

Context-Driven Data Filtering: A Methodology*

Cristiana Bolchini and Elisa Quintarelli

Politecnico di Milano, Italy

bolchini@elet.polimi.it, quintare@elet.polimi.it

Abstract. The goal of this paper is the introduction of a methodology for designing context-driven data selection, that is the possibility to tailor the available, usually too rich, data to be held on portable mobile devices, according to *context*. First of all, we will introduce the concept of context and its model, a data structure that expresses knowledge on the user, the environment and the possible scenarios. We will then focus on the proposed methodology for selecting, by means of such information, the relevant data to be made available on a user device. An application of the proposed methodology is the possibility to select data of interest for portable devices, where computation, memory, power and connectivity resources are limited, and thus, tailoring the available, usually too rich, data according to context is a mandatory task.

1 Introduction

Today the amount of available information is growing at high rates, making it difficult to be able to select in a simple way those data that are of interest, discarding all the other ones. Such a task becomes even more significant as the size of the devices devoted to managing such data decreases. Indeed the wide spread of portable devices, with limited resources such as computational power, battery life and memory, poses a quest for applications able to manage the most interesting data, keeping on board only the small portion that – in that moment – the user wants.

In this scenario, the criteria for performing either off-line or dynamic data tailoring plays a relevant role, being it the central element to determine which data should be kept and which should be discarded. In our opinion, such criteria can be expressed by means of the notion of *context*. Our model of context differs from the several ones available in literature, proposed in the past few years; some of them mainly concern the design and implementation of adaptive applications by introducing the notions of user profile and contexts [1,2,3,4,5,6,7]. However, these approaches often provide very specific solutions to data personalization that mainly consider either user preferences [1] or device features [3,5]. In this work we apply a very general notion of context that can include general aspects, at different levels of detail, mainly related to the application scenario. Indeed, our main purpose is to propose a methodology to support the data tailoring task by considering some dimensions as being part of the context and the application as the main point of view of the context description. More in detail, the methodology defines the main steps to identify the relevant portion of information, called *chunk*, on the basis of the current user's context.

* This work has been partially supported by MIUR projects Esteem and ARTDECO.

Among the possible scenarios that can take advantage from the proposed methodology we mention the Context-ADDICT (Context-Aware Data Design, Integration, Customization and Tailoring) project, aimed at defining and providing a framework for selecting and integrating the relevant information to be delivered on user's devices on the basis of the current context [8].

In this paper we present the methodology supporting *chunk* definition, that can be effectively adopted in such scenario, as well as in others.

The paper is organized as follows. The next section informally introduces our previous work, i.e., the context model used to represent the different aspects for tailoring data within an application scenario, whereas Section 3 presents the methodology for performing such data selection. The proposed approach is exemplified through a running example, discussed in Section 4. The last section draws some conclusions and highlights future research issues.

2 Context Modeling and Usage

The aim of the proposed context model is to allow the designer to express the various criteria that can be adopted to select a portion of the available data, considering the application environment and the active scenario. It is thus important to notice that this notion of context is strictly connected to the considered application and is not meant to model the general knowledge concerning one or more areas of interest, a situation where a data schema, or a domain-ontology may be better suited. In the following a brief description of the model and its representation is presented, for a formal and more detailed discussion please refer to [9].

The knowledge being modeled has neither the goal to be fully comprehensive, nor the necessity to be particularly detailed or homogeneous in its content; the important thing is that it models all the significant criteria that have a relevant impact on the choice of the interesting information within a defined application scenario. The set of primary criteria constitutes the so-called *ambient dimensions* of the context, and includes the following ones, identified throughout our experience as being significant: *holder* (expresses the role the user has within the application, modeling the different person types that may use the device); *interest-topic* (expresses the possible topics that will be considered within the application scenario); *situation* (indicates the different phases/circumstances the user, the environment and the application may be in); *interface* (identifies the kind of interaction with the stored data); *time* (expresses the possibility of filtering data with respect to a time instant, with levels of granularity depending from the specific application scenario); *space* (similarly to the *time* dimension, expresses the possibility of filtering data with respect to a location, with levels of granularity depending from the specific application scenario); *data-ownership* (expresses the type of access that can be performed on the available data based on the ownership permissions).

