

Smart Space a new dimension of context

Luca Lamorte, Claudio Venezia

Abstract Telco Networks are rich contextual information containers and the exploitation of contextual information is expected to be a key success factor for the next generation Telco services. On the other hand the Internet of things is driving us towards a scenario in which network connected intelligent objects will be capable of offering services to users. We expect that those services and their availability will be part of the context itself.

Smart spaces will therefore compound context and available services.

In this article we describe how a context aware architecture has been evolved towards a novel context aware SOA which provides smart services according to users' context. This paper presents a real use case for defining the interactions among the different actors, a description of the architecture and some future works to define the next steps.

1 INTRODUCTION

Every year manufactures market a plethora of new gadgets, technologies and devices, improving their functionalities and design. These devices are network enabled and capable of sharing information with each other within an ad hoc network or through the Internet.

In the future we expect to be able to achieve ambient services which span over different devices and provide humans with innovative interaction means. What is really missing in this futuristic scenario is a reference framework able to translate and adapt the different languages they are speaking and to define a suitable way of discovering them when needed.

Today people are surrounded by these kind of internet of things, which are everywhere, which would be ready to be used if you were aware of how to invoke their services.

They might be triggered by an explicit request of the user, or suggested considering user context or preferences.

With the term user context we intend the collection of information related to the user which may come from the available collecting sensors. Adding to the various sensor based information about a user the list of ambient available services we got to our definition of Smart Context.

The Smart Context represents a user status but also the potential interactions he might enact with the surrounding digital intelligence.

Our platform is able to:

- collect information from all available sensors with respect to a particular user and determine her context
- collect information about the available service
- perform a correlation between available services and user context

Each service, correlated to a location or context, becomes part of the context itself. This opens up a new scenario, not only

providing the user with the context sensible service when he/her needs it, but also to understand which ones he might want to use depending on his situation or kind of place and activity he's doing.

The actual research about user context aims at an exploitation of contextual information for customizing general purpose services according to the situation and expectation of the user in that context. But if he's surrounded by specific services and applications of any potential interest for him, this is a new way of pushing a new context to the user.

2 USE CASE: MULTIMODAL INTELLIGENT FRAME

Luca has invited Claudio in his high tech house, to see some pictures of Claudio's recent trip. Claudio carries his brand new mobile with all stored pictures. When Claudio enters Luca's house, a message is delivered to his mobile, triggering that Luca's house provides some interesting services that he can use. Along with the available services list the message provides a link to reach a service console for using them. This message has been delivered by the Context Aware Platform [CAP], which has discovered Claudio in Luca's house.

Claudio, that is a curious guy, tries the link. A custom mobile web page, lists all the available services which Luca's house provides and acts as a remote controller. As first choice, the service console suggests Claudio to use Luca's network. Claudio was set by Luca as a friend in his contacts list, and special services, like fast and free connection are available for friends in his house.

A second service is the HD Television which provides a channel to show pictures. After selecting this service, the console offers to Claudio an interface to remotely handle the frame by changing the displayed picture via buttons like next, stop, previous or defining preferences like output, layout and finally selecting an input source. Claudio can use his mobile as a source and finally display to Luca those pictures. Moreover with the handling frame buttons, Claudio can easily stop the presentation by his mobile to better describe the trip dynamics.

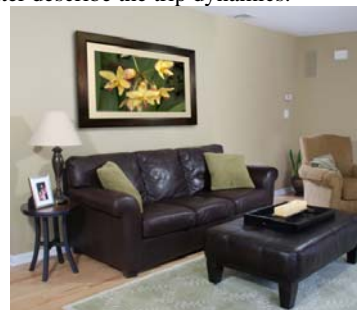


Figure 1. Luca's photo frame

3 Context Awareness Platform Architecture

In order to achieve the enablers for implementing a use case such as the one above we needed an architecture [6] capable of:

- collecting context information from different kinds of providers
- store contextual information and be able to aggregate them and provide to requestors

As said with we use the term context to indicate the collection of information available from user's surrounding environment, his/her terminal, network connectivity along with his/her profiles and preferences. It is a large amount of information which tends to grow proportionally to the observing time. The logical process of context high-level abstraction and extraction or inference is shown in Figure 2:

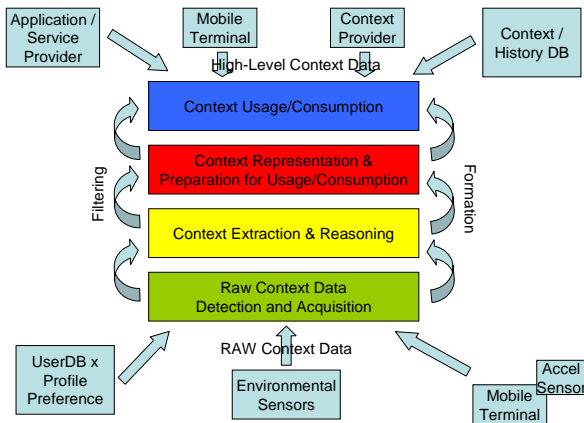


Figure 2: Context Processing

We've pursued a comprehensive and distributed context-aware system capable of aggregating and process a variety of context information. Despite that in context-aware system, domain knowledge is very much tied to applications; we resolved that building vertical solution was not effective. Therefore one of the goals of any context handling system is to split context and available applications. The Context Management Architecture shown in Figure 3 enables reuse and support for many applications from context acquisition to the context usage or consumption. This CMA designed according to the prosumer-consumer paradigm where some entities producing context are Context Providers (CP) while other entities consuming context are Context Consumers (CC). These entities are communicating each to other through a central entity named Context Broker (CB) which also provides some additional functions in the system.

The main characteristics of the entities shown in Figure 3 are the following:

Context Broker (CB) is the fundamental entity that creates and manages the relationship between CP and CC. The CB performs context source discovery and management and subject-based lookup service. Context Provider is the logical point where context is detected and acquired or extracted from other context

sources is the Context Provider (CP). A context provider provides mechanisms for on-demand queries and may optionally support subscription based notifications based on defined event occurrences, e.g. general context change or timer expiration.

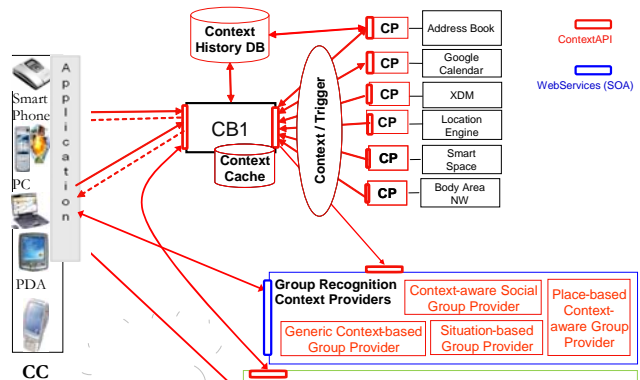


Figure 3 Context Management Architecture

Along with the acquisition of context data, the context provider performs data aggregation, fusion and inference. These activities are performed by different context providers, based on their individual internal logic and context output. A context provider is supposed to maintain context metadata and historical context. Within the CMA the context inference process is candidate to be embedded into CPs.

Reasoning and interpretation are seen as crucial tasks of the context-aware system, for deriving high-level contextual information from lower level context or raw data (e.g. databases or sensor output).

Most of the related research activities saw the context inference process as unidirectional process and rather static as far as the control or configuration are concerned. Due to the highly dynamic nature of mobile communication systems (e.g. mobility, activity change), reasoning and inference mechanisms need to be able to cope with the continuously change of contextual information;

Context Consumer is the logical point where context is used (or consumed) accordingly to context-aware service logic is the Context Consumer (CC) entity. The CC uses event-based model and uses query-based publish-subscribe mechanism for context data. In addition to context-aware applications, other architecture entities, like a CP or service enabler, can also assume the role of CC;

Service components or service enablers perform a set of generic control functions, used by applications and other service enablers, as illustrated in Figure 3.

These enablers usually act as a bridge between the CP and CC. Within the CMA, a service enabler assumes the role of CC and interacts with the CP. This initial interaction is always mediated by the CB.

Of course such architecture has a single point of failure represented in a Context Broker. However it has also many advantages respectively to other models based e.g. distributed or end-to-end paradigm. This model allows maintaining the control

over all the context information processed by the system and all the entities interconnected within the system and handling the context are connected over Context Broker. So the CB simplifies look-up based on required context for the entities within the system. Moreover, CB may have cache and history functions and then it may become the point of reference for already known and still valid context and for already expired context respectively. The weak points of this solution may be easily solved by common practices applicable for robustness within centralized systems.

