Pervasive systems

- The computer becomes part of the **infrastructure**
  - the middleware of a pervasive system hides the heterogeneity of hundreds of devices making them transparent to the application
- The perception of the environment makes the system **autonomic** and **proactive**
  - context-aware
  - reactive
  - self-adapting
The Internet of Things

- More objects are becoming embedded with sensors and gaining the ability to communicate.

- The resulting information networks promise to create new business models, improve business processes, and reduce costs and risks.

Motivations

- Many large applications require the decomposition of an information base into (possibly overlapping) subsets.
- Content and services available at different sources and places, and user is an integral part of numerous applications, interacting with:
  - service providers, product sellers, governmental organisations, friends and colleagues, sensing devices
  - users should not be confused by information noise
  - the information amount should be reduced in order to comply with the physical limitations of small systems

Keywords for pervasive systems: mobility, context-dependence, adaptability, multimodality, multi-channel delivery, embedded systems, ubiquity, navigation, incremental discovery, sensors, power-awareness, computational power, storage space...

Context

- derived from the Latin **CUM** (with or together) and **TEXERE** (to weave)
- a context is not just a profile, but an active process dealing with the way humans weave their experience within their whole environment, to give it meaning
- **Context is any information that can be used to characterize the situation of an entity** [Dey], where an entity is a person, a place, or an object that is considered as relevant for the interaction between a user and an application.
A motivating scenario

Natural history museums visitors, endowed with a portable device which reacts to changes of context:

- adapting the user interface to the different abilities of the visitor – from low-sighted people to very young children –;
- providing different information contents based on the different interests/profiles of the visitor (geology, paleontology, ... scholar, journalist, ...), and on the room s/he is currently in;
- learning, from the current situation and the previous choices performed by the visitor, what information s/he is going to be interested in next;
- providing the visitor with appropriate services – to purchase the ticket for a temporary exhibition, or to reserve a seat for the next in-door show on the life of dinosaurs –;
- deriving location information from sensors which monitor the user environment;
- providing knowledge of the surrounding people in terms of their roles and respective contexts, as related to the user;
- providing active features within the various areas of the museum, which alert visitors with hints and stimuli on what is going on in each particular ambient.

Workshop on Using Knowledge in its Context (IJCAI '93)

- It is generally accepted that knowledge has a contextual component
- <...> this is rarely represented explicitly in available knowledge representation systems and is not used in subsequent processing of knowledge
- Acquisition, representation and exploitation of knowledge in context would have a major contribution in:
  ✓ knowledge representation, ✓ knowledge acquisition, ✓ explanation, ✓ maintenance, ✓ documentation, ✓ learning, ✓ human-computer communication ✓ validation or verification
- A computational capability to understand, represent and reason about context will be very valuable for, and of immense benefit to, many (AI) problems.
Accordingly, in general...

- Context can contribute to the meaning that must be inferred from the adjacent world.
- Such meaning ranges from the references intended for indefinite indications such as “take that” to the shared reference frame of ideas and objects that are suggested by a situation.
- Context goes beyond immediate binding of variables (semantics) to the establishment of a framework for communication based on shared experience.
- Such a shared framework provides a collection of roles and relations to organize meaning for a piece of information.

Underlying, implicit principles

- **Principle 1 (Locality):** reasoning uses only part of what is potentially available (e.g., what is known, the available inference procedures). The part being used while reasoning is what we call context (of reasoning);
- **Principle 2 (Compatibility):** there is compatibility among the kinds of reasoning performed in different contexts.

*(Chiara Ghidini, Fausto Giunchiglia, Local Models Semantics, or Contextual Reasoning = Locality + Compatibility, Artificial Intelligence, 2001)*
Workshop on Using Knowledge in its Context (IJCAI ’93)

The COGNITIVE SCIENCE view:
• context is used to model interactions and situations in a world of infinite breadth, and human behavior is key in extracting a model.

The ENGINEERING view:
• context is useful in representing and reasoning about a restricted state space within which a problem can be solved.

The engineering view is subsumed by the cognitive science view.

The six "W" questions of context

• What is context?
• Who might benefit from an awareness of their context; whose context is important to whom, or to what?
• Where can an awareness of context be exploited?
• When is context-awareness useful?
• Why are context-aware applications useful?

• Answers to these five questions underpin the higher level, meta-question of:
• How do we implement context-awareness so that we can develop context-aware applications?

**What is context?**

Different meanings in different realms of Computer Science:

Is context a matter of….?....

---

**Is context a matter of….?**

COMMUNICATION:

- capability to *adapt content presentation* to different channels or to different devices (system communication WITH the users)

- agreement and shared reasoning among peers (communication AMONG users or systems)

- building *smart environments* (system communication with the ENVIRONMENT)
Is context a matter of….?

SITUATION AWARENESS:
• modeling location and environment aspects (physical situation)
• modeling what the user is currently doing
• supporting autonomous behaviours of the system
• making the user interaction implicit

Is context a matter of….?

MANAGING KNOWLEDGE CHUNKS:
• determining the set of application/situation-relevant DATA
• determining the set of application/situation-relevant SERVICES
• determining the set of application/situation-relevant BEHAVIORS
Summary of the “hoW’s”

MODELING and REPRESENTATION:
- Orthogonal context modeling
- Multi-application model
- Context elements (dimensions): time, space, role, more...
- Space/Time coordinates (relative or absolute)
- User profile (e.g. role or features based)
- Variable context granularity
- Type of formalism: parameter-values pairs, rule-based, ontological, graph-based NOT independent of the application
- Valid-context constraints

Summary of the “hoW’s”

DINAMICITY MANAGEMENT
- On-line context changes
- On-line model changes
- Inter-context (dynamic) constraints
- Context history
- Automatic learning features

FURTHER FEATURES
- Context construction (distributed or centralized)
- Context quality monitoring
- Ambiguity/Incompleteness mgmt.
Context management lifecycle

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

• Run-time loop:
  ◦ Context sensing (numeric observables)
  ◦ Context recognition (symbolic observables)
  ◦ Context feeding (binding)
  ◦ Context aware behavior