Owing to the fact that this context is strictly related to the application scenario, it cannot be a-priori defined and, more important, it may happen that not all the listed dimensions are applicable in certain situations, or others could become significant. Let us, for instance, consider a peer-to-peer scenario; the *holder* dimension is not applicable, since there are different roles a peer may assume with respect to the data it is interested

in. On the other hand, it could be useful to introduce other dimensions, to model other data selection criteria; in this same scenario it may be of interest to introduce a *service* dimension. Do note that dimensions are orthogonal among themselves, providing independent points of views driving the tailoring of data.

In order to model this kind of context a tree structure has been defined, where the root node represents the entire data space, and the listed ambient dimensions are the first-level nodes, to be further exploded into subtrees. This model has been dubbed the Context Dimension Tree (CDT), aimed at defining the elements used to tailor data; furthermore, the tree structure is then extended to define a Context Dimension Graph (CDG), which includes constraints and relationships among dimension values to remove meaningless combinations of elements, which will be discussed in the following.

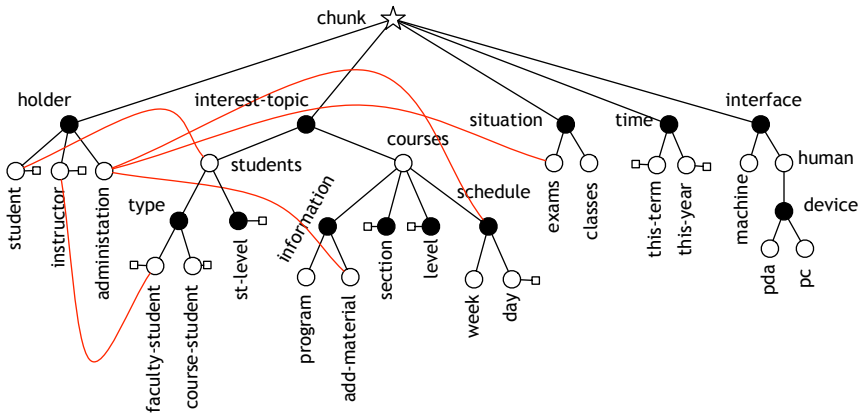


Fig. 1. The Context Dimension Graph for a university application scenario

For each one of the pertinent ambient dimensions a subtree is created, increasing the detail-level adopted to select data, thus allowing a refinement of the knowledge used to tailor data. An example of a Context Dimension Graph for a university application scenario, gathering contents about courses, students and related issues, is reported in Figure 1. Let us consider, as an example, the children of the *holder* dimension: *student*, *instructor*, *administration*; in each moment, and hence for each possible context, the device user may have only one of these roles. The same criterion applies to the children of the other ambient dimensions. For the *interest-topic* dimension the following mutually exclusive values can be envisioned: *courses* and *students*. Moreover, when considering the *courses* node, there are several aspects that can be identified for further refining the selection of interesting data: *information*, *section*, *level* and *schedule*; these aspects, which are graphically represented as sibling black nodes, are not mutually exclusive, since they provide different characterizations of the same concept.

To differentiate these two main types of nodes in the Context Dimension Tree two colors have been used; black nodes are *dimension nodes*, whereas white nodes are *concept nodes*. A concept may be characterized by several aspects, and each aspect may

be constituted by different concepts; as a result, there is an alternating pattern in the levels of the tree and at each level all siblings have the same color. In the context model there are also attribute nodes, which are leaves graphically represented as rectangles, expressing the desire to be able to select data according to a specific run-time generated value. Without considering attributes, leaf nodes are either concept or dimension nodes. When a dimension (black) node has not concept (white) children it must have an attribute child. In this case, it is possible to substitute the single attribute node, with a set of white children enumerating all possible choices, leading to the same information, although expressed in a less-compact form.

