ContextML

A Light-Weight Context Representation and Context Management Schema

Michael Knappmeyer[1,2], Saad Liaquat Kiani[1], Cristina Frà[3], Boris Moltchanov[3], Nigel Baker[1],
[1] Mobile & Ubiquitous Systems Group, Bristol Institute of Technology, University of the West of England, Bristol, UK
[2] Faculty of Engineering and Computer Science, University of Applied Sciences Osnabrück, Germany
[3] Telecom Italia Lab, via G.Reiss Romoli, 274, 10148-Torino, Italy
Email: {saad2-liaquat, nigel.baker}@uwe.ac.uk, m.knappmeyer@fh-osnabrueck.de, {cristina.fra, boris.moltchanov}@telecomitalia.it

Abstract—**Context representation is a fundamental process in developing context aware systems for the pervasive world. We present a light weight XML based context representation schema called ContextML in which context information is categorized into scopes and related to different types of entities (e.g. user, device). The schema is also applied for encoding management messages in order to allow for a flexible framework supporting gradual plug & play extendibility and mobility. ContextML is tailored to be used for REST-based communication between the framework components. Explanation of the schema is provided with the help of real world examples. Moreover, the European C-CAST testbed is introduced, embracing a variety of context providers and application domains.**

Keywords: Context Representation; Context Awareness

I. INTRODUCTION

In each context-aware communication system the context information needs to be represented and modeled for being machine interpretable and exchangeable using well-defined interfaces. The goals are to support easy manipulation (low overhead in keeping the model up-to-date), easy extension (cheap and simple mechanism for adding new types of information), efficient search, query access and scalability. In literature different approaches for representing contextual knowledge and semantic information can be found. On the one hand the representation is tightly coupled to the inference mechanism, e.g. probabilistic logic requires the modeling of probabilities. On the other hand the representation is often tailored to the problem domain and to the specific goal of the system.

Strang and Linnhoff-Popien [1] identify generic requirements: The modeling approach should (1) be able to cope with high dynamics and distributed processing and composition, (2) allow for partial validation independently of complex interrelationships, (3) enable rich expressiveness and formalism for a shared understanding, (4) indicate richness and quality of information, (5) must not assume completeness and unambiguousness, (6) be applicable to existing infrastructures.

Context models can be classified into six different model categories, namely Key-value models, markup scheme models, graphical models, object oriented models, logic based models and ontology based models [2].

Beside the context information itself, a model for representing related metadata is crucial. Such *context meta information* may include a quality of information quantifier, e.g. the degree of uncertainty, possibility, measurement accuracy, resolution or confidence interval. The required attributes are dependent on the inference and reasoning mechanisms. Especially when taking historic context into account, it is important to embed data related to the *time* such as time of creation (timestamp) and expiry time. Rapidly changing information can be differentiated from rather static information (such as gender, year of birth). Hyunjun Chang et al. [3] propose the modeling of a lifecycle for contextual information and an appropriate representation in meta data. The state of contextual information (e.g. ready, running, expired, suspended) enables flexible and fast transitions when the context changes temporarily.

Based on lessons learnt from existing efforts in this domain, we present here a light-weight markup based scheme, titled ContextML, that is being employed in the C-CAST project [4][5] for representation of context information and for encoding management messages. Before this scheme is discussed (Section III), it is necessary to briefly describe the context provisioning system (Section II) which utilizes the ContextML format. The schema is evaluated in Section IV by presenting a prototype framework in which several Context Providers are deployed.

II. CONTEXT PROVISIONING SYSTEM MODEL

Our reference context provisioning system, C-CAST, consists of three core components. Context producing components are titled Context Providers and components that use context are Context Consumers. A central brokering component, Context Broker, facilitates the flow of context information between providers and consumers. Detailed description of the C-CAST context provisioning system is provided in [4][5][6], a brief overview of core components is presented in the following paragraphs.

A Context Consumer (CxC) is a component that queries for and uses context data, e.g. a context-aware application. A CxC can retrieve context information by sending a subscription request to the Context Broker (CxB) and context information is delivered asynchronously once it is available or when it changes. A synchronous method of requesting context information also exists where a CxC requests the Context Broker (CxB) for a particular Context Provider (CxP) and can query the CxP directly.

