A Graph-Based Approach to Context Matching

Andrei Olaru

Computer Science Department
University Politehnica of Bucharest
313 Splaiul Independentei
060042 Bucharest, Romania
Email: cs@andreiolaru.ro

Adina Magda Florea

Computer Science Department
University Politehnica of Bucharest
313 Splaiul Independentei
060042 Bucharest, Romania
Email: adina@cs.pub.ro

Abstract—**This paper presents the work in progress towards a simple, flexible and decentralized representation of context and for the detection of appropriate context-aware action. Continuing our previous work on decentralized multi-agent systems for the context-aware exchange of information, we propose a representation for context inspired from concept maps and conceptual graphs, and also a formalism for context patterns, that allows the detection and solution of problems related to the user's context.**

I. Introduction

Domains like Ubiquitous Computing [1] and Ambient Intelligence [2] have brought context-awareness as a central issue for research. As the electronic environment that surrounds people must assist them in more and more of their daily activities, ambient applications must consider more aspects of the user's context in order to improve their performance and their adequateness to the principles of UbiComp and AmI.

A true AmI system must be non-intrusive, but also proactive. This is a balance that may be difficult to reach. Appropriate proactive action must fit the context of the user or the user will not have an optimal experience with the system [3]. Additionally, in order to be useful, many times proactive action must result from the anticipation, based on the current context, of future situations. Context-aware action and anticipation will also make an AmI system seem "intelligent".

There is a large body of research dealing with the subject of context-awareness. Context has been defined as [4]: "Any information that can be used to characterize the situation of entities (i.e. whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves". Most works deal with context as location, location and time or other physical conditions, like temperature [5], [6], [7]. Another aspect of context that is considered in some works is activity [8], [9].

However, most times the situation of the user is defined only by means of aspects that have been predefined. Situations may be described by means of certain types of associations [8] or by means of ontologies and / or rules [6]. However, these methods are not very flexible and the range of contexts that can be treated becomes limited with respect to what Ambient Intelligence should ideally be.

In our approach towards an implementation of Ambient Intelligence, we are trying to build mechanisms and representations that facilitate a more flexible approach to AmI and context-awareness, while in the same time are easy to implement and can work on resource-constrained devices. In previous work a decentralized multi-agent system for the distribution of information was built, that shared relevant information between agents, based on some simple measures of context-awareness [10]. However, context-awareness requires a more complex and powerful representation of context, while less capable devices require that this representation be flexible in size and also easy to process.

This paper deals with describing a simple formalism that allows agents in a multi-agent system, that have only local knowledge, to share and process context-related information and to solve problems by using *context matching*. We consider that there is one agent assigned to each user of the system (we call the system AmIciTy and the agents AmIciTy agents). Each agent has a representation of the context of its user, including models on other users. The focus of this paper is more on defining a manner of representation, and less on the algorithms used for context matching, the agent's behaviour or the protocol used in the communication between agents.

Context matching is based on representing information about the context of a user as a conceptual graph [11] and on the existence of context patterns, or, in short, patterns. Patterns can describe (in more or less detail) situations that the user has been in, they can describe common sense knowledge, or just associations between different pieces of information that are observed to appear frequently in the user's history. They resemble the notion of pattern in knowledge discovery in databases. In fact, patterns may be extracted by using data mining techniques, but this is not the focus of this paper.

Two matching context graphs (for two different users) mean a shared context, which is an occasion for further sharing of information between the users' agents. A pattern that matches the user's context means the user has been in the situation before (or it is a well-known, commonsense, situation) and this can allow the agent to anticipate possible problems, as well as to find solutions to problems already detected.

The next section presents related work in the field of context-awareness. The proposed context representation, as well as a scenario and examples of context-aware behaviour form the content of Section III. The last two sections give an insight on future work and draw the conclusions.

