The cGML: a XML language for mobile cartography.

Roberto Demontis, Emanuela de Vita, Andrea Piras, Stefano Sanna
CRS4, Center for Advanced Studies, Research and Development in Sardinia
Edificio 1, Loc. Piscinamanna, Polaris
09010, Pula (CA), Italy
Tel.: +39-070-92501
{ demontis, emy, piras, gerda}@crs4.it

ABSTRACT

Increasing processing power and storage capabilities encourage systematic adoption of high-end mobile devices, such as programmable cellular phones and wireless-enabled PDA to implement new exciting applications. The performances of modern mobile devices are bringing innovative scenarios, based on position awareness and ambient intelligence paradigms. The market is moving from old “Wireless Applications” approach to "Mobile Computing", which aims to exploit mobile host capabilities. This paper presents the compact Geographic Markup Language (cGML), an XML-based language defined to enable design and development of LBS applications specific for mobile devices, and an example of client-server architecture using it.

1 INTRODUCTION

As Internet services become pervasive in both for consumers and businesses, people ask for mobile access to e-mail, Web, instant messaging and multimedia delivery systems. Increasing processing power and storage capabilities encourage systematic adoption of high-end mobile devices, such as programmable cellular phones and wireless-enabled Personal Digital Assistant (PDA), to implement new exciting applications. Early mobile applications had been designed as downscaled browsers, which tried to port the desktop experience to the constrained device of the mobile phone. However, performances of modern mobile devices are bringing new innovative scenarios, based on active agents, position awareness and ambient intelligence paradigms. The market is moving from old “Wireless Applications” approach to "Mobile Computing", which aims to exploit mobile host capabilities. Basically, Mobile Computing refers to the ability of applications to run locally on the mobile device even without continuous network connection. Access to remote server is required to get updated information or to store data gathered on the move.

Position awareness is a key element for query data which strictly depends on user geographical context. The offer of navigation systems and electronic tourist guides is growing and market results show that mobile users look for geographic information and location-aware services, mobile cartography applications and context-sensitive data provisioning. Location Based Services (LBS) promise to be the basis for next year killer application for mobile devices. LBS implementation requires careful selection of standards and protocols, in order to guarantee interoperability and reuse of large existing databases. From the developer’s point of view, there are three main enabling technologies for LBS: connectivity, localization and visualization. Connectivity comprises remote and local resource access. Widely adopted GPRS infrastructure and upcoming UMTS network provide reliable yet fast wireless IP connectivity, so that designer can find a effective tradeoff between fully independent local data storage and browsing-oriented remote data access. At the same time, standards such as USB OTG (On the Go) and Bluetooth allow modularity and interconnection of small and inexpensive modules (phones, storage, multimedia). Localization is a key factor for LBS implementation, because it imposes main constraints for data filtering. Positioning systems like GPS, AGPS and cell-based are getting cheaper and easily embeddable in consumer devices. Finally, high resolution color displays provide effective visualization for images, maps and complex data.

On top of these enabling technologies, software frameworks support application development, by means of standard interfaces and protocols. Mobile devices running Symbian and Windows Mobile operating systems are widely supported by developer communities. On top of these systems, Java2 MicroEdition is market leader cross-platform runtime environment, especially on mobile phones area. Although hardware and software framework technologies can be considered mature and ready to implement mobile side of LBS applications, a standard language for geographical information encoding tailored for mobile devices is still missing.

2 LOCATION BASED SERVICE

An important business is focused onto provide to mobile users the right information, at the right position, at the right time, for the right context and therefore specific services are requested for specific purposes. In particular, two aspects must be considered: the functionalities and the market of niche product [1]. They are the objectives of LBS. In fact, it is a service able to handle geographical or geographical related information based on position.

The spread of PDAs and smartphones stimulates the development of LBS tailored on them. A lot of them have or will have a GPS module that let to automatically gather the user position without specify area, street, city names.

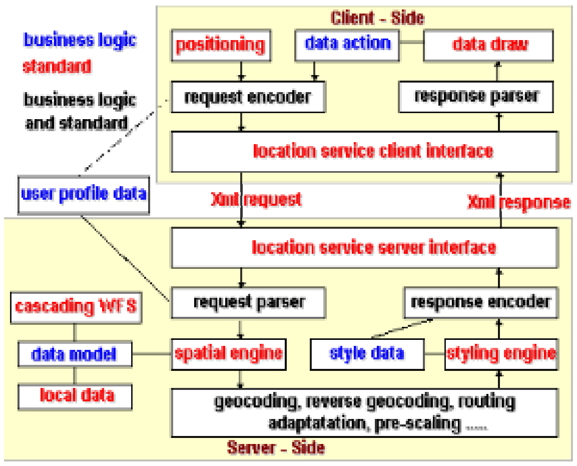

Figure 1. A possible mobile LBS structure.