4 CONTEXT INFORMATION PROCESSING MODEL

Context information processing and the main CA components' relationships within this architecture is shown in Figure 4.

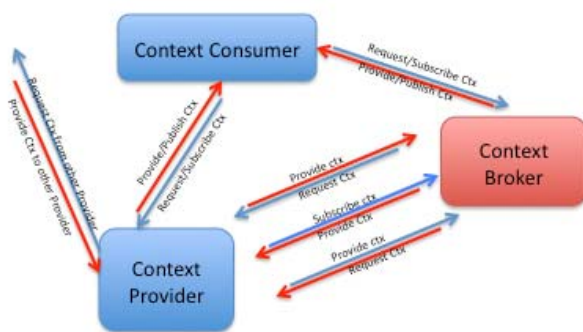


Figure 4: CMA entities relationships

Achieving a loosely coupled architecture was a critical requirement since the beginning.

Fixed Mobile Convergence of TLC and IP providers emphasizes the problem of heterogeneity and openness to 3rd parties applications using both types of networks. This issue is addressed within SOA concept [6] where enablers organize access to network's capabilities through standardized interfaces united within a service delivery platform, that may be opened by a service exposure layer towards 3rd parties applications respecting the identification, security and billing issues of external accesses in conventional systems.

The context management system has been created to avoid vertical context-aware applications and offer context-awareness to any application or service provider.

There is logic separation in context management, which remains within TLC domain, and application or service related functionalities provided by their respective providers and built on top of TLC features.

TLC features are abstracted in this case from service providers in a way that a service provider may not know how the functionality is implemented within the TLC domain.

This model creates a significant improvement in the business value creation chain; application logic inventors don't know all the particularities of TLC, while TLC are not always able to create services to attract users and create new revenue streams. The users take benefit of usage of heterogeneous network features without taking care to which networks they are connected to and which technology is used for data transmission.

In order to maximize interoperability between providers from different domains, a common language for user information representation has been defined: "ContextML", an XML-based language, which states a meta-model for the representation that all providers need to comply with, in order to register them with a User Information Broker and to enable potential Consumers to discover the user information they need.

For simplifying context management user's information have been subdivided by into scopes, namely sets related to the same information category. For example, the scope named "position" groups latitude, longitude and range with respect a certain entity's (e.g. user) location. Scopes can be atomic or aggregated, as union of different atomic scopes.

The ContextML schema is composed by the following elements:

- ctxEls: contains a set of information for a certain entity (response to the getContext method)
- ctxAdvs: contains the advertisement of provider features (providerAdvertising method) to the broker
- scopeEls: contains response from the Broker to the getAvailableAtomicScopes method
- ctxPrvEls: contains response from the Broker to the getContextProviders method

Any user information given by a provider is characterized by an entity and a specific scope. When a provider is queried, it returns the required data in a XML document, which contains the following elements:

- contextProvider: a unique identifier for the provider of the data.
- entity: the identifier of the entity which the data are related to.
- scope: the scope which the context data belongs to.
- timestamp and expires: respectively, the time in which the response was created, and the expiration time of the data part.
- dataPart: part of the document which contains actual user information data which are represented by a list of a features and relative values through the <par> element ("parameter"). They can be grouped through the <parS> ("parameter struct") or <parA> ("parameter array") elements if necessary.

For example, the getCivilAddress method of the Location Provider for latitude 45.112 and Longitude 7.67 for user "luca" is invoked through the following HTTP GET:

```
http://svrXXX/LP/LocInfo/getCivilAddress?username=x&lat=45.112&lon=7.67
```

and returns the following XML content:

```
<?xml version="1.0" encoding="UTF-8" ?>
<contextML xmlns="http://ContextML/1.1"
```

```

xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance"
xsi:schemaLocation="http://svr/schemas/ContextML-1.1.xsd">
<ctxEls>
<ctxEl>
<contextProvider id="LP" v="1.0.2" />
<entity id="luca" type="username" />
<scope>civilAddress</scope>
<timestamp>2007-01-24T16:05:19+01:00</timestamp>
<expires>2007-01-24T17:05:19+01:00</expires>
<dataPart>
<parS n="civilAddress">
<par n="street">Via Arrigo Olivetti 5</par>
<par n="postalCode">10148</par>
<par n="city">Torino</par>
<par n="subdivision">TO</par>
<par n="country">Italy</par>
</parS>
</dataPart>
</ctxEl>

```

Each specific domain of information is mapped by one or more scopes. As said a scope is a simple a priori aggregation of data with a semantic coherence, grouped together and identified by a name, which is used as parameter name (n attribute of <par> element) in ContextML. Such names could easily be mapped to concepts in an ontology. The scopes currently defined are related to: location, calendar and device information.