This work is supported by the European ICT project "Context Casting (C-CAST)". The schema and original concepts about ContextML have been developed by Telecom Italia Labs [9]; new scopes definitions have been contributed by collaborating partners.

A Context Provider (CxP) is a component whose task is to provide context information of a certain type, e.g. weather, location, activity, etc. Therefore, a CxP gathers data from a collection of sensors, network, services (e.g. web services) or other relevant sources. The CxP can use various filtering, aggregation and reasoning mechanisms to infer context from raw sensors, databases or other source data depending on the type of context it provides. A CxP provides context data only further to a specific invocation/subscription and is specialized on a particular context domain (e.g. location, weather etc). A Context Source (CxS) is a special CxP, only offering an asynchronous mode of communication. That is, a CxS directly pushes context to the broker without being queried.

Context Broker (CxB) is the main component of the architecture. It works as a handler and aggregator of context related communication and as an interface between architecture components. Primarily the CxB has to control context flow among all attached components which it achieves by allowing CxCs to subscribe to context information and CxPs to deliver notifications. For facilitating synchronous (on-demand) CxC context queries, CxB also provides a CxP lookup service and proxy query service by maintaining entries of context providers registered with the broker, their communication endpoints, and their capabilities.

III. CONTEXTML MODEL

ContextML is used in our context provisioning system to model context information, context subscription/notification and some control messages as well. The following paragraphs describe the core elements of ContextML. A XML schema of the language is also available [10].

A. Fundamentals

1) Entity

In the C-CAST system, every exchange of context data is associated to a specific *entity*, which can be a complex group of more than one entity. An entity is the subject of interest (e.g. user or group of users), which context data refers to, and it is composed of two parts: a type and an identifier.

The type refers to the category of entities; exemplar entity types are *username* (for human users), *imei* (for mobile devices), *SIP URI* (for SIP accounts), *room* (for a sensed room) and *group* (for groups of other entities e.g. usernames or IMEI numbers). The entity identifier specifies a particular item in a set of entities belonging to the same type. Every human user of C-CAST system could be related to many entities in addition to the obvious type *username*, therefore a component that provides identity resolution is necessary. In the C-CAST architecture it is performed by the CxB in collaboration with a User Profile CxP.

2) Scope

Specific context information in ContextML is defined as *scope* and is a set of closely related context parameters. Every context parameter has a name and belongs to only one scope. Using scope as context exchange unit is very useful because parameters in that scope are always requested, updated, provided and stored at the same time; it means that data creation and update within a scope are always atomic and that context data in a scope are always consistent. Scopes themselves can be atomic or aggregated in a union of different atomic context scopes. In a way, the scope design can be compared to object oriented modelling where the scope refers to an object class.

For example, consider scope 'position' which refers to the geographic location of an entity. This scope could be composed of the attributes latitude, longitude and accuracy and these are always changed at the same time. Updating the latitude value without updating longitude, if is changed, is obviously not correct. Entity-scope association is illustrated in Fig. 1.

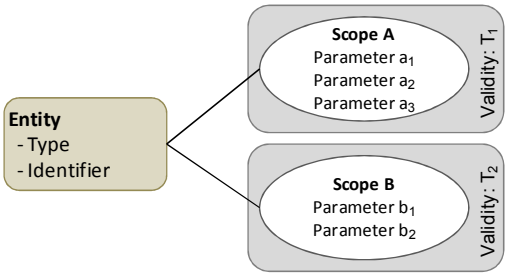

Figure 1. Entity scope relationship – an entity can have many context scopes associated with it, each with its own validity period

B. Context Management Messages

1) Advertisement

A Context Provider registers its capabilities to the broker by sending an advertisement message which is encoded in ContextML (cp. Fig. 2). The CxP informs a broker about how to access it (*urlRoot*) and what scopes it supports (*scopes*). Hence, in a single message several scopes can be registered simultaneously.