In order to identify a portion of data (called *chunk*) to be tailored from the entire data set, to be made available on the user's device, one or more nodes for each possible dimension are selected, choosing a single white sibling and any number of black siblings. This set of selected nodes/values is called *chunk configuration* and refers to a chunk of the entire data. Examples of chunk configurations for the Context Dimension Graph reported in Figure 1 are: $\{\{\text{student}\}, \{\text{schedule, program}\}, \{\text{classes}\}, \{\text{this_year}\}\}$ and $\{\{\text{instructor}\}, \{\text{students}\}, \{\text{classes}\}, \{\text{this_year}\}\}$.

Not all possible combinations of dimension values are worth considering, due to the fact that certain selections would never occur in the application scenario. Consider, as an example, the *interest-topic* dimension and the *students* value, this is not a significant value when *holder* is *student*; thus it is not necessary to generate and manage chunk configurations where $\text{holder}=\text{student} \wedge \text{interest-topic}=\text{students}$. This constraint is a *forbid constraint*, used to specify chunk configurations that would not be significant, and defined as a red solid relationships between the involved nodes; it is worth noting that this constraint would also discard all chunk configurations considering descendants from the involved nodes.

As a result, the tree structure is enriched leading to a graph one, to reduce the number of chunk configurations that will be generated, corresponding to the various possible application contexts that will be used. On the portable device, a user will load the portion of data corresponding to one or more configurations.

The proposed context model tree can be specified by means of different formalisms, in OWL-DL [10,11] as well as by means of an XML based (XML-SCHEMA or DTD) representation. The choice between the two possible representations is driven by the application scenario: in a situation where data sources are XML documents referring to an XML schema, an XML-SCHEMA representation may be more suitable and homogeneous (eventually translated from a DTD by means of one of the many available tools, e.g., *dtd2xml* [12]). On the other hand, in an ontology based application scenario, an OWL description may ease the integration with a Domain Ontology, as well as the possibility to use other available tools for manipulating the context model.

The next section discusses how the Context Dimension Graph is used to produce the significant chunk configurations, to be used to tailor the complete dataset.

3 Data Filtering Methodology

The Context Dimension Graph is the instrument to determine, based on the context, the portion of data to be held on a portable, mobile device. The following guidelines

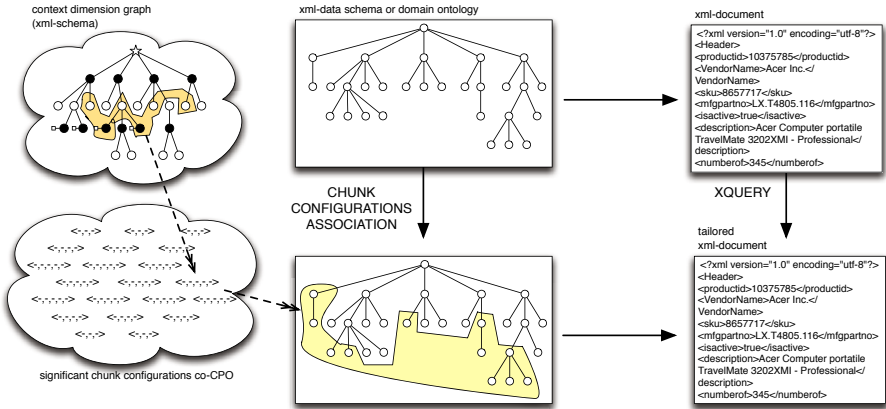


Fig. 2. The methodology flow in an XML data source environment

present the main steps to be followed in order to tailor data; an overall picture of the methodological approach is described in Figure 2. For each step a brief description summarizes the actions to be performed to pursue the final goal. Most of the actions are carried out by the designer, a few can be automated by means of tools that are being developed.