The use of the user position to determine the "right information" introduces the biggest topic for LBS: the lack of privacy security system that guarantees the protection of personal data. The solution is the use of user position inside the mobile devices and to provide it outside only when required. In this case, it is not possible to use a Location Server to provide the user position, by using cell sector triangulation or other techniques, to requiring applications. Regarding to the "right time", it is usually considered the instant when the user makes his request but LBS capable to predict future locations and prepare information are under development [2]. Finally, the "right context" is referred to the user context (device constraints, user's task). There are mechanisms available to fit the system to the current usage situation adapting it to the changing context [3].

To improve the LBS performance, the number of functionalities processed on mobile devices is increased but device constraints influence the service functionalities such as data format specification, data transferring and its visualization on client.

In last year, standards based on XML language for LBS were defined. Examples of them are the specification of the GeoMobility Server and the XML for Location Service (XLS) proposed by OpenLS [4]. The last one is derived from the concept of interoperability in GIS application based on the Geographic Markup Language (GML), the Web Features Server (WFS) and the Web Map Server (WMS). But the most relevant field of study for LBS is how the right information has to be showed. Usually, the response is delegated to the mobile cartography.

3 MOBILE CARTOGRAPHY AND DEVICES

Visualization properties of a cartographic application depend on data subset to represent and user expectations. By means of (implicit) user profile and (explicit) device profile, nomadic users can collect all pieces of information they need to make decisions regarding their activities (i.e. during tourism vs. work traveling) with reference to proximity of area of interest (AOI) and points of interest (POIs). The quantity and quality of the information provided to mobile devices depend on hardware capabilities. This section provides a brief overview about the LBSs, the mobile device constraints and some approaches to provide maps in mobile devices.

3.1 Device and runtime environments

The design and development of mobile applications are different from standard desktop approaches. Hardware and software constraints influence overall architecture and programmers have to balance performance requirements with actual limitations of host device. Hardware constraints are in terms of processing power, memory, user interface and connectivity. Processing power is strictly related to power consumption and battery life, and it determines what kind of computation can be performed locally. CPU-intensive algorithms may lock the device or come to an out-of memory error before completing task. On the other hand, although memory modules become small and cheaper, memory allocation has to be carefully managed. In fact, new multimedia extensions, such as cameras and video recorders, require the device to store large amount of data and keep it until the user will download it to a PC.

The user interface imposes other limitations that developers have to deal with:

- Display size and resolution are a fraction of modern PC screens, with reduced color palette;
- Most devices lack a (even reduced) QWERTY keyboard; user interaction is based on numeric keypad with multi-tapping character selection or, sometimes, on pen pointing.

The user himself imposes limitations. In fact, mobile devices are being used by non technical people and a "typical customer" can be hardly defined. Moreover:

- Predictive text assistant (for faster text input on numeric keypad) can be difficult to learn and use for senior people;
- GUIs must address the non-homogeneous users experience.

Mobile device connectivity is based on wireless networks and on-the-field operations are usually performed by means of packet switching cellular networks. Connectivity performances depend on network technology in use (GPRS, EDGE, and UMTS) and network coverage (inland or urban area). Moreover, wireless connectivity is often discontinued and network congestion may affect performances. Therefore, application designers try to limit the data size transferred using compression and data splitting techniques.

To split data over a wireless discontinued channel means "to split information", in order to transform information in an aggregate of small, atomic and self-contained items. For instance, considering one map as one single object, it can be split in small sets of bytes and recomposed to be able to process its information ("byte packaging") or it can be split in small map items, that can be separately processed and each one contains a part of the information of the whole map ("information packaging"). This approach is a valid solution for discontinued connection issues: since every data fragment is atomic and self-contained, there is no need to have a continuous network connection; any network failure will affect only current data item, while previously received items can be processed and shown to the user.

Software constraints reflect hardware limitations. More devices are equipped with low consumption processors, which provide poorer performance than desktop CPUs. Runtime memory is usually limited to 10 MB, which is shared with multimedia applications and messaging utilities.

Other constraints are being introduced by portability requirements. There two main software layers involved in application design and development: operating systems and runtime environments. Mobile device market leaders are PalmOS, Windows Mobile and SymbianOS. They provide state-of-the-art native development environments, based on C++ programming language and a code library tailored for mobile applications. However, the application programming interfaces (APIs) provided by runtime environments for mobile devices are poorer than their desktop counterpart and focused on the requirements for working on small displays and on the different strategies about GUI usability. System independent runtime environments such as Java2 MicroEdition (J2ME), In-fusio, Ewe are widely adopted for both consumer and business applications.

In our work, we chose J2ME platform because it is cross-platform, it does not require agreement to deploy applications and it is not related to a specific implementer. Mobile Information Device Profile (MIDP) is the leading technology for enhanced programmable mobile phones and provides a rich API for connectivity, visualization and localization modules.