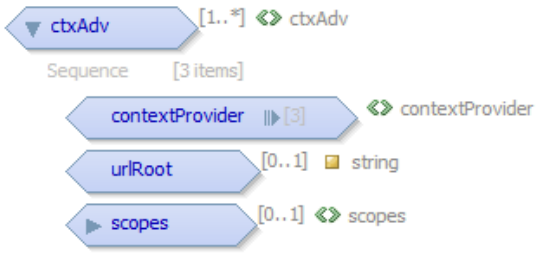

Figure 2. ContextML CxP Advertisement Schema Element

Fig. 3 shows the schema for a single scope. It consists of the *scope name*, the *url* where the scoped context can be requested from, the types of entities it supports, input context that is required to query context and dependency on other scopes (*depUrl*). The CxB keeps the entries in a lookup table. It is important to mention that these lookup entries are linked to an expiry timer. Therefore, a CxP needs to refresh the lookup entries by periodically invoking keep-alive advertisements. This mechanism serves also for basic mobility, in case a component is deployed on a mobile device. If the IP address of a mobile provider changes, the consecutive advertisement will automatically fix the connectivity loss. Moreover, the advertisement procedure enables plug and play behaviour of the framework. The evolution of context-aware applications can easily be supported by adding new scopes during runtime - without having to redesign or restart the system from scratch.

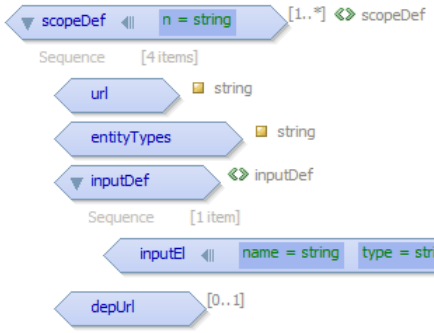

Figure 3. ContextML Adverisement Scope Schema Element

2) CxP Lookup

A CxC being interested in a specific scope can find out how to get the context information by querying a CxB for the access information and for determining the required input values. The CxB applies a matching algorithm with the data available in the lookup table and replies with a ContextML encoded message similar to what has been previously illustrated in the advertisement process. The schema is omitted here for the sake of brevity.

3) Acknowledgement

An acknowledgment (ACK) is being sent as control message to confirm the reception and completeness of various management invocations (e.g. advertisement, context update). According to Fig. 4 each ACK message contains not only the status (OK, ERROR) but also a numerical response code 0..999 mirroring HTTP status codes (e.g. 500 = Internal Error). The mandatory field *contextProvider* contains the ID of the component sending the ACK, *timestamp* contains the sending time. Optionally and dependant on the invoked method, the entity and scope information is encoded.

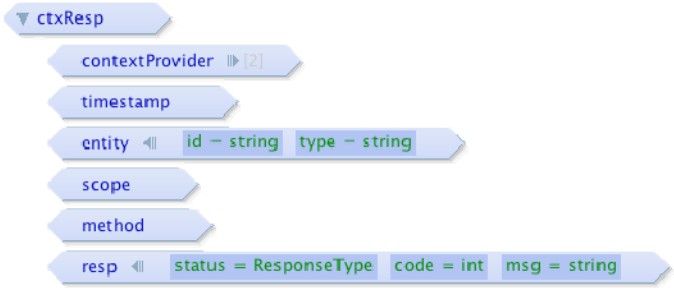

Figure 4. ContextML Acknowledgment Schema Element

C. Context Representation

Whenever a context consumer requests or subscribes to a specific context scope, it receives a response encoded in the ContextML element 'ctxEl' when context is available. *ctxEl* contains information about where the context has been detected and encoded (*contextProvider*), which entity it is related to (*entity*), what scope it belongs to, and the actual context data in the *dataPart* element. A graphical description of this element is given in Fig. 5. The elements *par*, *parS* and *parA* are simple constructs to store name-value pairs and attribute collections (structs and arrays) respectively. Every scope instance (context information) that is exchanged is tagged with a specific timestamp (time of context generation) and an expiry time. The expiry time tag states the validity of the context information. After this time, the information is considered invalid.

