

Distributed Context-Aware Systems

Massimo Benerecetti¹ Paolo Bouquet² Matteo Bonifacio³

¹University of Naples “Federico II”, Italy

²University of Trento, Italy

³Arthur Andersen Italia

Abstract

Currently, context-aware applications are defined as applications that react appropriately to information sensed in the environment, as opposed to applications that elaborate only information explicitly provided by users. Context is (implicitly or explicitly) thought of as a collection of features of the (physical or virtual) environment which can affect the behavior of an application. Though this notion of context is relatively unproblematic in systems with central control, it raises a number of challenging issues when applied to distributed systems, namely systems in which control is distributed over a group of heterogeneous, autonomous, interacting entities (typically, agents). Indeed, in distributed applications, we cannot assume that autonomous entities *share* a context, even though each of them uses contextual information for its operations. In this essay, we discuss in detail this claim and present a notion of context which seems to be adequate for distributed systems. For the sake of illustration, we outline how this notion of context can be used to design distributed context-aware systems.

1 Introduction

Currently, context-aware applications are defined as applications that appropriately react to information sensed from the environment, as opposed to applications that just elaborate information explicitly provided by users (Dey et al. 2001). Context is viewed as a collection of features of the physical – or virtual – environment which can affect the behavior of the application. Typical examples of contextual features are: location, time, speaker, hearer, other salient agents and objects. The main advantage of context-awareness is that it allows designers to create applications that can use information about contextual features to automatically adapt their behavior to a dynamic environment.

For centrally designed, stand-alone applications, the notion of context as a collection of features that can be automatically sensed from the environment is relatively unproblematic. Designers can choose whatever collection of features can be helpful for the application they have in mind, and decide how to use contextual information to implement complex functionalities. However, the situation is radically different when we consider scenarios in which a multiplicity of context-aware applications – each independently designed by different people – interact with each other to achieve goals which were not given in advance. In such a scenario, each application can have a different goal, and control can be distributed over a collection of autonomous, heterogeneous entities (software components, human users, etc.), henceforth called *agents*. This is what we call a *distributed context-aware system*, namely a system where a collection of autonomous context-aware applications (co-)operate with each other (or with human agents) to achieve their own objectives or, as a special case, joint goals.

Distribution brings forward many interesting issues. First of all, each autonomous application may in principle operate in a different physical environment, and thus cannot directly access the context in which other applications are operating. Second, if two (or more) applications operate in a different context, meaningful interaction between them in general may require the ability to communicate to each other information about their current context, and to establish relationships between their context and the context of the other applications. Third, even if the environment is the same, in general independently designed applications will not share the representation of the environment (e.g. because they were designed to perform different tasks or to achieve different objectives). Indeed, each application will have its own partial and perspectival representation of the environment, which reflects the particular perspective and objectives of their designers and/or users: (i) distinct applications may consider different contextual features as relevant in a particular circumstance (e.g. current time can be relevant for some applications and not others); (ii) different applications may assign different values to the same contextual feature (e.g. if they get time from different sources); (iii) different applications may use representations which are semantically heterogeneous (e.g. the same expression may have different meaning, or the same object can be referred to by different terms).

The main thesis of this essay is that, for distributed context-aware systems, an “objective” notion of context as a collection of environmental features (as proposed in (Dey et al. 2001)) is no longer sufficient. Context is better thought of as a “subjective” representation of an application’s environment, suitable to achieve the application’s objectives. In more philosophical terms, context has more to do with an application’s internal state than with a collection of “objective” features of the surrounding environment. The major consequence of this is that, to interact with other context-aware applications, each application needs the ability to use (and, for complex applications, to reason about) the relationship between its own context-dependent representation and other applications’ context-dependent representations. In short, in distributed systems, context-aware applications not only must be aware of their own context, but also need to take into account the fact that other applications operate in different contexts, and that this has many important consequences in their interaction.