II. RELATED WORK

In previous work in the field of context-awareness there are usually two points of focus: one is the architecture for capturing context information; the other is the modeling of context information and how to reason about it.

Ever since the first works on context-awareness for pervasive computing [12], certain infrastructures for the processing of context information have been proposed [13], [14], [15], [8], [5], [16]. There are several layers that are usually proposed, going from sensors to the application: sensors capture information from the environment, there is a layer for the preprocessing of that information, the layer for its storage and management, and the layer of the application that uses the context information [5]. This type of infrastructures is useful when the context information comes from the environment and refers to environmental conditions like location, temperature, light or weather. However, physical context is only one aspect of context [17]. Moreover, these infrastructures are usually centralized, using context servers that are queried to obtain relevant or useful context information [12], [15]. In our approach [18], we attempt to build an agent-based infrastructure that is decentralized, in which each agent has knowledge about the context of its user, and the main aspect of context-awareness is based on associations between different pieces of context information.

Modeling of context information uses representations that range from tuples to logical, case-based and ontological representations [6], [19]. These are used to determine the situation that the user is in. Henricksen et al use several types of associations as well as rule-based reasoning to take context-aware decisions [8], [20]. However, these approaches are not flexible throughout the evolution of the system – the ontologies and rules are hard to modify on the go and in a dynamical manner. While ontologies make an excellent tool of representing concepts, context is many times just a set of associations that changes incessantly, so it is very hard to dynamically maintain an ontology that describes the user's context by means of a concept. In this paper we propose a more simple, but flexible and easy-to-adapt dynamical representation of context information, based on concept maps and conceptual graphs. While our representations lacks the expressive power of ontologies in terms of restrictions, a graph-based representations is very flexible and extensible, so support for restriction may be added as future work.

Our approach to context representation is rooted in existing knowledge representation methods like semantic networks, concept maps [21] and conceptual graphs [11]. These structures can be used to describe situations (and context) in a more flexible manner and using less memory than ontological representations. While graph matching has been previously used, for instance for image processing [22], we attempt to use it for the matching of context graphs, also improving the graphs by means of special notation elements that allow the definition of patterns.

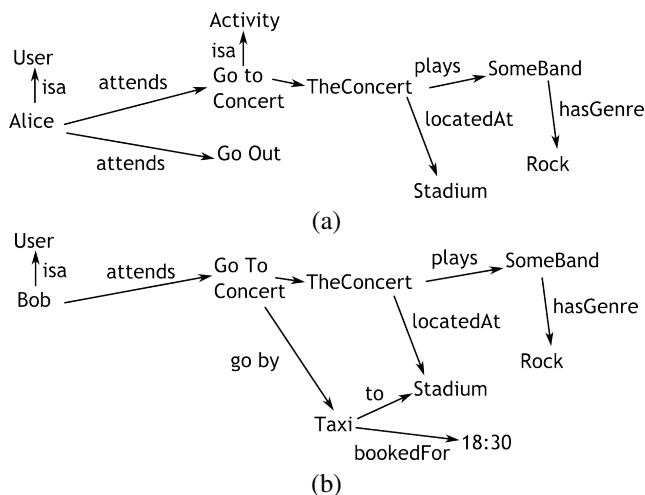There has been a significant body of work in the domain of

Fig. 1. The knowledge of Alice's and Bob's agents, respectively: (a) Alice will go to a concert where a rock band is playing and which is located at the stadium, and then to go out with friends; (b) Bob will be going at the same concert, and has also booked a taxi to get there. Part of the relations and concepts that appear in the graphs may come from ontologies.
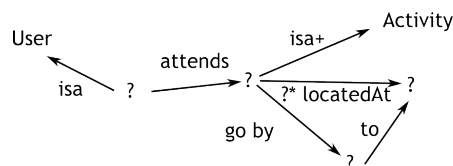

Fig. 2. A pattern that says that if the user attends an activity that has a location, then the location of the activity should be reached in some way.

ontology alignment, which is vital for a viable implementation of Ambient Intelligence systems [23]. However, this is not the subject of this paper. We assume that all agents in the system work with terms from the same ontology (where it is the case), or that ontologies have already been aligned.

