

# Context-Based Data Tailoring for Mobile Users\*

Letizia Tanca

Dipartimento di Elettronica e Informazione – Politecnico di Milano  
Piazza Leonardo da Vinci, 32—20133 Milano (Italy)  
tanca@elet.polimi.it

**Abstract:** Independent, heterogeneous, distributed, sometimes transient and mobile data sources produce an enormous amount of information that should be semantically integrated and filtered, or, as we say, tailored, based on the users' interests and context. Since both the user and the data sources may be mobile, and communication unreliable, caching the information on the user device is certainly useful. Thus, we propose to exploit knowledge about the user, the adopted device, and the environment - altogether called *context* - to the end of *information tailoring*. The key is *context-aware data design* where the notion of *context* must be formally defined, together with its role within the process of information tailoring. This paper presents a context model, called *Context Dimension Tree*, which plays a fundamental role in tailoring the information domain model within the framework of the *Context-ADDICT* project, currently under development at Politecnico di Milano. To conclude, we report on other context models, along with a brief comment on their relevant features.

## 1 Introduction

Today we are living an epochal change, whereby the advent of the internet and the development of the communication technologies have completely modified the focus of information retrieval, from the struggle for finding information and organizing it to that of appropriately reducing the enormous stream of available data. While the traditional problems typical of the data integration field are far from being solved, new challenges have also to be faced: integration of unknown-in-advance data sources, automatic semantic extraction, data filtering. Mobility is, at the same time, becoming crucial for people, emphasising old challenges while bringing to the surface new ones.

The Context-ADDICT system's aim is to address the above mentioned challenges, with particular emphasis on the notion of *context*: indeed, database design for mobile applications must model two different realms: the reality of interest, which is captured by the information domain model, and the user/device context. Classical data models, at a conceptual or at a logical level, are perfectly suited to represent the former, while context modelling presents different challenges and needs appropriate consideration.

“Context” is a rather general concept and, although commonly accepted and seemingly clear, has been interpreted in many different ways in various fields of research such as

---

\*This research is partially supported by the Italian MIUR projects: ARTDECO (FIRB), and ESTEEM (PRIN).

psychology, philosophy and computer science: Section 4 is devoted to a quick review of the various meanings of context in Computer Science.

Our context model, called *Context Dimension Tree*, plays a fundamental role in tailoring the target application data according to the user information needs. The tree can be specified in several ways; in particular we propose, in [CQT06], an ontological representation in the OWL language [MvH04], and an XML representation [BQ06]. In the Context Dimension Tree, the root's children are the *context dimensions* which capture the different characteristics of the device users and of the context they are acting in; by specifying a value for each dimension, a point in the multidimensional space representing the possible contexts is identified. A dimension value can be further analyzed w.r.t. different viewpoints, generating a subtree in its turn. However deep the tree, the complex structure originating from the composition of the Context Dimension Tree leaves is called a *context* or *chunk configuration*, and determines a portion of the entire data set, i.e., a *chunk*, specified at design-time, to be selected later, at run-time, when the corresponding context becomes current. Figure 1 shows an example – discussed below – modeling the possible contexts of an archaeological application. Black nodes represent dimensions and sub-dimensions.

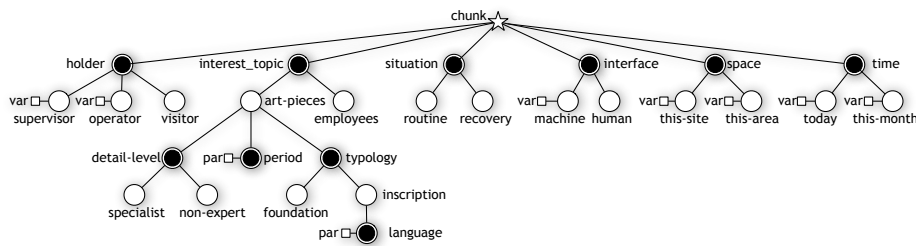


Figure 1: The Context Dimension Tree for the running example on archaeological sites.

The paper is organized as follows: a brief introduction to the Context-ADDICT project and system architecture is presented in Section 2. Section 3 presents the Context Dimension Tree, along with some considerations on its use. Section 4 briefly reports on other context models, considering their relevant features and some considerations comparing those proposals to ours, and finally the conclusions are drawn.

## 2 Context-ADDICT

Goal of *Context-ADDICT* is to discover and wrap data sources whose contents are accessible and relevant w.r.t. the application, and make their data available on the users' mobile devices, appropriately tailored on the user's current context and information needs. In this section we briefly describe the Context-ADDICT architecture [BCST06] and context model [BQ06].

