Implementing a Knowledge-driven Hierarchical Context Model in a Medical Laboratory Information System

Jaan Pruulmann and Jan Willemson

Abstract— **This paper discusses a context model developed for medical laboratory information system of Tartu University Hospital laboratories (in Tartu, Estonia). Due to the size, structure and complex history of the laboratory infrastructure, managing user communications and even achieving a common vocabulary has become a serious task. Most of the concepts used by the laboratory personnel depend on the particular cultural, organizational or other types of context, thus reducing the problem of user communications to context management. In this paper we present a cross-linked hierarchical approach to building a context management engine that uses open context space and a flexible attribute value delegation mechanism. We discuss development objectives of the laboratory information system together with the design decisions made and summarize the first user experience. Even though the context engine is currently implemented for use in laboratory environment, its architecture is general enough to support a considerably larger variety of application domains.**

Keywords: context hierarchy, information systems, self-management

I. Introduction

The concept of context is not a very clearly specified one. There have been several attempts to define it proposed in the last decade. The first one, based on formal predicate logic, was started by John McCarthy [1]. His model has been developed by numerous authors and used in different AI systems, see e.g. [2], [3].

The second approach was motivated by information search domain and data warehousing. Specific problems arise here with gigantic amounts of legacy data available on the Internet, mostly without any reasonably formalized semantic structure or metadata. A possible solution is to apply heuristic algorithms and context models for backward "coding time context" reestablishment [4], [5].

The third approach was motivated by a very simple definition of context as the "surrounding environment" which can be represented by a number of external attributes such as user's position, external temperature etc. An important motivating scenario for this development has been guiding people in an interactive space [6]. For example, the research directions include context-aware pocket museum guides [7], clever pill-boxes and smart beds in hospitals [8]. Recent study by Kofod-Petersen and Cassens extends this research also by introducing context-awareness and context-sensitivity [9]. The current status of the whole approach is well described by Hardian [10].

All of the approaches presented above try to express the context as a set of specific attributes, occasionally together with some processing rules, but disregarding application data and business logic.

This gap was filled by the process-centric view on context management originally introduced in 1990s by Brézillon [11] and further developed by himself and his collaborates (see [12], [13], [14], [15]).

Brézillon uses a three-level model for context-related knowledge, dividing it first into *external* (essentially irrelevant or unavailable) *knowledge* and *contextual knowledge*. Parts of the contextual knowledge are in turn focused by running processes, thus extracting a *proceduralized context*.

On top of this model, process modeling is carried out using *contextual graphs* with arcs corresponding to sequential actions and nodes corresponding to context-dependent branching (and later recombination of the branches).

It may happen that the available contextual knowledge is not sufficient for selecting the branch. Some external knowledge is then needed and it may be obtained e.g. via user interaction. As a result of knowledge acquisition, the contextual graph may be modified as well.

Thus, Brézillon treats context as situation-specific knowledge that is partially focused by processes and at the same time drives the process selection in the contextual graph. The structure of the knowledge itself is not important in the Brézillon model. However, the authors of the current paper believe that imposing some internal structure to the knowledge space (which it intrinsically anyway has) can significantly support process-centric context modeling as well.

In this paper we will consider an extensive case study of a real medical laboratory information system and show, how, extending the hierarchically structured context space, it is also possible to embed process information in there to the extent, where the process centric context graph as a separate entity can be dropped altogether.

We will represent the available knowledge in an abstract way as a set of relations between different basic items (called taxons), thus forming a certain graph as well (in a way very much like our brain does [16]). On top of that graph, the knowledge space will be built in a way that all the process control decisions can be taken based on the local contextual information only. In our proposed model, processes not only use knowledge for control, but they can also cause new

Supported by Estonian Science Foundation grant #7081

Jaan Pruulmann is with Cybernetica, Aleksandri 8a, Tartu, Estonia, and University of Tartu, Institute of Experimental Physics and Technology, Tähe 4, Tartu, Estonia `jp@cyber.ee`

Jan Willemson is with Cybernetica, Aleksandri 8a, Tartu, Estonia, and University of Tartu, Institute of Computer Science, Liivi 2, Tartu, Estonia `janwil@cyber.ee`

links between taxons, thus developing the knowledge space further. As a result, neither of the processes and knowledge can be considered superior to the other, they are rather equally important aspects of the same phenomenon.