This essay explores in some detail the problems involved in distributed context-aware systems. We discuss the problem of what notion of context is adequate to design distributed context-aware systems, and propose a model inspired to the framework of *Local Models Semantic* (Giunchiglia & Ghidini 1998) and *MultiContext Systems* (Giunchiglia & Serafini 1994); as an illustration, we describe the architecture of a multi-agent system for information retrieval in distributed environments.

2 Context in distributed applications

In many types of context-aware applications, context can be thought of as a collection of features of the surrounding environment. Thus, a context-aware navigation system will have a geographic representation of a region, but such a representation cannot be used if the application does not know the current location (a contextual feature of the environment); a calendar application has a representation of time and of a user’s agenda, but an alarm function cannot work if the application does not know the current time (another contextual feature of the environment). More complex applications require the ability to sense other features, such as the identity of relevant people, the position of salient objects, and so on; consider for example applications that must be able to assign the right content to indexical or

demonstrative expressions such as ‘I’, ‘here’, ‘now’, ‘this’, ‘that’, and so on¹.

The “objective” notion of context (i.e. as a collection of environmental features) seems to rely on three implicit assumptions:

- that there is an objectively definable environment in which the application runs;
- that there is a single list of relevant features that the application needs to sense from such an environment;
- that each feature can get a unique (unambiguous) value.

For non distributed applications, the three assumptions above can be regarded as a useful abstraction². However, they cannot be applied to distributed context-aware systems. Agents, which in general are designed and implemented by different people, may use different subjective representations of the “same” environment (for example may consider different collections of salient objects in it, or may need to represent different relations among them), may need different ways of assigning values to contextual features. In other words, different applications operate with different context-dependent representations of the environment, each suitable for their own objectives. As a consequence, the three assumptions above cannot be straightforwardly applied. Indeed:

- different agents can be given the ability to represent different features of the environment in which they interact with other agents. This makes quite problematic the idea that agents share an environment, since an agent’s environment is whatever aspects of the world the agent can represent (i.e. what the agent can know about the world). In the literature on context in artificial intelligence, this problem is well-known. A standard example of different representations of the “same” environment is John McCarthy’s *AboveThe-ory* (McCarthy 1993): the fact that an object A is on an object B can be represented using a binary or a ternary version of the predicates *on* (that is, $on(A, B)$ and $on(A, B, t)$), depending on whether an agents needs to explicitly take into account time (ternary predicate) or not (binary predicate). Much more complex examples can be found in the literature³;
- different agents may be designed to take into account different collections of contextual features. Research in knowledge representation shows that we are extremely unlikely to come up with a list of contextual features that are relevant for any possible application, or even for a single application which can operate in a rich environment (see for example (Guha 1991, McCarthy 1993, Bouquet & Giunchiglia 1995) w.r.t. to the so called qualification problem in AI). There is no list of objectively relevant contextual features, as relevance is strictly dependent on what an agent is designed for and to the problem it is trying to solve (so that even “natural” candidates, such as time, location, or agent, are not always relevant in context-aware applications);
- different agents can give a different interpretation to the “same” feature of the surrounding environment. Location, for example, can be expressed at different

¹The problem of the context dependence for indexical and demonstrative expressions is well-known in philosophy of language. See for example D. Kaplan’s well known logic of demonstratives (Kaplan 1978) and J. Perry (Perry 1997) for a more general philosophical discussion.

²But see conclusions for a short comment on this.

³For instance, R. Guha (Guha 1991) discusses the representation of an object’s price with the predicate $price(x, y)$ (the price of x is y), in which many parameters are left implicit (e.g. currency, date, net or gross price, VAT – if applicable, depending on the country –, ...). In section 3 we will discuss in detail the *Magic Box* example, a scenario which involves agents with different viewpoints.

levels of granularity (is location a geometrical point, a room, a building, a city, a region, a country?), and therefore can have a different range of possible values; or can be given using different coordinate systems, making it usable for some agents, and useless for others. Moreover, different agents can assign different values to the “same” feature. As we said, this may depend on a different semantics for the a parameter (e.g. location). However, there are also structural reasons, depending on the nature of the parameter. A standard example is the spatio-temporal location of an agent (e.g. the value of location will be different for different agents, the value of time will be different for agents in different time zones, and so on). Finally, we cannot disregard the fact that there can be errors, so different agents may end up with different values sensed for a parameter.