In addition to the above We've added the list of the ambient services which would be triggerable by the user in a particular context (e.g. location plus activity and social information). Basically ambient services publish their interfaces as contextual information. At any point in time a user can ask the platform if there is a smart service he can use for the time being. In the case of our use case when Claudio approaches Luca's television he is notified of its availability and provided with the information for using it.

In fact the photo frame declares several interaction interfaces in the form of URI like next, stop, refresh, set interval, choose playlist.

The system stores these information and associate them to a physical location (e.g. room which hosts the photoframe) and/or a suitable fruition context (e.g user's is in leisure status).

Most of portable devices can be located via cell triangulation, Global Positioning System (GPS) or bluetooth and when a user approaches the photo frame the presence of the photo frame becomes actually part of his context. The contextML document will provide the following in addition:

```

<contextML>
<ctxEls>
<ctxEl>
<contextProvider id="AmbientServProv" v="1.0.2" />
<name>photoframe</name>
<op>Http://smartspaceIP/frame/Home</op>
<op>Http://smartspaceIP/frame/upload</op>
<op>Http://smartspaceIP/frame/next</op>
<op>Http://smartspaceIP/frame/stop</op>
<op>Http://smartspaceIP/frame/refresh</op>
<op>Http://smartspaceIP/frame/setInterval</op>
<op>Http://smartspaceIP/frame/changePlaylist</op>

```

```

</ambService>
</ctxEl>
</ctxEls>
</contextML>

```

The URIs above points to the functionalities implementations which are exposed via REST interfaces, and available for use, if the requestor has the right privileges. Whether these REST functionalities are implemented as a Web application which resides on the photoframe or a network Web application which than instructs the photoframe via proprietary protocols is just an implementation detail which depends on the device capabilities.

5 SMART CONTEXT CLIENT GUI AND IMPLEMENTATION

From the really beginning of this work We had clearly in mind that users should have been the primary actors of this user centric contextual services ecosystem.

Given the importance of the mobility aspects it was a requirement to provide mobile friendly interfaces to this ecosystem. Basically the requirements are:

- mobile enabled experience
- platform independence
- a basic knowledge of the mobile capabilities [9]

The choice was then to achieve a mobile web based client and provide a mobile Web application. Users interact with a Web based Mobile Desktop. This mobile desktop periodically reloads as users change location and displays different services and information accordingly. Those data could also be overridden manually by users by expressing explicit preferences. This is a very dynamic phase of the scenario, because as the user moves around the mobile application changes.

The photo frame is a static element in the scenario, because its services, its location and space is fixed. The only thing that it has to do it is to announce itself as a service available for a context/space. This is done by sending its registration the the CAP as a context provider.

The Web based mobile desktop periodically asks a context broker for updated context information (receiving back a XML such as the one described in the previous chapter) and displays to the user the information about potential available surrounding services. The process is more complex because before sending back the XML result, the CAP detects the user identity, and the requesting client profile. Matching and crossing the information stored in the user profile, his/her preferences and context, it creates a customized result to fit that specific user and his/her situation. The page displayed on the mobile web desktop after the update will reflect exactly the context, services and the user's preferences.

When a user gets in proximity of a photo frame the mobile desktop GUI shows a section which displays the interaction possibilities. Basically several available remote commands are executable via HTTP/REST interfaces.

For what concern privacy and trust issues in our implementation the requirement for being able to interact with the photo frame is being part of the owner's social network, which automatically grants the right to use it. That's why Claudio's Web Mobile

desktop notifies him of Luca's photo frame presence and Claudio can choose to use this service. If the web mobile desktop is not running, CAP can notify the user in different ways according to the mobile device capabilities (MMS, SMS, Instant Message).