The overall system is composed by three main subsystems, each one devoted to a specific task:

- The *Design-Time subsystem* supports the designer in the context-modeling activity and in modeling the information domain. The latter can be represented as an ontology or, alternatively, by any data model: if the data model is not an ontology, the support of a domain ontology will still be needed, since it will become a precious tool for dynamic integration support.
- The *Run-Time Schema Level subsystem*, composed of several modules, performs data source discovery and wrapping, schemata integration and tailoring.
- The *Run-Time Data Level subsystem*, once the schemata have been integrated and tailored, is devoted to the actual data movement, to synchronizing and integrating the data instances only for the pieces of information considered relevant, and to query processing (both local and remote).

The *Design-Time subsystem* must, first of all, support the designer in the modification of an existing Domain Ontology or in its creation from scratch. The Domain Ontology captures the main concepts of the information domain; we expect it to provide a well accepted general taxonomy enriched of the main relations among concepts, even if not all the concepts' attributes and relations are detailed.

If the data model chosen to represent the information domain is not an ontology, also the information schema, represented in the chosen data model, must be present – either already existing or produced from scratch–. Along with it goes a specification of the relationships of its concepts with those of the Domain Ontology. The Domain Ontology supports the system at run-time, in the integration of the global information schema with the data source schemata.

The *Design-Time subsystem* also supports the designer in the context design activity. The Context Dimension Tree is used to represent the knowledge of "*What may the activities and interests of the various users be, within this application domain?*". It is created by the designer via a syntax oriented *Dimension Tree Editor*. Following a precise methodology [BQ06], the tool guides the designer in the process of defining the context dimensions and their values.

Once the Context Dimension Tree has been created, the designer has to associate each context (or *chunk configuration*), with the schema portion representing the data which are relevant w.r.t. that context, thus knowledge about "*What part of the domain is relevant for a given user in a certain context?*". This process is heavily application dependent, and cannot be performed automatically in any sense; the *Context Integrator* is a highly interactive tool, meant to give the designer an important role: setting the relationship between the context and the information domain schema means to be able to capture, for each context, the specification of the actual data to be delivered in that given context, discarding unnecessary information. This association is done in a semi-automatic way: the designer specifies the schema portions declared relevant to each context element, represented in the tree by a white node – e.g., in the archaeological example, a supervisor will be assigned a portion of data, different (at least partially) from the data related to the visitors' role –. The

system completes the specification by appropriately combining, in each context, the data relevant to its elements. The result is the definition of a view over the domain information schema for each possible context.

*Run-Time Schema Level subsystem:* ambition of *Context-ADDICT* is being able to capture datasources which might be fully heterogeneous in terms of schemata, data format and access interfaces. At run time, a *Data Source Discovery Service* will be in charge of actually discovering datasources and making them reachable. This module may heavily vary depending on the specific scenario we are considering, from a centralized server to a set of fully distributed discovery procedures (see [CST04] for a detailed discussion), from a mere syntactic matcher to a semantical filter. The datasources may range from Relational Databases to XML datasources, to Web Services, to sensor networks, but the key point is that their schemata will be transformed into a common format and then operated upon in a uniform global representation, as shown in [CLN99]. Some of these datasources will be *cooperative*, i.e., they will be aware of their participation to *Context-ADDICT*; in this case they will provide a description of the available data in the common format. Other, occasional data sources may offer heterogeneous interfaces such as the DDL specification of the database schema or a set of web pages: in the latter scenario, *Context-ADDICT* provides a set of wrappers and wrapper generators to automatically translate the schema into the common model. The domain ontology supports this phase, by mediating between the semantics of datasources' and domain concepts.

In general, data sources may appear and disappear during system working time; however, considering a snapshot of the system life at a given time, a set of datasources schemata, a global information schema and a Domain Ontology are available; thus integration is needed. This operation is performed at run-time either on the user's device or on a dedicated machine, depending on the deployment choices of the designer.

The integration operation is a rather general and well known problem, though far from having been solved. A lot of research effort has been devoted to make this process as automatic and precise as possible (see [EdBMMR04] for a survey). In *Context-ADDICT*, the *Integration Module* makes intensive use of an ontology matcher we have developed: X-SOM [Ors06] At the end of the integration, all the datasources' information is coherently integrated with the global schema. Since the correspondences drawn between the Context Dimension Tree and the global schema are available, at this point the data tailoring part comes into play: to reduce the amount of data to be managed on the user device, the view definitions produced in the design phase are mapped to appropriate queries over the datasource's and used to tailor the information coming from them, possibly instantiated by means of actual values describing the current user context. For instance, in the archaeological example, if the user (device *holder*) is an operator during on-site work, the data should contain information about the art pieces s/he is in charge of on that day, or in that period. Considering a medical database example, if the device user is a doctor during house call time, the data should be tailored to contain his/her patients' information related to that day's envisaged calls. Thus, from each data source, only the information which is relevant to that context will be physically integrated on the device. During the described process, several metadata have been recorded, together with the data schema, in order to enable query processing and synchronization, in charge to the Data Level subsystem.