The paper is organized as follows. Section II describes the background application domain of a laboratory information system together with the development objectives. Sections III and IV describe the design principles and several characteristic subtaxonomies of LIS. Section V covers the structure of our context space together with the process management framework, and Section VI summarizes the results obtained in practical implementation. Finally Section VII makes some conclusions and sets directions for further work.

II. Background

Our research was initially motivated by the development needs of Laboratory Information System (LIS) for Tartu University Hospital (Estonia). When LIS development started, the first analysis exhibited serious inconsistencies in user requirements. Further analysis showed that this was just a result of ignoring individual context of situations where these requirements were stated and the inherent diversity of such situations.

In this section we discuss the application domain of LIS, its development objectives and requirements in more detail.

A. Application domain

This subsection gives a short description of the laboratory environment our system needs to operate within.

The main laboratory routine starts with an order from a doctor who sends a specimen from a patient to the lab for particular analyses. The analyses are then performed, the results are validated, and sent back by a laboratory doctor. Problems with communication arise mostly from the size and heterogeneity of the medical infrastructure. Some illustrative parameters are presented below to give the reader an idea about the complexity of the whole system.

- Laboratory of Tartu University Hospital serves approximately 1250 doctors with very different background who work for approximately 470 medical organizations including both private doctors in distant rural areas and top-level medical centers in large towns.
- The list of analyses has more than 800 items, but the taxonomy used to describe them is not very uniform. As a general rule, laboratories describe the results of the analyses in much more detail than the ordering doctors expect. The situation is complicated even further because of the need to interface with many third-party systems using proprietary code sets instead of widely accepted ones. Altogether 55 different code sets are used (including LOINC [17] and SNOMED[1]).
- An order usually consists of much more than one analysis and there is technological interference between many combinations of analyses – common resources can be used, they take different time or require different ways for specimen preparation, etc. As a result, besides the well-formalized main laboratory workflow there exist many rarely occurring exceptions that make up more than a half of all the cases.
- LIS does not cover one single laboratory, but three 24×7 ones and nine business-hours laboratories located at the distance of up to 5 km from one another partially in specialized clinics. There are 62 different workplaces, 87 laboratory workers and 32 external systems that communicate with LIS (including three large hospital information systems of their own right).

[1]`http://www.snomed.org`

B. Development objectives

The laboratory business processes in Tartu University Hospital existing before LIS project were the result of a long and complicated historical evolution. Different parts of the laboratory had very different traditions, often non-formalized and not uniform across the whole establishment. One of the design goals for the new LIS was to achieve a higher level of process specification and uniformness. Of course, it was not possible to change the state of affairs overnight, both because of long-standing traditions and 24×7 requirements. Rather there was need for a system which could start from the present state and evolve over time. Thus the respective context model had to support this kind of evolution as well.

Based on the ideology described in the introduction, it was clear that the knowledge base had to contain enough information to model the whole laboratory process. A closely related development objective was inclusion of the requirements concerning the communication between the clients and the services offered by LIS. It was also envisioned that the system should be able to manage itself to a certain extent based on the user interaction (corresponding to external knowledge acquisition and the implied graph rearrangement in the Brézillon model). We will see in Section VI how it worked out in practice.

III. Design principles

In order to keep the context well-manageable and well-usable, we will introduce some structure to our taxonomy. First we define the set S of all items relevant in our context space (see also Section V-A) and assign a suitable set of valued attributes to each item. Next we apply a hierarchical top-down clustering algorithm to the set S (see [18] for a good overview of different clustering methods). The resulting clusters will be called *taxons*; all taxons in our hierarchy will thus essentially correspond to some subset of S with the root corresponding to S itself.