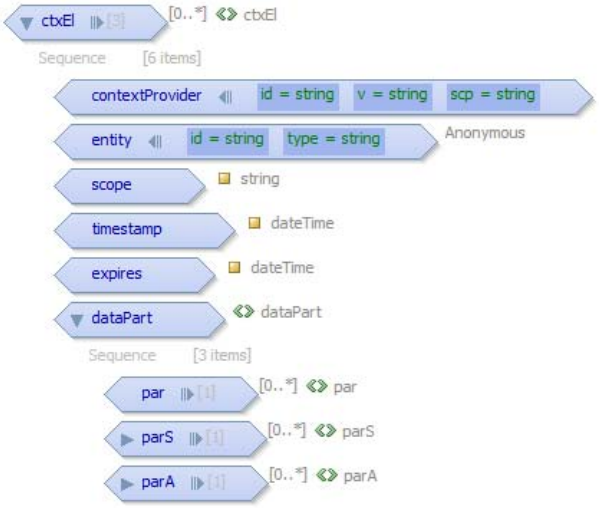

Figure 5. ContextML Context Representation Schema

D. Context Query Models

Our basic context query model is comparable to a key-value query concept. The triple of entity identifier, entity type and scope serves as key whereas the associated context refers to the value. Together with the access information, the required input data (i.e. instantiated context of a more primitive scope) has been communicated in the advertisement process. This circumstance highlights the distributed context processing and the layered abstraction and allows for several implemented query methods.

1) Synchronous Direct CxP Invocation

Context consumers can formulate a simple context query for a particular context scope by invoking the context provider through RESTful (Representational State Transfer [8]) HTTP and encoding the request parameters in the HTTP URL directly. Figure 6 depicts the structure of a context query that requests weather information related to a user in a particular location. Hence, in this example, the geographic coordinates (scope *position*, parameters *latitude* and *longitude*) as required as input parameters.

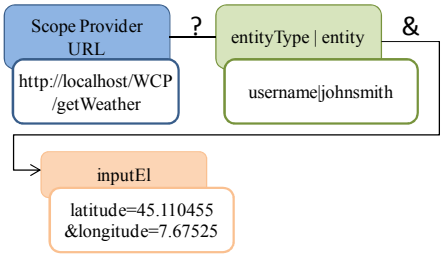

Figure 6. Structural representation and example (weather of a user's location) of a context query

2) Synchronous CxB Proxy Query

Another mode of on-demand query is provided by the CxB. Its proxy query service queries the required providers on behalf

of the CxC. In our example, the CxC does not have to invoke the position provider at first for retrieving its longitude and latitude. Instead the CxC can simply query a broker for the desired scope and the complexity is hidden from the CxC. This mechanism is especially useful for resource constraint CxC (e.g. context-aware applications running on mobile phones) and becomes efficient if the broker is equipped with a local context cache which keeps context until its expiration.

3) Asynchronous Event-based Publish/Subscribe

As third option, a CxC can utilize an event-based publish/subscribe interface. It can formulate the event conditions in a specific schema, the Context Query Language (CQL), which shows similarities to the well known Structured Query Language (SQL). In addition, a callback URL is provided which is addressed if the conditions become true.

Listing 1 provides an example where notifications are requested in case user 'john' is located in Bristol (scope: *civilAddress*; parameter: *city*). Except from the equals (EQ) operator, CQL supports: unequals (NEQ), starts with (STW), contains (CONT), ends with (ENW). Each subscription is bound to a specified validity time (in seconds) but can easily be renewed. Subscriptions are acknowledged by the CxB/CxP.

```
<contextQL>
 <ctxQuery>
  <action type="SUBSCRIBE"/>
  <entity>username|john</entity>
  <scope>civilAddress</scope>
  <validity>180</validity>
  <conds>
   <cond type="ONVALUE">
     <constraint param="civilAddress.city" op="EQ" value="Bristol"/>
   </cond>
  </conds>
 </ctxQuery>
</contextQL>
```

Listing 1. Exemple Context Subscription

IV. EVALUATION

The presented schema has been developed and trialed in the European C-CAST [7] evaluation testbed. In order to strengthen its applicability to numerous service domains, we present some examples of Context Providers (cp. Fig. 7) and their respective context representation. The purpose of each provider is easier to explain when following the scope dependency graph in Fig. 8.