When notified Claudio can access simply clicking the FRAME button of his Mobile Web page, entering a page dedicated to the frame interaction where he can find functionalities like those listed below:

- more details about the frame like size, number of pictures available, refresh frequency, layout type, ...
- changing the output layout, if the user has the authority. Some example can be: slideshow, split display by 2,4,6 ...
- Handle the display experience using those operation:
 - auto
 - next
 - previous
 - play/stop.
 - More/less info

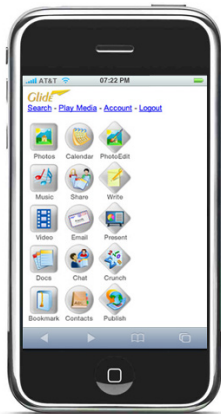


Figure 6 Smart Context client

6 RELATED WORK

There is a lot of work related to SOA and context management. Both the i-Room [1] and Gaia [2] present smart office application scenarios. i-Room focuses on human computer interaction (HCI) in a single interactive meeting room. Gaia defines Active Spaces as physical spaces coordinated by a responsive context-based infrastructure. This infrastructure is made available to service applications by means of an operating system (Gaia OS), which provides context and event management services to running programs. As a future work, the Gaia team plan to federate Gaia Services to aggregate different active spaces. Cooltown [3] uses the technologies behind the Web to provide pervasive nomadic computing in urban environments. In Cooltown, interest places and resources are tagged with URLs or other identifiers that can be retrieved by users' personal devices by means of bar codes, RFIDs or IR transceivers. URLs may be used to access the different services related to their associated points of interest, and other identifiers may be resolved to URLs which link to the services related to the identified item. Resources are grouped in places, and for

each defined place there is a place manager, which maintains directories of resources, acting both as a resolver for looking up resources in that place from their identifier and as a Web server providing information about resources. COBRA [4] takes advantage of multiagent systems to develop context-aware applications. It is based on a brokercentric architecture used to provide runtime support for context awareness in an Intelligent Meeting Room. In COBRA, the environment is divided in domains, and there is a broker for each domain, which is an autonomous agent that manages and controls the context model of the specific domain. Though COBRA brokers are intended mainly for context sharing, its centralized approach to management in each domain and the possibility of sharing context between domains through context federation is closely related to the approach we have taken to develop our hierarchical architecture for smart spaces

7 CONCLUSIONS & FUTURE WORK

Ambient intelligence and context awareness seem to be the ideal ground for the exploitation of the already consolidated SOA principles.

In the meantime, interoperability, lack of standards, and privacy are still important challenges to be mastered.

For the future our idea is evolving this architecture towards an "increasing" social awareness. Ambient services are often delivered to groups or communities of people. If the communication style is changing from p2p to group communication thanks to Social Networking applications we could expect that this might also apply to the service experience. For people belonging to the same community, who also share the same physical spaces, could be indeed fascinating to create and provide ambient services to their friends. Telecom Italia is participating to PERSIST exactly for the sake of researching on Smart Spaces technologies and pursuing a platform for achieving highly flexible and interoperable Personal Smart Spaces.

REFERENCES

- [1] Johanson, B., Fox, A., Winograd, T.: The interactive workspaces project: Experiences with ubiquitous computing rooms. *IEEE Pervasive Computing* (2002) 67–74
- [2] Román, M., Hess, C.K., Cerqueira, R., Ranganathan, A., Campbell, R.H., Nahrstedt, K.: Gaia: A middleware infrastructure to enable active spaces. *IEEE Pervasive Computing* (2002) 74–83
- [3] Kindberg, T., Barton, J.: A web-based nomadic computing system. *Computer Networks* 35 (2001) 443–456
- [4] Chen, H.: An Intelligent Broker Architecture for Pervasive Context-Aware Systems". PhD thesis, University of Maryland, Baltimore
- [5] 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*, 19:12-20, (2004).
- [6] L. Lamorte, C.A. Licciardi, M. Marengo, A. Salmeri, P. Mohr, G. Raffa, L. Roffia, M. Pettinari, T.S. Cinotti, "A platform for enabling context aware telecommunication services," in: Proc. Third Workshop on Context Awareness for Proactive Systems, Guildford, UK (2007)
- [7] C.Venezia, C.A. Licciardi, A. Salmeri, L.Buriano; "Rule based dynamic adaptation of mobile services based on context"; in the proceedings of ICIN 2008.
- [8] W3C Ubiquitous Web Applications. <http://www.w3.org/2007/uwa/> [Online; accessed 10-Feb-2007].
- [9] W3C delivery context ontology. <http://www.w3.org/TR/dcontology/> [Online; accessed 10-Feb-2007].