3.2 Approaches

There are mainly three approaches to mobile cartography people can find in commercial products: one based on stand-alone architecture and two on client/server architecture.

The first approach requires installing a native application and related data, generally defined in a proprietary format, on the client device. Such an approach does not require any Internet connection and it is able to show only maps based on preloaded data (i.e. maps, points of interest).

The second approach can be defined "Mobile Browsing". The user needs to have a mobile device with an Internet browser to access raster and vectorial maps. However, interactivity implies a new connection with the remote server, because no computation is expected to be performed locally. Such an approach requires a permanent Internet connection to submit queries (zoom, path) and get results but it could be expensive.

The last one is defined "Mobile Computing", where increasing performances of upcoming mobile devices permit to run smart applications locally on cellular phones and PDAs. The client is able to use the device to both visualize a map and manage and elaborate data locally. In this case the client does not need a permanent connection.

cGML research activity fits into mobile computing area and it has been designed to be part of web services technology and XML-based languages.

4 THE GEOGRAPHY MARKUP LANGUAGE

The Geography Markup Language (GML) has been adopted as "de facto" standard based on XML language for encoding geographic information. It has been defined by the Open GIS Consortium to encode vector geographical information together with metadata on spatial and non-spatial resources.

The GML defines a set of concrete and abstract element but not a root element. To use GML data in a valid XML document, it is required to develop a GML application schema that fixes a vocabulary for a particular domain by defining and describing the terms of such vocabulary. An application schema declares new elements and attributes in its own namespace using types, attributes and elements defined by GML to give vocabulary-specific names to their content models.

There are three main GML releases and other three subversions: GML 1.0 [5], GML 2.0 [6], GML 2.1.1 [7], GML 2.1.2 [8], GML 3.0 [9] and GML 3.1[10].

The first two main releases and related subversions are based on the OpenGis Abstract Specification [11] and have the objective to encode geographic information in XML format in way to make possible modeling, storage and transport using simple geographic features defined as features containing geometric properties. Each geometric property contains a set of two-dimensional coordinates of its vertexes.

Version 3.0 introduces several new features and concepts to improve and extend the GML capabilities. The GML is not only based on the OpenGIS Abstract Specification but also on the ISO 19100 Series of Geographic Information Standards [12] specification and it becomes from an XML encoding to an XML grammar for the manipulation of geographic information. The set of geometric primitives is enriched by new numerous one- and two-dimensional primitives and by the introduction of the three-dimensional ones. Examples of new one- and two-dimensional geometric objects are curves and their segments, arcs, Bezier curves, geodesic curves, triangles, rectangles, surfaces and gridded surfaces like spheres, cones and cylinders. We have solids in the three dimensional geometry.

New sets of geometric elements allow defining aggregations and composite geometries. The new primitives let available a wide set of items to have as much as possible close representation of the real world.

Same interesting features included in GML 3.0 and GML 3.1 are:

- The definition of coordinate reference systems and coordinate operations; topology; temporal information (including temporal reference systems, temporal topology and geometry);
- Dynamic features representation;
- The possibility to insert definitions and dictionaries, unit of measure, measuring systems, direction (orientation, direction, heading, bearing or other directional aspects of geographic features);
- Observation feature to describe the information related to a capture event and the value for the result of the observation, coverage model and representations.

Although several default styles are defined for top-level elements, such as Feature, Geometry, Topology and Label, the styling description may be completely ignored because GML has the strict separation of data and presentation and the internal styling mechanism is been though as a separate model that can be "plugged-in" to a GML data set.

The common mechanism to obtain a map from GML data is to convert it into an XML graphical format such as VML [13], X3D [14], SVG [15] or Mobile SVG [16], using standard XML transformation facilities (XSLT [17]). Usually, the GML is translated in SVG for desktop applications (see [18] e [19]) and Mobile SVG for mobile devices ones. Mobile SVG is a subset of SVG's elements, attributes and events selected for being used in mobile devices.

There are commercial and open source tools to translate GML in SVG and for SVG/Mobile SVG rendering. Some renders are natively embedded into web browser (i.e. Mozilla [20]), some distributed as plug-ins for many browsers (i.e. Adobe [21] and Corel [22] SVG Viewer) and some available as stand-alone viewers (i.e. Apache Squiggle [23]).

4.1 Some drawbacks to use GML in mobile devices

The direct use of GML for mobile devices is not possible. In fact, processing and saving GML data require a considerable memory space which cannot be provided by the major part of mobile devices. Furthermore, the transfer of GML documents of hundreds of kilobytes through unreliable wireless connection is expensive and boring for users. Thanks to the spread of UMTS in Europe, the bandwidth for smartphone is at least of 358 Kbps, so it could be possible to download data as the same speed as the first releases of ADSL connections. But the UMTS has not the same geographic coverage of the GSM network: it is available only in city areas. To be used on the mobile network slower then UMTS, the GML document could be compressed before the transfer but the decompressing process in mobile device requires considerable processing power and extra memory capabilities.