Design time

CONTEXT MODEL

DATA

MOBILE APPLICATIONS

RULES

DOCUMENTS

SERVICES
Run-time

Context-aware behavior

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

Symbolic variables (context dim. values)

Observable, numerical variables gathered from sensors

Other sources of context elements

Software and service “hooking”

Possible associations:

• **Context** \(\rightarrow\) *(associated to)* \(\rightarrow\) set of ECA rules
  - Rules activate and deactivate behaviors

• **Context** \(\rightarrow\) *(associated to)* \(\rightarrow\) set of pairs <class, role>
  - Each object takes a context-aware role

• **Context** \(\rightarrow\) *(associated to)* \(\rightarrow\) set of program “variations”
  - An agent triggers context-aware variations which change process behavior

• ....
The data-centric point of view
A contribution from the database research area

CHALLENGES:
• Involved data volumes
• Data heterogeneity
• Data dynamicity
• Data distribution
• Scalability of the personalization solutions

ANSWERS:
• Reduction of data volumes
• Data integration

DESIGN-TIME
Context-aware data tailoring

- Tailoring based on context analysis
- Ambient (or Context) Dimensions:
  - different points of view the device data are viewed from
  - they drive the portion of data to be stored on the portable device
  - views over the global schema

The medicare application: a simple schema

PATIENT (SSN, FName, LName, Sex, BirthD, DeathD, Address, City, State, Zip, Phone, BloodType, Notes, MCUID, Booklet, DocID)

MEDICAL CARE UNIT (ID, Name, Address, City, State, Zip, Phone, Type)

SERVICE (ID, Name, Tipology, Difficulty, Period)

USES (MCUID, SERVICEID)

PRESCRIPTION (SSN, DRUGID, Mode, Dosage, Administration, StartDate, EndDate, Comments)

DRUG (ID, Name, Posology, Ingredients, SideEffects, Manufacturer, Comments)

DRUG IN PHARMACY (Drugid, Pharid)

PHARMACY (ID, Name, Address, City, State, Zip, Phone, OpeningHrs)
CONTEXT DIMENSIONS

INTUITIVE DIMENSIONS

• HOLDER
  ✓ TYPE (ROLE) OF USER CARRYING/USING THE DEVICE
• INTEREST TOPIC
  ✓ PARTICULAR ASPECT/SUBJECT THE USER IS INTERESTED IN AT A CERTAIN MOMENT
• SITUATION
  ✓ PHASE OF THE APPLICATION’S LIFE
• INTERFACE
  ✓ TYPE OF ACCESS TO THE DATABASE CONTENTS
• TIME
  ✓ RELATIVE OR ABSOLUTE
• SPACE
  ✓ RELATIVE OR ABSOLUTE
• DATA OWNERSHIP
  ✓ ACCESS RIGHTS TO THE STORED DATA

DIMENSION VALUES (1)

*holder* dimension in the medical care application

✓ *doctor*
  • doctors will hold information about all their patients,

✓ *patient*
  • patients will only hold information related to themselves, maybe at a finer level of detail.

✓ *each of the possible other values of this dimension (e.g. hospital administrator)*.
DIMENSIONS VALUES (2)

*interest topic* dimension
- prescriptions
- chronic diseases
- ...

*situation* dimension
- regular, i.e. ordinary patient’s state
- hospitalized
- rehabilitation state
- ...

*interface* dimension
- human
- machine
- ...

Context dimension tree
Medical Database interest topics

Relevant areas

*Contexts* are derived from the CDT

- `<patient, chronic diseases, human, hospital>` contains all the information needed by a patient at the hospital w.r.t. his/her chronic diseases (if any).
- `<patient, prescriptions, human, regular>` contains all the information needed by a patient in a normal situation, w.r.t. his/her prescriptions (if any).
- `<doctor, prescriptions, human, regular>` contains all the information needed by a doctor regarding all his/her regular patients’ prescriptions.

!! attention to the context’s actual significance

- `<doctor, accounting, machine, hospital>` makes little sense in view of the application semantics
Relevant areas

<patient,prescriptions,hospital,*,*,*>  

CREATE VIEW PAT-PRESC-HOSP AS

SELECT P.SSN, P.FName, P.LName, DRUG.Name AS DrugName, Posology, SideEffects, Mode, Dosage, Administration, StartDate, EndDate, Comments, MCU.Name, MCU.Address, MCU.City, MCU.State, MCU.Zip, MCU.Phone, MCU.Type

FROM PATIENT P, DRUG, PRESCRIPTION, MEDICAL CARE UNIT MCU

WHERE P.SSN = PR.SSN AND PR.DRUGID = DRUG.ID
AND P.MCUID = MCU.ID AND MCU.Type = "hospital"

Relevant area instantiation
(specific user and device)

SELECT *
FROM PAT-PRESC-HOSP
WHERE SSN = "930029747"
AND MCUID.NAME = "Mt. Sinai"
Relevant area instantiation
(runtime)

w.r.t. time and space:
SELECT *
FROM PAT-PRESC-HOSP
WHERE SSN = "930029747" AND
StartDate < Now() AND EndDate > Now()
AND City = ThisCity()

the order_by clause can be used to limit the cardinality to the topmost n entries
• the most recent n (on time dimension)
• the nearest n (on space dimension)
• the cheapest n (on interest_topic dimension)
• ...