Taxons may be thought of as more or less general concepts concerning all the elements of the corresponding subset (also called *elements of the taxon*). In any given business situation, a certain set of taxons is relevant. This set will be further referred to as *context*.

In order to refer to the taxons, we give every one of them a globally unique identifier and an identifier unique in the set of sibling taxons (also called *code*). The codes from the root of the hierarchy to the given taxon make up a *full path* *identifier*. Full path identifier fpi of a taxon has the form $fpi = fpi_{parent} + PathSeparator + code$ (where + stands for concatenation and fpi_{root} is an empty string).

We can split the taxon's concatenated fpi into two parts at any $PathSeparator$ obtaining $fpi = fpi_{CS} + PathSeparator + code_{CS}$. In such a case we say that this taxon has the code $code_{CS}$ in the codeset fpi_{CS}.

As noted above, items of S may have valued attributes. If all elements of some taxon have the same attribute, then it is natural to consider this attribute being assigned to the whole taxon. There are two types of taxon attributes. First, an attribute may have been used as a selection criterion for creating this taxon; we will refer to these as *classification attributes*. Naturally, the value of a classification attribute is fixed for the respective taxon and all its subtaxons.

Second, if an attribute was not used for classification, it can still be evaluated for the whole taxon. This value is automatically delegated to all the elements and subtaxons of the given taxon. This value can be overridden for some subtaxon by simply giving that attribute a new (say, more specific) value.

NULL is an acceptable value for non-classification attributes and its semantics is blocking the value delegation coming from higher taxonomy levels. Note that value delegation is not implemented by copying the values, but rather by runtime hierarchy traversal. This approach allows us to consider parts of the taxonomy tree in different contexts and evaluate the attributes accordingly.

As an example use scenario of value delegation approach consider the case of organizational relocation of a laboratory within the institution. All the internal relations between employees remain the same, but relations to the new management are introduced automatically with the described value delegation mechanism.

Taxonomic hierarchy in our approach is not a fixed entity. As noted in Section II-B, the LIS system is meant to support natural evolution of context space; hence the users must be provided with means to modify the taxonomic hierarchy themselves. In our framework, these modifications are generally performed by changing classification criteria in the nodes or by further classification of leaf nodes. As a result, new branches of the tree can be introduced or old ones moved to other locations.

User credentials for changing the classification are determined by the type of the classifier. In our taxonomic hierarchy, three types of classifiers can occur.

a) Fixed: Fixed classifier means that the classification for the particular node is predefined and can not be changed by the user.

b) List-based: List-based classifiers are used when it is natural to list all the possible values of the attribute. The user can add classifier values to the list and move elements between the lists. As a result, new branches of the hierarchy are created or some existing ones are relocated.

c) Value-based: Value-based classifiers are mostly used for numerical attributes. The classification is represented by non-intersecting union of intervals of real numbers. The user may extend the intervals and split them, inducing the respective changes in the underlying hierarchy as well.

One may note that deletion of the nodes (and the respective branches in the tree) is currently not supported at all. There are two reasons for that. First, functionality of our taxonomy is not affected if it contains some unnecessary nodes. Second, determining unnecessity of nodes may turn out to be rather complicated. It is, however, possible to join two existing sibling nodes in our hierarchy if their common parent uses list-based classification. Another way to get rid of an unused taxon in a given context is to link it somewhere else in the hierarchy.

Besides attribute values, taxons may also have some meta-parameters. First, effective lifetime of a taxon may be given using $valid\_since$ and $valid\_until$ parameters. Second, in our taxonomic hierarchy it is possible to use a depth modifier that hides all sub-levels below the depth determined by the modifier value. This property is useful when building up a selection list for the user interface based on the given taxonomy or when the taxonomy has a large number of taxons on deeper levels. See Section VI-A for a more detailed description of the class model for context data.

IV. Hierarchical taxonomy of LIS

In this section we will describe some subtaxonomies of our hierarchical LIS taxonomy relevant in the context model.