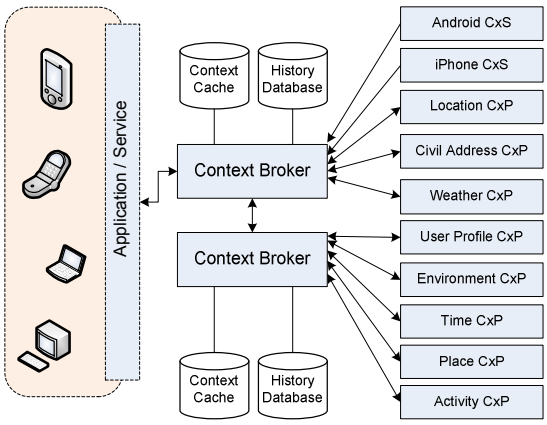

Figure 7. Context Provisioning Testbed

For Apple's iPhone and for Android devices a Context Source Application has been developed to publish primitive context into the broker cache (cp. Listing 2). It provides the following scopes: deviceStatus, deviceSettings, wf (=wifi), cell (=cellular), bt (=Bluetooth) and motion. Exemplar output is depicted in Listing 2. The *Android application* installs itself as background service and is restarted with every reset of the phone. It is non-intrusive to the user and allows several settings to respect her privacy, e.g. the publication of nearby Wifi access points can be suppressed whereas the phone continues to send its device capabilities. In addition, the user can monitor which information is made public. Furthermore, the update interval can be easily configured per scope. A screenshot of the context source application is presented below in Fig. 9.

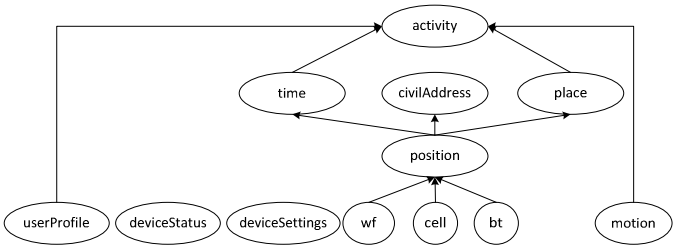

Figure 8. Scope Dependency Graph

The scopes *deviceStatus* and *deviceSettings* provide device specific information, e.g. the screen resolution, the battery capacity, the screen size and the orientation of the display. Such information can be taken into account by multimedia applications for adapting their content delivery to the capabilities of the device.

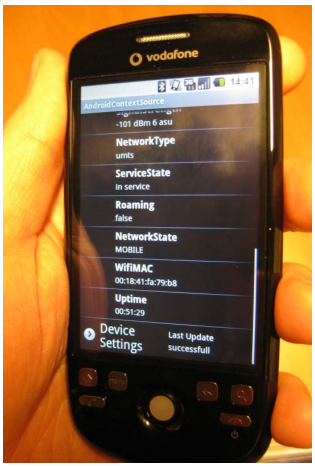

Figure 9. Android Phone Publishing Context

Within the scope *wf* -amongst other attributes- information about the MAC addresses of WiFi access points in proximity and the respective RSSI (Received Signal Strength Indicator) is contained. The same applies to the cell *scope* which provides information about serving and neighboring cell towers.

The *Location Provider* (cp. Listing 3) requires these three scopes (wf, bt, cell) as input context to determine the geographic position of the entity. This way, the position can be estimated independently from GPS sensors, hence battery costs are reduced and indoor coverage is significantly increased. The

prototype queries the Google GEARS Geolocation API and in addition an own in-door location service being based on a Naïve Bayesian Classifier for determining the exact room within selected campus buildings.

The *Civil Address Provider* (cp. Listing 4) in turn takes the derived geographic coordinates and identifies the civil address (street name, city, country, room; scope: civilAddress) based on an external Web Service query. Our *Weather Provider (Listing 5)* provides the current weather information (raining/sunny/..., humidity, temperature) and a three-day forecast based on various weather service queries. As input requirement it takes either the position or the civilAddress.

The *User Profile Provider* (cp. Listing 6) is realised as JavaEE enterprise application running on a JBoss server. Under the scope *userProfile* attributes such as the user's forename, surname, occupation, birth date, contact details and her devices are available. This provider works in synchronous mode only and is able to associate the entities of the type user to the type imei, thus linking a user to his devices.

The *Environment Provider* is based on a Java EE server component connected via Java RMI to multiple SunSPOT gateways. The association between user and/or device is performed via the room identifier (scope: civilAddress). The mobile spots can measure temperature, light intensity, and noise. The raw noise values are pre-processed by calculating the moving average.