Data ownership

• Concerns read, update, delete, and insert access rights to the vsdb information, which might be different depending on the user categories
• Ownership views on relevant areas
A more complex example: MSA

- The scenario is the University Everyday Life (MSA)
- Users (in this small example): Professors, Students, Visitors
- Provide context-aware data on mobile terminals (Smartphones, PDAs) and standard devices (Desktop, Laptop) about:
  - Restaurants and bars in the area surrounding the university (each subdivision)
  - Free rooms (both to be reserved or just to be used)
  - Courses
  - Information about seminars and events at the Department
  - News about professors (schedule changes, new materials)
  - Data sources are heterogeneous, distributed, independent, possibly transient, possibly partially overlapping

Context dimension tree: a more expressive representation

Context-Dimension Tree:
- representation independent
- extensible
- granularity and (useless-context) constraints support
The archaeological site example

The real-estate example

OWNER(IdOwner, Name, Surname, Type, Address, City, PhoneNumber)
ESTATE(IdEstate, IdOwner, Category, Area, City, Province, RoomsNumber,
    Bedrooms, Garage, SquareMeters, Sheet, CadastralMap)
CUSTOMER(IdCustomer, Name, Surname, Type, Budget, Address, City, PhoneNumber)
AGENT(IdAgent, Name, Surname, Office, Address, City, PhoneNumber)
AGENDA(IdAgent, Data, Hour, IdEstate, ClientName)
VISIT(IdEstate, IdAgent, IdCustomer, Date, ViewDuration)
SALE(IdEstate, IdAgent, IdCustomer, Date, AgreePrice, Status)
RENT(IdEstate, IdAgent, IdCustomer, Date, RatePrice, Status, Duration)
PICTURE(IdPicture, IdEstate, Date, Description, FileName)
Real-estate context dimensions

<table>
<thead>
<tr>
<th>dimension</th>
<th>Meaning</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>role</td>
<td>The actors using the system.</td>
<td>CEO, agency manager, agent.</td>
</tr>
<tr>
<td>interest topic</td>
<td>The areas of interest for the possible users of the application.</td>
<td>&quot;agencies&quot;, i.e., information about agencies (and one in particular) that can be controlled by the CEO, &quot;agents&quot;, i.e., information about agents that can be viewed by the CEO or by the agency’s manager, &quot;customers&quot;, i.e., information about sellers and buyers that can be viewed by the agent and manager, and &quot;properties&quot;, i.e., all the knowledge about estates to be sold or rented. This last interest topic can be further decomposed w.r.t. two different criteria: commercial/residential estates or rented/sold properties.</td>
</tr>
<tr>
<td>situation</td>
<td>Phases of the application life.</td>
<td>The user is consulting historic data when at the office, i.e., inoffice as opposed to the onsite situation, when, for instance, an agent is showing an apartment to a prospective customer.</td>
</tr>
<tr>
<td>time</td>
<td>Temporal indication based on the current time. Time can be relative or absolute and its granularity may vary.</td>
<td>In our example we have chosen a relative view of time, and allowed two choices: the current day, or a variable time interval centered on the current instant, suitable for data analysis.</td>
</tr>
<tr>
<td>space</td>
<td>A location indication, normally referring to the place where the user is currently located. Space can be relative or absolute, and its granularity may vary.</td>
<td>In our example, &quot;here&quot; or &quot;this city&quot; are relative space data, while &quot;Marina del Rey district in L.A.&quot; is absolute.</td>
</tr>
<tr>
<td>interface</td>
<td>Indication of channel or presentation for delivering information.</td>
<td>In some application cases, some data have to be used by humans, directly perusing text and multimedia information, but in others data could be managed by electronic devices solely, requiring only compact codes.</td>
</tr>
</tbody>
</table>

The real-estate context dimension tree

[Diagram showing the real-estate context dimension tree]
Real-estate context examples

<table>
<thead>
<tr>
<th>Context</th>
<th>Corresponding data</th>
</tr>
</thead>
<tbody>
<tr>
<td>[col:agent</td>
<td>Agent.id], interest:topic:estate, situation:site, time:today, location:city($c.id)]</td>
</tr>
<tr>
<td>[col:agent</td>
<td>ID], interest:topic:sales, time:month, location:city($c.id)]</td>
</tr>
<tr>
<td>[col:manager</td>
<td>ID], interest:topic:agents, situation:in.office, time:range(10am-5pm)]</td>
</tr>
<tr>
<td>[col:agent</td>
<td>Agent.id], interest:topic:customers, situation:in.office, location:city($c.id)]</td>
</tr>
</tbody>
</table>

The agency manager, when in the office
Terminology and notation

\[ T = (N, E, r) \]

\[ N = N_D \cup N_C \quad \text{dimensions (black), concepts (white)} \]

- The root is a *concept node*
- The root’s children are the *top dimensions*
- Each generation contains *nodes of the same color*

Formal definition

- Colors are *alternated* while descending the tree:
  \[ \forall e = (n, m) \in E, \quad \text{either } n \in N_D \land m \in N_C \text{ or } n \in N_C \land m \in N_D \]
- It is possible to add a parameter to concept (white) nodes and leaf dimension (black) nodes.
- White node parameters indicate how to select a specific set of data instances
  e.g. the *agent_id* for the *agent* role
- Leaf black node parameters indicate a selection parameter whose instances represent the possible values of that (sub-)
  dimension
  e.g. the *price_range_var* for the dimension *price range*
- Dimension nodes without concept children *must have* a parameter
- Dimension nodes with concept children *do not* have a parameter
• A context is formalized as a conjunction of propositions expressing context elements, each of them of the form 
dim name = value
• value can be
  ➢ a concept node or
  ➢ a concept node filtered by the value of its parameter or
  ➢ the value for a parameter of a sub-dimension (black) leaf node
• The values for each context element may be at any level in the tree, thus allowing for different levels of granularity.

• Sibling white nodes are mutually exclusive in a context, since they represent orthogonal concepts.
• When building a context, for each top dimension, at each level, only one white node among each set of siblings, and any number of black siblings may be included.
• A context can lack some dimension value(s): this means that those dimensions are not taken into account to tailor data, i.e., the view corresponding to that context does not filter the data for these dimension(s).
Example

- Let us consider the situation of an agent (whose id. is 23564), ready to take prospective buyers to visit the residential estate properties located in the “Piola” area ($zone id="Piola")
- The current context C is the conjunctive propositional formula:

\[
\text{role} = \text{agent}(\text{"23564")}
\land \text{category} = \text{residential}
\land \text{type} = \text{sale}
\land \text{zone} = \text{"Piola"}
\land \text{situation} = \text{on site}
\land \text{time} = \text{today(getdate())}
\]

Context Modeling

- Not all configurations make sense: e.g., there is no point in combining the visitor’s role with the courses data
- Constraints are specified over the tree by means of a standard logical formula
- The most common constraints are the forbid (or “useless-context”) constraints
Constraints

• The designer can express constraints or preferences on the possible combinations of the context elements.

