Context-Aware Computing: A Guide for the Pervasive Computing Community

Ghita Kouadri Mostéfaoui
University of Fribourg
Software Engineering Group
Rue Faucigny 2, CH-1700 Fribourg, Switzerland
Ghita.KouadriMostefaoui@unifr.ch

Jacques Pasquier-Rocha
University of Fribourg
Software Engineering Group
Rue Faucigny 2, CH-1700 Fribourg, Switzerland
Jacques.Pasquier@unifr.ch

Patrick Brézillon
University of Paris VI
LIP6
8 rue du Capitaine Scott 75015 Paris, France
Patrick.Brezillon@lip6.fr

Abstract

There is a high interest on context-aware computing expressed by the pervasive computing community, which considers context as a key to design more adaptive applications. Due to this interest, a huge amount of contributions have been made in the field as we can notice from the increasing amount of publications on context and context-aware computing. However, the user interested in integrating context in its specific application will rapidly find himself in a jungle of various tools and techniques. This paper aims at structuring this information by providing a survey on the main context-aware categorization, acquisition and modeling techniques. "Main" refers to the most promising techniques in terms of their genericity in a way that they can be used in various application domains. This work tends to provide a guide for the pervasive computing community in order to select the best technique to use in a specific application or -more interesting- to justify the development of a new one.

1. Introduction

Heterogeneity is an intrinsic characteristic of pervasive environments. The plug and play facility provided for such environments allows clients from different locations to access remote services by the mean of various devices. These devices have different functionalities and do not necessary implement all the standards used by the requested service such as communication protocols, discovery mechanisms and so one. Pervasive environments are then designed in a way that facilitates adaptability and run-time reconfiguration. Adaptability is guided by a set of information about the current state of the environment, or formally "context". Due to the great interest of the pervasive community in context and context-aware computing, many techniques and works have been proposed. Some of them deal with the integration of context in existing pervasive systems; others deal with context acquisition, etc. But most of these contributions cope with a specific application and no generic tools are actually achieved. As a result, a user wishing to add context-aware functionality to its pervasive application will face a huge amount of alternatives and techniques which makes selecting the most useful one a cumbersome task. This paper is thus intended to surveys the context-aware computing field including the main acquisition and modeling techniques. The aim of this study is to provide the pervasive computing community with a detailed view of the current state of the art.

The paper begins by a set of definitions of context (Section 2). Section 3 highlights the importance of context for pervasive systems with a definition of context-aware computing. Section 4 discusses context categorization. Context acquisition is presented in Section 5. Context modeling techniques are detailed in Section 6. Section 7 concludes this paper with a summary of elements drawn from the context-aware computing area.

2. What is Context?

Context has an intuitive connotation in human reasoning. It generally refers to what surrounds the center of in-

0-7803-8577-2/04/$20.00 ©2004 IEEE

terest, provides additional sources of information "where, who, what" and increases understanding. Work on context has a long history in literature, philosophy, artificial intelligence, and linguistics. But, due to its intuitive connotation, the term "context" remains a general word with a vague meaning. Because it is heavily used in different domains, context enjoys multiple definitions. In the following, we try to capture the meaning of context by presenting the main ones.

2.1. Definitions of Context

The first step in studying context-aware computing is to define its main component; "context". Since the earlier, until the most recent conferences and workshops on context [11], [9], [8] have focused on this question, without yet a much agreement. There are a great many different conceptions and uses of the term "context". With the aim of working out a concrete definition, we begin by presenting the general dictionary definition and continue with a selection of definitions proposed in the engineering field in a chronological order.

According to the Merriam-Webster dictionary [4], the term 'context' usually has two primary meanings.

1. the parts of a discourse that surround a word or passage and can throw light on its meaning.
2. the interrelated conditions in which something exists or occurs.

The first meaning is closely related to linguistics and is the most used definition, whereas, the second meaning is more generic. Other synonyms of the word exist as: circumstances, situation, conditions, position, posture, attitude, surroundings, and environment [7].

From an engineering perspective, many definitions have been proposed and are included between the two sides' definitions above.

The term context-aware computing was first introduced by Shilit and Theimer in [34], where they refer to context as:

Definition 1: the location of use, the collection of nearby people and objects, as well as the changes to those objects over time.

A similar definition is given by Brown et al in [13]:

Definition 2: We define context to be any information that can be used to characterize the situation on an entity, where an entity can be a person, place, or physical computational object.

Brézillon and Pomerol [12] define context as:

Definition 3: all the knowledge that constrains a problem solving at a given step without intervening in it explicitly.

Dey in [19] proposes a more generic definition that states:

Definition 4: Context is any information that can be used to characterize the situation of an entity. An entity is a person, or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves.