The *Time CxP* (cp. Listing 7) enriches the raw time context (i.e. system time) and calculates the entity's local time and their local season. A spatial equivalent is the *Place CxP* which enriches the raw position by assigning place labels to it. The prototype allows for user feedback. An Android Place Reporter App can be used to mark the landscape and add meaning to various places (e.g. university, shoe shop, supermarket, etc.). Furthermore, labels can be imported from Open Street Map [11]. The provider tries to infer which place label is most relevant for the user. All relevance values sum up to one. This way, the system knows the main business of the place the user is currently located at.

The prototype *Activity Provider* (Listing 8) relies on Bayesian Inference and applies a basic Bayesian Network as user activity model. The outcome is a probability of four selected activities and two exemplar moods (Lst. 11).

```
<contextML><ctxEls><ctxEl> <contextProvider id="DC" v="0.2.1"/>
  <entity id="xxx" type="imei"/> <requestEntity id="john" type="username"/>
  <scope>deviceStatus</scope>
  <timestamp>2010-02-08T16:17:38+01:00</timestamp>
  <expires>2010-02-08T16:47:38+01:00</expires>
  <dataPart>
    <par n="BatteryLevel">100%</par> <par n="BatteryOnCharge">false</par>
    <par n="NetworkType">umts</par> <par n="ServiceState">in service</par>
    <par n="Roaming">false</par> <par n="NetworkState">MOBILE</par>
    <par n="WifiMAC">00:aa:bb:cc:dd:ee</par> <par n="Uptime">06:50:59</par>
  </dataPart></ctxEl><ctxEl>
  <contextProvider id="DC" v="0.2.1"/>
  <entity id="xxx" type="imei"/> <requestEntity id="john" type="username"/>
  <scope>deviceSettings</scope>
  <timestamp>2010-02-08T16:17:37+01:00</timestamp>
  <expires>2010-02-08T16:47:37+01:00</expires>
  <dataPart>
    <par n="Ring Mode">normal</par> <par n="Screen Orientation">portrait</par>
    <par n="Screen Height">480</par> <par n="Screen Width">320</par>
  </dataPart></ctxEl><ctxEl> <contextProvider id="DC" v="0.2.1"/>
  <entity id="8xxx" type="imei"/> <requestEntity id="john" type="username"/>
  <scope>wf</scope>
  <timestamp>2010-02-08T16:24:28+01:00</timestamp>
  <expires>2010-02-08T16:30:28+01:00</expires>
  <dataPart>
    <par n="wfList">00:aa:bb:cc:dd:ee;00:aa:bb:cc:dd:ff</par>
    <par n="wfRSSIList">-83;-84</par>
    <parA n="wfDevices">
      <parS n="wfDevice">
        <par n="wfName">eduroam</par> <par n="wfBssid">00:aa:bb:cc:dd:bb</par>
        <par n="wfType">[WPA-EAP-TKIP+CCMP]</par>
      </parS>
      <parS n="wfDevice">
        <par n="wfName">FH-Osnbr</par> <par n="wfBssid">00:aa:bb:cc:dd:11</par>
        <par n="wfType">[WEP]</par>
      </parS>
</parA></dataPart></ctxEl> <ctxEl> <contextProvider id="DC" v="0.2.1"/>
  <entity id="xxx" type="imei"/> <requestEntity id="john" type="username"/>
  <scope>cell</scope>
  <timestamp>2010-02-08T16:17:38+01:00</timestamp>
  <expires>2010-02-08T16:32:38+01:00</expires>
  <dataPart>
    <par n="cgiList">262-07-31516-12345678</par> <par n="rssiList">-89</par>
  </dataPart>
</ctxEl> <ctxEl>
  <contextProvider id="DC" v="0.2.1"/>
  <entity id="xxx" type="imei"/> <requestEntity id="john" type="username"/>
  <scope>motion</scope>
  <timestamp>2010-02-08T16:25:42+01:00</timestamp>
  <expires>2010-02-08T16:28:42+01:00</expires>
  <dataPart>
    <par n="accSum">1.057</par> <par n="accDelta">-0.001</par>
    <par n="accX">0.042</par> <par n="accY">0.012</par>
    <par n="accZ">1.002</par> <par n="probes">129</par>
    <par n="interpretation">device is lying on table</par>
  </dataPart>
</ctxEl></ctxEls></contextML>
```