Software agents and multi-agent systems have been used many times in the implementation of AmI environments in the past [15], [24], [25], however context-awareness in these systems is limited to location-related associations and simple forms of representation.

III. CONTEXT MATCHING

The main idea of this paper is to propose a graph representation for contexts, designed in order to facilitate the detection of compatible contexts.

Scenario. Alice will go to a rock concert in the evening of the current day. The concert is located at a stadium outside the city, therefore she should find some means of transportation to get there, but she hasn't yet given thought about that. Bob, her roommate, will go to the same concert but he has not talked to Alice about that yet. However, he has already booked a taxi to get to the concert. This is a typical situation for our approach [18]: insufficient communication between people leads to a lack of otherwise relevant information that could be easily obtained by means of an AmI system.

Alice and Bob are both users of the AmIciTy Ambient

Intelligence system. What we want is that the system (1) detects the need for a means of transportation for Alice, (2) based on information on Bob's agenda, suggest that a taxi may be an appropriate solution for Alice as well, and (3) based on the existing shared context, propose to Alice that she uses the same taxi that Bob has already booked.

Each user of AmIciTy has an associated agent. Figure 1 shows the concept graph for the knowledge of the two agents (agent A for Alice and agent B or Bob) that is relevant to the scenario. Formally, the knowledge of each agent can be represented as a graph:

$G = (V, E)$
$V = \{v_i\}, E = \{e_k\}, e_k = (v_i, v_j, value)$
where $v_i, v_j \in V, i, j = \overline{1, n}, k = \overline{1, m}$

The values of vertices and edges can be either strings or, better, URI identifiers that designate concepts, relations, people, etc. The value of an edge may be null.

The graph that an agent has contains the knowledge that the agent has about the user and about the user's context. The graph represents the context of the user, in the measure in which the agent has perceived it (or been informed of by the user or by another agent).

First, we want the system to detect the fact that it is necessary to know how Alice will be getting at the concert. This can be done by means of the following pattern: if the user intends to attend something that is an activity, and that has a location (it's not, for instance, making a phone call), then there should also be a means for the user to reach that location. The pattern is represented in Figure 2.

A pattern is also a graph, but there are several additional features that makes it match a wider range of situations. The graph for a pattern s is defined as:

$G_s^P = (V_s^P, E_s^P)$
$V_s^P = \{v_i\}, v_i = string \mid URI \mid ?, i = \overline{1, n}$
$E_s^P = \{e_k\}, e_k = (v_i, v_j, E\_RegExp), v_i, v_j \in V_s^P, k = \overline{1, m}$

where E_RegExp is a regular expression formed of strings or URIs.

A pattern represents a set of associations that has been observed to occur many times and that is likely to occur again. Patterns may come from past perceptions of the agent on the user's context or be extracted by means of data mining techniques from the user's history of contexts. Commonsense patterns may come from public databases, and patterns may also be exchanged between agents. However, the creation or extraction of patterns is not the subject of this paper.

The agent has a set of patterns that it matches against the current context (graph G). We will mark with the P superscript the graphs or vertex / edge sets that contain special pattern features (like ? nodes, for instance).