Dey's definition includes both explicit input (as information explicitly provided by a user via a user interface) and implicit input that needs a computation process before effective use (as information about the state of a local network by drawing up the number of connected users).

Despite some similarities between the proposed definitions, one can imagine how difficult it is to find a common ground. These definitions are far away from a mathematical precision, and strongly rely on the authors' research field and focus.

3 The Importance of Context

Context has been considered in different fields of computer science, including natural language processing, machine learning, computer vision, decision support, information retrieval, pervasive computing and more recently computer security. By analogy to human reasoning, the goal behind considering context is to add adaptability and effective decision making. In the following we highlight the main works that have been considering context in the domains of pervasive computing in general and security in these types of environments in particular.

3.1. Context in pervasive computing

Due to its nature, pervasive computing is closely related to context. The principal reason of this correlation comes from the high heterogeneity and ubiquity of communicating entities in these kinds of environments. These two aspects require run-time adaptation of provided services and of users' devices depending on their location, role and task at hand, where, adaptation is mainly dependent on the situation of use or "context". As a consequence; it is very

rare to actually find a conference or a journal call for paper without context and context-aware computing as a topic of interest.

Based on the work of Dey and Abowd [21] and Schilit et al. [35], Chalmers identifies five uses of contextual information in pervasive environments [14].

1. Contextual sensing - where the context is sensed and information describing the current context, e.g. location, temperature, can be presented to the user.
2. To associate context with data, known as contextual augmentation, e.g. records of objects surveyed can be associated with location, meeting notes can be associated with people in the meeting and the place the meeting was held.
3. To enable contextual resource discovery, e.g., to cause printing to be on the nearest printer.
4. Context triggered actions to trigger actions such as loading map data for an area to be entered, or exchange business cards [37].
5. Contextual mediation - using context to modify a service. For instance to describe limits and preferences over a large range of offered data, in order to display the most appropriate parts. The request for the data being mediated need not arise from the context.

3.2. Context in security for pervasive environments

Considering context in security is a recent research direction. Most of the efforts are directed towards securing context-aware applications. Covington's team explores new access control models and security policies to secure both information and resources in an intelligent home-environment [15], [16], [17]. Their framework makes use of environment roles. In the same direction, Masone [27] designed and implemented RDL (Role-Definition Language), a simple programming language to describe roles in terms of context information. There have also been similar initiatives in [39] and [29]. It is interesting to observe that all previous work on combining security and context-aware computing follow the same pattern: using contextual information to enrich the access control model in order to secure context-aware applications with a focus on specific applications. However, the unique contextual elements considered in this approach are collected through sensors (e.g. location by GPS), but other contextual elements as the user's preferences are not really considered. The second main observation is that security decisions follow an old-fashioned rule-based formalism which does not consider systems and networks dynamics.

This overview shows the broad use of context in different domains. The heterogeneity of these domains generates different approaches for context categorization, acquisition and modeling. These approaches are discussed in the following sections.

3.3. Context-aware computing

According to [20], it is commonly agreed that the first research investigation of context-aware computing was the Olivetti Active Badge system. The Active Badge was conceived, designed and prototyped between 1989 and 1992. It is intended to provide direct location information of staff members in a building for automatic telephone call routing. For this purpose each staff member wears a badge that transmits periodical signals to a centralized location system providing information about his location. The largest Active Badge system is actually at Cambridge University Computer Laboratory , with more than 200 badges and 300 sensors used daily.

But the term context-aware computing was first introduced by Shilit et al in 1994 [34] as a software that "adapts according to its location of use, the collection of nearby people and objects, as well as the changes to those objects over time". Another definition given by Dey [20] states that "A system is context-aware if it uses context to provide relevant information and/or services to the user, relevancy depends on the user's task".

Now context awareness is a well established community with conferences as ubiquitous computing, pervasive computing, etc.

4 Context Categorization

Contextual information can be classified differently according to two aspects. The first aspect refers to the application domain while the second depends on the point of view. Schilit et al. in their anchor paper on context-aware computing applications [35] divides context into three categories: user context (user's profile, user's location, social situation, etc), computing context (network connectivity, communication costs, communication bandwidth) and physical context (lighting, noise levels).

In the same perspective, but mainly focused on mobile applications, [36] suggests four "primary" types of context. Namely, location, identity, time and activity. He claims that many other types of contextual information may be deduced from this primary set. For example, the user's identity may help determining user's preferences, her e-mail address, and her relationships with other people in the environment. Location can also be used to deduce what other people and objects are in the surroundings and what activity is taking place nearby.

From a user's point of view, Gwizdka [22] makes a distinction between two types of context; internal and external. Internal context describes the state of the user and may include the work context (e.g. current projects and their status, status of to-dos, project team), personal events (i.e. events experienced by the user), communication context (i.e. state of interpersonal email communication), and emotional state of the user. External context describes the state of the surrounding environment. It may include location, proximity to other objects (both people and devices), and temporal context.