• When we combinatorially generate the complete set of contexts, many contexts get discarded.

• Constraints are expressed by means of formulae of propositional logic:

$$\neg(\bigwedge {\text{context\_element\_proposition}})$$

where a context element proposition is:

✓ a proposition of the form \( \text{dim name}=\text{value} \) or

✓ a disjunction of such propositions.

Interesting constraints

• *useless context (forbid)* constraints allow the context designer to specify configurations that are not significant, thus discarding those that would represent semantically meaningless context situations or would be irrelevant for the application.

In the MSA:

$$\neg (\text{role}=\text{visitor} \land (\text{Desc(type}=\text{courses}) \lor \text{type}=\text{courses}))$$

In the real estate example:

$$\neg (\text{role}=\text{CEO} \land (\text{Desc(situation}=\text{on\_site}) \lor \text{situation}=\text{on\_site}))$$
Interesting constraints

- **dimension independent constraints** are used when the values of a (sub-) dimension do not influence the views associated with contexts containing some values for other dimensions. In the agency example:

\[
\neg (role = \text{CEO} \land \\
\land (situation = \text{in\_office} \lor situation = \text{on\_site}))
\]

The data associated with the CEO's role are independent of the situation.

Interesting constraints

- **preferred-detail constraints** allow the designer to express the level of detail to be preferred for a dimension, considering the other dimension values. In the agency example:

\[
\neg (role = \text{CEO} \\
\land \text{Desc(interest\_topic = estates)})
\]

The CEO has access to all the data related to estates, other roles may be interested in lower level details.
Recall what we want to do

Node-based area assignment

- The relevant area, or view, related to a context $C$, is denoted by $R_{cl(C)}$.
- Assigning relevant areas to all possible (valid) contexts of a tree is a very time-consuming task.
- Less time-consuming, but more difficult from a conceptual viewpoint, is deriving the context view from the composition of relevant areas of the component nodes. Thus:
  - Relevant area assignment: a partial view, expressed as a set of relational algebra expressions, is associated with each context element.
  - View composition by algebraic operators: the previously defined partial views, associated with the context elements in a configuration, are properly combined by means of an appropriate integration operator, to automatically obtain a final view defining all the information relevant for that configuration.
Composing relevant areas

methodology 1 (relational data)

Context-Dimension tree

Global Relational Schema

Valid contexts

Relevant area association

View

Rel. area
Context refinement via view composition

Relevant area assignment

- The designer is in charge of assigning to each context element its relevant area, or partial view.
- Such attribution has to be made in a coherent fashion.
- We say that

\[ \text{Rel}(w) \subseteq \text{Rel}(k) \quad \text{iff} \quad \forall R_i \in \text{Rel}(w) \exists R_j \in \text{Rel}(k) \quad \text{s.t.} \quad \text{Att}(R_i) \text{ included in Att}(R_j) \text{ and } \Pi_{\text{Rel}(R_i)} R_i \text{ included in } \Pi_{\text{Rel}(R_j)} R_j \]

Assumption:

For each pair of context elements \( n \) and \( m \),

if \( n < m \) then \( \text{Rel}(n) \supseteq \text{Rel}(m) \)

Where \( n < m \) means that \( n \) is more abstract than \( m \), i.e., \( m \) is a descendant of \( n \).
Relevant area assignment

Assignment strategy:

- **navigating the CDT top-down**
- Specifying, for each node $m$, a *partial view* $(\Re(n))$ over the (previously defined) view $(\Re(l(m)))$ pertaining to $n$, with $n \prec m$
- Requirement: when assigning relevant areas, keys must be kept
Example

\[ R_{el}(\text{estate}) = \{ \text{estate, owner, visit, sales, rent, agenda, picture} \} \]

Consider now the estates belonging to the “residential” category:

\[ R_{el}(\text{residential}) = \{ \sigma_{\text{Category} = \text{Residential}} \text{estate}, \text{owner} \bowtie (\sigma_{\text{Category} = \text{Residential}} \text{estate}), \text{visit} \bowtie (\sigma_{\text{Category} = \text{Residential}} \text{estate}), \text{sales} \bowtie (\sigma_{\text{Category} = \text{Residential}} \text{estate}), \text{rent} \bowtie (\sigma_{\text{Category} = \text{Residential}} \text{estate}), \text{agenda} \bowtie (\sigma_{\text{Category} = \text{Residential}} \text{estate}), \text{picture} \bowtie (\sigma_{\text{Category} = \text{Residential}} \text{estate}) \} \]

Relevant area combination

The operators we use to combine partial views need to satisfy two fundamental requirements:

- **Query resilience**: the application of our operators should leave the input relational schemas “as intact as possible”. This requirement is important because, given their nature, context-aware views are dynamic, thus the queries embedded within the application code should be still applicable when context changes. If this requirement is satisfied, the application code can refer once and for all to the global schema.

- **Set orientation**: we need to compose views formed by several relations with other such views, where the output view has been obtained by applying the same operation to all elements. Thus, better introduce set-oriented operators, which act on all pairs of input relations in a uniform way.
Context-view composition by double intersection

The **Double Intersection**, between two sets of expressions A and B, applies the intersection operator of relational algebra to the maximal common sub-schemata produced by the pairs of expressions $e_A$ and $e_B$, belonging to A and B, respectively, having the same origin:

```
Input: Two partial views A and B.
Output: The view obtained by combining A and B.
Procedure DoubleIntersection (A, B)
{
  1. $S = \emptyset$
  2. for all $e_A \in A$, $e_B \in B$, such that $O(e_A) = O(e_B)$
  3. $S = S \cup \{ \pi_X(e_A) \cap \pi_X(e_B) \}$, where $X = \text{atts}(e_A) \cap \text{atts}(e_B)$
  4. output(S);
}
```