The *Run-Time Data Level subsystem* deals with the actual data transfer, thus, together with on-line query processing, it is responsible for data synchronization and local data management. Each datasource schema will contain a description of how each concept of the datasource has been stored. This can be done by means of metadata or be explicitly coded within the transformation rules used by the source wrapper. For example, the fact that the concept “teacher” of the datasource 1 is stored in a table named “professor” of a database named “university” reachable at a given URL is captured by transformation rules between the relational model and the *Context-ADDICT* internal model. In the current implementation, where the global information schema is represented as an ontology, we have a translator which interprets the DDL of a relational database and produces an ontology, in its turn integrated with the Domain Ontology. Here, the concepts “teacher” and “professor” are recognized as synonyms, and a special mechanism generates virtual identifiers for tuples of the relational database, in order to see them as ontology concepts and to be able to translate ontological queries into relational queries. Because of the tailoring phase, data synchronization and local data management are performed on a manageable amount of data, actually relevant to the user. A Local Data Management service such as the one presented in [BCG<sup>+</sup>04], as well as mechanisms for data synchronization are required together with a set of caching policies. Apart from the caching and storage policies, the process data synchronization will be similar to distributed and heterogeneous query processing [Kos00, MHMM05], with the addition of local data storage.

### 3 The Context Dimension Tree

The Context Dimension Tree is an advanced user profile and context descriptor based on the concept of *dimension*, an extension of the context and user interest model presented in [BST07]. It is used to describe systematically the user needs, and to capture the context the user is acting in. In [BST04] and [BST07] it has been shown how the designer can use an array of dimensions to capture the different characteristics of a user profile and of the context of an application; here the array has been refined into a tree to enrich the modeling possibilities.

A dimension captures an aspect of a context or of a user profile. In our experience, some dimensions are very common in most applications: here we list them, although it may happen that only a subset of them be needed, or that other dimensions come into play.

- **Holder:** The various user categories involved. In our examples the device holders may be supervisors, operators and visitors.
- **Interest Topic:** The various areas of interest for the possible users of an application; in the archaeological example we consider “art pieces” and “employees”.
- **Situation:** Different phases of the application life, which might be different even for a same interest topic. In the archaeological case we have considered a routine situation as opposed to a recovery phase, which becomes current in case of emergency.

In a personal medicare application, “hospitalized” and “at-home” may be different situations to be taken into account.

- **Space:** refers to the place where the user is currently located. It might be represented by GPS coordinates or by any other location information, and its granularity may vary depending on the application: for example, a museum “room” or a whole “city” or a “region”. Depending on the application need, the space dimension can be taken as relative (e.g. “this site”, in our example) or absolute (e.g. “Piazza Duomo in Milan”).
- **Time:** a temporal indication based on the current time. Its granularity may vary: for example, one could choose the “current month” or “last year”. Like in the case of the space, the time dimension can be taken as relative (e.g. “today”, in our example) or absolute (for example, “January 2007”).
- **Interface:** a dimension describing the kind of readings that will be done. In the running example we have considered that some data will be used by humans, directly perusing text and multimedia information, but also that some specific sites could offer totems or other electronic devices that, when put in contact with the mobile device (for instance via an RFID reader), might enact procedures or simply update the data contained in the device. Another good example of the use of this dimension is the smartcard database of a patient in a medical application; in this case, a hemodialyzed patient could simply insert his/her card into the hemodialysis machine, which would react by self-scheduling the hemodialysis intervention to be carried on the patient. In both cases, in order for the machine to be synchronized with the mobile device there is no need of human-interpretable information; rather, codes and identifications are in order.

As said before, not all the listed dimensions are always necessary, and more might be required. For example, in many applications a dimension “Ownership” might be needed, describing the access rights to the data, and used to enforce privacy and security issues that may come into play. It will be the designer’s task to establish, following the methodology guidelines, which dimensions are appropriate for the application s/he is designing.