Deriving from their experience in designing and implementing location-based systems, and from a human-computer interaction view, Petrelli et al. [31] distinguish between two types of context; material and social. Material context refers to location (in an office, at home, in a museum, etc), the device (desktop computer, handled device, etc) or available infrastructure (networks, infrared, GPS). Social context may include the user's traits, interests, preferences, if he is alone, who the others are, etc.

Based on the previous review, it is clear that there is no generic categorization of context. Relevant information differs from a domain to another and depends on the effective use of this information. A concise set of contextual information is not achievable because whatever is the application, there are always some information that needs to be included while others has to be neglected. This process is performed throughout the evolution of the application, and according to the users' feedback and the administrator's observations.

5 Context Acquisition

Depending on the field of application, various ways have been used to obtain context. In the following, and based on our observations, we distinguish between three ways to context acquisition.

- *Sensed context:* this type of information is acquired by the mean of physical or software sensors such as temperature, pressure, lighting and noise level.
- *Derived context:* this kind of contextual information can be computed on the fly. The most illustrative examples are time and date.
- *Context explicitly provided:* For example, user's preferences when they are explicitly communicated by the user to the requesting application.

Context acquisition from sensors is however not an easy process. This situation is due to the following main reasons: information may be acquired from different sensors and may require an additional step (interpretation) in order to be useful to an application. Plus, contextual information may be dynamic in nature and these changes need to be grabbed by the sensing infrastructure. Additionally, contextual information originates from heterogeneous and distributed sources and its corresponding sensing technology is generally fixed to a specific one. This is a poor practice from a software engineering point of view since it prevents reuse of application code when the sensors change.

A set of architectures for acquiring sensed context can be found in the literature. For instance, the Context And Location Aware Information Service (CALAIS) architecture [28] relies on a uniform interface in order to hide the details of sensors from applications. This interface is based on CORBA [5]. The Sulawesi framework [6] - aimed at supporting user's interaction with its wearable computer - , relies on a set of GPS units in order to acquire context information. But there is no support for context interpretation. Schmidt et al. in [38] propose a layered architecture for a context-aware system based on sensors. The low-level data obtained from physical and logical sensors is transformed into a set of cues that form the context using real-time recognition software.

The most consequent attempt at developing a reusable solution for context acquisition is the context toolkit [18] [2]. It is a generic Java framework that enables rapid prototyping of context-aware applications. The context toolkit is inspired from GUI toolkits in the way they mediate between the application and the user. Mediation in the context toolkit is performed between the application and the sensed information. It is realized by a set of abstract components (widgets, interpreters and aggregators) that support the acquisition of context data from sensors and to process it into high-level context information. A widget is a software component that provides access to context information through a uniform interface. It hides the complexity of sensors and provides reusable building blocks for context sensing. Applications interested in specific context information may subscribe to the corresponding widgets. The interpretation problem is solved by the mean of interpreters. An interpreter is responsible for mapping between context representation and a useful form to be used by the requesting application. For instance, an interpreter has the ability to convert between GPS coordinates to a street location. An additional software component is proposed; the aggregator. An aggregator is responsible about the context of a particular entity (person, place or object) and obtains information from multiple widgets. Dey in [20] reports a set of limitations of the context toolkit architecture. The main ones are the non-support for continuous context, unreliable data and context privacy.

6 Context Modeling

Context modeling is about providing a high level abstraction of context information. The diversity of contextual information and its use in diverse domains leads to different ways for modeling context. The following surveys main context models in various domains.

1. *Key-value pairs:* One of the earliest works on modeling context is the contribution of Schilit, Theimer and Welsh [36]. They propose the use of dynamic environment servers to manage contextual information for an environment (person, place or community). Contextual information is modeled as key-value pairs called environment variables and is used for mobile application customization. Here is an example environment variable; listing the names of occupants of a room:
   OCCUPANTS = adams:schilit:theimer:weiser:welch

2. *Web-based descriptions:* The Cooltown project [24] considers that everything has a web presence; people, places, and things. This vision allows these entities to participate in web services in order to become more personalized, more spontaneous, and more responsive to the wide variety of contexts in which people live their lives [3]. The project relies on a web-based context model in which each entity (person, place or thing) has a corresponding description that can be retrieved via a URL [23]. Entity descriptions are included in unstructured web pages, intended more to be used by human rather than applications.