Double intersection

Two sets of tables

Their double intersection
Example

Let:
\[ \mathcal{R}_{el}(\text{agent}($\text{ag}_\text{id}$)) = \{\sigma_{\text{id}_\text{Agent}=$\text{ag}_\text{id}$}^{\text{AGENT}}, \sigma_{\text{id}_\text{Agent}=$\text{ag}_\text{id}$}^{\text{AGENDA}}, \text{VISIT}, \text{SALES}, \text{KENT}, \text{OWNER}, \text{CUSTOMER}, \text{STATE}, \text{PICTURE}\} \]

and:
\[ \mathcal{R}_{el}(\text{residential}) = \{\sigma_{\text{Category}=$\text{Residential}^\text{ESTATE}$}, \text{OWNER} \land (\sigma_{\text{Category}=$\text{Residential}^\text{ESTATE}$}, \text{VISIT} \land (\sigma_{\text{Category}=$\text{Residential}^\text{ESTATE}$}, \text{SALE} \land (\sigma_{\text{Category}=$\text{Residential}^\text{ESTATE}$}, \text{RENT} \land (\sigma_{\text{Category}=$\text{Residential}^\text{ESTATE}$}, \text{AGENDA} \land (\sigma_{\text{Category}=$\text{Residential}^\text{ESTATE}$}, \text{PICTURE} \land (\sigma_{\text{Category}=$\text{Residential}^\text{ESTATE}$}))\)\}

Then for: \[ C = \{\text{role} : \text{agent}($\text{ag}_\text{id}$), \text{category} : \text{residential}\} \]
we can apply double intersection, and have:
\[ \mathcal{R}_{el}(C) = \mathcal{R}_{el}(\text{agent}($\text{ag}_\text{id}$)) \cap \mathcal{R}_{el}(\text{residential}) \]
\[ = \{\sigma_{\text{Category}=$\text{Residential}^\text{ESTATE}$}, \text{OWNER} \land (\sigma_{\text{Category}=$\text{Residential}^\text{ESTATE}$}, \text{VISIT} \land (\sigma_{\text{Category}=$\text{Residential}^\text{ESTATE}$}, \text{SALE} \land (\sigma_{\text{Category}=$\text{Residential}^\text{ESTATE}$}, \text{RENT} \land (\sigma_{\text{Category}=$\text{Residential}^\text{ESTATE}$}, \text{AGENDA} \land (\sigma_{\text{Category}=$\text{Residential}^\text{ESTATE}$}, \text{PICTURE} \land (\sigma_{\text{Category}=$\text{Residential}^\text{ESTATE}$}))\)\}) \]

View design restrictions

When defining partial views, the designer should observe the following restrictions:

a) for each expression \( e \) of the form \( \Pi_{\text{att}} (ei) \), the set of attributes \( \text{att} \) contains the key attributes of the relation \( O(ei) \);

b) for each \( R \) of the global schema, a partial view may contain at most one \( ei \) such that \( O(ei) = R \)
Consequences

- Requirement (a) is introduced because keys are necessary to identify and possibly connect tuples of two expressions.
- Requirement (b) means that partial views cannot contain more than one expression with the same origin.

$\rightarrow$ The partial view associated with a context element has a tailored schema with respect to the global one, where some relations do not appear and those that appear may have a reduced number of attributes. The same holds also for the final view associated with a context.

$\rightarrow$ This property and this naming convention guarantee that the query resilience requirement is satisfied.

Some operator properties

Double intersection is associative:

\[ \mathcal{R}_\mathcal{C}l((V_1 \ldots, V_k, V_{k+1})) = \mathcal{R}_\mathcal{C}l((V_1 \ldots, V_k)) \cap \mathcal{R}_\mathcal{C}l(V_{k+1}) \]

i.e., we can obtain the view for a context composed by \( k+1 \) context elements, by combining the partial view for the context formed by the first \( k \) elements with the view of the last one.

$\rightarrow$ Useful when the designer has to modify the CDT: if e.g. a dimension is added, relevant areas can be automatically re-computed.

$\rightarrow$ Favours dynamicty.
More operator properties

Since double intersection is *commutative*, we have:

\[ \mathcal{R} \cap (V_1, V_2, \ldots, V_k) = \mathcal{R} \cap (V_1, V_2, \ldots, V_k) \]

i.e., for any permutation of \(<1, \ldots, k>\), i.e., the view for a context composed by any number of context elements *is independent of the order in which it has been composed*.

More properties

If:

- \( C = <V_1, \ldots, V_k> \)
- \( V, W \) two context elements such that \( V < W \)
- we have:

\[ \mathcal{R}(V_1, \ldots, V_k) \supseteq \mathcal{R}(V_1, \ldots, W, \ldots, V_k) \]

This is the extension to final views of the assumption on containment for partial views of context elements.
methodology 1 (XML data)

methodology 2 (XML data)
MORE APPLICATIONS OF CONTEXT MODELING

ContextADDICT ARCHITECTURE
Ontology-driven, context-aware data integration for mobile devices
(see in Technologies for IS)

- Dynamic data integration, within a scenario where:
  - data sources are not known a-priori
  - user queries are dealt with in a context-aware fashion
  - information fruition is fostered by
    - handing it to the user in a semantics-aware, integrated fashion
    - eliminating non-interesting information, thus reducing the "information noise"
    - controlling the problem’s dimension via context-based reduction of the current information space

- Queries are issued to the system in SPARQL and translated into CA-DL for internal processing

- CA-DL is an ontology language which can uniformly represent the application domain and the context

Data Sources
- We assume (in the most general case) to deal with data sources that are:
  - Heterogeneous
  - Distributed
  - Independent
  - Transient (not in this example)
  - Partially Overlapping
Semantic Extraction

- Data-source heterogeneity is solved by extracting the semantics in an ontology-based format
- Automatic Wrapper generation will make the actual data accessible

Ontology extraction from relational data sources

Data Source Ontology:
- Automatic Extraction: semantic ontology + structural ontology
- Models structural/semantic independence (the different models can be used separately)
- Data-source constraints are supposed to hold
Domain Modelling

We need a Domain Ontology to enable automatic data integration.