Let us refer, for the moment, only to the topmost part of the tree of Figure 1, which contains exactly the dimensions listed above. For each dimension, the designer defines a set of admissible values, called *dimension values*. A dimension value is an instantiation of the concept represented by that dimension; by means of dimensions and dimension values, the context designer describes all the possible user roles (holders) and contexts, i.e. the dimensions define a multidimensional space where each point represents a potential user role and context. The table of Figure 2 lists, for each dimension, a set of possible values. In principle, these values could be combined in every possible way, each representing a possible (component of a) context, i.e., a *chunk configuration*; we will see later that this is not always the case. An example of possible combination is the following:

*holder*=supervisor, *interest\_topic*=art-pieces, *situation*=recovery, *interface*=human,  
*time*=this-month, *space*=this-area

Such a configuration defines the data about art-pieces needed by the site supervisor when performing a recovery action. The data must be human readable (text or other), and concern the whole area of the current archeological site that are listed in the database, for the current month. Each configuration will be related to the set of relevant concepts of the domain, thus to a set of relevant data in the datasources: the so-called *data chunk*.

holder	interest_topic	situation	interface	time	space
operator	art-pieces	routine	human	today	this-site
supervisor	employees	recovery	machine	this-month	this-area
visitor					

Figure 2: An example of dimensions and their possible values.

This level of granularity may suffice for some of the dimensions (such as for instance the *situation* and *interface* dimensions), but in a more general case a further refinement through a hierarchy of values (such as the case of the *interest\_topic*) may be needed. Let us consider, in the archaeological information domain, the concept of *art-piece*. Here the user might be interested in different categories of historical data (e.g., foundations rather than inscriptions), or different levels of detail in the information (e.g., specialized or non-expert). This is the reason why the Context Dimension Tree allows the hierarchical specification of further levels of detail.

Let us now explain, on the example of Figure 1, the Context Dimension Tree morphology. The root of the tree corresponds to the entire dataset which needs to be tailored. First-level nodes are the top dimensions; a sub-tree rooted in a dimension expresses the specialization of the values of that dimension, towards a fine-grained characterization that allows a more refined tailoring of the data.

Two main types of nodes are used: *dimension nodes* (in black) and *concept nodes* (in white). Each “generation” contains nodes of the same colour, and colours are alternated while descending the tree. A concept may be characterized by different aspects, which explains why the corresponding node becomes the father of new sub-dimensions. Consider Figure 1: while the children of the *situation* dimension are the values already considered in the table of Figure 2, the *interest\_topic* value *art-pieces* is subject to further refinement: it can be analysed in terms of *detail-level*, *period* and *typology*, which become the new sub-dimensions, admitting new values. Note that the process may obviously be iterated any number of times, and choosing the right level of detail is a design trade-off.

Consider now the sub-dimensions of *art-pieces*: while *detail-level* and *typology* have white children (i.e., their values), *period* only features a small square, an *attribute node*; this attribute is a selection parameter whose instances represent the possible values of the *period* dimension, e.g., the art piece century, like VI b.c., V b.c., IV b.c.. Thus, dimension branching can be represented in both ways: by explicitly drawing the values as white nodes, or by indicating a parameter whose instances are the dimension values. This does not modify the expressive power of the model: rather, it makes it more readable, and more usable for the designer.

Sibling white nodes are mutually exclusive since they represent orthogonal concepts, while sibling black nodes represent the different (sub-)dimensions which define a concept. There-

fore, in descending the tree to build a chunk configuration, at each level, only one white node, but any number of black ones, among siblings, may be selected.

Leaf nodes can be either black or white: both kinds of nodes may feature an attribute: the former case was introduced above; in the latter case the attribute represents a variable indicating how to select a specific set of data instances. As an example, consider the variable associated to the `supervisor` holder: it is an identifier used, at run-time, to select exactly the data related to each specific supervisor.

Once the tree has been defined, the list of chunk configurations is derived; a chunk configuration on the Context Dimension Tree is expressed in terms of a set of values, one for each (sub-)dimension: when the tree is deeper than two levels, the values may be at any level in the tree. Equation 1 shows the chunk configuration for tailoring data for a visitor. The data must be human readable and are related only to the site the visitor is currently seeing. Moreover, the considered situation is a routine one. In this example, the `interest_topic` dimension has been instantiated with two white nodes, `non-expert` and `inscription` (values, for *detail-level* and *typology* resp.), that refine the `art-pieces` concept.

$$\langle \{\text{visitor}\}, \{\text{non-expert}, \text{inscription}\}, \{\text{routine}\}, \{\text{human}\}, \{\text{this-site}\} \rangle \quad (1)$$

We also found useful to express constraints or preferences on the allowed combinations of the dimension values; in fact, not all of them make sense for a given scenario, thus the meaningless ones must be discarded. For example, the interest topic `employees` (meaning the administrative data about employees) is significant when the holder is the `supervisor`, but not when s/he is a `visitor` or an `operator`.