In a distributed system, context is better thought of as a “subjective” representation of the environment that an agent uses to solve a particular problem, and thus is much more directly related to its internal state than to “objective” features of the surrounding environment. This opens a number of interesting research issues. Not only an agent needs a representation of its environment and the ability to get values for some relevant features, but it must also possess the ability to interact with other agents that may use different representations of the “same” environment, may assign a different value to a contextual feature, may be aware of different contextual features. Since – by definition – the internal state of an autonomous entity is not directly accessible by other agents (and human agents are a typical example), the interaction among agents cannot be based on the assumption that they can access a shared context-dependent representation of the environment. Rather, the contents of an internal representation must be inferred using external indicators, such as observed behavior, actions and linguistic communication, together with (default) assumptions on what other agents can perceive of the environment. In other words, a context-aware agent must be capable of performing more or less complex forms of *contextual reasoning*, namely reasoning about the relationships between its own context-dependent representation and other agents’ context-dependent representations of the environment (see (Benerecetti et al. 2000) for a theoretical analysis of this notion of contextual reasoning). And this without having direct access to other agents’ internal state or to a “God’s eye” representation of an environment. This is what we call the problem of context in distributed systems. In the next section, we outline the intuition underlying a formal model of context and contextual reasoning that can be suitably applied to design context-aware applications in distributed environments.

3 A model of context and contextual reasoning

A satisfactory model of context for distributed context-aware systems should provide a model of the relationship between partial and perspectival representations of an environment that autonomous agents use to achieve their goals and to interact with other agents. An adequate formal model does provide a very good framework for studying and designing appropriate interactions between context-aware autonomous agents in a distributed application.

In what follows, we outline the ideas underlying such a formal model, largely inspired to *Local Models Semantics* (LMS) (Giunchiglia & Ghidini 1998) – and its proof-theoretical counterpart called *MultiContext Systems* (MCS) (Giunchiglia & Serafini 1994). A technical presentation of LMS is outside the scope of this essay. Therefore, we only present the main intuitions through a simple example, and refer the interested reader to the technical papers in the bibliography at the end of the essay.

The LMS model can be described as an attempt to capture the following intuitions:

- a context is a partial and approximate representation of the world (Giunchiglia 1993). It is partial, because it “covers” only a portion of the world; it is approximate, because a portion of the world is represented at a given level of granularity;
- each context is an autonomous representation of the world from an individual’s perspective. However, since in general different contexts represent bits and pieces of the same world, different contexts can be (sometimes systematically) related to each other.

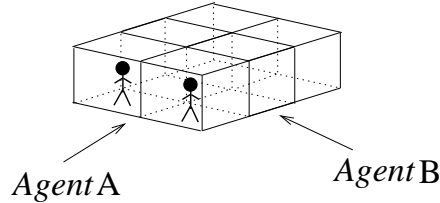


Figure 1: The conference room example

As a simple illustration, let us consider the following scenario, which is isomorphic to the so-called *Magic Box Example* (MBE) (Giunchiglia & Ghidini 1998, Benerecetti et al. 2000). Suppose two independently designed agents, A_1 and A_2 are monitoring a room for the presence of people. The two agents are connected to sensors located in different positions in the room, as depicted in Figure 1. The two agents can’t tell the depth of the room (for example, A_1 cannot tell whether the two people it sees are at depth one, two or three, or even if they are at the same depth in the room). Each agent has a partial representation of the room. A_1 represent the room as a two sectors (left and right) box, each containing a person in the example, whereas A_2 represents it as a three sectors (left, center and right) box, with a single person present in the leftmost one. Even though the two agents can’t see the entire room, the physical properties of the room impose a collection of compatibility constraints on what they can see at a given time (for example, if A_1 doesn’t see anybody, A_2 won’t see anybody either).

