Abstract—Context representation is a fundamental process in developing context aware systems for the pervasive world. We present a lightweight XML based context representation scheme called ContextML in which context information is categorized into scopes and related to different types of entities. Provision is made for addition of meta-information to improve the usability and quality of context information. Explanation of the ContextML schema is provided with the help of real-world examples.

Keywords: Context Representation; Context Awareness

I. INTRODUCTION

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 unambiguosity, (6) be applicable to existing infrastructures and frameworks.

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 lightweight 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. Some examples of ContextML are presented in Section IV and we conclude our discussion in Section V.

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 Consumer (CxP) and queries the CxP directly.

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).

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. Entities

In the C-CAST system, every exchange of context data is associated with a specific entity, which can be a complex group of more than one entities. 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. Every Context Provider (CxP) supports one or more entity type and this information is published to a Context Broker (CxB) from which Context Consumers (CxC) can query context. A type is an a category of entities; for example entity types are *username* (for human users), *imei* (for mobile devices), *mobile* (GSM phone number), *SIP uri* (for SIP accounts) 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.

B. Scopes

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.

For example, consider scope ‘position’ which refers to the geographic location of an entity. This scope could be composed of attributes latitude, longitude and accuracy and these are always changed at the same time. Updating the latitude value without updating longitude, if is changed, and vice versa 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](image1)

Every scope instance (context information) that is exchanged is tagged with a specific timestamp (time of context detection) and an expiry time. The expiry time tag states the validity of the scope. After this time, the information is considered invalid. Fig. 2 shows the schema for defining a scope. It consists of scope name, *url* where the scoped context can be requested from, types of entities it supports, inputs that are required to request context and dependency on other scopes (*depUrl*).

![Figure 2. ContextML scope schema element](image2)

A context consumer can find out how to get particular context information by requesting a context broker. The Context broker can provide this lookup service as all context providers in our system register with a broker by sending a ContextML message, depicted in Fig. 3, informing a broker about the CxP, how to access it (*urlRoot*) and what scopes it supports (*scopes*).

![Figure 3. ContextML CxP advertisement schema element](image3)
C. Context Data

Whenever a context consumer requests or subscribes to a specific context scope, it receives a response encoded in ContextML element ‘ctxEl’ when context is available. ctxEl contains information about where the context is generated (contextProvider), which entity it is related to (entity), what scope it belongs to, validity duration (timestamp, expires) and the actual context data in the dataPart element. A graphical description of this element is given in Fig. 4. The elements par, parS and para are simple constructs to store name-value pairs and attributed collections (arrays and structs).

D. Context Query

Context consumers can formulate a simple context query for a particular context scope by sending a ContextML message to the context provider or invoking the context provider through HTTP and encoding the request parameters in the HTTP URL. Figure 5 depicts the structure of a context query that requests weather information related to a user in a particular location.

IV. EXAMPLES

This section describes ContextML with the help of an example where a context provider announces its capability of providing weather context and a context consumer requests such a context from this provider.

A. Weather and Position Scope

Listing 1 shows the announcement of a weather CxP. Note that it contains a strict requirement of specifying latitude and longitude (from position scope) when requesting weather context.

```
<ctxAdv>
  <contextProvider id="WCxP" v="0.0.1" />
  <urlRoot>http://localhost/WeatherCP</urlRoot>
  <scopes>
    <scopeDef n="weather">
      <url>/getWeather</url>
      <entityTypes>username</entityTypes>
      <inputDef>
        <inputEl name="lat" type="position:latitude" />
        <inputEl name="lon" type="position:longitude" />
      </inputDef>
    </scopeDef>
  </scopes>
</ctxAdv>
```

Listing 1. ContextML based announcement from a CxP specifying where and how to request weather context

B. Weather Context Response Data

When a consumer wants to request weather context, the announcement from weather CxP will show the requirement of providing position information. The consumer will therefore request position information from a context provider that has advertised the scope position. The position context response from such a CxP is shown in Listing 2.

```
<contextML>
  <ctxEl>
    <contextProvider id="gLCB" v="2.2_3"/>
    <entity id="35xx59x219xxx45" type="imei"/>
    <requestEntity id="johnsmith" type="username"/>
    <scope>position</scope>
    <timestamp>2009-12-14T16:48:33+01:00</timestamp>
    <expires>2009-12-14T17:19:33+01:00</expires>
    <dataPart>
      <par n="locMode">GPS</par>
      <par n="longitude">7.67159035</par>
      <par n="accuracy">102.0</par>
      <par n="latitude">45.1118874</par>
    </dataPart>
  </ctxEl>
</contextML>
```

Listing 2. Position context information from a CxP executing on a mobile device

C. Weather Context Query

With the input requirements met, the context consumer can now invoke the weather context provider to retrieve the weather information about a specific location. Listing 3 shows the request along with a constraint on validity of the context information.

```
<contextML>
  <ctxEl>
    <contextProvider id="gLCB" v="2.2_3"/>
    <entity id="35xx59x219xxx45" type="imei"/>
    <requestEntity id="johnsmith" type="username"/>
    <scope>position</scope>
    <timestamp>2009-12-14T16:48:33+01:00</timestamp>
    <expires>2009-12-14T17:19:33+01:00</expires>
    <dataPart>
      <par n="locMode">GPS</par>
      <par n="longitude">7.67159035</par>
      <par n="accuracy">102.0</par>
      <par n="latitude">45.1118874</par>
    </dataPart>
  </ctxEl>
</contextML>
```

Listing 3. Weather context request from a CxP to a context consumer
The response of this query containing the weather context about a particular location is shown in Listing 4.

```
<contextML>
  <ctxEls>
    <ctxEl>
      <contextProvider id="WCP_SLK" v="0.0.2" />
      <entity id="johnsmith" type="username" />
      <scope>weather</scope>
      <timestamp>2009-10-05T16:15:00+02:00</timestamp>
      <expires>2009-10-05T17:00:00+02:00</expires>
      <dataPart>
        <par5 n="currentWeather">
          <par n="summary">Mostly Cloudy</par>
          <par n="tempC">26</par>
          <par n="humidity">30%</par>
          <par n="windSpeed">4 mph</par>
          <par n="windDirection">North</par>
        </par5>
        <par8 n="forecastedWeather">
          <parS n="weather">
            <par n="day">2009-10-04</par>
            <par n="lowTemp">8</par>
            <par n="highTemp">21</par>
            <par n="summary">Chance of Rain</par>
          </parS>
        </par8>
      </dataPart>
    </ctxEl>
  </ctxEls>
</contextML>
```


V. CONCLUSION AND FUTURE WORK

We have presented a brief overview of ContextML schema for representing context information. Context is categorised in scopes which are related to different entity types. Currently our system has definitions of following scopes: calendar, civilAddress, environment, movement, myPlaces, nearbyPlaces, nearbyVotedBuddiesPoi, position, socialNetwork, socialPreferences, userProfile, userProximity, usualPlaces weather, userGroup, userActivity and situation. Some basic scopes are aggregated with the help of a rule based reasoning engine to generate complex scope information e.g. situation context.

One of the strong features of this representation scheme is the possibility of adding new scopes gradually and extending the domain of the context provisioning system. New providers can be added and consumers can start using new context by updating the ContextML schema. 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 provider and consumer applications for these mobile platforms are being tested including location, weather, Shopping Mall scenario, 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.

We are in the process of extending ContextML schema by modeling event subscription and notification constructs. This will allow the C-CAST system to handle subscriptions to context events and allow providers to notify context consumers.

REFERENCES