Another drawback is related to the coordinate format. The coordinates express geographic locations with high definition and with more details respect to actual user needs and visualization capabilities of device.

Finally, projecting and scaling geographic data can be impracticable on small appliances, since many devices (i.e. all MIDP-compliant mobile phones) have not a native support for floating-point math.

5 THE COMPACT GEOGRAPHIC MARKUP LANGUAGE

The considerations exposed in previous paragraph and the wish to eliminate the translation from GML to SVG in favor to draw directly the geometric data and keep the geographic information that disappear with that translation, driven us to define the compact Geographic Markup Language (cGML).

5.1 Language features

First of all, it is important to say the cGML is inspired by GML 2.1.2 so all the references of GML items in these sections must be considered related to this version and the features introduced by versions 3 are not included in the current cGML release.

The word "compact" in the cGML definition summarizes some its key aspects. We consider the cGML a compact version of the GML because it uses shorter elements names, removes elements without attributes that have only the target to contain other elements, reduces the number of available attributes. Although the cGML elements names are shorter than GML ones, they preserve the human-readability feature of the XML documents because their names are chosen in way to make possible to guess their related words (see **Table 1**).

To make possible that a cGML document could be a stand-alone XML document, we have introduced the `cGML` root element and defined concrete elements based on several GML's abstract ones.

The correct sequence of cGML children elements starts with `Head` followed by features and/or feature collections elements.

Head and its children elements define the spatial reference system (SRS) and the viewport size where the geometric information will be drawn. The SRS provides indication regarding coordinates transformation from the real world to the cartographic data and it is specified by the value of the `srsName` attribute in the `RealBox` element. Respect to GML, the SRS is unique in the cGML and it is applied to all geometry primitives. The `RealBox` value is the coordinates of the interest area expressed by integers.

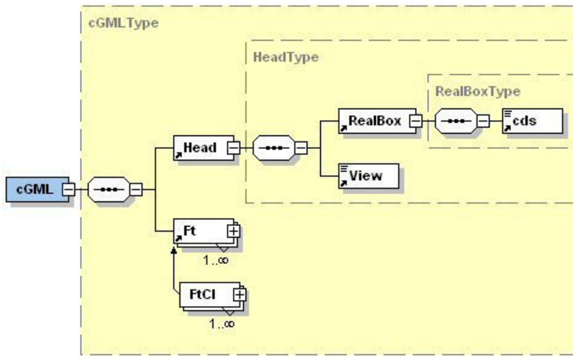

Figure 2. Structure of a cGML document.

The `View` element contains the view size of the device screen. Its `zoom` attribute is the scale factor between the real box and the view and represents the inverse of the number of units expressed in the SRS associated to one pixel.

In the following example of cGML document, the Head block is contained in rows 3 – 8. The value of the `srsName` attribute is `EPSG:32632` and it is the European Petroleum Survey Group coordinate reference system code [24] for the UTM (WGS84) Zone 32 North CRS.

The information in the `Head` element allows determining the scale factor between the geographic coordinates of the AOI and the coordinates expressed in pixels of the view area. Furthermore, for obtaining the real coordinates of one point of the view, it is necessary to consider that the plotting plane is mirrored respect to horizontal axis respect to the real plane. Defining the view area as (0, 0, width, height) and the real box area as (bx1, by1, bx2, by2), the real coordinates Preal(x, y) of a point Pview(xv, yv) in the view are defined by the rules:

$$x = \text{round}(zoom * xv) + bx1$$

$$y = \text{round}(zoom * (height - yv)) + by1$$

Using the values in the following example of the `Head` code, one pixel represents approximately 3.75 meters and the real coordinates Preal(x, y) for the point Pview(103, 211) of the view are:

$$x = \text{round}(0.2667 * 103) + 510775 = 510803$$

$$y = \text{round}(0.2667 * (400 - 211)) + 4339616 = 4339666$$

```
 1: <?xml version="1.0
          encoding="UTF-8"?>
 2: <cGML
     xmlns:xsi="http://www.w3.org/
     2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation=
     "cgml_base.xsd"
     version="1.0">
 3:   <Head>
 4:    <RealBox
         srsName="EPSG:32632">
 5:          <cds>510775,4339616
                  512275,4341116</cds>
 6:    </RealBox>
 7:    <View zoom="0.2667">
              400,400</View>
 8:   </Head>
 9:   <FtCl id="stuffA"
            name="figures">
10:      <Ft id="x100" name="zxwy">
11:         <LnSt>
12:            <cds>267,39
                   276,42</cds>
13:          </LnSt>
14:          <info>info for feature
               x100 in stuffA</info>
15:      </Ft>
16:      <Ft id="z54">
17:          <LnSt>
18:            <cds>326,64 353,78
                   396,102</cds>
19:          </LnSt>
20:          <info>info for feature
              z54 in stuffA</info>
21:      </Ft>
22:    </FtCl>
23:    <Ft id="target" name="peak">
24:           <Point>
25:              <cds>84,55</cds>
26:           </Point>
27:    </Ft>
28: </cGML>
```

Code 1. Example of cGML document. Rows are numbered to simplify references.