A pattern G_s^P *matches* a subgraph G' of G, with $G' = (V', E')$ and $G_s^P = (V_s^P, E_s^P)$, iff an injective function $f : V_s^P \rightarrow V'$ exists, so that
(1) $\forall v_i^P \in V_s^P, v_i^P = ?$ or $v_i^P = f(v_i^P)$ (*same value*)
and
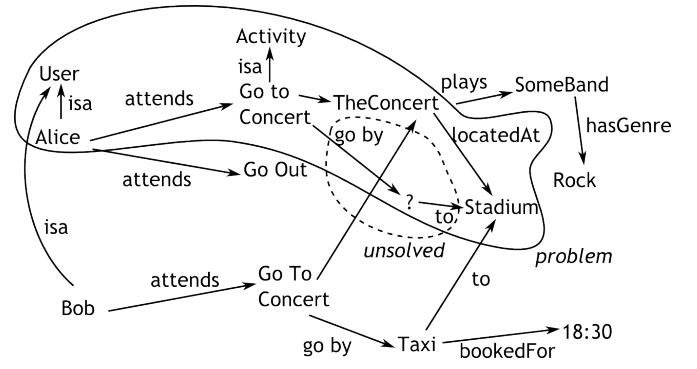(2) $\forall e^P \in E_s^P, e^P = (v_i^P, v_j^P, value)$ we have:

Fig. 3. The knowledge base of agent A, completed with the information on Bob's agenda. Also the problem and its unsolved part are circled with a continuous and a dashed line respectively. Although the unsolved part is displayed together with the rest of the context graph, it is not a concrete or known fact so it would not be used in pattern-matching.
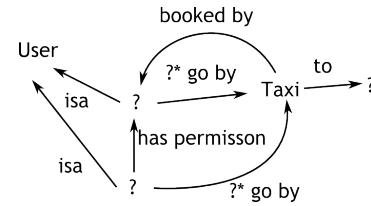

Fig. 4. A second pattern, specifying that two people can use the same taxi to get to the same location if the person who has not booked the taxi has permission from the other to ride the same taxi.

if $value$ is a string or an URI, then the edge $(f(v_i^P), f(v_j^P), value) \in E'$

if $value$ is a regular expression, then it matches the values $value_0, value_1, ..., value_p$ of a series of edges $e_0, e_1, ..., e_p \in E'$, where

$e_0 = (f(v_i^P), v_{a_0}, value_0),$
$e_k = (v_{a_{k-1}}, v_{a_k}, value_1), k = \overline{1, p-1}$
$e_p = (v_{a_{p-1}}, f(v_j^P), value_p),$
$v_{a_l} \in V'.$

In other words, every non-? vertex from the pattern must match a different vertex from G'; every non-RegExp edge from the pattern must match an edge from G'; and every regular expression edge from the pattern must match a series (that can be void, if the expression allows it) of edges from G'. Subgraph G' should be *minimal*. A graph (or subgraph) G' is minimal with respect to a matching pattern G_s^P iff there is no edge in G' that is not the match (or part of the match) of an edge in G_s^P.

A pattern G_s^P *k-matches* a subgraph G' of G, if condition (2) above is fulfilled for $m - k$ edges in E_s^P, $k \in [1, m-1]$, $m = \|E_s^P\|$ and G' remains connected and minimal. The relationship of *k-matching* should be interpreted as *matching except for k edges*. Non-matching vertexes imply non-matching edges. We consider the number of edges as relevant (as opposed to number of vertices, for instance), because context is a set of associations, so it is the edges that matter.

For the example in Figure 1 (a), the pattern in Figure 2 2-matches the knowledge about Alice that her agent has, G'

```
pattern G_s^P  k − matches G →
    if k > 0
        put problem in the list
    if there is a problem and this can be a solution
        is the solution is certain / complete
            let user know
            user confirms solution
                increase confidence in pattern
            otherwise
                decrease confidence in pattern
        otherwise
            link possible solution to problem
```

Fig. 5.   Pseudo-code of the agent's behaviour related to context matching.

containing the information that the user (Alice) will attend a concert (which is an activity) which is located at the stadium. The missing edges are the *go by* edge and the *to* edge, and there is also a vertex that is missing – the means of transportation. Because the pattern fits in a percentage of 66%, it means that Alice is in the situation described by the pattern, but something is missing, so the agent should ask Alice about that piece of information or to try to find it itself (see also Figure 5). This is defined as a *problem*.