As a consequence, we should neither include, in the set of possible chunk configurations, those having `holder=visitor`  $\wedge$  `interest_topic=employees`, nor those with `holder=operator`  $\wedge$  `interest_topic=employees`.

Figure 3 shows the context design and its association with the information domain schema (in this example, an XML schema). The top-leftmost corner of the figure represents the Context Dimension Tree, whereby all the possible chunk configurations are derived (bottom-left corner). Each chunk configuration is associated by the designer to the data portion which is relevant to the corresponding context (bottom, in the middle), and then the system automatically generates the X-Query which will tailor the corresponding data from the actual XML dataset (rightmost, bottom).

However, as can be expected, even after excluding the meaningless configurations, a medium Context Dimension Tree originates a huge number of chunk configurations, thus the task of associating relevant chunks to each of them is unpractical. Thus we have defined a set of policies which, once the designer has selected the schema portion to be associated to *each white node*, drive the system to combine them within each chunk configuration. The possible policies involve the use of different combination operators, such as intersection, join or semijoin [BQRT06]. The process is shown by Figure 4, where the central area displays all the node/subschema associations, to be combined into the chunks corresponding to the combination of those nodes.

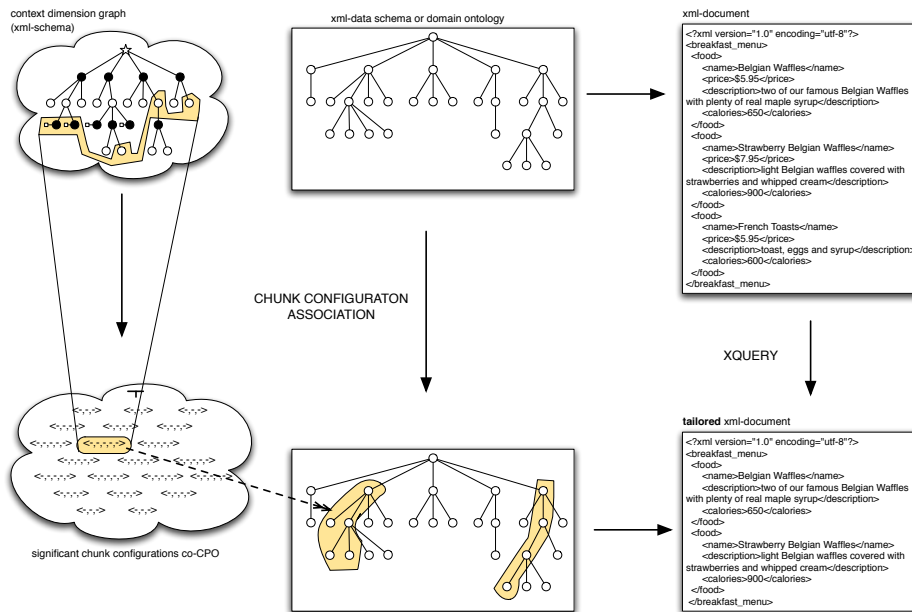


Figure 3: Design time, manual *Chunk* definition.

## 4 Related work on context models

Many approaches defining the notion of context have been proposed and several adaptive applications have been designed and implemented starting from them [TC03, AW00, BHH04, MAI, BST06, Aea04, Ouk03]. Although interesting context models already exist [SLP04, KO04, BDRar, RTA05], when the target application is *data tailoring*, [BCQ<sup>+</sup>06], some features become relevant which are not all present within a unique context model among the ones we have reviewed. We analyze the most well known models w.r.t. their main features, and, based on this, try to classify them into five categories. Some of the reviewed systems can be considered as fitting the features of all the considered categories.

In many systems, context awareness is perceived as the *capability, on the system's part, to adapt content presentation to different channels or to different devices* [KO04, CoD03, PdBW<sup>+</sup>04, GPZ05, BHH04, VTH06, BST06]. Most systems of this class are able to model the context at different levels of granularity, but do not necessarily include a refined mechanism of location and time awareness. User profiling is often present, normally feature-based, and the context specification is generally quite informal. Context models of this category are not very flexible, meaning that they are normally designed for specific applications. Sometimes automatic context learning features are available, but not to the point of allowing to reason about context.