In the situation depicted in Figure 1, A_1 sees two people in the room. There are a number of actual configurations of the room which are ruled out once this information is provided. Namely, there cannot be less than two people in the room, even though there can be more. However, things are slightly different when we consider what situations are admissible from A_2 ’s perspective under the same circumstances. A_2 has a only a partial view of the room and from a different perspective. It may well be the case that, while A_1 sees two people, A_2 sees only one (this is exactly the situation of Figure 1). Similarly, it is possible that A_2 sees two people, while A_1 sees only one (e.g. if they stand in two different sectors, both of which lie in the same triple of sectors). In particular, A_1 ’s representation of the situation depicted in Figure 1 is “compatible” with all the situations, as perceived by A_2 , in which she sees either one or two people in (her view of) the room. Indeed, this view of A_1 is compatible with all the views of A_2 but one, namely the one in which A_2 sees no people. An analogous reasoning can be made for the view of what A_2 can see in the figure. Figure 2 shows the compatibilities among all the possible views the two agents may perceive from their perspective.

LMS (and MCS) have been devised to serve as a formal model of these kind of phenomena. They represent a very good framework for studying the possible

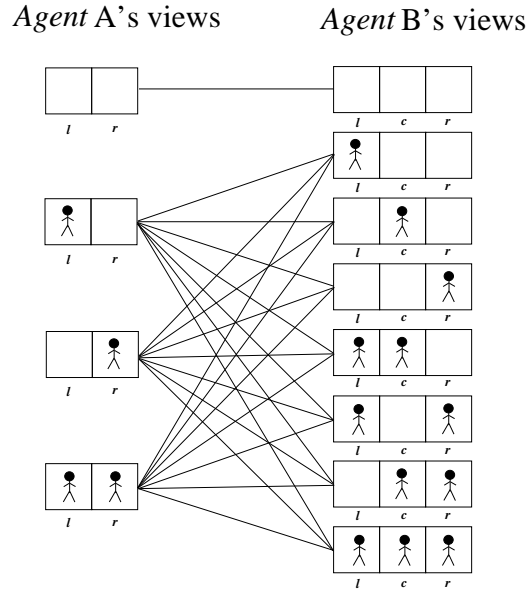


Figure 2: The compatibility between agent 1 and agent 2 views

interactions between context-aware autonomous applications in a distributed environment. Indeed, with respect to the three implicit assumptions of traditional approaches, we can say that:

- there is not such a thing as a shared representation of the environment: each context is an autonomous representation where different aspects of the world are taken into account (for example, A_1 does not have the power of expressing the concept of “center” in its local language);
- as a consequence, there is no need of assuming that there is a list of “objectively” relevant features: we can easily imagine a situation in which A_1 uses a richer language in which current time is taken into account, without being forced to change anything in the way A_2 sees the room (of course, we might want to work a bit on the compatibility relation, but see below for this issue);
- the semantics of what is represented by the two agents (including values sensed from the environment) is local: what l means to A_1 is not a priori related to what l (left) means to A_2 . The relation between the symbol l in the two representation language is given via compatibility constraints. Nothing prevents us from changing A_1 's language and use u (up) for left (l). This would not affect in any way the semantics of *left* in A_2 's language.

However, it's worth remarking that LMS was thought of as a general model of contextual reasoning, independently of any application. So, for example, the compatibility relations between the situations that the two agents are able to represent is given a priori. This would allow us to model a simple version of the scenario above, in which we imagine that the properties of the room the two agents are observing are known to both of them, and that the problem is simply to determine the position and the number of the people in the room by communicating with each other⁴, reasoning about the compatibility relations mentioned above. However, this is not realistic in distributed systems, where agents may not know what

⁴Incidentally, the problem is not as simple as one might think, since the information jointly available to A_1 and A_2 can be insufficient to determine the configuration inside the room.

the relationship is between their representation and other agents' representations of the world. In many situations, agents need the ability to learn such a relationship through the interaction (typically, communication) with other agents. In a more complex scenario, we can imagine that (part of) the problem that A_1 and A_2 are facing is discovering what sort of object they are observing; this means that they do not know what the relation is between their perspectives, and that such a relation can only be learned by exchanging messages (e.g. every time the configuration changes, they tell each other what they see). Moreover, A_1 and A_2 need not have a common goal; even worse, they might have conflicting objectives, and be unaware of such a situation.