A *Problem* is a graph G^P that contains features that are specific for patterns (like ? nodes for instance) and that is a partial instantiation of a pattern G_s^P, according to the current context. A problem G^P is the union between the subgraph G' (of the context graph G) that k-matches pattern G_s^P and the part of G_s^P that is not matched by G'. The latter is the *unsolved* part of the problem. A problem also remains associated with the pattern that generated it. Therefore, formally, if a pattern $G_s^P = (V_s^P, E_s^P)$ k-matches the subgraph $G' = (V', E')$ of G, we can define a problem p as a tuple (G_s^P, G_p^P), where G_p^P is the problem's graph:

$$G_p^P = G' \cup G_x^P$$
$$G_x^P = (V_x^P, E_x^P)$$
$$V_x^P = \{v \in V_s^P, v \notin dom(f)\}$$
$$E_x^P = \{e \in E_s^P \text{ for which condition (2) is not fulfilled}\}$$

Note that G_x^P (the unsolved part of the problem) is a subgraph of G_s^P. Also note that the unsolved part may contain edges whose vertices are not both in the unsolved part. The *problem* from our example is circled in Figure 3 with a continuous line, and its unsolved part is circled with a dashed line.

The agents in AmIciTy are not single agents. They are part of a multi-agent system. Agents A and B communicate frequently due to the fact that Alice and Bob live in the same place and exchange a lot of data. At some point in this communication, they exchange data about Alice's and Bob's agendas, which is normal for two people that share an apartment. Agent B will send the subgraph $agenda \rightarrow Concert$ and A will send $agenda \rightarrow Concert / \rightarrow Go\ Out$ (agent A will only send the $GoOut$ activity if Alice has not designated it as private).

Agent A receives the subgraph $agenda \rightarrow Concert$ and

matches it against Alice's context, detecting the compatibility (a full match). So it responds by building upon this common context: it sends a larger subgraph, containing the band playing at the concert, as well as the location of the concert. Agent B does the same operations as A (they share the same context regarding the concert, so each one's context matches the other one's), just that it also sends to A the associations $Concert - go\ by \rightarrow Taxi - to \rightarrow Stadium$.

The communication between agents as described above is done based on shared context. Starting from sharing their agendas, at each step agents detect matches between the two contexts and respond with a subgraph that is larger with one level (breadth-first).

All the data that agent A has about other agents (here, agent B) is stored in the agent's knowledge base as its model of the other users. The model for the other users is not necessarily separate though: if the same concept appears in both models (provided the concept ha the same URI, or the agent is able to detect by means of common sense knowledge that it is the same concept), both subgraphs will contain the corresponding node. Figure 3 shows the knowledge of agent A regarding users Alice and Bob. The model for Bob's agenda contains the same $Concert$ node that is contained in the graph for Alice. When matching patterns from its pattern set, A detects that the pattern mentioned above fully matches the model for Bob. Agent A also has a *problem* that is linked to this pattern. Since Bob's context fully matches the pattern, it means it may be a *solution* to Alice's problem: Alice may also use a taxi to reach the concert. But that would mean booking a different taxi (use a different instance of the concept).

Another pattern may be used in this context: agent A may know that two people may share the same taxi to get to the same destination, if one has permission from the person that booked the taxi (we have somewhat simplified the problem and we do not mention that the two people must leave from the same location and need to reach the destination at the same time). This pattern is shown in Figure 4. Matching this pattern against the knowledge of agent A about Alice's context in Figure 3 (remember that unsolved parts are not matched), a 2-match is obtained (missing relations are *has permission* and Alice's *go by*). Not only that, but adding those relation would solve the problem that Alice has. Therefore, the agent can suggest to Alice to ask permission from Bob to use the same taxi.

In this particular case, with the given knowledge and patterns, there is only one solution to the problem that arose. But in a more realistic case, where context is more complex and there are more patterns, more solutions to the same problem may be found. In case they fit equally well in the current context, then the agent must prompt the user with all of them and the user must be given the choice.