3. *ConteXtML:* In [30], the author proposes an architecture known as Stick-e Note. The role of the latter is to make it possible to easily allow applications to adapt their user interfaces based on the current context. Stick-e Note relies on ConteXtML in order to model contextual information. ConteXtML is an XML-based protocol used as a standard format to exchange contextual information between a server and a mobile user. The following is an example cited in [33]. ConteXtML messages are grouped within a `<context>` tag or element. For instance, the following might be sent by a client to indicate its current location (the `<spatial>` element) and that it requires notes containing a data item called "landuse" with the value "pasture" (the `<require>` element).

```
<context session="123" action="update">
<spatial proj="UTM" zone="33"
    datum="Euro 1950 (mean)">
<point x="281993" y="4686790" z="205" />>
</spatial>
<require>
<note>
<data name="landuse" value="pasture" />
</note>
</require>
</context>
```

4. *Object-oriented model; contexts as object states:* In a recent work of Henriksen et al. [23], the authors investigate the modeling of context in pervasive systems. They propose a set of modeling concepts based on an object-oriented approach. This model is intended to overcome existing issues in previous models such as temporal aspects, context quality and relationships amongst context information.

   Based on a context-aware communication example system, Figure 1 explains the new modeling approach. In this approach, contextual information is built around a set of entities. Each entity represents a physical or a conceptual object such as a person, device or communication channel. Entities' properties such as the person's name and a device type are represented by attributes. Entities link to their corresponding attributes by the mean of associations. There are many types of associations (static, derived, temporal, etc). Each type models one of the requirements established so far such as: temporal aspects, context quality and relationships amongst contextual information (refer to [23] for the graphical representation of each association type).

   Henriksen's model does not handle yet contextual information distribution in a pervasive environment and does not address yet privacy issues of context. But, these two issues are actually under consideration.

5. *Contextual graphs:* A contextual graph [10] (CxG for short hereafter) allows a context-based representation of a given problem solving for operational processes by taking into account the working environment. Contextual graphs have been initially designed for an application for incident solving on a subway line. In order to solve an incident, the operator applies a set of curative actions called by the authors a practice. There are different practices for a given incident depending on the context in which the incident must be solved. CxGs support incremental acquisition of new practices, while, the initial structure of a CxG (its skeleton) is defined by the procedure that is established by the company for the problem solving. CxGs are now the object of studies by their own.

   Contextual graphs elements: A contextual graph is the representation of the set of practices to perform in order to solve a given problem. A path, from the input to the output of the contextual graph represents a practice with the contextual elements explicitly considered.

   A contextual graph is an acyclic graph with a unique input, a unique output, and a serie-parallel organization of nodes connected by oriented arcs. A node can be an action (square boxes in Figure 2), a contextual node (large circles) or a recombination node (small black circles), sub-graphs and parallel grouping.

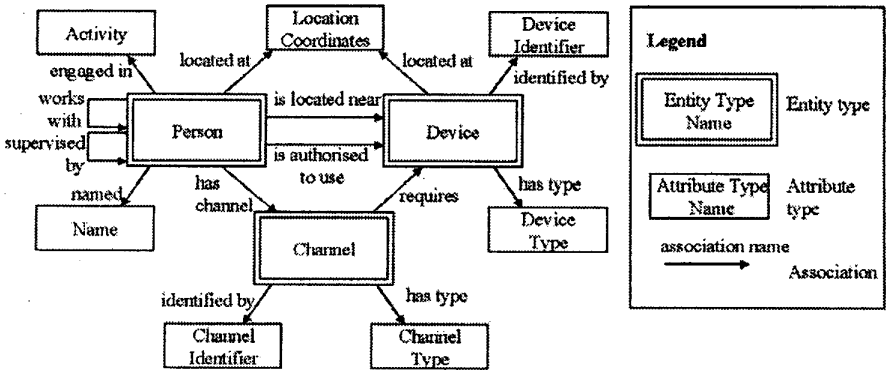

Figure 1. Graphical notation of Henriksen et al. modeling approach [23]