Starting from the abstract GML feature and feature collection elements (`_Feature` and `_FeatureCollection`), cGML defines concrete and simpler features (`Ft` and `FtCl` respectively), while the `featureMember` element has been removed. The optional attribute `fid` of the GML feature is substituted by the required `id` attribute. The optional child element description is renamed in `info`, while name becomes an optional attribute of Ft and FtCl. In the following example of cGML document, there is one feature collection, with the id equals to stuffA (rows 9 – 22), containing two features, with the id attributes are x100 (rows 10 – 15) and z54 (rows 16 – 21), and another feature having the id attribute equal to target (rows 23 – 27).

GML	cGML	Characters saved
_FeatureCollection	FtCl	78%
_Feature	Ft	75%
Description	Info	64%
LineString	LnSt	60%
LinearRing	LnRn	60%
Polygon	Plgn	43%
MultiGeometry	MlGeo	62%
MultiPoint	MlPoint	30%
MultiLineString	MlLnSt	60%
MultiLinearRing	MlLnRn	60%
MultiPolygon	MlPlgn	50%
outerBoundaryIs	ex	87%
innerBoundaryIs	in	87%
Coordinates	cds	73%

Table 1. Comparison between some GML and cGML tags.

The cGML feature elements contain the elements for defining the geometries. They are defined in two-dimensional SRS and use linear interpolation between coordinates. The primitive and aggregate geometry elements are the same for cGML and GML 2.1.2 except for the GML Box and the cGML Arc, MlArc (multi-arc) and MlLnRn (multi-linear ring). The names of geometry elements are compacted in shorter ones, following the cGML philosophy, and the GML geometry members are not used.

GML allows defining the coordinates of the vertexes of the geometries in two manners:

- A sequence of coord elements (that encapsulate the X, Y and Z elements)
- A single string contained within a coordinates element.

For cGML, the second encoding system has been chosen: it allows reducing the number of characters for expressing the same coordinate list. The name coordinates is modified in the shorter cds and its value is one sequence of tuples separate by one or more spaces. Each tuple is composed from one to four integers separated by commas. Furthermore, the cGML coordinates are related to the screen device. Rows 5, 12, 18, 25 of previous cGML document are examples of cds values.

5.2 The XML Schemas

The cGML is defined by three XML Schema files: cgml_base.xsd, cgml_feature.xsd and cgml_geometry.xsd. The first file contains the specification of the cGML root element, the sub-tree related to the Head element and defines the sequence of features and feature collections. It is the file name specified in the xsi:noNamespaceSchemaLocation attribute of the cGML documents. Row 3 of the previous example of cGML document depicts the start tag of the root element. cgml_feature.xsd file defines the structure for Ft and FtCl while cgml_geometry.xsd specifies the list of geometry primitives and the coordinates. The files are downloadable from the cGML support site [25].

The cGML is thought to be able to work as a stand-alone XML document and so its three XML Schema files are required for defining one cGML instance. Thanks to its modularization in three base schemas, it is possible to define application schemas that use only cGML geometries or cGML features and geometries importing the required subsets. The modularization is a feature common to the cGML and the GML.

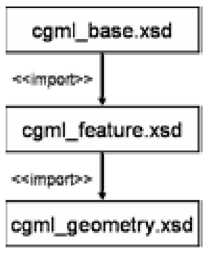

Figure 3. cGML schema dependencies.

5.3 Advantages for application development

The compact notation and the deletion of some elements produce make concrete to have cGML documents an average of about 64% shorter then corresponding GML ones. It provides several advantages in the data manipulation on mobile devices:

- The document transfer on wireless networks requires less time and it is cheaper for user;

- Using a static and well-known vocabulary allows the development of optimized parsers;
- No compression on server side and decompression on client side are required. The second one is an operation power processing consuming in device with limited capabilities;
- Decrease the depth of Document Object Model (DOM) tree allowing reducing the memory space required by cGML parser for their processing and it can work in entry-level smartphones too.

6 A USE CASE

In this part of the paper we describe a service that exports geographic data, encoded in cGML format, provided from a server powered by a Web Feature Server to a Java client.

6.1 The scenario

The service is a task-centered LBS which gives information about pressure and water flow of the conduits in the aqueduct system of Cagliari city. The scenario's actors are mobile workers and the control center.

The goal of the service is to serve real-time information to the workers through their mobile phone in order to solve emergency situations due to breakdowns. The mobile worker can communicate with the control center by phone call, SMS, or using the client application installed in his device. Furthermore, the worker is able to move in the working area and the application does not require a continuous connection with the server: it downloads information only when necessary and uses the last data received between a request and the following.