To illustrate this crucial point, in the final section we describe an application in which various collections of documents are available to an information seeker, each provided by a different information provider and each associated with an independently defined conceptual structure (e.g. a taxonomy, a RDF description, an ontology). We interpret such independent structures as descriptions of different local contexts. In our application, the seekers' and the providers' agents cooperate to perform context-based retrieval on these collections of documents without knowing *a priori* the relationship between the different local contexts.

4 Towards context-aware multi-agent systems

Multi-agent systems (MAS) are a very promising technology for distributed systems (see (Weiss 1999) for a reference book on MAS). In a very broad sense, agents are autonomous, computational entities that can be viewed as perceiving their environment through sensors and acting upon their environment through effectors. For example, in (Wooldridge & Jennings 1995), an agent is defined as “a computer system that is *situated* in some *environment*, and that is capable of *autonomous action* in this environment in order to meet its design objectives”. The goal of this section is to illustrate the issues raised in the previous sections about distribution by introducing a simple traveling scenario in which a collection of context-aware agents support two tourists in their visit to Venice.

Suppose that Laura and Tom are visiting Venice using their new Personal Traveling Assistant (PTA), a personal agent running on their palmtop whose goal is to support them while traveling across the world. Suppose also that the local administration in Venice has set up a Venice Information System (VIS), a system which supports provides access to information about transportation, history and art in Venice, food and accommodation, entertainment). In our scenario, the VIS has not been designed as a single portal, but as a “federation” of pre-existing information providers (e.g. single hotels and restaurants, museums, transportation companies, cinemas). The idea of the federation is that each provider takes care of his own piece of information (collects documents, use his own technology to store, categorize, retrieve, present information, make independent contracts for advertisement on his site, and so on), without the necessity of creating a single access to information (e.g. a web portal). Technically, the VIS is designed as a collection of information sources (the providers), each with its own agent (the provider's agent) that provides access to locally stored data (“wraps” its information source) and guarantees communication with other provider's agents and with each visitor's PTA. For the sake of this essay, we assume that agents exchange messages in a shared syntactic format (say XML documents) and use a shared communication protocols, such as the Knowledge Query and Manipulation Language (KQML) or the FIPA Agent Communication Language (ACL).

Let us consider two simple situations in which the problems of distributed context-aware systems may emerge in such an application. The first has to do

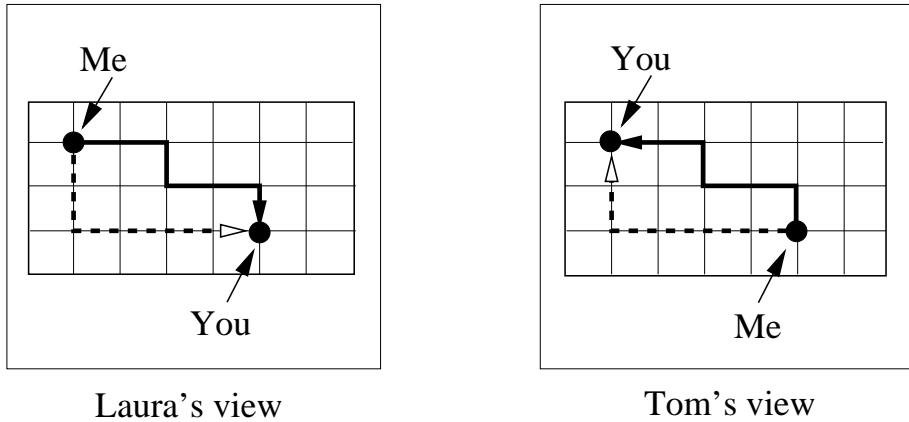


Figure 3: The traveling scenario

with physical context, the second with cultural context.