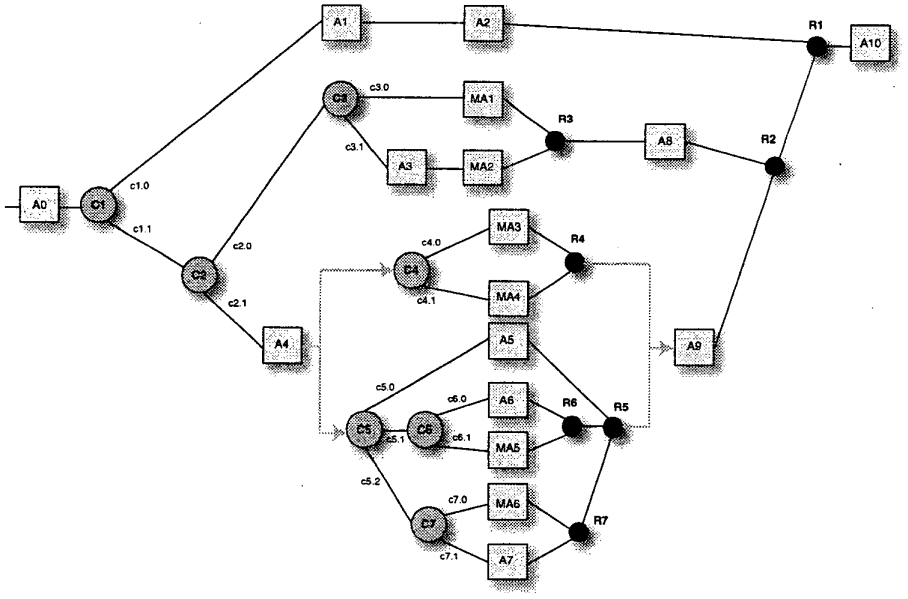

Figure 2. An example contextual graph

Actions and activities: an action is a basic executable method defined as being so by the person that solves the problem. The same action may be considered as a complex activity by another person in another context. Thus, actions are defined from a given point of view. An action is represented by *Ai*.
An activity is a complex action represented by a sub-graph with a unique input and a unique output. Activities are identified as recurring structures appearing in different graphs. Macro-actions represented in the CxG by *MAi* are special cases of activities reduced to a simple sequence of actions.

Contextual nodes: A contextual node is a decision node that directs the process throw one path among the set of all possibilities. A contextual node corresponds to the explicit instantiation of the contextual element. It is represented by *Cn*. For each contextual node *Cn*, a set of branches emerges. The branches are exclusive (one goes from the input to the output by only one of the n branches). The instances of the contextual elements are introduced in the contextual graph when needed, i.e. when a new practice is added in the contextual graph.

Recombination nodes: Upon instantiation of a contextual element, the natural behavior of the person that solves the problem is to comeback to a normal state (by eliminating the causes of the problem and coming back to a stable state of the system). Additionally, some contextual elements may change during a small period of time. Recombination nodes allow representing the convergence of the different strategies. A recombination is thus introduced as soon as the action sequences become again identical on the two paths.
Contextual and recombination nodes exist by pairs *(Cj, Rj)*. They give to contextual graphs a general structure of spindles or series of spindles, with a divergence of branches at contextual nodes initiated by a diagnosis and a convergence at recombination nodes, thanks to actions or activities performed.

Sub-graphs: A sub-graph is itself a contextual graph with one input and one output. A sub-graph encapsulates a local reasoning (a diagnosis/action structure) corresponding to intermediate goals. If a sub-graph is on a branch of a (Cj,Rj) pair, then it contains at most all the items on the branch. The same rules apply for the relationships between a sub-graph and a parallel action grouping.

Parallel action grouping: A parallel action grouping represents a set of *m* steps of the problem solving (actions or activities) that can be realized in parallel or in any order but all must be accomplished before to continue. They are illustrated in Figure 2 in thick arrowed lines.

The three types of context in contextual graphs: Since the set of contextual information is too big, Brézillon and Pomerol distinguish three types of context [32], namely,

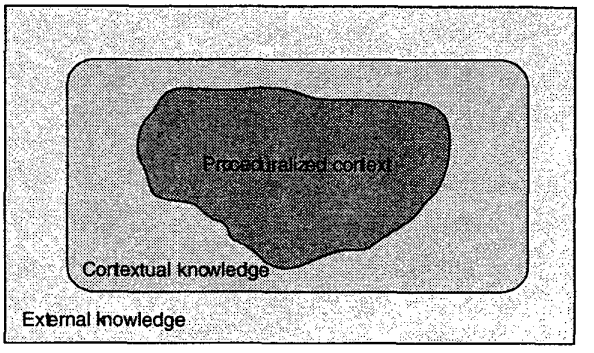

Figure 3. Three types of context

contextual knowledge, proceduralized context and external knowledge. To illustrate the three types of context, we consider a decision making process, whose actions to activate depend on a set of contextual information. The process is modeled using contextual graphs. Thus, contextual knowledge is a subset of context that directly intervenes in the decision making process. Moreover, only a subset of the contextual knowledge is used at a given step in the decision making process, this is known as proceduralized context. The remaining subset which includes information which is not relevant to the situation is called external knowledge. Consider the example presented in Figure 2, and more specifically the path with the action A3. The context of the contextual graph represented in Figure 2 is given by the elements C1, C2, C3, C4, C5, C6, C7. All that is not represented in the contextual graph is called external knowledge (i.e. knowledge in other contextual graphs in the database or not in the database). The context of an action (e.g. A3) is composed of two parts: the contextual elements used on the path from the input to the action and the other elements. On the path, some of the contextual elements has a value that intervenes in the practice (C1 with the value C1.1, C2 with the value C2.0 and C3 with the value C3.1) and others not (the value of C4 does not matter). The first ones intervene in an ordered way and is called the proceduralized context. The second one is the set of elements called the contextual knowledge. Thus, the context of the action A3 is defined by:

The proceduralized context C1 with the value C1.1, C2 with the value C2.0, and C3 with the value C3.1.