Identification of the CDT ambient dimensions. The set of dimensions modeling the application scenario is determined, referring to the set of common criteria previously listed. Some of them may be omitted, others may be added.

Identification of each dimension sub-tree. For each dimension, a sub-tree of relevant concepts and classes is created, providing an increasing level of detail as the depth of the sub-tree increases.

Introduction of the constraints. Specification of undesired chunk configurations to prevent the generation of insignificant portions of data that would never be used (top left corner of Figure 2).

Generation of significant chunk configurations. By traversing the Context Dimension Graph all chunk configurations, considering one or more values according to the stated rules and constraints roles, are computed. The result is a co-complete partial order (co-CPO) where the greatest (top) chunk configuration is represented by no value per dimension (i.e., the Context Dimension Graph root itself corresponding to the entire data) (bottom left corner of Figure 2). The partial order relation \succ between two chunk configurations $A = \langle A_1, \dots, A_n \rangle$, $B = \langle B_1, \dots, B_n \rangle$, we say that $A \succ B$ (A is more abstract (i.e. less detailed) of B) iff $(\forall j \in \{1, \dots, n\})(\forall a_i \in A_j \exists b_k \in B_j$ such that $b_k \in \text{Descendant}(a_i) \cup \{a_i\}$).

Identification of the data schema. The data schema is determined (or designed from scratch), and when more data sources are available their schemata are integrated.

Chunk configurations and XML-data schema areas association. For each one of the derived chunk configurations, an area corresponding to all interesting elements of the XML-data schema is selected (see the bottom center of Figure 2). The designer is in charge of determining the relationships between the chunk configurations and the elements of the schema (domain ontology or entity relationship or relational database/XML schema) of the data to be tailored.

XQuery generation. Given the XML-data schema elements corresponding to a chunk configuration, a set of XQuery expressions is generated to produce tailored data.

XQuery application. The XQueries, derived according the XML-data schema as representing the desired data portions, are applied to XML documents producing a *tailored XML-data document*, that is a chunk (in the bottom right corner of Figure 2).

Chunk collection. One or more chunks are collected, based on the selected applied context, and loaded on the user's device.

The information associating the chunk configuration with the XQuery is stored in a repository called *chunk dictionary*, that can be used on- and off-line to select and load on the portable device the desired chunks.

In three cases it may be necessary to update the loaded data: i) context changes (i.e., the user's interests change, the situation changes, ...) or ii) new data sources are available, or iii) both events occur. In these situations, the chunk dictionary is used to find the set of XQueries to be applied to the available data sources that are assumed to be valid w.r.t. the XML-data schema. It is worth noting that the underlying assumption is that the XML-data documents refer to the XML-data schema used during the association phase between chunk configurations and the schema itself. If this is not the case, the applied query will retrieve unpredictable data, as semantically distant as the document is not valid w.r.t. the used schema and is structurally very different from this schema.

Deriving the association between chunk configurations and the XML-data schema areas is one of the most complex tasks within this methodology. Indeed the number of significant chunk configurations may be relevant and the selection, for each one of them, of the related XML schema portion may be both sensitive and time-consuming. To cope with this issue, we are currently working on the possibility to automate the association between chunk configurations and the XML-data schema, by requesting the designer to perform an association between Context Dimension Tree nodes and XML-data schema elements. More precisely, for each node in the tree, a set of XML-data schema elements are identified independently of the other dimension nodes. The elements corresponding to a chunk configurations are thus derived as a combination (intersection) of the elements associated with the nodes expressing the dimensions' values. The goal is to reduce the complexity of the association phase, because the number of nodes in the tree is smaller than the number of significant chunk configurations. The automated association result can be considered as a starting point to be refined to better tailor data, adding or pruning elements that are not really interesting. More details of this alternative approach will be provided in the application example.

