Context Modeling and Reasoning: An Overview

http://www.cs.unibo.it/difelice/cas/

Context Aware Systems

Prof. Marco Di Felice
Department of Computer Science and Engineering
University of Bologna
Life-cycle of CA systems

Life-cycle of CA systems typically involve four main steps for their deployment.

- Context Acquisition
- Context Dissemination
- Context Modeling
- Context Reasoning
Context Modeling

- The development of context aware applications must be supported by adequate software engineering practices and methods.
  - The design of application functionalities must not be intertwined with the definition of context data.
  - The usage of context modeling formalism improves the maintainability and evolvability of the applications.
  - The usage of context modeling formalism allows the re-use and sharing of context information between applications.
Context Modeling

EXAMPLE: Designing context-aware app for museums

✧ **Domain** context
  Content personalization

✧ **Physical** context
  Physical positions of the devices

✧ **System** context
  Peer discovery and related services


D. Raptis, N. Tselios and N. Avouris, Context-based Design of Mobile Applications for Museums: A survey of existing practice
Context Modeling

- **Requirement set for context models**

  - **Heterogeneity and mobility**
    - Context models have to deal with context sources that might differ in their update rate and semantic level.
  
  - **Relationships and dependencies**
    - Relationships between types of context information have to be captured in order to ensure the correct application behaviour.
  
  - **Timeliness**
    - Context-aware applications might need to access past states as well as to predict new states (prognosis).
Context Modeling

- **Requirement set for context models**

  - **Imperfection**
    Context modeling approach must include modeling of the context information quality to support reasoning functions.

  - **Reasoning**
    Reasoning can be used to derive new facts from existing knowledge and/or to provide high-level context abstractions.

  - **Usability**
    Designers must easily translate real world concepts to the modeling constructs, applications must easily manipulate context information.
Context Modeling

- Main formalisms proposed so far
  - Key-value Models
  - Markup Schema Models
  - Object-role Models
  - Ontology-based Models
Context Modeling

- **Context-Triggered Software** (1994)
  - Based on the Active Badge system previously defined.
  - IF-THEN-ELSE rules used to specify how context-aware system should behave/react to any user location-change event.
  - Context $\rightarrow \langle$user/badge, location$\rangle$
  - Action $\rightarrow$ Unix shell command to execute

<table>
<thead>
<tr>
<th>badge</th>
<th>location</th>
<th>event-type</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schilit</td>
<td>Kitchen</td>
<td>attention</td>
<td>“play-v 50 /sounds /rooster.au”</td>
</tr>
</tbody>
</table>
Context Modeling

- Main formalisms proposed so far
  - Key-value Models
  - Markup Schema Models
  - Object-role Models
  - Ontology-based Models
Context Modeling

- **Composite Capabilities/Preference Profile (CC/PP)**
  - [https://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/](https://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/)
  - A **CC/PP profile** is a description of device capabilities and user preferences that can be used to guide the adaptation of content presented to that device.
  - A set of CC/PP attribute **names**, permissible **values** and associated **meanings** constitute a CC/PP vocabulary.
  - A CC/PP profile contains **one or more components**, and each component contains one or more attributes.
Context Modeling

- **Composite Capabilities/Preference Profile (CC/PP)**
  - [https://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/](https://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/)
  - Each component is represented by a resource of type `ccpp:Component`
  - CC/PP profiles are constructed using the RDF notation.
  - The RDF data model represents CC/PP attributes as named properties linking a subject resource to an associated object resource or RDF literal value.
Context Modeling

RDF version

XML version

<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xhtml="http://www.w3.org/1999/xhtml"
  xmlns:ex="http://www.example.com/schema#">
  <ex:MyProfile>
    <ex:component rdf:resource="http://www.example.com/schema#TerminalHardware" />
    <ex:terminalHardware>
      <ex:displayWidth>320</ex:displayWidth>
      <ex:displayHeight>200</ex:displayHeight>
    </ex:terminalHardware>
  </ex:MyProfile>
</rdf:RDF>
Context Modeling

- Main formalisms proposed so far
  - Key-value Models
  - Markup Schema Models
  - Object-role Models
  - Ontology-based Models
Context Modeling

- Context Modeling Language (CML) [1]

ER/ORM variant, conceptual modeling of databases via a graphical notation:

- **Direct graph**, entities are nodes, relationships among nodes are arcs
- Each entity can contain attributes
- Additional modeling constructs for representing the context data

Context Modeling

Context Modeling Language (CML) [1]

 Associations based on the context type:

- Static association
- Dynamic association
- Sensed association
- Derived association
- Profiled association

Legend:
- Static association
- Derived association
- Profiled association
- Sensed association
- Simple association
- Collection association
- Alternative association
- Temporal association
Context Modeling

- Context Modeling Language (CML) [1]
  - Modeling dependencies among context data

Context Modeling

- Context Modeling Language (CML) [1]

- Modeling context quality (how it is evaluated)

Context Modeling

- Main formalisms proposed so far
  - Key-value Models
  - Markup Schema Models
  - Object-oriented Models
  - Ontology-based Models
The Resource Description Framework (RDF) is a data model for representing information (especially metadata) about Web resources.

The broad goal of RDF is to define a mechanism for describing resources that makes no assumptions about a particular application domain, nor defines (a priori) the semantics of any application domain.

RDF is intended for situations in which information about Web resources needs to be processed by applications, rather than being only displayed to people.
The basic data model consists of three object types:

- **Resources**
  All things being described by RDF expressions are called resources. Resources are always named by URIs: anything can have a URI.