4.1 Physical contexts

Suppose Laura and Tom split up to visit different parts of the city and at noon want to meet for lunch. We assume that their PTAs can communicate with each other, and that each of them can get information about the current location using techniques and tools similar to those described in (Dey et al. 2001). Laura and Tom are in different areas of the city, so their PTAs get different values for the contextual feature corresponding to location. In other words, the PTAs do not share the respective current context. How can they support Laura and Tom in finding a meeting point?

A first idea is that Laura’s PTA might try to get the same contextual information as Tom’s PTA (e.g. by using the same widgets that Tom’s PTA is using, in the sense of (Dey et al. 2001)). However, this wouldn’t help that much, as the problem is not accessing the same contextual information, but assessing the relationship between the two contexts. Using the terminology of section 3, this means that the two agents need to reason about the relationship between their viewpoints. In the case of physical contexts, we can imagine many ways in which this can be done. For example, they might try to get a map in which both locations are indicated (e.g. by asking the VIS to do this for them). Notice, however, that this is not the same as saying that now they share a representation of the (enlarged) context. Indeed, to really use the information on the map, the two agents need to read the map from their perspective, namely must be able to tell who is who (this problem is discussed from a philosophical point of view by J. Perry in his paper on essential indexicals (Perry 1979). This is represented in Figure 3: on the left (Laura’s view), Laura’s location is represented as Me, and Tom’s position as You; analogously for Tom on the right. To show that this is not just a detail, consider now the problem of finding a path on the map such that Laura and Paolo will eventually meet. The two PTA’s may each suggest a path from one location to the other, but the two paths must be compatible (e.g. the same path); so, for example, the dashed and the solid lines in the figure represent two paths that are not compatible (Laura and Tom would not meet). To be precise, we should not have said the same path, but two paths that describe the same path on the map from the two locations. And this is the only way the two PTAs can reason about the problem. Indeed, the collection of instructions that the two PTAs give to Laura and Tom will be different, as the assume a different starting point. So again the problem is to check whether the two

sets of directions describe two paths that will achieve the goal.

A final remark is the following. One might object that the system can be designed in such a way that it's the VIS that gives directions to the two PTAs; this would eliminate the necessity for the two PTAs to reason about the relationship between the two contexts. But this is just as saying that the VIS is a “mediator” that can build an external representation of the two contexts. The VIS would use this representation to reason about the relationship between the two contexts, and between the two contexts and its own point of view. This is the only way we have to explain how the VIS would be able to give appropriate directions to Laura and Tom, that is directions from their respective starting point.

4.2 Cultural contexts

In the model we proposed, context is not only a collection of features of the physical environment, but a “subjective” view of the world used by an agent to achieve a goal. This may include physical features, but also a user profile, a list of preferences, a domain description, a collection of background beliefs, a record of past conversations, and so on. All these aspects are included in what we call a cultural context. Our goal here is to show that cultural context plays an important role even in applications that are embedded in a physical environment.

Suppose, for example, that Laura and Tom want to query the VIS to find a restaurant near to the point where they met. Even if we disregard the subtleties involved in the notion of “nearness”, still we can imagine that Laura is vegetarian, and that Tom likes good wine. This is an essential part of their context, and their PTAs will use this information to filter out any suggestion from the VIS that do not comply to these requirements. Or suppose that Laura and Tom, who are American tourists, want to know the cost of food and accommodation in US dollars (and not the local currency, e.g. in Italian Lire).

Another interesting situation is the following. Laura wants to retrieve from the VIS documents about S. Marco. If the VIS had a traditional search engine that creates a keyword-based index of all documents available on the federated information sources, Laura could query the system as we do today with search engines on the web. However, this would suffer from all the limitations that we all daily experience with current search engines. For example, if Laura meant S. Marco the Cathedral, then all documents about Piazza S. Marco, or about pub named after the saint, would be useless for her. The fact that she meant the cathedral, not the Piazza or the pub, is part of what we called her cultural context. How can we enable Laura to make a search which is sensitive to her cultural context?