The contextual knowledge C4, C5, C6 and C7.

Note that the proceduralized context is an ordered sequence of contextual-knowledge pieces considered through their instantiations: (C1, C1.1), (C2, C2.0), (C3, C3.1).

Figure 3 illustrates the containment relationships between the three types of context.

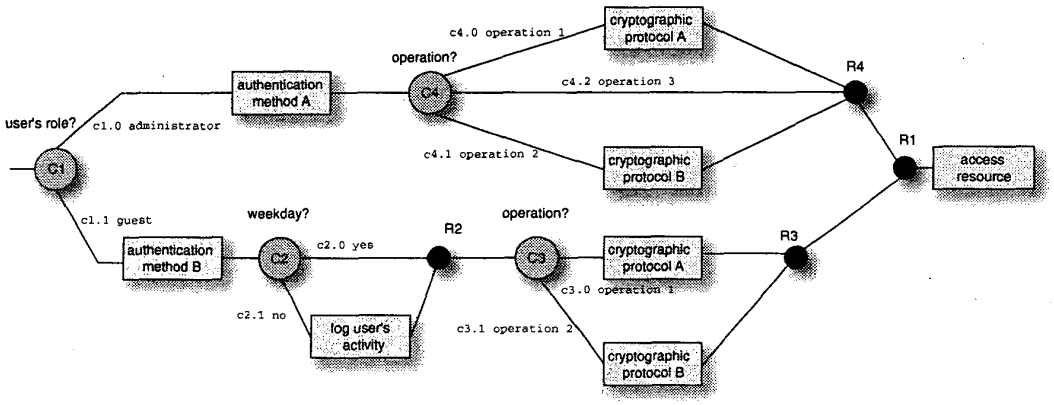

Figure 4. Contextual graphs-based security policy

An example: Modeling context-based security policies with contextual graphs: As an illustrative example, we use contextual graphs to model the context-based security policy that manages security levels in a distributed environment (Figure 4). In our case, the input corresponds to the user entering into the environment. The output corresponds to the user leaving the environment with no security incidents on both the environment and the user as long as the user is connected. In this example, a task corresponds to calling a method on a resource. The user wishing to access the resource first enters the distributed system that allows him to specify its role. Depending on this context (C1), a specific authentication method is applied. For an administrator user the cryptographic protocol that will be set between the user application and the resource application relies on the type of the requested operation. For a guest user, an additional step is needed in order to know if its corresponding activity must be logged or not depending on the day of request (C2). As one can observe, for a guest user, only two operations are allowed for him to perform on the resource in contrast to the administrator. Not allowed operations are not shown on the contextual graph. This is to allow specifying only safe paths in order to perform a secure action and is commonly known as closed security policy "which is not explicitly permitted is denied". According to the user's role, the time, and the type of requested operation, the contextual graph represents the decisions to undertake according to the current context. These decisions are security actions. These actions include authentication methods (code authentication, username/password, etc), cryptographic protocols and more specific operations as to log user's activity.

Practices acquisition in contextual graphs: Contextual graphs have the capacity of evolving by accommodation and assimilation of practices. An operator's practice may differ from a practice known by the CxG system because their contexts are slightly different and thus the operator has used a different action at a step of the problem solving. Generally, the new practice differs by few changes (an action instead of an existing one, the addition or lack of an action). Then, the CxG system enters a phase of acquisition of the new practice from the operator. The practice acquisition concerns the new action to integrate and the contextual element that discriminates that action with the previous one. The integration of the new practice requires either the addition of a new branch on an existing contextual node (just before the diverging part of the practice), or the introduction of a new contextual node to distinguish the alternatives. In all the alternatives, the piece of contextual knowledge to add must be instantiated. The phase of incremental acquisition of practices relies on interaction with operators in order to acquire operators' expertise, which consists of a context-based strategy and its evolution along the process of the problem solving. Concretely, the operator shows the system which path is concerned by the new practice, and makes the wished changes among the actions in the practice. Then, the system interacts with the user to check the novelty of the practice provided by the user. When a discrepancy is encountered at a step of the sequence of actions, the system asks for the difference of contexts (the name of the contextual node and the two different values of the contextual elements) between the previous action and the action to introduce, and where to put the recombination node. The contextual element that is added, generally comes from the external knowledge. The reason is that the instantiation of this contextual element was not relevant before, but is considered explicitly in the new practice. Thus, the movement from the external knowledge to the contextual knowledge of a contextual graph goes through its use in a proceduralized context.