- **Properties**
  A property is a specific aspect, characteristic, attribute, or relation used to describe a resource. Each property has a specific meaning, defines its permitted values, the types of resources it can describe, and its relationship with other properties.

- **Statements**
  A specific resource together with a named property plus the value of that property for that resource is an RDF statement.
Context Modeling

- **RDF (Resource Description Framework)**
  - RDF statements can be written down using **triple notation**. In this notation, a statement is written as follows:
    
    subject predicate object

    *Ora Lassila is the creator of the resource http://www.w3.org/Home/Lassila.*

    - **Subject** (Resource)  http://www.w3.org/Home/Lassila
    - **Predicate** (Property)  Creator
    - **Object** (literal)  "Ora Lassila"
Context Modeling

- **RDF (Resource Description Framework)**

RDF can be drawn as **labeled graphs** ("arcs diagrams").

- **ovals** represent resources
- **arcs** represent named properties.
- **rectangles** represent string literals
Context Modeling

- RDF (Resource Description Framework)

RDF labeled graphs can be serialized into XML documents.

```
<rdf:RDF>
  <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator>Ora Lassila</s:Creator>
  </rdf:Description>
</rdf:RDF>
```
Context Modeling

- **RDFS (Resource Description Framework Schema)**
  - Meaning in RDF is expressed through reference to a schema. A schema defines the terms that will be used in RDF statements and gives specific meanings to them.

RDF Schema can be considered as a sort of **vocabulary**:
- The vocabulary gives “extra meaning” to particular RDF predicates and resources (such as subClassOf).
- The “extra meaning”, or semantics, specifies how a term should be **interpreted** and validated.
Context Modeling

- **RDFS** *(Resource Description Framework Schema)*
  - RDF Schema terms (just a few examples):
    - rdfs:Resource
    - rdfs:Class
    - rdf:Property
    - rdf:type
    - rdf:range
    - rdf:domain
    - rdfs:Literal
    - rdfs:subClassOf
Context Modeling

RDFS (Resource Description Framework Schema)

RDF Schema terms (just a few examples):

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdfs:Resource</td>
<td>the class of all resources</td>
</tr>
<tr>
<td>rdfs:Class</td>
<td>the class of all classes</td>
</tr>
<tr>
<td>rdfs:Literal</td>
<td>the class of all literals (strings)</td>
</tr>
<tr>
<td>rdfs:Property</td>
<td>the class of all properties</td>
</tr>
<tr>
<td>rdfs:Datatype</td>
<td>the class of datatypes</td>
</tr>
</tbody>
</table>
Context Modeling

- **RDFS (Resource Description Framework Schema)**
- **RDF Schema terms** (just a few examples):

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdfs:subClassOf</td>
<td>Relates class to one of its superclasses, declare hierarchies of classes, is transitive by definition</td>
</tr>
<tr>
<td>rdfs:subPropertyOf</td>
<td>relates a property to one of its superproperties, is transitive by definition</td>
</tr>
<tr>
<td>rdfs:domain</td>
<td>declares the class of the subject in a triple</td>
</tr>
<tr>
<td>rdfs:range</td>
<td>declares the class or datatype of the object in a triple</td>
</tr>
<tr>
<td>rdfs:comment</td>
<td>typically provides a longer text description of the resource</td>
</tr>
<tr>
<td>rdfs:label</td>
<td>associates the resource with a human-friendly name</td>
</tr>
<tr>
<td>rdfs:isDefinedBy</td>
<td>relates a resource to the place where its definition, typically an RDF schema</td>
</tr>
<tr>
<td>rdfs:seeAlso</td>
<td>relates a resource to another resource that explains it</td>
</tr>
</tbody>
</table>
Context Modeling

- **RDFS (Resource Description Framework Schema)**
  - RDF Schema *vocabularies* (just a few examples):
    - `<Person,type,Class>`
    - `<hasColleague,type,Property>`
    - `<Professor,subClassOf,Person>`
    - `<Carole,type,Professor>`
    - `<hasColleague,range,Person>`
    - `<hasColleague,domain,Person>`
Context Modeling

- RDFS (Resource Description Framework Schema)
  - RDF Schema in labeled graphs
Context Modeling

- **RDFS (Resource Description Framework Schema)**
  - RDFS are too weak to describe resources in sufficient detail (e.g. no localised range and domain constraints)
  - RDFS do not provide adequate reasoning support

Web ontology languages are built on top of RDF and RDFS
Context Modeling

OWL (Web Ontology Language)

https://www.w3.org/OWL/

“Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relations between things”.

- Three versions of OWL language
  - OWL full (OWL + RDF), OWL-DL, OWL-Lite
- Well defined semantics
- Reasoning algorithms
- Several supporting tools available
Context Modeling

- **OWL (Web Ontology Language)**

Classes are a collection of individuals, a way of describing part of the world. They are defined in an OWL document with the `owl:Class` element.

For example, the class “Teacher” can be define as follows,

```xml
<owl:Class rdf:ID="Teacher">
  <rdfs:subClassOf rdf:resource="#Person"/>
</owl:Class>
```
Context Modeling

- **OWL (Web Ontology Language)**

Differently from RDFS, in OWL it is possible to establish relationships among classes, for instance:

```xml
<owl:Class rdf:ID="faculty">
  <owl:equivalentClass rdf:resource="#academicStaffMember"/>
</owl:Class>
```

```xml
<owl:Class rdf:about="#AssociateProfessor">
  <owl:disjointWith rdf:resource="#FullProfessor"/>
</owl:Class>
```