A. Taxonomy of locations

Human actions mostly take place in some physical location and this location is often an important piece of contextual information. However, the level of detailization of location data may greatly vary depending on the particular context. For example, when addressing a message to a neighboring office we rarely tend to start the address with something like "Europe, European Union, Estonia ...". At the same time, when sending a letter from a different continent to the same office, this prefix becomes essential.

Our context model provides a convenient solution for this addressing problem. Noting that locations can be hierarchically described in a very natural way, initializing the location taxonomy is rather an easy task. Each user operates in his/her own context that usually also determines the current location taxon (office table, part of the laboratory, etc). If the address of some remote taxon is needed, we can follow the two paths from these taxons towards the root of the hierarchy until we hit a common node on both paths. The taxon corresponding to that node determines the common codeset fpi_{CS} that does not need repeating in the address.

B. Taxonomy of attributes

We have declared a number of attributes for location taxonomy above and obviously we need to add other attributes, too. It will be helpful if we will be able to handle attributes in a systematic manner. Our approach is to create a separate taxonomy "attribute" classifying all attributes used to describe some properties of taxons.

Figure 1 illustrates our taxonomic hierarchy of attributes created at system startup. It can later be extended by the users according to business logic and user requirements.

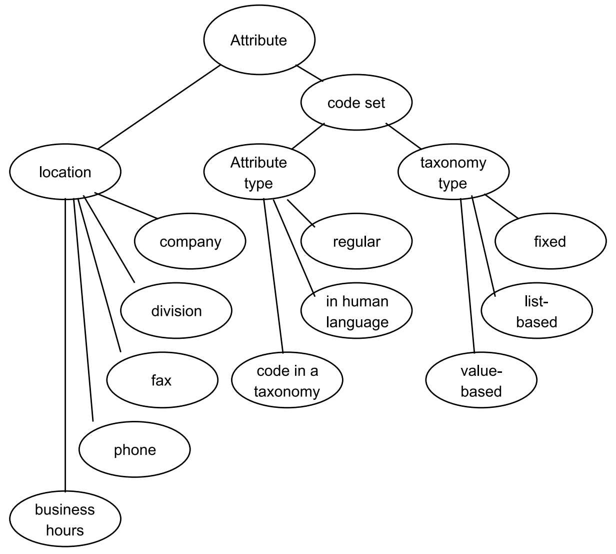

Fig. 1. Initial fragment of the attribute type taxonomy

C. *Taxonomy of human languages*

When building user interfaces for information systems, the problem of adequate object naming often occurs. This problem is especially important in a heterogeneous and multi-cultural environment, as is the case with our LIS example.

In order to carry human-language information about the taxons, our taxonomy allows a special subclass of attributes, namely textual attributes. Besides containing a (textual) value, such an attribute must also refer to a specific language (see Figure 3 in Section VI-A for a more detailed class model).

It is of course natural to include language information into the current context, but defining what exactly is a language is far from being trivial. Languages may have different variations (dialects, slangs, local terminology) that are understood only within a limited context (e.g. in one particular laboratory or by a certain group of users). Some words may have different meanings or even have no meaning at all in some language variants.

In order to address these problems, all the identified language variations can be gathered into the hierarchical taxonomy of human languages. Each user may be given his/her preferred language from the taxonomy. If the user does not have the preferred language attribute set, the *working-language* attribute of location taxonomy defined for the particular work place is used. If the value of some textual attribute is not found in a lower level language variation, we can keep moving up the hierarchy until we find the first suitable language and use the name in that language for user communication.

Figure 2 illustrates a fragment of the language taxonomy of our LIS.

V. Knowledge and processes

In this section we will discuss the structure of our knowledge/context space and the corresponding process management framework.