4 Methodology Application: An Example

The application of the proposed methodology to the university scenario, partially introduced in the previous discussion, is now presented.

As a *first step*, the designer of the application has to *produce the Context Dimension Tree of the application*, by means of one of the formalism we have discussed in Section 2. Figure 1 shows the CDG of our application; we consider three *holders*: the student, the instructor, and the employee of the administration office. Two possible *situations* may be relevant for the application: we can imagine to have the necessity of tailoring relevant information for a given *holder* during the session when exams are held, or in the other periods. To complete the model, constraints are introduced, deriving the graph.

Starting from this representation of the context model, as a *second step*, the *relevant chunk configurations* are automatically extracted by a Java tool. In our scenario, the forbid constraints used to reduce the chunk configurations (504 for the represented CDT) and remove the meaningless ones are:

holder = student \leftrightarrow *interest_topic* = students
holder = instructor \leftrightarrow *interest_topic* = faculty_students
holder = administration \leftrightarrow *interest_topic* = schedule
holder = administration \leftrightarrow *interest_topic* = add_material
holder = administration \leftrightarrow *situation* = exams

Once the relevant chunk configurations have been extracted (208 in the present example), as *third step*, the designer has to *select, from the global schema of data sources, for each meaningful chunk configuration the portion of data that are relevant*; in this way the chunk schema is defined for each chunk configuration. As an example of data source for the university scenario, we consider in this section an extension of the XML dataset of the Reed Courses ([13]), whose graphical representation of the XML Schema of the dataset is shown in Figure 3a.

As anticipated, we are investigating two ways for defining the chunk schemata (the former is described in Figure 2), described here in the following.

Relevant area per chunk configuration. The designer, for each significant *chunk configuration*, starting for the global schema (e.g., the XML schema) of the data sources, selects – manually – the relevant sub-schema. As an example, consider the chunk configuration

$$\langle \{ \text{instructor} \}, \{ \text{course_student} \}, \{ \text{classes} \}, \{ \text{this_year} \} \rangle$$

the portion of the source XML schema that is relevant for the considered combination of dimension values is depicted in Figure 3a, where grey nodes represent the chunk schema. This chunk configuration is used by the instructor to collect, for each course (s)he is teaching during the current year, the list of the students attending it. The following chunk configuration

$$\langle \{ \text{instructor} \}, \{ \text{course_student} \}, \{ \text{exams} \}, \{ \text{this_term} \} \rangle$$

is used to select for a given instructor all the information about the exams of the courses (s)he is teaching during the current term, and about students who are attending those exams. Figure 3b reports the chunk schema for the chunk configuration: note that in this case the information about the students are those related to an exam (and not a course).