A different class of models has the task of *modeling location and environment aspects*

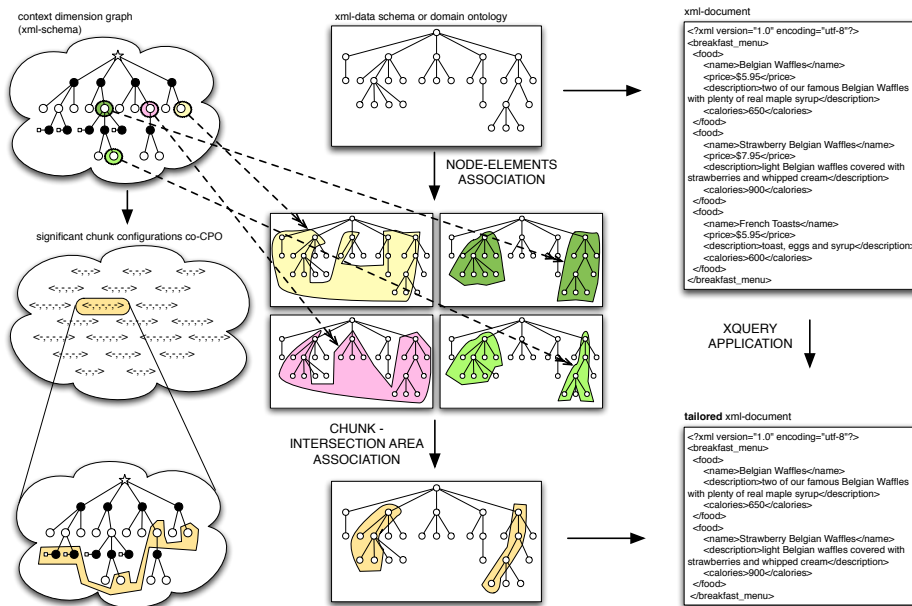


Figure 4: Semi-automatic *Chunk* definition.

[KO04, FC04, CoD03, PdBW<sup>+</sup>04, GPZ05, SRA06, PNS<sup>+</sup>00]. In this case, a precise treatment of the time and space coordinates is needed. Most of these systems can model the context in a highly flexible way, and also context reasoning may be provided, offering a powerful abstraction mechanism. Since in this case the system acquires location and time information from various kinds of sensors, these systems are also able to control information quality, and to deal with information ambiguity.

There is also a class of models whose focus is on “*what the user is doing*” [KO04, SRA06, SSR03, PNS<sup>+</sup>00]. Here, the context history becomes relevant and thus must be modeled, and also reasoning mechanisms are needed; time and space are taken into account if they provide information about the user’s current activity. The different models are described with different levels of formality; when available, automatic learning is used to guess user activity from sensor readings.

Interesting approaches model context as a matter of *agreement and sharing among groups of peers* [KO04, CPFJ04, Ouk03]. The focus of this group is on reaching an agreement about a context shared among peers. Differently from the other classes, here the context definition is reached in a distributed fashion; context reasoning, quality monitoring, ambiguity and incompleteness management are also key issues. Because of these needs, the context model must be rather well formalized. On the other hand, sophisticated location, time and user profiling treatment is not of primary importance.

The last category concerns the *tailoring problem* [KO04, SHC<sup>+</sup>06, BCQ<sup>+</sup>06]. Actually,

besides data tailoring, also relevant functionalities and services may be selected according to contextual information. Here, context history and reasoning are often not provided, while time, space and user profile are in general highly developed and well formalized. These models provide the possibility to describe different kinds of contexts, thus flexibility is high, the level of granularity in describing the context is variable, and constraints on valid contexts can be expressed.

Although a lot of work has been done, the representation and management of the context can hardly be considered an assessed issue. Our experience has convinced us that, due to the complexity of the problem as a whole and to the multitude of different applications, the best models are those that, although being quite general, have a well defined focus, and try to support only one of the above mentioned context-modeling categories. By contrast, the systems whose aim is to be completely general, and to support the context modeling problem as a whole for any possible application, tend to lack practical applicability and usability.

## 5 Conclusions

The *Context-ADDICT* system faces a very challenging scenario where distributed, heterogeneous, independent, maybe mobile and transient data sources come into play. The goal is to automatically integrate and tailor the available data, based on a context model named Context Dimension Tree.

The ultimate goal of *Context-ADDICT* is to support specific applications with an appropriate integration and tailoring layer, offering the application designer the chance to focus on the business logic. A design methodology guides the application designer in the task of modeling the information domain as well as the possible application contexts.

During the work on *Context-ADDICT*, we have seen that different context subproblems and applications have almost incompatible requirements, and common solutions are still not available, and possibly not useful. As a consequence, the context model should be chosen according to the target application, or possibly defined from scratch, based on the specific requirements. For this reason, after an accurate review of the existing models, we have decided to design the Context Dimension Tree. To this day our experience has proven it to be well suited for the data tailoring task; however, we plan to test it on larger, industrial applications, and also to apply it to service configuration.