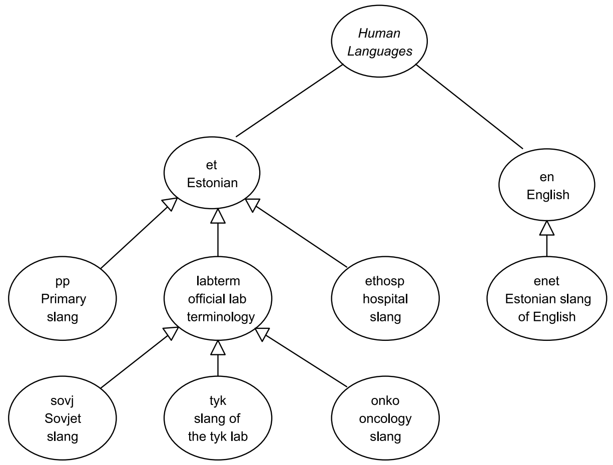

Fig. 2. Fragment of the taxonomy of human languages

A. *Context space*

In Sections IV-A, IV-B and IV-C we saw several mechanisms that help us building up individual taxonomies. There are still a few other steps required to reach the full context space.

First we need to take into account third party code sets/taxonomies. For example, our LIS uses Logical Observation Identifiers Names and Codes (LOINC) code set [17] for external communications. Such code sets can also be viewed as subtaxonomies in our taxonomic hierarchy. However, they can of course not be changed by the users run-time and hence the framework of context evolution is not applicable to them.

The second step is creating just one taxonomy that contains all others as subtaxons somewhere in it. There are several ways of doing so – one can create a large taxonomy just by adding the component taxonomies as descendants under a common root, but also by including some taxonomies into others, etc.

Additionally, it may be appropriate to create links between some taxons in different subtaxonomies, hence establishing some extra structure besides the basic one. For example, in our LIS case we may need to exhibit the fact that some piece of equipment is able to perform certain analyses on some specimen using certain chemical reactives. In such a way we may even define several parallel hierarchies on the same set of taxons, and our LIS uses such alternative taxonomies extensively e.g. to achieve greater flexibility for the process management framework (see Section V-B)

Given the current state of the research, no good general methodologies are proposed for the procedure of joining different taxonomies into one and this remains the subject for future research. In the case of LIS, human expert knowledge was used when creating and joining the taxonomies.

B. *Process management*

As stated already in the introduction, our model does not really have a notion of a business process. The actual effect of a process is achieved as a result of communication with the involved parties (human users, other ISs, lab devices, etc.). Such a communication act begins by obtaining a message from some actor and ends after processing the message and giving a reply. The incoming messages carry information concerning the surrounding environment (e.g. sensor readings) and outgoing messages endeavor some changes there (e.g. contain actuator control commands).

Processing of all the messages in the system is carried out independently. During the processing, the taxons referred to by the incoming message get a focus in the context space. Additionally, some new taxons or links may be created as well in order to keep the description of the surrounding environment (i.e. the knowledge base) adequate and consistent. The default target of message processing is to free the objects in focus from the encumbrance of the associated links.

For example, we may have a message "In the workplace wp there is a lab worker p holding an object with bar code bc in his/her hand". Assuming that all the referred taxons are known in the system, they will be focused forming the current context. Within this context, the most encumbered taxon is selected. In the given example, this turns out to be the lab worker's hand, since it can only have up to one link with the bar-coded objects. The system draws a conclusion that the object with bar code bc (say, a test tube) must be moved somewhere else, and by possibly extending the context using the available links, reply message of the form "the object with bar code bc should be moved to location c" is generated. If location c is, say, a stand, the lab worker may get this message e.g. by seeing the stand led flashing. The system gets the information about successful movement of the test tube from another message of the form "the object with bar code bc reached the machine g" or a message stating that the lab worker now has something else in his/her hand.

Our experience with the LIS shows that this kind of "local" process management enables one to describe arbitrarily complex work flows. At the same time, the hierarchical nature of our knowledge space allows us to tune the level of detailization in user communication very flexibly.

VI. Implementation

A. Core class model for context data

The core part of our context engine data model is shown in Figure 3.

There are two central classes.