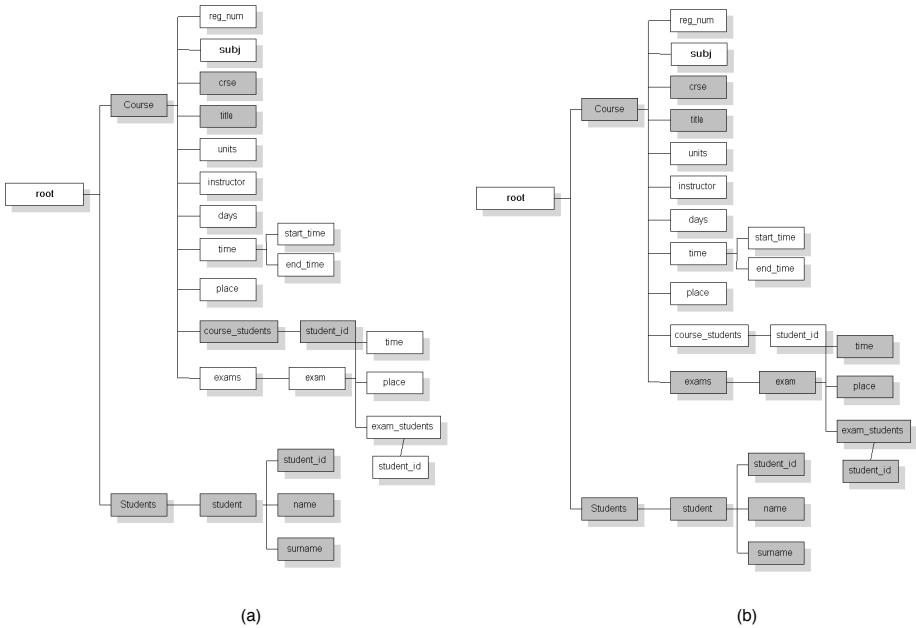


Fig. 3. The portions of XML Schema for the chunk configurations (a) $\{\{instructor\}, \{course_student\}, \{classes\}, \{this_year\}\}$ and (b) $\{\{instructor\}, \{course_student\}, \{exams\}, \{this_year\}\}$

Relevant area per node. Another possibility allows the designer, starting from the CDT, to manually select for each *dimension* value the relevant portion of the global schema, and automatically obtain a first approximation of the chunk schema as intersection of the relevant sub-schemata of each dimension value composing a chunk configuration. In this case, when considering intersection to combine different sets of relevant data, the portion of source schema relevant for each dimension value must include all the data that are relevant for that value. Thus, given the chunk configuration $C = \langle D_1, \dots, D_n \rangle$, where D_i is the set of values for the i -th ambient dimension, if $Rel(D_i)$ is the subset of the data source schema relevant for the set of values D_i , then the chunk schema of C is obtained as $\bigcap Rel(D_i)$. For example, in Figure 4 we annotated each node with a number 1, 2, 3 or 4 assessing the relevance of that node for the *instructor* value of the *holder* dimension, for the *course_student*, for the *classes* situation, and for the value *this_year* of the *time* dimension, respectively. In this last case, we notice that the XML document does not contain information about the validity of the published information, thus, we suppose they refer to the current year. The portion of XML Schema relevant for the chunk configuration

$$\{\{instructor\}, \{course_student\}, \{classes\}, \{this_year\}\}$$

is thus obtained as intersection of the relevant areas and corresponds to the set of grey nodes.

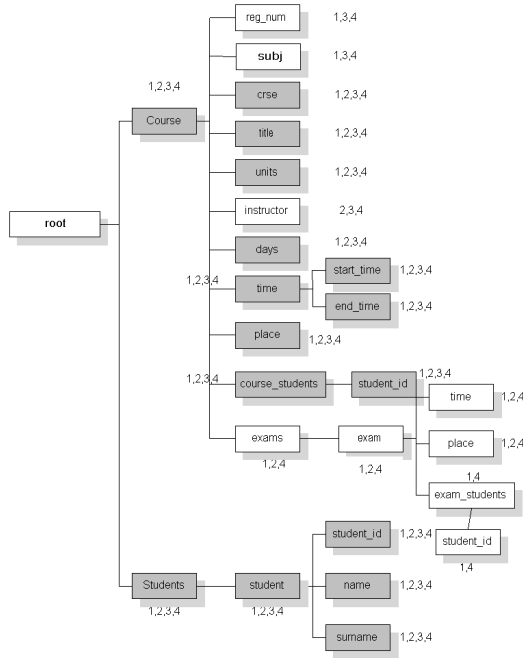


Fig. 4. The annotated xml schema: each number corresponds to a node in the CDT

For the *last step*, given a chunk configuration and its chunk Schema, the XQuery expressions to *retrieve the XML document related to that chunk* are produced. This task can be automatically performed similarly to [14], where a Java tool is presented, a tool that a) selects the chunk schema starting from the E-R representation of the structure of the data source, and b) produces the XQuery expression to retrieve the chunk data.