Domain Modelling

Data-source ontologies are mapped to the Domain Ontology (by our X-5OM tool).
Data Tailoring

- The domain model is related to the context
- Context-aware views are defined
- At run-time views are extended to data sources

CDT → domain ontology → source ontologies
Relevant areas, or projections

Projection:
- is the set of relevant data for a given user in a given context
- projected from the ADO to the data sources
- is context-aware
- possibly materialized on the user device

Recall PerLa: Pervasive query Language

High-level interface: the (declarative) language
- SQL-like syntax
- Three levels of queries
  - High level query (HLQ)
    - Equivalent to SQL for streaming DBs
  - Actuation query (AQ)
    - Executes commands, set parameters on devices
  - Low level query (LLQ)
    - Defines the behavior of a single or of a group of devices

Low-level interface: the hardware abstraction layer provides genericity
- Device as a Functionality Proxy Component (FPC)
- An FPC provides:
  - Attribute reading (id, temperature, pressure, power level, last sensed RFID reader, ...)
  - Event notification (last sensed RFID reader changed, ...)
  - Meta-description (name, data type, ...)
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 queries

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
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.

A CDT schema for the wine production process
CDT declaration

CREATE DIMENSION <Dimension Name>
[CHILD OF <Parent Node >]
{CREATE CONCEPT <Concept Name>
WHEN <Condition>
[EVALUATED ON <Low Level Query>]]*

In our case

CREATE DIMENSION Role
CREATE CONCEPT Farmer WHEN getuserole() = 'farmer'
CREATE CONCEPT Oenologist WHEN getuserole() = 'oenologist'
CREATE CONCEPT Driver WHEN getuserole() = 'driver'
CREATE DIMENSION Risk
CREATE CONCEPT Disease WHEN getinteresttopic() = 'disease'
CREATE CONCEPT Overheat WHEN
    temperature > 30 AND brightness > 0.75;
CREATE DIMENSION Phase
CREATE CONCEPT Growth WHEN getphase() = 'growth'
CREATE CONCEPT Ageing WHEN getphase() = 'ageing'
CREATE CONCEPT Transport WHEN getphase() = 'transport'
Context creation

CREATE CONTEXT <Context Name>
ACTIVE IF <Dimension>=<Value> [AND
<Dimension>=<Value>]*
ON ENABLE <PerLa Query>
ON DISABLE <PerLa Query>
    /*one shot only */
REFRESH EVERY <Period>

In our case

CREATE CONTEXT Transport Monitoring
ACTIVE IF Phase = 'transport' AND Role='driver' AND Risk='overheat'
ON ENABLE:
SELECT temperature , gplatitude , gplongitude
WHERE temperature > 30
SAMPLING EVERY 120 s
EXECUTE IF location = 'truckdepartingzone'
SET PARAMETER ' alarm ' = TRUE;
ON DISABLE:
DROP Transport Monitoring ;
SET PARAMETER 'alarm' = FALSE;
REFRESH EVERY 24 h ;
Context detection phase

- Here the Context Manager discovers if one of the declared contexts has become active or has been deactivated.
- A context is active if the dimensions that define it assume the values specified by the Context statement:
  \[ \text{Context} \equiv \bigwedge_{i,j} (\text{Dimension}_i = \text{Value}_{i,j}) \]
- The system recognises as active all the contexts whose CDT concepts contain a not-empty device list.
- In fact, if one ID has been associated with a concept it means that, for at least one device, a CDT dimension is currently assuming that value.
- If true for every <Dimension >=<Value> used to define C, then the environment is in the situation expressed by C, and C is considered as active.

Context modeling literature
Summary of the “hoW’s”

MODELING and REPRESENTATION:
- **Orthogonal** context modeling
- **Multi-application model**
- Context elements (dimensions): time, space, role, more...
- Space/Time coordinates (relative or absolute)
- User profile (e.g. role or features based)
- Variable context granularity
- Type of formalism: parameter-values pairs, rule-based, ontological, graph-based → **NOT** independent of the application
- Valid-context constraints

DINAMICITY MANAGEMENT
- On-line context changes
- On-line model changes
- Inter-context (dynamic) constraints
- Context history
- Automatic learning features

FURTHER FEATURES
- Context construction (distributed or centralized)
- Context quality monitoring
- Ambiguity/Incompleteness mgmt.
### Context model features and systems exposing them

<table>
<thead>
<tr>
<th>System</th>
<th>Space</th>
<th>Time</th>
<th>Spatio/Temporality (Relations of Asynchrony)</th>
<th>Contextuality</th>
<th>Subject (Type of Information)</th>
<th>Valid context constraints</th>
<th>Type of formulation (Structure)</th>
<th>Type of formulation (Semantics)</th>
<th>Type of formulation (Inference)</th>
<th>Metadependency</th>
<th>Context comparison (Determination of Context)</th>
<th>Context summarizing</th>
<th>Authenticity assurance</th>
<th>Automatism assurance</th>
<th>Multimodal model</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASS</td>
<td>+++</td>
<td>+</td>
<td>A</td>
<td>U</td>
<td>F</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>C aixaMoS</td>
<td>+</td>
<td></td>
<td>R/A</td>
<td>A</td>
<td>F</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>COMANDO</td>
<td>+</td>
<td></td>
<td>A</td>
<td>U</td>
<td>F</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Conceptual CM</td>
<td>+</td>
<td></td>
<td>+</td>
<td>A</td>
<td>F</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>CSMP</td>
<td>+</td>
<td></td>
<td>R</td>
<td>A</td>
<td>F</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>EFFECTS</td>
<td>+</td>
<td></td>
<td>A</td>
<td>A</td>
<td>F</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Graphical CM</td>
<td>+</td>
<td></td>
<td>A</td>
<td>A</td>
<td>F</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>H2O/HyperAudio</td>
<td>+</td>
<td></td>
<td>A</td>
<td>A</td>
<td>F</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>MAE</td>
<td>+</td>
<td></td>
<td>R/A</td>
<td>A</td>
<td>F</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SCOPES</td>
<td>+</td>
<td></td>
<td>R/A</td>
<td>A</td>
<td>U</td>
<td>F</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SOCAM</td>
<td>+</td>
<td></td>
<td>+</td>
<td>A</td>
<td>U</td>
<td>F</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>U-Learn</td>
<td>+</td>
<td></td>
<td>A</td>
<td>U</td>
<td>F</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