- "Taxon" class contains all taxonomies we have in the system and they are joined into just one taxonomy. The primary tree structure is set up by the relation *belongs to* and there are only three original mandatory attributes: `code`, `id` and `description`. The `description` attribute is used as a default value for the textual attributes of the taxon if some attribute's name is not set in the language of current context, nor any of the higher level languages.
- All possible attribute values are collected into the second context engine core class "Attribute". This class has two mandatory relations to the Taxon class. Note that the *belongs to* link uses taxon's id, but the *links with* uses taxon's full-path id. This keeps attributes "on place" even if taxons are moved within the taxonomic hierarchy. The attributes *valid_since* and *valid_until* control the actuality of the attribute value at a particular time. The parameter *modifier* is optional and can modify the taxonomy pointed by *value* as described in Section IV.

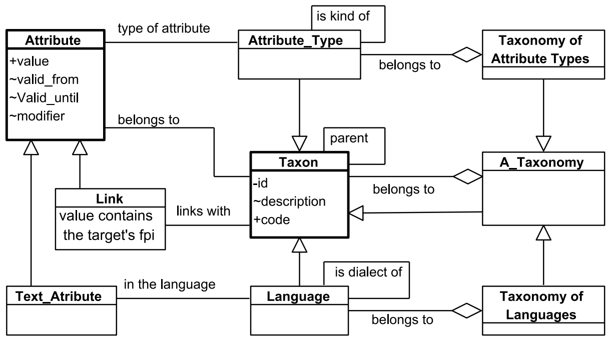

Fig. 3. Context engine core

B. Practical results

The described hierarchical taxonomy management mechanism was implemented for LIS of Tartu University Hospital and released in December 2006. The release took place in two 24×7 and one smaller business-hours laboratory. By the time of this writing (January 2008) three more laboratories have implemented the system and two launches are currently being prepared.

It was originally envisioned that the end users will manage the taxonomies themselves and they were provided with the appropriate software tools. However, taxonomic hierarchy as a concept for context management turned out to be rather a dramatic paradigm shift in practice, and the users in Hospital did not adopt it easily. Therefore, an initial taxonomy was created for system release time. After some acquaintance period (about 3 months), the users started to feel more comfortable with LIS and started developing the taxonomic hierarchy themselves.

Table I summarizes development of the taxonomic hierarchy during the first year of use. The first row indicates the number of taxons, the next two rows display the numbers of two 'simple' (i.e. textual and non-textual) types of taxon attributes and the last one the number of link attributes between the taxons in the whole hierarchy.

Category	Release-time	After 12 months	Increase %
Taxons	3377	6038	78.8
Non-text attributes	20010	28050	40.2
Text attributes	4928	7844	59.2
Links	29586	46866	58.4

TABLE I
Evolution of the taxonomic hierarchy

The hierarchic nature of the context space also enables evolutionary optimization of the underlying graph. For example, if all subtaxons of a higher level taxon are connected to some other taxon, the links from the lower level can be dropped in favor of one link from the higher level taxon. Via user interaction, it is also possible to unlink unused taxons and move them outside of the active hierarchy. As a result, the hierarchy becomes smaller and more efficient to process. For example, consider the numbers of containers and materials together with the links between these two types of taxons. Besides these numbers Table II also contains the product of the numbers of different containers and materials (i.e. the potential number of links), which has basically remained unchanged, but nevertheless the number of actual links has decreased more than two times.

Category	Release-time	After 12 months	Increase %
Containers	86	63	-26.7
Materials	231	317	37.2
Cont.×Mat.	19866	19971	0.5
Links	755	312	-58.7

TABLE II
SELF-OPTIMIZATION OF THE HIERARCHY

VII. CONCLUSIONS AND FUTURE WORK

This paper discussed a data model created for context space management in context-aware Laboratory Information System. The model has been implemented in the laboratory infrastructure of Tartu University Hospital (Estonia). The user feedback has been mixed, mainly due to long-standing habits and traditions in many laboratories. However, the users are gradually starting to accept the context management based on hierarchical taxonomies. It is the belief of the authors of the current paper that this approach will eventually turn out to be much more powerful and this power will be appreciated by the end users as well. Of course, improving usability of hierarchical taxonomies from user perspective is the key prerequisite for wider appreciation.

There are several theoretical issues that will need further studies as well. First, the data model needs to be formalized better in order to allow development and analysis of more complicated attribute value delegation mechanisms.