7 Conclusions

Having presenting the importance of context-awareness in different computing domains, we observe that this paradigm is not yet widely adopted. In our opinion, the first reason is the lack of reusable architectures/mechanisms

for context acquisition and storage. Most of existing architectures are built in an ad hoc manner with the sole desire to obtain a working system. As a consequence, context acquisition is highly tied up with the remaining infrastructure; the resulting systems are thus, difficult to adapt and to reuse. Historical values of contexts are generally left to the consuming applications and do not benefit from a separate infrastructure. The second point is the difficulty to predict the relevant contextual information that may be considered in the system. Even if this process mainly depends on human intelligence and experience, most existing systems do not allow easy inclusion of new contexts. The latter may be discovered based on users' feedback, or on the systems administrators' observations. Scalability of context-aware applications is a supplemental problem rarely addressed. We suggest that the main reason is that most proposed solutions are at the prototype stage and thus, not care yet about the number of users. Additionally, unambiguous or missing contextual information is rarely addressed. Another main issue in context-aware computing is context privacy, in about all reviewed works; privacy is always left to future improvements of the system.

Additionally, many of the works in context-aware computing focus only on one aspect, either context sensing or context modeling. It is far from finding a framework that gets together these two aspects. As a consequence, there is a need for a generic context-based framework which can support the needs discussed earlier for context-aware applications.

In this paper we presented the main aspects of context-aware computing including context categorization, acquisition and modeling. The current work is based on the most recent contributions in the field and provides a structured guide for the pervasive computing community in order to evaluate the best technique to adopt in a specific application. We concentrate on the most generic techniques even if a standard framework is not actually achieved as mentioned earlier.

Even if it is out of scope of this paper, it is worth mentioning that this work constitutes the investigation part of the CoDiS project. CoDiS stands for Context-Based security for Distributed Systems (see CoDiS website [1]). The project is a collaborative work between the two universities of Fribourg, Switzerland and Paris VI, France. It aims at providing a generic infrastructure for managing context-based security in pervasive environments. Security is not limited to access control but may be extended to include intrusion detection systems or to determine which security level to apply. The conclusions we have drawn from this study drive us to build our own framework from scratch, for context gathering, management and modeling. Also, this work allows us to compare available modeling techniques in order to select the most suitable one for modeling context-based security policies. We refer the interested reader to the following references [25] and [26].

References

[1] Codis project. *http://diuf.unifr.ch/people/kouadri/phdthesis.html*, accessed April, 29th 2004.

[2] The context toolkit. *http://www.cs.berkeley.edu/ dey/context.html*, accessed April, 29th 2004.

[3] The cooltown project. *http://www.cooltown.com/cooltown/index.asp*, accessed April, 29th 2004.

[4] Merriam-webster online dictionary. *http://www.m-w.com/home.htm*, accessed April, 29th 2004.

[5] Object management group homepage. *http://www.omg.org/*, accessed April, 29th 2004.

[6] Sulawesi framework. *http://wearables.essex.ac.uk/sulawesi/*, accessed April, 26th 2004.

[7] thesaurus.reference.com. *http://thesaurus.reference.com/*, accessed April, 29th 2004.

[8] V. Akman, P. Bouquet, R. Thomason, and R. A. Young. Modeling and using context. *Proceedings of the Third International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT'01)*, Springer Verlag, Lecture Notes in AI 2116, July 2001.

[9] P. Bouquet, L. Serafini, P. Brézillon, M. Benerecetti, and F. Castellani. Modeling and using context. *Proceedings of the Second International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT'99)*, Springer Verlag, Lecture Notes in AI 1688, September 1999.

[10] P. Brézillon. Context-based modeling of operators' practices by contextual graphs. *Proceedings of the 14th Mini Euro Conference: Human Centered Processes*, pages 129–137, May 5-7 2003.

[11] P. Brézillon and M. Cavalcanti. Modeling and using context: Report on the first international and interdisciplinary conference (context'97). http://citeseer.nj.nec.com/386811.html, 1997.

[12] P. Brézillon and J. C. Pomerol. Contextual knowledge sharing and cooperation in intelligent assistant systems. *Le Travail Humain*, 62(3):223–246, 1999.

[13] P. G. Brown, J. D. Bovey, and X. Chen. Context-aware applications: From the laboratory to the marketplace. *IEEE Personal Communications*, 4(5):58–64, October 1997.

[14] D. Chalmers. *Contextual mediation to support ubiquitous computing*. PhD thesis, Department of Computing, Imperial College, 2002.

[15] M. J. Covington, M. Ahamad, and S. Srinivasan. A security architecture for context-aware applications. *Technical Report GIT-CC-01-12, College of Computing, Georgia*, May 2001.

[16] M. J. Covington, P. Fogla, Z. Zhan, and M. Ahamad. A context-aware security architecture for emerging applications. *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, pages 249–258, December 2002.