It can be argued that context-matching is a very difficult problem in the case of large graphs and complex situations. However, resource-constrained devices will work only with smaller pieces of context information (i.e. smaller graphs), that are relevant to their function. Second, algorithms inspired from

data-mining allow for incremental matching, starting from common nodes and growing the matching sub-graph (similar to the algorithm for matching Rule Schemas [26]).

Another problem that may appear in realistic situations (as opposed to our simple example) is the abundance of simultaneous matching context patterns, possibly describing contradictory situations. This is where more refined measures must be found that will allow calculating the relevance of each match. This too will be part of our future work.

IV. FUTURE WORK

The work presented in this paper is in progress. We are in the process of identifying an efficient matching algorithm, as well as deploying the described formalism into a previously implemented multi-agent system.

Besides detecting compatible contexts, there is a very interesting potential in detecting incompatible contexts, or contexts that the user should not be in. Also, uncertainty has yet to be included in our work. Both these problems have been researched in the domain of conceptual graphs and graph matching.

One last question that must be further researched is if a subgraph k-matching a pattern should really be connected. Moreover, should a pattern be necessarily connected, and how could unconnected patterns and matches be interpreted.

V. CONCLUSION

So far, work in context-awareness for pervasive environments has been based predominantly on location-awareness and physical conditions. The use of ontologies or rules does not bring much dynamical flexibility and they are not easy to modify automatically, at runtime.

This paper presents the work in progress towards the development of a more simple – suitable for resource-constrained devices – and more flexible manner of representing context and context patterns, that allows the agents to take decisions without the need for a centralized structure, by means of their knowledge, their history and local communication alone.

ACKNOWLEDGMENT

REFERENCES

[1] M. Weiser, "The computer for the 21st century," *Scientific American*, vol. 272, no. 3, pp. 78–89, 1995.

[2] K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten, and J. Burgelman, "Scenarios for ambient intelligence in 2010," Office for Official Publications of the European Communities, Tech. Rep., February 2001.

[3] G. Riva, F. Vatalaro, F. Davide, and M. Alcañiz, Eds., *Ambient Intelligence*. IOS Press Amsterdam, 2005.

[4] A. Dey, "Understanding and using context," *Personal and ubiquitous computing*, vol. 5, no. 1, pp. 4–7, 2001.

[5] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 2, no. 4, pp. 263–277, 2007.

[6] M. Perttunen, J. Riekki, and O. Lassila, "Context representation and reasoning in pervasive computing: a review," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 4, no. 4, pp. 1–28, October 2009.

[7] H. Hagras, V. Callaghan, M. Colley, G. Clarke, A. Pounds-Cornish, and H. Duman, "Creating an ambient-intelligence environment using embedded agents," *IEEE Intelligent Systems*, pp. 12–20, 2004.

[8] K. Henricksen and J. Indulska, "Developing context-aware pervasive computing applications: Models and approach," *Pervasive and Mobile Computing*, vol. 2, no. 1, pp. 37–64, 2006.

[9] M. Kaenampornpan and E. ONeill, "An integrated context model: Bringing activity to context," in *Proceedings of the Workshop on Advanced Context Modelling, Reasoning and Management*, 2004, pp. 7–10.

[10] A. Olaru, C. Gratie, and A. M. Florea, "Context-aware emergent behaviour in a MAS for information exchange," *Scalable Computing: Practice and Experience - Scientific International Journal for Parallel and Distributed Computing*, vol. 11, no. 1, pp. 33–42, March 2010, iSSN 1895-1767.

[11] J. Sowa, *Knowledge representation: logical, philosophical, and computational foundations*. MIT Press, 2000.

[12] A. Dey, G. Abowd, and D. Salber, "A context-based infrastructure for smart environments," *Proceedings of the 1st International Workshop on Managing Interactions in Smart Environments (MANSE'99)*, pp. 114–128, 1999.