In a centrally designed system, one might think of organizing all available documents using a shared concept-based classification, and to allow Laura to navigate across such a classification. However, this approach has many practical and theoretical drawbacks⁵: first of all, it does not scale up to systems with a very large number of (autonomous, independent) information sources (such as the web); second, it presupposes that information providers can find an agreement on what classification is “the right” classification; finally, it does not exploit the added value of having many autonomous viewpoints on a collection of documents (one can learn a lot from the way other people classify documents!!).

In a distributed system, we may try to exploit the notion of compatibility between contexts (see (Bonifacio et al. 2000) for a more technical description of the high level architecture of such a system). On the one hand, suppose Laura had a

⁵experimental work and practical experience show that a category system which is perfectly suitable for some users can be almost unintelligible – or irritating – for others, since people tend to have different perspectives on the world. See (Bonifacio et al. 2000) for a discussion on these issues in relation with the development of the *Knowledge Space* in Arthur Andersen.

simple way of describing her viewpoint to her PTA (for example, using a simple ontology of historical buildings in Venice as the domain of her search); on the other hand, suppose that each information provider had one (or more) categorization(s) of available documents, and that the provider’s agent could use these categorizations as the provider’s context (the provider’s point of view on documents). Then Laura’s PTA and each provider’s agent might start a simple “meaning negotiation” protocol to decide whether the context provided by Laura and the context provided by a categorization of documents are compatible. For example, when Laura makes a query on S. Marco in the context of historical buildings in Venice, the contextual information contained in the ontology of historical buildings can be compared with the classification of documents of each information provider in the VIS, and all documents associated with classifications that do not match (at least partially) with Laura’s context can be rejected (or get a very low rate).

5 Conclusions

The notion of context needed in distributed context-aware systems is much more general than the notion of context currently used in centrally designed, stand-alone applications. *Context is not simply a collection of features of the surrounding environment, but a partial and approximate representation used by an agent to interact with the environment and with other agents.* Obviously, part of this representation may have to do with features of the surrounding environment, but what features – and what use is to be made of them – depends on factors such as the agent’s goals and intentions, available sensors, reasoning capabilities, and so on.

If context is an agent’s viewpoint on the environment, it cannot be shared with other agents. This is not to say that there can be no relationship between different autonomous viewpoints; indeed, the possibility of this relationship is based on the intuitive fact that contexts are viewpoints on a common world, and therefore there can be some compatibility constraints between autonomous representations of it. However, it is essential that these constraints are not given *ex ante*, and can only be learned (inferred, conjectured, guessed, . . .) *ex post* by observing the behavior of other agents and by interacting with them.

What we propose can be described as a shift from an “objective” to a “cognitive” approach to context-aware systems (see (Giunchiglia & Bouquet 1997) for a conceptual and formal analysis of this shift). Interestingly enough, this shift is supported by important theories in cognitive science, linguistics, and philosophy, where the notion of context as a subjective viewpoint has been recognized and studied from different perspectives (Fauconnier 1985, Dinsmore 1991, Sperber & Wilson 1986, Kokinov 1995). This cognitive approach is very clearly described in the following quote from Sperber and Wilson’s well-known book on relevance (Sperber & Wilson 1986): “[t]he set of premises used in interpreting an utterance [...] constitutes what is generally known as the context. A context is a psychological construct, a subset of the hearer’s assumptions about the world. It is these assumptions, of course, rather than the actual state of the world, that affect the interpretation of an utterance. A context in this sense is not limited to the information about the immediate physical environment or the immediately preceding utterances: expectations about the future, scientific hypotheses or religious beliefs, anecdotal memories, general cultural assumptions, beliefs about the mental state of the speaker, may all play a rôle in interpretation”. This paradigmatic shift which happened in theories of context needs to be done also in applications. We believe that it will be a key step toward the design and the implementation of distributed context-aware systems.