The same cGML data is used to describe the situation on worker phone and on PCs of the control center. The data showed in the client side includes static and dynamic information. The static data refers to the Cagliari city viability and permits to calculate minimal path between locations. The dynamic data is the conduit path and its values in terms of pressure and flow. These real-time data is used by the worker to locate the breakdown.

6.2 The server side

The server gathers various kinds of data divided in geographical feature and their no-geographical attributes. All of these data items are referenced by name and contains planar coordinates.

Data processing is performed by means of a predefined GML model stored in the GeoServer [26], a full transactional Java implementation of the WFS specifications. The data returned by the WFS are in the GML format and refers to the UTM (WGS84) Zone 32 North CRS.

A client can work with the server using the HTTP protocol through GPRS connection. Each request contains the specification of the new features context based on device screen size (the `View` element) and on the AOI (the `RealBox` element) and the SRS name.

```
<?xml version="1.0"
      encoding="UTF-8"?>
<Request>
  <RealBox srsName="EPSG:32632">
    <cds>510505,4339272
       512871,4341638</cds>
  </RealBox>
<View>400,400</View>
</Request>
```

Code 2. An example of client request.

When the server receives a request, it performs the following transformation operations on data model items:

- Clipping. The server sends a request to the WFS which contains the AOI and its response is a selection of the data in the database. This response is composed by generic XML data for the non-geographical attribute and GML data for the geographical attribute.
- Translation. The XML data returned by the WFS is translated to cGML through XSLT processing. The result of such a transformation is used to define cGML geometric elements and their attributes. Final cGML document is completed by adding the `Head` block. Each geographic feature is related to one cGML feature element by the value of their `id` attribute.
- Scaling. The real coordinate values are adapted for plotting in the viewport deleting collapsed or overlapped lines. The selected geographic data items are projected and scaled by the server because the runtime environment for smartphones has not a native support for floating-point mathematical operations.

The several parts of the server side are developed in Java.

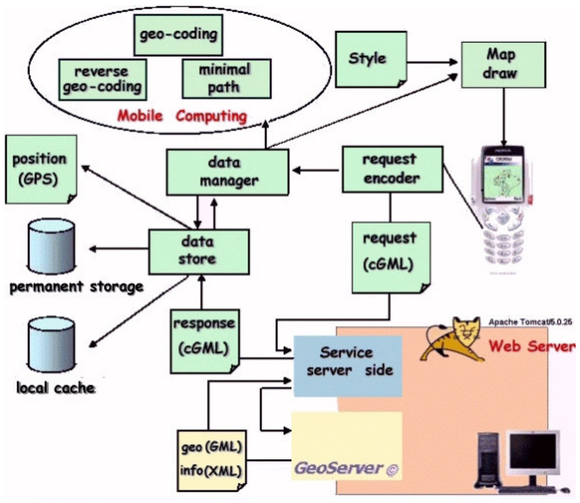

Figure 4. The architecture implementing the scenario.

6.3 The cGML viewer

Java technology allows next-generation devices to offer new capabilities such as enhanced interactivity, rich user interface, off-line processing, local data storage, and networking. J2ME provides an application environment that specifically addresses the needs of cellular phones, thanks to its MIDP.

The viewer composes a request (like **Code 2**) specifying the AOI based on the current user position. Such a position is retrieved by a Bluetooth GPS hardware connected to the device. The request is sent to server establishing an HTTP connection and it returns the cGML map file. The viewer performs calculations and display information locally reducing interaction over the wireless network. It parses the cGML document using the kXML SAX parser [27] and instances objects to manage the `Header`, `Ft` and `FtCl` elements and geometry information. These objects are the input of the plotting routines. Plotter has been optimized to reduce allocation of new objects: double buffering and flyweight pattern to share a common set of elementary components.

The viewer exposes the common set of operations for managing maps. For example: map saving/loading in/from device memory or downloading from a server; zooming; panning; searching a point of interest by name; marking POIs by means of virtual cursor; the visualization of the minimum path and the distance (by meters). Other features have been implemented to improve user accessibility. One of them consists of writing the label related to selected object (i.e. locations name) in the upper corner of the display with horizontal text, providing good experience even in bad light conditions. The application can process these operations in offline modality without downloading other cGML documents, according to the mobile computing paradigm.