[13] J. Hong and J. Landay, "An infrastructure approach to context-aware computing," *Human-Computer Interaction*, vol. 16, no. 2, pp. 287–303, 2001.

[14] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster, "The anatomy of a context-aware application," *Wireless Networks*, vol. 8, no. 2, pp. 187–197, 2002. [Online]. Available: http://www.springerlink.com/content/l42022u2k266658p/

[15] T. C. Lech and L. W. M. Wienhofen, "AmbieAgents: a scalable infrastructure for mobile and context-aware information services," *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), July 25-29, 2005, Utrecht, The Netherlands*, pp. 625–631, 2005.

[16] L. Feng, P. M. G. Apers, and W. Jonker, "Towards context-aware data management for ambient intelligence," in *Proceedings of DEXA 2004, 15th International Conference on Database and Expert Systems Applications, Zaragoza, Spain, August 30 - September 3*, ser. Lecture Notes in Computer Science, F. Galindo, M. Takizawa, and R. Traunmüller, Eds., vol. 3180. Springer, 2004, pp. 422–431.

[17] G. Chen and D. Kotz, "A survey of context-aware mobile computing research," Dartmouth College, Technical Report TR2000-381, November 2000.

[18] A. Olaru, A. E. F. Seghrouchni, and A. M. Florea, "Ambient intelligence: From scenario analysis towards a bottom-up design," in *Proceedings of IDC'2010, the 4th International Symposium on Intelligent Distributed Computing*, 2010, accepted for publication.

[19] T. Strang and C. Linnhoff-Popien, "A context modeling survey," *Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp*, pp. 1–8, 2004. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.2.2060

[20] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni, "A survey of context modelling and reasoning techniques," *Pervasive and Mobile Computing*, vol. 6, no. 2, pp. 161–180, 2010.

[21] J. D. Novak and A. J. Cañas, "The origins of the concept mapping tool and the continuing evolution of the tool," *Information Visualization*, vol. 5, no. 3, pp. 175–184, 2006.

[22] E. Bengoetxea, P. Larrañaga, I. Bloch, A. Perchant, and C. Boeres, "Inexact graph matching by means of estimation of distribution algorithms," *Pattern Recognition*, vol. 35, no. 12, pp. 2867–2880, 2002.

[23] J. Viterbo, L. Mazuel, Y. Charif, M. Endler, N. Sabouret, K. Breitman, A. El Fallah Seghrouchni, and J. Briot, "Ambient intelligence: Management of distributed and heterogeneous context knowledge," *CRC Studies in Informatics Series. Chapman & Hall*, pp. 1–44, 2008.

[24] G. Cabri, L. Ferrari, L. Leonardi, and F. Zambonelli, "The LAICA project: Supporting ambient intelligence via agents and ad-hoc middleware," *Proceedings of WETICE 2005, 14th IEEE International Workshops on Enabling Technologies, 13-15 June 2005, Linköping, Sweden*, pp. 39–46, 2005.

[25] C. Muldoon, G. M. P. O'Hare, R. W. Collier, and M. J. O'Grady, "Agent factory micro edition: A framework for ambient applications," in *Proceedings of ICCS 2006, 6th International Conference on Computational Science, Reading, UK, May 28-31*, ser. Lecture Notes in Computer Science, V. N. Alexandrov, G. D. van Albada, P. M. A. Sloot, and J. Dongarra, Eds., vol. 3993. Springer, 2006, pp. 727–734.

[26] A. Olaru, C. Marinica, and F. Guillet, "Local mining of association rules with rule schemas," in *Proceedings of CIDM 2009, the IEEE Symposium on Computational Intelligence and Data Mining, March 30 - April 2, Nashville, TN, USA*, ser. IEEE Symposium Series on Computational Intelligence, 2009, pp. 118–124.