## 6 Acknowledgements

This work was not conceived as a one-person task, and indeed it has not been. Besides the project pioneers, Cristiana Bolchini, Fabio A. Schreiber, and myself, the *Context-ADDICT* project involves a number of researchers and PhD, graduate and undergraduate students among which I only mention some: Carlo Curino, Giorgio Orsi, Elisa Quintarelli, Antonio

Penta and Rosalba Rossato. To all of them I wish to express my thanks for being there and making our work such a pleasant experience.

## References

- [Aea04] K. Aberer and et al. Emergent Semantics: Principles and Issues. In *Invited paper at 9th International Conference on Database Systems for Advanced Applications (DASFAA 2004), Jeju Island (Korea)*, pages 25–38. Springer-Verlag Lecture Notes in Computer Science (LNCS 2973), ISSN 0302-9743, ISBN 3-540-21047-4, March 2004.
- [AW00] R. Agrawal and E. L. Wimmers. A Framework for Expressing and Combining Preferences. In *Proc. of the 2000 ACM SIGMOD Int. Conference on Management of Data*, pages 297–306. ACM, 2000.
- [BCG<sup>+</sup>04] Cristiana Bolchini, Carlo Curino, Marco Giorgetta, Alessandro Giusti, Antonio Miele, Fabio A. Schreiber, and Letizia Tanca. PoLiDBMS: Design and Prototype Implementation of a DBMS for Portable Devices. In *Proc. of the 12th Italian Symposium on Advanced Database Systems (SEBD 2004)*, pages 166–177, 2004.
- [BCQ<sup>+</sup>06] Cristiana Bolchini, Carlo Curino, Elisa Quintarelli, Fabio A. Schreiber, and Letizia Tanca. Context-ADDICT. Technical Report 2006.044, Dipartimento di Elettronica e Informazione, Politecnico di Milano, 2006.
- [BCST06] Cristiana Bolchini, Carlo Curino, Fabio A. Schreiber, and Letizia Tanca. Context integration for mobile data tailoring. In *Proc. IEEE/ACM of Int. Conf. on Mobile Data Management*. IEEE, ACM, May 2006.
- [BDRar] M. Baldauf, S. Dustdar, and F. Rosenberg. A Survey on Context-Aware systems. *Int. Journal of Ad Hoc and Ubiquitous Computing*, to appear.
- [BHH04] S. Buchholz, T. Hamann, and G. Hübsch. Comprehensive Structured Context Profiles (CSCP): Design and Experiences. In *2nd IEEE Conf. on Pervasive Computing and Communications Workshops (PerCom 2004 Workshops)*, pages 43–47. IEEE Computer Society, 2004.
- [BQ06] Cristiana Bolchini and Elisa Quintarelli. Filtering mobile data by means of context: a methodology. *Springer-Verlag, LNCS 4278*, pages pp. 1986–1995, 2006.
- [BQRT06] Cristiana Bolchini, Elisa Quintarelli, Rosalba Rossato, and Letizia Tanca. A comparative study and a formal framework. . . . Technical Report D.42, Esteem Project - Progress Report, 2006.
- [BST04] Cristiana Bolchini, Fabio A. Schreiber, and Letizia Tanca. A context-aware methodology for very small data base design. *SIGMOD Record*, 33(1):71–76, 2004.
- [BST06] Cristiana Bolchini, Fabio A. Schreiber, and Letizia Tanca. Data management. In Barbara Pernici, editor, *Mobile Information Systems - Infrastructure and Design for Adaptivity and Flexibility*, chapter 6, pages 155–176. Springer, 2006.
- [BST07] Cristiana Bolchini, Fabio A. Schreiber, and Letizia Tanca. A Methodology for Very Small DataBase Design. *Information Systems (Available on-line)*, 32(1), March 2007.

