overheat})\]
Example 2: vineyard monitoring

\[ \text{Growth Monitoring} \equiv A \oplus G \oplus E \oplus F \]

\[ \text{Transport Monitoring} \equiv B \oplus I \oplus D \]
Example 2: vineyard monitoring

**Growth Monitoring** \( \equiv A \oplus G \oplus E \oplus F \)

**Transport Monitoring** \( \equiv B \oplus I \oplus D \)

```sql
CREATE CONTEXT Growth_Monitoring
ACTIVE IF phase = 'growth' AND role='farmer' AND
Disease.Type=3 AND
Disease.Affected_Hectares = 200 REFRESH EVERY 1 d;
ON ENABLE (Growth_Monitoring)
SELECT humidity,temperature
WHERE humidity > 0 AND temperature > 0
SAMPLING EVERY 6 h
EXECUTE IF EXISTS humidity,temperature AND
location='vineyard'
ON DISABLE (Growth_Monitoring)
DROP CONTEXT Growth_Monitoring;

CREATE CONTEXT Transport_Monitoring
ACTIVE IF phase = 'transport' AND role='driver' AND
Risk='overheat' REFRESH EVERY 24 h;
ON ENABLE (Transport_Monitoring)
SELECT temperature,gps_latitude,gps_longitude
WHERE temperature > 30
SAMPLING EVERY 120 s
EXECUTE IF location = 'truck.departing_zone'
SET PARAMETER 'alarm' = TRUE;
ON DISABLE (Transport_Monitoring)
DROP Transport_Monitoring;
SET PARAMETER 'alarm' = FALSE;
```
## Comparison with Active DB

<table>
<thead>
<tr>
<th><strong>ACTIVE DATABASES</strong></th>
<th><strong>PerLa FOR CONTEXT</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>EVENT data modification: insert, delete, update</td>
<td>EVENT general system events, clock, …</td>
</tr>
<tr>
<td>CONDITION (optional) SQL predicate</td>
<td>CONDITION context definition formula</td>
</tr>
<tr>
<td>ACTION sequence of SQL statements (or extensions, e.g. PL/SQL in Oracle)</td>
<td>ACTION Data, code, services tailoring, whatever action on the physical system</td>
</tr>
</tbody>
</table>

Context in PerLa
Comparison with Active DB

<table>
<thead>
<tr>
<th><strong>ACTIVE DATABASES</strong></th>
<th><strong>PerLa FOR CONTEXT</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>COUPLING (immediate/deferred)</td>
<td>ONLY IMMEDIATE</td>
</tr>
<tr>
<td>ATOMIC/INTERRUPTIBLE ACTIONS</td>
<td>ONLY ATOMIC</td>
</tr>
<tr>
<td>EVENT CONSUMPTION (never, local, global)</td>
<td>EVENT CONSUMPTION (never, only at context change)</td>
</tr>
<tr>
<td>CONFLICT RESOLUTION (serial/parallel)</td>
<td>CONFLICT RESOLUTION (serial, managed by priority policies)</td>
</tr>
</tbody>
</table>

Context in PerLa
Comparison with programming languages

Philosophy without Science is empty, Science without Philosophy is blind

I. Kant

PARAPHRASE

Programs without Data are empty, Data without Programs are blind

F. A. Schreiber
## Comparison with programming languages

<table>
<thead>
<tr>
<th></th>
<th><strong>PerLa</strong></th>
<th><strong>COP</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Context</strong></td>
<td>Numeric observables $\rightarrow$ Symbolic observables (Coutaz, CACM, 2005)</td>
<td>Any computationally accessible information (Hirschfeld, JOT, 2008)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Numeric observables $\rightarrow$ Symbolic observables</td>
</tr>
<tr>
<td><strong>Context model</strong></td>
<td>Context Dimension Tree (CDT)</td>
<td>Left to application software</td>
</tr>
<tr>
<td></td>
<td>Context Element $\equiv {Dimension_i = Value_i}$</td>
<td>Multiple active contexts</td>
</tr>
<tr>
<td></td>
<td>Multiple active contexts</td>
<td></td>
</tr>
<tr>
<td><strong>Context declaration</strong></td>
<td>Contextual Block</td>
<td>Left to application software</td>
</tr>
<tr>
<td></td>
<td>• activation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Enabling</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Disabling</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Change detection</td>
<td></td>
</tr>
<tr>
<td><strong>Context sensing and recognition</strong></td>
<td>LLQ from sensors</td>
<td>Layer activation mechanisms</td>
</tr>
<tr>
<td></td>
<td>User declared variables GET_ACTIVE IF REFRESH EVERY</td>
<td></td>
</tr>
<tr>
<td><strong>Contextual actions</strong></td>
<td>ON ENABLE ={TRUE/FALSE} $\rightarrow$ LLQ/HLQ/AQ</td>
<td>Behavioural variations</td>
</tr>
<tr>
<td></td>
<td>ON DISABLE ={TRUE/FALSE} $\rightarrow$ LLQ/HLQ/AQ</td>
<td>Partial methods</td>
</tr>
<tr>
<td></td>
<td>Partial components associated with each Context-element</td>
<td>WITH</td>
</tr>
<tr>
<td></td>
<td>WITH {ENABLE/DISABLE/REFRESH} COMPONENT</td>
<td>WITHOUT</td>
</tr>
</tbody>
</table>

Context in PerLa
Conflict resolution

Context in PerLa
Performance evaluation

- In its original configuration, PerLa’s middleware scales linearly w.r.t. the operations (i.e.: LLQs, HLQs, AQs) that are performed on the deployed devices.

- The creation of the CDT and the search for active contexts scale linearly too (simple lookup control in every table) and thus do not impact PerLa’s linear behaviour.
## Conclusions

- PerLa allows for an easy and rapid passage between *numeric* and *symbolic* observables.

- Moreover it allows to model and define the context with the preferred granularity, and to actuate context-aware actions within the same language.

- It offers design support tools through the contextual block composition both at design and at run time.

- The PerLa system is operating in a rockfall monitoring project in Mt. San Martino in Lecco (MI) since April 2010.

- We are currently focused on the following issues:
  - the management of possible context conflicts.
  - the management of context evolution
  - Assessing C-A systems stability
Further readings

• **on CDT:**
  
  
  – Bolchini C., Quintarelli E., Tanca L. - Carve: Context-aware automatic view definition over relational databases - Information Systems, Accepted manuscript (unedited version available online: 12-MAY-2012).


• **on PerLa:**
  

THANK YOU