Second, even though our present research was motivated by the laboratory environment, the authors feel that the resulting hierarchical context model is applicable for a larger variety of settings. Still, it is necessary to conduct more case studies and to find out whether one can state good general rules for composing larger taxonomies out of smaller ones.

Third, the issue of deletion of certain attribute values together with the respective branches of the hierarchy remained outside of the scope of this paper. This feature was not needed for functionality of the system; however, deletion operation may be required for refactoring purposes and achieving better usability of the taxonomy. Deleting something from the taxonomic hierarchy must be done with extreme care – even if some branch is not needed in the local fragment of the hierarchy, there may be some links or context pointers to it from other taxons. Handling these kinds of situations remains the subject for future research as well.

REFERENCES

[1] John McCarthy. Notes on formalizing context. In *IJCAI*, pages 555–562, 1993.

[2] Vipul Kashyap and Amit Sheth. Semantic and schematic similarities between database objects: a context-based approach. *The VLDB Journal*, 5:276–304, 1996.

[3] M. Benerecetti, P. Bouquet, and C. Ghidini. Contextual reasoning distilled. *Journal of Experimental & Theoretical Artificial Intelligence*, 12:279–305, 2000.

[4] Soumen Chakrabarti, Byron Dom, Rakesh Agrawal, and Prabhakar Raghavan. Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies. *The VLDB Journal*, 7(3):163–178, 1998.

[5] Shui-Lung Chuang and Lee-Feng Chien. Taxonomy generation for text segments: A practical web-based approach. *ACM Trans. Inf. Syst.*, 23(4):363–396, 2005.

[6] Anthony Jameson. Modelling both the context and the user. *Personal Ubiquitous Comput.*, 5(1):29–33, 2001.

[7] Daniela Petrelli, Elena Not, Massimo Zancanaro, Carlo Strapparava, and Oliviero Stock. Modelling and adapting to context. *Personal Ubiquitous Comput.*, 5(1):20–24, 2001.

[8] Jakob E. Bardram. Applications of context-aware computing in hospital work: examples and design principles. In *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*, pages 1574–1579. ACM Press, 2004.

[9] Anders Kofod-Petersen and Jörg Cassens. Explanations and context in ambient intelligent systems. *Lecture Notes in Computer Science*, 4635:303–316, 2007.

[10] Bob Hardian. Middleware support for transparency and user control in context-aware systems. In *MDS '06: Proceedings of the 3rd international Middleware doctoral symposium*, page 4, New York, NY, USA, 2006. ACM Press.

[11] Patrick Brézillon. Contextualized explanations. In *Proceedings of International Conference on Expert Systems for Development*, 1994.

[12] Jean-Charles Pomerol and Patrick Brézillon. Operational knowledge representation for practical decision making. In *Proceedings of the 34th Hawaii International Conference on System Sciences*, 2001.

[13] P. Brézillon, L. Pasquier, and J.C. Pomerol. Reasoning with contextual graphs. *European Journal of Operational Research*, 136(2):290–298, 2002.

[14] P. Brézillon. Context Dynamic and Explanation in Contextual Graphs. *Modeling and Using Context: 4th International and Interdisciplinary Conference, CONTEXT 2003, Stanford, CA, USA, June 23-25, 2003: Proceedings*, 2003.

[15] Ghita Kouadri Mostefaoui, Jacques Pasquier-Rocha, and Patric Brézillon. Context-aware computing: A guide for the prevasive computing community. In *Proceedings of the IEEE/ACS International Conference on Prevasive Services*, 2004.

[16] Boicho Kokinov. Context-sensitivity of human memory: Episode connectivity and its influence on memory reconstruction. *Lecture Notes in Computer Science*, 4635:317–329, 2007.

[17] Marilyn Stark. Look at LOINC: The established standard for lab data gains visibility as data exchange increases. *Journal of AHIMA*, 77(7):52–55, July/August 2006.

[18] A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: a review. *ACM Computing Surveys (CSUR)*, 31(3):264–323, 1999.