References

- Benerecetti, M., Bouquet, P. & Ghidini, C. (2000), ‘Contextual Reasoning Distilled’, *Journal of Theoretical and Experimental Artificial Intelligence* **12**(3), 279–305.
- Bonifacio, M., Bouquet, P. & Manzardo, A. (2000), A distributed intelligence paradigm for knowledge management, in ‘AAAI Spring Symposium Series 2000 on Bringing Knowledge to Business Processes’, AAAI. Submitted.
- Bouquet, P. & Giunchiglia, F. (1995), ‘Reasoning about Theory Adequacy: A New Solution to the Qualification Problem’, *Fundamenta Informaticae* **23**(2–4), 247–262. Also IRST-Technical Report 9406-13, IRST, Trento, Italy.
- Dey, A., Salber, D. & Abowd, G. (2001), ‘A Conceptual Framework and Toolkit for Supporting the Rapid Prototyping of Context-aware Applications’, *Human-Computer Interaction* **16**([this special issue]), xxx–xxx.
- Dinsmore, J. (1991), *Partitioned Representations*, Kluwer Academic Publishers.
- Fauconnier, G. (1985), *Mental Spaces: aspects of meaning construction in natural language*, MIT Press.
- Giunchiglia, F. (1993), ‘Contextual reasoning’, *Epistemologia, special issue on I Linguaggi e le Macchine XVI*, 345–364. Short version in Proceedings IJCAI’93 Workshop on Using Knowledge in its Context, Chambery, France, 1993, pp. 39–49. Also IRST-Technical Report 9211-20, IRST, Trento, Italy.
- Giunchiglia, F. & Bouquet, P. (1997), Introduction to contextual reasoning. An Artificial Intelligence perspective, in B. Kokinov, ed., ‘Perspectives on Cognitive Science’, Vol. 3, NBU Press, Sofia, pp. 138–159. Lecture Notes of a course on “Contextual Reasoning” of the European Summer School on Cognitive Science, Sofia, 1996.
- Giunchiglia, F. & Ghidini, C. (1998), Local Models Semantics, or Contextual Reasoning = Locality + Compatibility, in ‘Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR’98)’, Morgan Kaufmann, Trento, pp. 282–289. Short version presented at the AAAI Fall 1997 symposium on context in KR and NL. Also IRST-Technical Report 9701-07, IRST, Trento, Italy.
- Giunchiglia, F. & Serafini, L. (1994), ‘Multilanguage hierarchical logics (or: how we can do without modal logics)’, *Artificial Intelligence* **65**, 29–70. Also IRST-Technical Report 9110-07, IRST, Trento, Italy.
- Guha, R. (1991), Contexts: a Formalization and some Applications, Technical Report ACT-CYC-423-91, MCC, Austin, Texas.
- Kaplan, D. (1978), ‘On the Logic of Demonstratives’, *Journal of Philosophical Logic* **8**, 81–98.
- Kokinov, B. (1995), A Dynamic Approach to Context Modelling, in P. Brezillon & S. Abu-Hakima, eds, ‘Working Notes of the IJCAI-95 Workshop on “Modelling Context in Knowledge Representation and Reasoning”’, Montreal (Canada).
- McCarthy, J. (1993), Notes on Formalizing Context, in ‘Proc. of the 13th International Joint Conference on Artificial Intelligence’, Chambery, France, pp. 555–560.
- Perry, J. (1979), ‘The Problem of the Essential Indexical’, *Nous* **13**, 3–21.

- Perry, J. (1997), Indexicals and Demonstratives, *in* R. Hale & C. Wright, eds, 'Companion to the Philosophy of Language', Blackwell, Oxford.
- Sperber, D. & Wilson, D. (1986), *Relevance: Communication and Cognition*, Basil Blackwell.
- Weiss, G., ed. (1999), *Multiagent Systems*, The MIT Press.
- Wooldridge, M. & Jennings, N. (1995), 'Intelligent agents: Theory and practice', *The Knowledge Engineering Review* **10**(2), 115–152.