Listing 2. Exemplar Context Originating from Android Context Source

```
<contextML><ctxEls><ctxEl> <contextProvider id="PP" v="1.0.0"/>
  <entity id="john" type="username"/><requestEntity id="john" type="username"/>
  <scope>position</scope>
  <timestamp>2010-02-08T16:21:20+01:00</timestamp>
  <expires>2010-02-08T16:26:20+01:00</expires>
  <dataPart>
    <par n="latitude">52.281571</par> <par n="longitude">8.024918</par>
    <par n="accuracy">150.0</par>
  </dataPart>
</ctxEl></ctxEls></contextML>
```

Listing 3. Exemplar Context Originating from Location Provider

```
<contextML><ctxEls><ctxEl> <contextProvider id="CAP" v="0.0.2"/>
  <entity id="john" type="username"/> <requestEntity id="john" type="username"/>
  <scope>civilAddress</scope>
  <timestamp>2010-02-08T16:24:30+01:00</timestamp>
  <expires>2010-02-08T16:29:30+01:00</expires>
  <dataPart>
    <par n="street">Caprivistraße 81</par> <par n="zipcode">49076</par>
    <par n="city">Osnabrück</par> <par n="country">Deutschland</par>
    <par n="countryISO">DE</par> <par n="accuracy">8</par>
</dataPart></ctxEl></ctxEls></contextML>
```

Listing 4. Exemplar Context Originating from Civil Address Provider

```
<contextML><ctxEls><ctxEl> <contextProvider id="WCP" v="0.0.2"/>
  <entity id="john" type="username"/> <requestEntity id="john" type="username"/>
  <scope>weather</scope>
  <timestamp>2010-02-08T16:25:54+01:00</timestamp>
  <expires>2010-02-08T16:30:54+01:00</expires>
  <dataPart>
    <parS n="currentWeather">
      <par n="summary">Mostly Cloudy</par> <par n="tempC">-4</par>
      <par n="humidity">86%</par> <par n="windSpeed">9 mph</par>
      <par n="windDirection">East</par>
    </parS>
    <parA n="forecastedWeather">
      <parS n="weather">
```

```
    <par n="day">2010-02-09</par>          <par n="lowTemp">-7</par>
    <par n="highTemp">0</par>              <par n="summary">Mostly Sunny</par>
   </parS>[…] </parA>
 </dataPart> </ctxEl></ctxEls></contextML>
```

Lising 5. Exemplar Context Originating from Weather Provider

```
<contextML><ctxEls> <ctxEl> <contextProvider id="PCP" v="0.0.2" />
   <entity id="john" type="username" /> <requestEntity id="john" type="username" />
   <scope>userProfile</scope>
   <timestamp>2010-02-08T23:10:11+01:00</timestamp>
   <expires>2010-02-08T23:11:11+01:00</expires>
   <dataPart>
    <par n="username">john</par>          <par n="firstname">Michael</par>
    <par n="lastname">Knappmeyer</par> <par n="gender">m</par>
    <par n="birthday">02.08.1981</par>   <par n="email">m.k@fh-osnabr.de</par>
    <par n="imei">354059020634927</par><par n="phone">+4917638060865</par>
    <parS n="address"> […] </parS>
    <parS n="messenger"> <par n="msn" /> <par n="skype" />          </parS>
   </dataPart> </ctxEl> </ctxEls> </contextML>
```

Listing 6. Exemplar Context Originating from User Profile Provider

```
<contextML><ctxEls><ctxEl> <contextProvider id="TimeCP" v="0.0.1"/>
 <entity id="john" type="username"/> <requestEntity id="john" type="username"/>
 <scope>time</scope>
 <timestamp>2010-02-08T16:25:55+01:00</timestamp>
 <expires>2010-02-08T16:26:55+01:00</expires>
 <dataPart>
  <par n="timezone">Europe/Berlin</par> <par n="localTime">16:25</par>
  <par n="localDate">2010-02-08</par>   <par n="localDayOfWeek">Mon</par>
  <par n="weekend">0</par>              <par n="season">winter</par>
 </dataPart>
 </ctxEl></ctxEls></contextML>
```