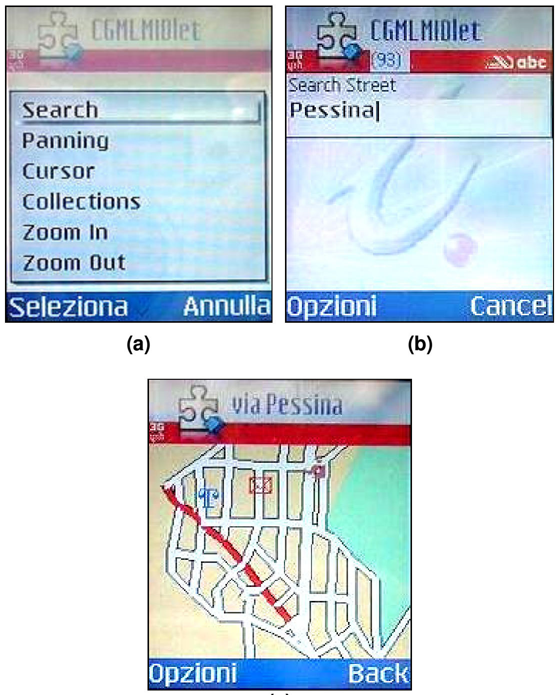

Figure 5. The viewer running on Nokia 6630. The three images show how finding a street by name. Image (a) shows the list of available functionalities. Image (b) displays the GUI where inserting the name of the street. While image (c) shows the map with the street highlighted.

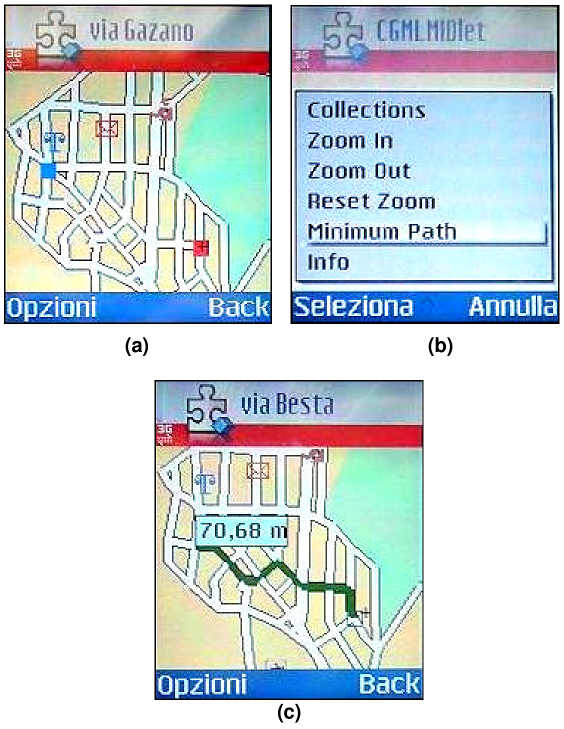

Figure 6. Three images about the minimum path between two POIs. Image (a) shows the two points (red and blue) selected by the virtual cursor. In Image (b) is selected "Minimum Path" item. Image (c) shows the minimum path (green) and the distance by meters.

The design of viewer GUIs keeps special attention on the accessibility because smartphones have not a pointing system such as mouse or a touch-screen. To provide simple user interface we: add a virtual cursor, reduce the amount of information displayed on the device; make input sequences concise in way to minimize the number of clicks; offer the user selection lists; use colors with high contrast.

Another important issue is related to limit the memory usage and to reduce the number of time-consuming operations. We kept in account optimization techniques applying design patterns with little changes (trusted, i.e., for the accessor methods to the private fields) and some common properties of the single elements have been delegated to the objects collecting them (i.e., the object related to feature collection collects the common properties of the contained features).

7 CONCLUSIONS

The GML has been adopted as "de facto" standard to exchange geographical data for state-of-the-art LBS ([28], [29]). Its richness and complexity make it not suitable for mobile devices. In this paper, we have proposed the cGML, a compact version of GML 2.1.2, based on short tags and encoded with pre-projected and pre-scaled coordinates.

Our work has been focused on finding a tradeoff between requirements of geographic information encoding and visual representation on mobile devices equipped with J2ME runtime environment. Since target platform has stronger limits than high-end native environments, such as frameworks provided by SymbianOS or WindowsCE operating systems, design phase has required investigation of device limits and extensive tests on commercial products. This phase has shown that models running on emulator where totally impracticable on real devices.

The result of this process is cGML 1.0 and an application prototype. cGML acts as both model and view. The geographic information can be totally transferred to client device for drawing, caching, and local operations without a permanent connection to the server, keeping some XML key features (platform independent, easily extensible, human readable).

Other works [30] have shown that map provisioning for mobile devices requires implementing a complex infrastructure. cGML enable to simplify application implementation and deployment, by means of XML-based language and web services infrastructure.

In the field of User-Adaptative Maps [31], it has been shown device adaptation does not cover all aspects of mobile provisioning of cartographic data, since maps dynamically generated depend on too many variables. cGML enables on-the-fly generations of model, leaving to the client application the responsibility of data visualization and it can also be tailored according to any other user (or device) property specified in the request to the server.

cGML and Mobile SVG share the same objective: they are designed to be used in applications for mobile devices. However, cGML and Mobile SVG keep the same main scope of languages they come from: cGML encodes geographic information, even if it could be directly showed, and Mobile SVG encodes vector graphics.