The XQuery expression to retrieve the data chunk for the chunk configuration

$\{\{instructor(VAR)\}, \{course_student\}, \{regular\}, \{this_year\}\}$

where parameter *VAR* is instantiated at run-time with the instructor name, e.g., I1, is

```

<chunk> {
  for $c in document("reed.xml")//Course[./instructor[text()='I1']]
  for $s in $c/course_students/student_id
  return <Course>{
    $c/crse
    $c/title
    <Students>
      for $st in document("reed.xml")//Students/student/student_id[text()=$s]
      <Student>
        $st/student_id
        $st/name
        $st/surname
      </Student>
    </Students>
  }
} </chunk>

```

The XQuery expression retrieves for each course taught by the instructor I1 the information about the students attending that course. Note that here the `Students` element is a sub-element of each `Course` taught by instructor I1.

The same process is iterated for each significant chunk configuration, leading to a set of data chunks, to be then collected on the user's device.

5 Conclusions and Future Work

In this work we have presented a general methodology to support the designer of context-aware applications in all the phases related to the context definition and the chunk extraction.

As an ongoing work we are integrating the algorithm to extract chunk configurations in the visual tool that automatically generates XQuery expressions to collect data to be stored in the user's device, starting from an Entity Relationship representation of the global schema of the data sources. Moreover, we plan to extend the tool in order to manage XML Schemata and OWL ontologies as well.

References

1. Torlone, R., Ciaccia, P.: Management of user preferences in data intensive applications. In: Proc. of the 11th Italian Symp. on Advanced Database Systems, SEBD. (2003) 257–268
2. Agrawal, R., Wimmers, E.L.: A framework for expressing and combining preferences. In: Proc. of the 2000 ACM SIGMOD Int. Conference on Management of Data, ACM (2000) 297–306
3. De Virgilio, R., Torlone, R.: A general methodology for context-aware data access. In: Proc. of the Int. Workshop on Data engineering for wireless and mobile access, ACM (2005) 9–15
4. Buchholz, S., Hamann, T., Hübsch, G.: Comprehensive structured context profiles (cscp): Design and experiences. In: 2nd IEEE Conf. on Pervasive Computing and Communications Workshops (PerCom 2004 Workshops), (2004) 43–47
5. MAIS: Multi channel adaptive information system (<http://www.mais-project.it/>)
6. Aberer, K., et al.: Emergent semantics: Principles and issues. In: Invited paper at 9th International Conference on Database Systems for Advanced Applications (DASFAA 2004), Springer-Verlag Lecture Notes in Computer Science (LNCS 2973), (2004) 25–38
7. Ouksel, A.M.: In-context peer-to-peer information filtering on the web. *SIGMOD Record* 32(3) (2003) 65–70
8. Bolchini, C., Curino, C., Schreiber, F.A., Tanca, L.: Context integration for mobile data tailoring. In: Proc. IEEE/ACM Int. Conf. on Mobile Data Management, (2006)
9. Bolchini, C., Curino, C., Quintarelli, E., Schreiber, F.A., Tanca, L.: Context-addict. Technical Report 2006.044, Dip. di Elettronica e Informazione, Politecnico di Milano (2006)
10. McGuinness, D.L., van Harmelen, F.: OWL Web Ontology Language Overview, W3C Recommendation (2004)
11. Curino, C., Quintarelli, E., Tanca, L.: Ontology-based information tailoring. In: Proc. IEEE 2nd Int. Workshop on Database Interoperability (InterDB 2006). (2006) 5–5
12. dtd2xml: A Conversion Tool from DTD to XML Schema (2004)
13. The XML document of Reed Courses: (<http://www.cs.washington.edu/research/xmldatasets/data/courses/reed.xml>)
14. Pigozzo, P.: Metodologia e strumento per la traduzione di diagrammi E-R in Schemi XML normalizzati e la generazione automatica della porzione di dati da memorizzare su un dispositivo mobile. Master's thesis, Politecnico di Milano (2005)