Listing 7. Exemplar Context Originating from Time Provider

```
<contextML><ctxEls><ctxEl> <contextProvider id="ActivityCP" v="0.0.1"/>
 <entity id="john" type="username"/> <requestEntity id="john" type="username"/>
 <scope>activity</scope>
 <timestamp>2010-02-08T16:25:55+01:00</timestamp>
 <expires>2010-02-08T16:26:55+01:00</expires>
 <dataPart>
  <parS n="activity">
   <par n="working">0.5946</par>      <par n="shopping">0.2396</par>
   <par n="dining">0.0829</par>       <par n="partying">0.0829</par>
  </parS>
  <parS n="mood">
   <par n="frustrated">0.6650</par>  <par n="happy">0.3350</par>
  </parS>
 </dataPart>
</ctxEl>
</ctxEls></contextML>
```

Listing 8. Exemplar Context Originating From Activity Provider

V. CONCLUSION

In this paper we presented a light weight XML based approach for modeling context information and for encoding context management messages. In our testbed we proved that ContextML can be applied for processing and providing high-level context and organizing context in various abstraction layers from primitive scopes up to high-level scopes.

One of the strong features of this representation scheme is the possibility of adding new scopes gradually and extending the domains of the context provisioning system. New providers can be added in plug and play fashion and evolving applications can be supported in their entire lifecycle. While the use of ContextML schema provides a common structural definition of context representation, it lacks the semantic strengths of OWL or RDF based ontologies which are used in some context representation models. ContextML is used for light weight representation ensuring fast processing also on resource constraint mobile devices. However, defining ContextML as schema for exchanging context between various network entities does not hinder specific high-level context providers from applying more sophisticated knowledge representation models, e.g. based on ontologies.

Further, ContextML benefits from the advantages of object oriented context modelling. In a way, a scope can be interpreted as class embracing various (elementary, arrayed or structured) attributes. From a different angle, one context scope can be seen as the attribute of an entity. Entities are related to each other by considering their specific type (imei, username, etc.).

ContextML parsers have been developed for two mobile device platforms (Android and iPhone) and a number of context providers and consumer applications for these mobile platforms are being tested including a Shopping Mall scenario, a Party scenario and the Train station scenario, all of which utilize ContextML for producing and consuming contextual information related to context scopes mentioned earlier in this section. In these scenarios, the C-CAST context provisioning framework was applied for providing multimedia content recommendation, tailored advertisement and social networking purposes. The interested reader is referred to [7] for details.

REFERENCES

[1] T. Strang, and C. Linnhoff-Popien, A Context Modeling Survey, The Sixth International Conference on Ubiquitous Computing, Nottingham, UbiComp 2004

[2] M. Baldauf, S. Dustdar, and F. Rosenberg, A Survey on Contextaware Systems. International Journal of Ad Hoc and Ubiquitous Computing, 2(4), 263-277

[3] H. Chang , S. Shin and C. Chung, Modeling Context Life Cycle for Building Smarter Applications in Ubiquitous Computing Environments,LNCS Volume 5333/2009, Springer Berlin / Heidelberg, ISBN: 978-3-540-88874-1, Pages 851-860

[4] M. Zafar, N. Baker, B. Moltchanov, J.M. Goncalves, S. Liaquat and M. Knappmeyer, Context Management Architecture for Future Internet Services, ICT MobileSummit 2009, Santander, Spain. June 2009.

[5] B. Moltchanov, M. Knappmeyer, C. A. Licciardi, "Context-Aware Content Sharing and Casting, 12th ICIN, Bordeaux, France, October 2008

[6] M. Knappmeyer, R. Tnjes, N. Baker, "Modular and Extendible Context Provisioning for Evolving Mobile Applications and Services", ICT Mobile Summit 2009, Santendar, Spain, June 2009

[7] Context Casting (C-CAST), FP7 ICT Research Project, Web Site: http://www.ict-ccast.eu/

[8] Fielding, R. T., Architectural Styles and the Design of Network-based Software Architectures, PhD Thesis, University of California, Irvine, 2002.

[9] M. Valla, C. Fra', W. L. Goix, M. Marchetti, E. Paschetta, A. Salmeri: "Architettura e moduli della Context Awareness Platform", Telecom Italia Lab - Technical Report, DPC2006.01818, December 2006

[10] ContextML XML schema – available on: http://contextml.tilab.com

[11] OpenStreetMap Project, http://www.openstreetmap.org