8 REFERENCES

[1] D. H. Williams: *It's the (LBS) applications, stupid!* URL: http://www.wirelessdevnet.com/features/williams_lbs01 Retrieved: 31 March 2005

[2] H.A. Karimi, X. Liu: *A Predictive Location Model for Location-Based Services*. In Proceedings of GIS'03, pp. 126 – 133, April 2003.

[3] T. Reichenbacher: *Adaptive methods for mobile cartography*. In Proceeding of ICC 2003, August 2003.

[4] *OpenGIS Location Services (OpenLS).* URL: http://www.opengeospatial.org/functional/?page=ols Retrieved: 31 March 2005

[5] *Geography Markup Language (GML) v1.0.* URL: http://www.opengis.org/docs/00-029.pdf Retrieved: 31 March 2005

[6] *Geography Markup Language, v2.0.* URL: http://www.opengis.org/docs/01-029.pdf Retrieved: 31 March 2005

[7] *Geography Markup Language, v2.1.1.* URL: http://www.opengis.org/docs/02-009.pdf Retrieved: 31 March 2005

[8] *OpenGIS® Geography Markup Language (GML) Implementation Specification, version 2.1.2.* URL: http://www.opengis.org/docs/02-069.pdf Retrieved: 31 March 2005

[9] *Geography Markup Language (GML) Implementation Specification, Version 3.0.* URL: https://portal.opengeospatial.org/files/?artifact_id=7174 Retrieved: 31 March 2005

[10] *Geography Markup Language (GML) Implementation Specification, Version 3.1* URL: http://portal.opengis.org/files/?artifact_id=4700 Retrieved: 31 March 2005

[11] *OpenGis® Abstract Specification.* URL: http://www.opengeospatial.org/specs/?page=abstract Retrieved: 31 March 2005

[12] *International Standard Organization, Technical Committee 211-Geographic information/Geomatics.* URL: http://www.iso.org/iso/en/stdsdevelopment/tc/tclist/TechnicalCommitteeDetailPage.TechnicalCommitteeDetail?COMMID=4637 Retrieved: 31 March 2005

[13] *Vector Markup Language (VML).* URL: http://www.w3.org/TR/NOTE-VML Retrieved: 31 March 2005

[14] *X3D.* URL: http://www.web3d.org/x3d/ Retrieved: 31 March 2005

[15] *Scalable Vector Graphics (SVG) 1.1 Specification.* URL: http://www.w3.org/TR/SVG/ Retrieved: 31 March 2005

[16] *Mobile SVG Profiles: SVG Tiny and SVG Basic.* URL: http://www.w3.org/TR/SVGMobile12/ Retrieved:

[17] *XSL Transformations (XSLT) Version 1.0.* URL: http://www.w3.org/TR/xslt Retrieved:

[18] R. Lake: *Making Maps With Geography Markup Language (GML).* Galdos Systems Inc., 2000.

[19] Z. Guo, S. Zhou, Z. Xu, A. Zhou: *G2ST: A Novel Method to Transform GML to SVG.* In Proceedings of ACM, pp. 161 – 168, ACM Press, 2003.

[20] *Mozilla SVG Project.* URL: http://www.mozilla.org/projects/svg Retrieved: 31 March 2005

[21] *Adobe SVG Zone.* URL: http://www.adobe.com/svg/ Retrieved: 31 March 2005

[22] *Corel® SVG Viewer.* URL : http://www.corel.com/servlet/Satellite?pagename=Corel2/Products/Content&pid=1047023276653&cid=1047023286996 Retrieved: 31 March 2005

[23] *Apache Squiggle – the SVG Browser.* URL: http://xml.apache.org/batik/svgviewer.html Retrieved: 31 March 2005

[24] *EPSG Geodesy Parameters database of geodetic parameters and Coordinate Reference Systems.* URL: http://ocean.csl.co.uk Retrieved: 31 March 2005

[25] *cGML.* URL: http://www.crs4.it/nda/cgml Retrieved: 31 March 2005

[26] *GeoServer Project.* URL: http://geoserver.sourceforge.net Retrieved: 31 March 2005

[27] kXML Project. URL: http://kxml.enhydra.org/ Retrieved: 31 March 2005

[28] J. Chang-Won, Y. Suk-Dae, K. Myung-Sam, C. Yeong-Jee, L. Joon-whoan: *Development of LBS Application using GML.* In Proceedings of GMLDays 2004, July 2004.

[29] M. Kyoung-Wook, J. In-Sung, C. Dae-Soo, H. Eun-young: *The GML Data Processing in Open LBS Platform.* In Proceedings of GMLDays 2004, July 2004.

[30] A. Zipf: *User-adaptive maps for location-based services (lbs) for tourism.* In Proceeding of ENTER 2002, June 2002.

[31] T. Reichenbacher: *The world in your pocket towards a mobile cartography*. In Proceedings of ICC'01, June 2001.