### Context models in computer science

**Conceptual CM considers the context not only as a state, but as a part of a process**

- Context as “any information that can be used to characterize a situation”
- Situation refers to the current state of the environment
- Context specifies the elements that must be observed to model situations
- A process-based software architecture

→ **Conceptual CM tailors services and information to the current situation**
Conceptual CM
(Coutaz, Crowley, Dobson, Garlan)

- Context is about evolving, structured, shared information spaces, each information space designed for a specific purpose. Challenges:
  - Recognize users’ goals and activities
  - Map these goals and activities adaptively to the population of available service and resources

entity Bob at the café plays the role of an architect while at the train station he is a traveller; entity “lump of sugar” plays the role of a building while Bob shows it to Jane on the table

- Danger:
  - Mismatch between the system’s model of interaction and the user’s mental model of the world

Conceptual CM
(Coutaz, Crowley, Dobson, Garlan)

- The runtime infrastructure is a middleware that instantiates entities, roles and relations for the current state of a context, with different levels of abstraction, by allowing the collection of all the information required to identify current context values and predict changes in the situation or in the actual context.
A context is a composition of situations that concerns a set of roles and relations.

The "state" of an environment is defined as a conjunction of predicates.

The run-time environment must act so as to render and maintain each of these predicates to be True.

A context determines the configuration of processes necessary to detect and observe the entities that can play the roles and the relations between roles that must be observed.

\[ \text{Context} \rightarrow \{ \text{Role}_1, \text{Role}_2, \ldots, \text{Role}_n ; \text{Relation}_1, \ldots, \text{Relation}_m \} \]

Relations are predicate functions evaluated over the properties of the entities that have been assigned to roles by the RTE.

The design problem is to single out the appropriate entities (resp. relations) that must be determined (resp. verified) with respect to a task or service to be provided, in a potentially infinite set.

Entities are assigned to roles by role assignment processes. The context model specifies which roles are to be assigned and launches the necessary role assignment processes.

A meta-supervisor determines what kind of entities can play each role, and launches processes to detect and observe these entities. The most suitable entities are assigned to the roles.

Relations are evaluated, and the set of relations determines the situation.

The relations in the situation are computed on the entities assigned to roles. Situation changes when the relations between entities change.

If the assignment of entities to situations changes, the situation remains the same. However, the system may need to act in response to a change in role assignment.
Conclusions:
- A context is a network of situations concerning a set of roles and relations.
- Roles are services or functions relative to a task.
- Roles may be "played" by one or more entities.
- A relation is a predicate defined over the properties of entities.
- A situation is a configuration of relations between the entities.

Observations are provided by perceptual processes defined by a tracking process or transformation controlled by a reflexive supervisor. Perceptual processes are invoked and organized into hierarchical federations by reflexive meta-supervisors. A model of the user’s context makes it possible for a system to provide services with little or no user intervention.
Context models in computer science

CoBrA is an agent-based Context Broker Architecture, whose goal is to provide a shared context model for event/meeting management

- Based on ontologies (Context Knowledge Base)
- Context built by a distributed effort (rules describing how shared context is built)
- It takes into account privacy issues
- Presence of a context broker
- Detecting inconsistent beliefs about certain contexts
- Limited flexibility (mono-target context model)

→ CoBrA models agreement and sharing

CoBrA

(Chen, Finin, Joshi)

- An agent based architecture for supporting context-aware computing in intelligent spaces
- Intelligent spaces are physical spaces (e.g., living rooms, vehicles, corporate offices and meeting rooms) populated with intelligent systems
- Intelligent context broker maintains and manages a shared contextual model on behalf of a community of agents. It can be federated with other context brokers
- Agents can be
  - Applications hosted by mobile devices that a user carries or wears (e.g., cell phones, PDAs and headphones)
  - Services that are provided by devices in a room (e.g., projector service, light controller and room temperature controller)
  - Web services that provide a web presence for people, places and things in the physical world (e.g., keeping track of people’s and objects’ whereabouts)
A *context broker* has four main functional components:

1. **Context Knowledge Base**: a persistent store for context knowledge in an intelligent space. This knowledge base provides a set of APIs for other components to assert, delete, modify, and query stored knowledge.

2. **Context Reasoning Engine**: a reactive inference engine that reasons over the knowledge base. Its main function is to deduce additional knowledge from information acquired from external sources and to maintain the consistency of the knowledge base.

3. **Context Acquisition Module**: a middleware abstraction for acquiring contexts from external, heterogeneous sources (e.g., physical sensors, web services, databases, devices, and agents).

4. **Privacy Management Module**: a set of communication protocols and behavior rules that the broker follows when performing privacy management tasks (i.e., negotiate privacy policies with new users and enforcing these policies when sharing information with agents in the community).
CoBrA Intelligent Meeting Room Scenario

1. R210 is an intelligent meeting room equipped with RFID sensors. As Alice enters the room, sensors inform the R210 broker that a cell phone belonging to her is present and the broker adds this fact in its knowledge base.

2. The agent on Alice’s Bluetooth enabled cell phone discovers R210’s broker and engages in a “hand shake” protocol (e.g. authenticates agent identities and establishes trust) after which it informs the broker of Alice’s privacy policy.

3. Based on the policy, the broker (i) acquires - and reasons about - Alice’s location and activity contexts, (ii) informs Alice’s personal agent at home when Alice’s contexts change, and (iii) shares her contexts with agents in the meeting room.

4. Knowing Alice’s cell phone is currently in R210, the broker concludes Alice is also there. R210 is a part of the Engineering building. These conclusions are asserted into the broker’s knowledge base.

5. The broker informs Alice’s personal agent of her whereabouts. Personal agent attempts to determine why Alice is there. Her Outlook calendar has an entry indicating that she is to give a presentation on Campus about now.

6. The personal agent concludes that Alice is in R210 to give her talk and informs the R210 broker. Broker shares this information with the projector agent and the lighting control agent in the ECS 210. Projector agent downloads the slides from Alice’s personal agent and sets up the projector, the lighting control agent dims the room lights.