- [CLN99] D. Calvanese, M. Lenzerini, and D. Nardi. Unifying Class-Based Representation Formalisms. *Journal of Artificial Intelligence Research*, 11:199–240, 1999.
- [CoD03] CoDaMoS development team. The CoDaMoS Project, 2003.
- [CPFJ04] Harry Chen, Filip Perich, Tim Finin, and Anupam Joshi. SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications. In *International Conference on Mobile and Ubiquitous Systems: Networking and Services*, Boston, MA, August 2004.
- [CQT06] Carlo Curino, Elisa Quintarelli, and Letizia Tanca. Ontology-based Information Tailoring. In *Proc. IEEE of 2nd Int. Workshop on Database Interoperability (InterDB 2006)*, pages 5–5, April 2006.
- [CST04] Augusto Celentano, Fabio A. Schreiber, and Letizia Tanca. Requirements for Context-Dependent Mobile Access to Information Services. In *Proc. 10th Int. Workshop on Multimedia Systems (MIS)*, pages 60–65, 2004.
- [EdBMMR04] M. Ehrig, J. de Bruijn, D. Manov, and F. Martín-Recuerda. D4.2.1 State-of-the-art survey on Ontology Merging and Aligning V1. Technical report, Institut AIFB, Universität Karlsruhe, 2004.
- [FC04] P. Fahy and S. Clarke. CASS - Middleware for Mobile Context-Aware Applications. In *Mobisys 2004 Workshop on Context Awareness*, 2004.
- [GPZ05] T. Gu, H. Keng Pung, and Da Qing Zhang. A service-oriented middleware for building context-aware services. *Journal of Network and Computer Applications*, 28(1):1–18, 2005.
- [KO04] M. Kaenampornpan and E. O’Neill. An Intergrated Context Model: Bringing Activity to Context. In *Workshop on Advanced Context Modelling, Reasoning and Management - UbiComp 2004*, 2004.
- [Kos00] D. Kossmann. The state of the art in distributed query processing. *ACM Comput. Surv.*, 32(4):422–469, 2000.
- [MAI] MAIS. MAIS: Multi Channel Adaptive Information System.
- [MHMM05] N. Marsit, A. Hameurlain, Z. Mammeri, and F. Morvan. Query Processing in Mobile Environments: A Survey and Open Problems. In *Proc. IEEE 1st Int. Conference on Distributed Frameworks for Multimedia Applications (DFMA)*, pages 150–157, 2005.
- [MvH04] D. L. McGuinness and F. van Harmelen. OWL Web Ontology Language Overview, W3C Recommendation, 2004.
- [Ors06] Giorgio Orsi. An ontology based data integration system: solving semantic inconsistencies. Master’s thesis, Politecnico di Milano, 2006.
- [Ouk03] A. M. Ouksel. In-context peer-to-peer information filtering on the Web. *SIGMOD Record*, 32(3):65–70, 2003.
- [PdBW<sup>+</sup>04] D. Preuveneers, J. Van den Bergh, D. Wagelaar, A. Georges, P. Rigole, T. Clerckx, Y. Berbers, K. Coninx, V. Jonckers, and K. De Bosschere. Towards an Extensible Context Ontology for Ambient Intelligence. In *Proc. 2nd European Symp. Ambient Intelligence (EUSAI 2004)*, Lecture Notes in Computer Science, pages 148–159. Springer, 2004.

- [PNS<sup>+</sup>00] D. Petrelli, E. Not, C. Strapparava, O. Stock, and M. Zancanaro. Modeling Context Is Like Taking Pictures. In *Proc. of the Workshop "The What, Who, Where, When, Why and How of Context-Awareness"* in CHI2000, 2000.
- [RTA05] Dimitrios Raptis, Nikolaos Tselios, and Nikolaos Avouris. Context-based design of mobile applications for museums: a survey of existing practices. In *MobileHCI '05: Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, pages 153–160, New York, NY, USA, 2005.
- [SHC<sup>+</sup>06] Yang S.J.H., A.F.M. Huang, R. Chen, Shian-Shyong Tseng, and Yen-Shih Shen. Context Model and Context Acquisition for Ubiquitous Content Access in ULearning Environments. In *Sensor Networks, Ubiquitous, and Trustworthy Computing, 2006. IEEE International Conference, vol.2*, pages 78–83, 2006.
- [SLP04] Thomas Strang and Claudia Linnhoff-Popien. A Context Modeling Survey. In *1st Int. Workshop on Advanced Context Modelling, Reasoning and Management*, 2004.
- [SRA06] Maria Strimpakou, Ioanna Roussaki, and Miltiades E. Anagnostou. A Context Ontology for Pervasive Service Provision. In *20th International Conference on Advanced Information Networking and Applications (AINA 2006), 18-20 April 2006, Vienna, Austria*, pages 775–779, 2006.
- [SSR03] H. Sridharan, H. Sundaram, and T. Rikakis. Computational models for experiences in the arts, and multimedia. In *Proc. of the 2003 ACM Workshop on Experiential Telepresence (ETP)*, pages 31–44, 2003.
- [TC03] R. Torlone and P. Ciaccia. Management of User Preferences in Data Intensive Applications. In *Proc. of the 11th Italian Symp. on Advanced Database Systems, SEBD*, pages 257–268, 2003.
- [VTH06] Roberto De Virgilio, Riccardo Torlone, and Geert-Jan Houben. A Rule-based Approach to Content Delivery Adaptation in Web Information Systems. In *7th International Conference on Mobile Data Management (MDM)*, page 21. IEEE, 2006.