[17] M. J. Covington, W. Long, S. Srinivasan, A. K. Dey, M. Ahamad, and G. D. Abowd. Securing context-aware applications using environment roles. *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies (SACMAT'01)*, pages 10–20, May 2001.

[18] 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)*, pages 114–128, December 1999.

[19] A. K. Dey. *Providing Architectural Support for Building Context-Aware Applications*. PhD thesis, College of Computing, Georgia Institute of Technology, 2000.

[20] A. K. Dey. Supporting the construction of context-aware applications. *Dagstuhl Seminar on Ubiquitous Computing*, September 2001.

[21] A. K. Dey and G. Abowd. Towards a better understanding of context and context-awareness. *Proceedings of Workshop on The What, Who, Where, When, and How of Context-Awareness, Conference on Human Factors in Computing Systems (CHI 2000)*, April 2000.

[22] J. Gwizdka. What's in the context? *Proceedings of Workshop on The What, Who, Where, When, and How of Context-Awareness, Conference on Human Factors in Computing Systems (CHI 2000)*, April 2000.

[23] K. Henricksen, J. Indulska, and A. Rakotonirainy. Modeling context information in pervasive computing systems. *Proceedings of Pervasive 2002*, (Zurich, Switzerland):167 – 180, 2002.

[24] T. Kindberg et al. People, places, things, web presence for the real world. *Proceedings of the Third Annual Wireless and Mobile Computer Systems and Applications*, December 2000.

[25] G. Kouadri Mostéfaoui. Towards an architecture and new modeling approach for dynamic security in emerging applications. *Advances in Pervasive Computing, Alois Ferscha, Horst Hortner, Gabriele kotsis (eds.), Austrian Computer Society*, ISBN 3-85403-176-9:87–92, April 2004.

[26] G. Kouadri Mostéfaoui and P. Brézillon. Modeling context-based security policies with contextual graphs. *Workshops Proceedings of the 2nd IEEE International Conference on Pervasive Computing and Communication (PerCom04)*, pages 28–32, March 2004.

[27] C. Masone. *Role Definition Language (RDL): A Language to Describe Context-Aware Roles*. TR2002-426, Dartmouth College, Computer Science. Hanover, N H, 2002.

[28] G. J. Nelson. *Context-Aware and Location Systems*. PhD thesis, University of Cambridge, 1998.

[29] P. Osbakk and N. Ryan. Context privacy, cc/pp, and p3p. *Proceedings of UBICOMP2002 - Workshop on Security in Ubiquitous Computing*, 2002.

[30] J. Pascoe. The stick-e note architecture: Extending the interface beyond the user. *Proceedings of the International Conference on Intelligent User Interfaces*, pages 261–264, January 1997.

[31] D. Petrelli, E. Not, C. Strapparava, O. Stock, and M. Zancanaro. Modeling context is like taking pictures. *Proceedings of Workshop on Context Awareness (CHI 2000)*, 2000.

[32] J. C. Pomerol and P. Brézillon. Dynamics between contextual knowledge and proceduralized context. *Proceedings of the Second International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT'99)*, Springer Verlag, Lecture Notes in AI 1688:284–295, September 1999.

[33] N. Ryan. Contextml: Exchanging contextual information between a mobile client and the fieldnote server. *http://www.cs.kent.ac.uk/projects/mobicomp/fnc/ConteXtML.html*, accessed April, 29th 2004.

[34] B. Schilit and M. Theimer. Disseminating active map information to mobile hosts. *IEEE Network*, 8(5):22–32, July 1994.

[35] B. N. Schilit, N. Adams, and N. Want. Context-aware computing applications. *Proceedings of the IEEE Workshop on Mobile Computing System and Application*, pages 85–90, December 1994.

[36] B. N. Schilit, M. M. Theimer, and B. B. Welch. Customizing mobile applications. *Proceedings USENIX Symposium on Mobile and Location-Independent Computing*, (USENIX Association):129–138, August 1993.

[37] A. Schmidt, K. A. Aidoo, A. Takaluoma, U. Tuomela, K. Van Laerhoven, and W. Van de Velde. Advanced interaction in context. *Proceedings of First International Symposium on Handheld and Ubiquitous Computing (HUC99)*, pages 89–101, September 1999.

[38] A. Schmidt, K. A. Aidoo, A. Takaluoma, U. Tuomela, K. Van Laerhoven, and W. Van de Velde. Advanced interaction in context. *Proceedings of the First International Symposium on Handheld and Ubiquitous Computing HUC99*, pages 89–101, May 1999.

[39] N. Shankar and D. Balfanz. Enabling secure ad-hoc communication using context-aware security services. extended abstract. *Proceedings of UBICOMP2002 - Workshop on Security in Ubiquitous Computing*, 2002.