---

<table>
<thead>
<tr>
<th>Place Related</th>
<th>Agents’ Location Context</th>
<th>Place Related</th>
<th>Agents’ Location Context</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lobby</td>
<td>PersonalRoom</td>
<td>Classroom</td>
<td>PersonalRoom</td>
</tr>
<tr>
<td>Conference</td>
<td>PersonInBuilding</td>
<td>Location</td>
<td>RestrictedTo</td>
</tr>
<tr>
<td>Engineering</td>
<td>PersonalRoom</td>
<td>Location</td>
<td>RestrictedTo</td>
</tr>
<tr>
<td>Engineering</td>
<td>PersonInBuilding</td>
<td>Location</td>
<td>RestrictedTo</td>
</tr>
<tr>
<td>Engineering</td>
<td>PersonalRoom</td>
<td>Location</td>
<td>RestrictedTo</td>
</tr>
<tr>
<td>Engineering</td>
<td>PersonInEngineering</td>
<td>Location</td>
<td>RestrictedTo</td>
</tr>
<tr>
<td>Engineering</td>
<td>PersonInEngineering</td>
<td>Location</td>
<td>RestrictedTo</td>
</tr>
</tbody>
</table>

---

**Agent’s Activity Interface**

- presentation schedule
- send evolutionary
- send presentation
- send presentation
- send presentation
- send presentation
- presentation
- presentation
- presentation
- presentation

---

**Agent’s Activity Contact**

- participate in
- send email
- send presentation
- send presentation
- send presentation
- send presentation
- send presentation
- send presentation

---

CoBrA (Chen, Finin, Joshi)
CoDaMoS (Context-Driven Adaptation of Mobile Services) offers a rather sophisticated context model.

- The system provides a context reasoning service, based on user-defined rules, and a context source discovery service.
- No constraints to limit the number of admissible contexts.
- Context represented by an extensible structure based on 4 categories: user, environment, service (in CoDaMoS) and platform.
- A point in the multidimensional context space is represented as a graph of concept instances – high representation complexity.

CoDaMoS models the problem as a whole, but are a little more focused on channel, device and location.

| Space | Time | Space/Time coordinate (Relative or Absolute) | Context history | Subject (User or Application) | User profile (Role or Features based) | Verbal context granularity | Valid context constraints | Type of formalism: Key-value-based | Type of formalism: Markup-based | Type of formalism: Logos-based | Type of formalism: Graph-based | Type of formalism: Ontology-based | Formality level (High or Low) | Adaptability (High or Low) | Context construction (Distributed or Centralized) | Context reasoning | Context quality monitoring | Ambiguity/Incompleteness agent | Automatic learning features | Multi-context model |
CoDaMoS  
(Preuveneers, Van den Bergh et al.)

Context ontology overview

Every device will contain its own context specification with a full description of its provided services, plus pointers to relevant information on the devices in its environment

- **User**: properties include a user’s profile, but also his preferences, mood and current activity
- **Environment**: time and location information, and environmental conditions, such as temperature and lighting
- **Platform**: hardware and software description of a specific device
- **Service**: specifying semantic and syntactic information sustains easy service discovery and service interaction using a well-defined service interface

![Context ontology diagram]

User ontology

![User ontology diagram]
CoDaMoS
(Preuveneers, Van den Bergh et al.)

Environment ontology

CoDaMoS
(Preuveneers, Van den Bergh et al.)

Platform ontology
**CoDaMoS**
(Preuveneers, Van den Bergh et al.)

Service ontology

**Context management in CoDaMoS**
(Preuveneers, Van den Bergh et al.)

**Context storage components:**
- Context representation
- History of context
- Outdated and redundant info management
- Context repository
- Adaptation support

**Context manipulation components:**
- Context transformation
- Context reasoning
- Context-based decision making and adaptation
- Component-based context manipulation
- Support for adaptation
CoDaMoS

• ECORA (A. Padovitz, S. W. Loke, A. B. Zaslavsky):
  A hybrid architecture for context-oriented pervasive computing

• In ECORA: incremental awareness and compositionality [Loke 09]

• emphasis on systematically extending over time the contexts and situations a system can be aware of

• a formalism of operators for building context-aware pervasive systems incrementally

  → ECORA is focused on location and environment
Homework

- Survey and presentation of ECORA and the previous and subsequent work by the same group
- The same for another, chosen context-aware system

Conclusion on context models in computer science

- Due to the complexity of the problem as a whole and to the multitude of different applications, the best models have a well defined focus, and try to support only one of the above mentioned context-modeling categories
- Context in information systems: the viewpoint abstraction
Context as abstraction

- abstraction mechanisms are the indispensable means to deal with complexity in information system design (see abstractions for database design).
- the viewpoint abstraction has received little attention
- context as a viewpoint mechanism that takes into account implicit background knowledge

Context in Information Systems

- Context as a selector of workspaces  
  - information-system oriented
  

- Context as a selector of views or facets  
  - database oriented

  Y. Roussos Y. Stavrakas V. Pavlaki: Towards a Context-Aware Relational Model, in "Contextual Representation and Reasoning" Workshop (CRR'05), held in conjunction with CONTEXT'05, Paris, 2005
  Ours
Cooperation Scenario
(Theodorakis, Analyti, Constantopoulos, Spyrou)

User Anastasia’s interactions with the various workspaces

Context Relational Model
(Roussos, Stavrakas, Pavlaki)

- Information entities are multi-facet
- Each facet $f_{ij}$ is the variant of an entity
- A set of entities is a context relation
- For a context relation, a number of (possibly different) attributes is defined for each possible world
- The value of an attribute $A_i$ in world $w_j$ is denoted by $A_i(w_j)$
Conclusions

- Even within Information Systems, context is used for different purposes:
  - to provide access to different facets of the same object or group of objects
  - to provide/equip different users with specific functions in various situations
  - to tailor data or services in different “shapes”
  - to associate data with different preference for values in different situations
- Fundamental underlying concepts:
  - VIEW
  - CONTEXT DIMENSION

More possible homework topics

- For non-db students: using the CDT as context model, suggest a high-level system design for context and context-aware behaviour in a field of your interest
- Review of the literature on sensor-based context detection and subsequent enforcement
- “Deep review” of a system proposing context-awareness and comparison with features discussed here