Compact GML: merging mobile computing and mobile cartography

Andrea Piras, Roberto Demontis, Emanuela De Vita, Stefano Sanna
CRS4, Center for Advanced Studies, Research and Development in Sardinia
Edificio 1, Loc. Piscinamanna, Polaris
09010, Pula (CA), Italy
{piras, demontis, emy, gerda}@crs4.it
http://www.crs4.it

Abstract. The use of portable devices is moving from "Wireless Applications", typically implemented as browsing-on-the-road, to "Mobile Computing", which aims to exploit increasing processing power of consumer devices. As users get connected with smartphones and PDAs, they look for geographic information and location-aware services. While browser-based approaches have been explored (using static images or graphics formats such as Mobile SVG), a data model tailored for local computation on mobile devices is still missing. This paper presents the Compact Geographic Markup Language (cGML) that enables design and development of specific purpose GIS applications for portable consumer devices where a cGML document can be used as a spatial query result as well.

1 Introduction

The use of portable devices is moving from "Wireless Applications", typically exploited as browsing-on-the-road, to "Mobile Computing", which aims to exploit increasing processing power of consumer devices. As users get connected with smartphones and Personal Digital Assistants (PDAs), they look for geographic information and location-aware services. The spread of hand-held devices, new generation of programmable cellular phones and the availability of development environments for such devices have made possible the design and development of new kind of software that satisfies the users' needs about mobility support and personal information management.

Many of such software applications are specialized for geographic data management and cartographic presentations (i.e. navigation systems, interactive maps for tourism and commerce). Current mobile cartographic systems are based on ad-hoc approaches, tailored for a specific device. Moreover, they are stand-alone applications, aimed only to support user mobility. At the same time, software development for mobile devices and web applications is moving from digital mapping and travel assistant applications to Location Based Services (LBSs), in which content and information are filtered according to user position.

The need for embedded maps into general-purpose application is emerging. LBS can be considered as an extension of general purpose Web Services, where execution context consists of both user profile and user position. While user profile is intended to be manually provided by the user (by means of registration form or automatic agent-based profiling systems), user position is automatically gathered by dedicated hardware.

Positioning is key element for LBS: Global Positioning System (GPS), aGPS (provided by some network operators on 3G European cellular networks), CellID (based on the number that enables to identify each cell/tower on GSM network) are widely available on consumer market, with different form factors and connections. Bluetooth devices make it possible to add positioning facilities to any Bluetooth master device able to link to a serial port (the RFCOMM protocol), opening new application scenarios. Positioning is the enabling technology for client-side; as said before, GPS data can be exploited thanks to processing and programmability of new exciting mobile devices. Technologies such as Java2 MicroEdition (J2ME) and SymbianOS, specifically designed for mobile phones, and Microsoft Windows Mobile platform (PocketPC and Smartphone) enable design and development of powerful client side applications, able to access to LBS and provide customized, accessible graphical user interfaces (GUIs).

Convergence of positioning and programmability has brought to real implementation of LBS, mobile computing paradigm and mobile cartography systems. While browser-based approaches have been explored (using static images or graphics formats such as Mobile SVG), a data model tailored for local computation on mobile devices is still missing.

This paper presents design and implementation of the Compact Geographic Markup Language (cGML), a custom version of GML tailored for mobile devices and mobile LBS applications.

2 Mobile cartography and devices

The basic properties of cartography depend on the answers to questions “What should it be represented?”, “Who is the end user?”, and “Where is data being displayed?”. With the spread of mobile devices and the convergence of Internet and wireless communications, the cartography is being adopted to such devices. Therefore, the basic properties are addressed by the service specific purpose, the user profile and the mobile device constraints.

The mobile cartography permits to nomadic users to collect all pieces of information for satisfying their needs and taking decision in different situations (i.e. during tourism vs. work traveling) with reference to their proximity.

In this context, the representation of proximity is focused on describing an area of interest (AOI) where same points of interest (POIs) are evidenced. Each POI can have text information for providing detailed description. The map of one AOI is not only the result of a request but it can be used for composing next request. The quantity and

quality of the information that can be provided to the mobile device cannot be supported by its hardware capabilities.
This section provides a brief overview about the LBSs, the mobile device constraints and some approaches to provide maps in mobile devices.

2.1 Location Based Service

A generic service that provides geographical or geographical related information based on a position is called Location Based Service (LBS). There are two different kinds of LBSs: the task-centered services (i.e. emergency management and electricity line repair) and the user-centered services (i.e. travel assistant and people finder).
An important business is focused onto provide to mobile users the right information, at the right position, at the right time, for the right device 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].
The GPS-enabled mobile devices avoid users to specify their "right position" through the tuple street/city/country: it will be gathered by the mobile application and sent automatically to the server. This scenario introduces the biggest obstacle topic for LBS: the lack of privacy security system that guarantees the protection of customer personal data. Ubiquitous computing paradigm would be considered as a reliable solution to have service flexibility and user data protection: private data (such as user position and user path) is kept inside mobile devices and provides outside only if needed. There is no Location Server to store users location.

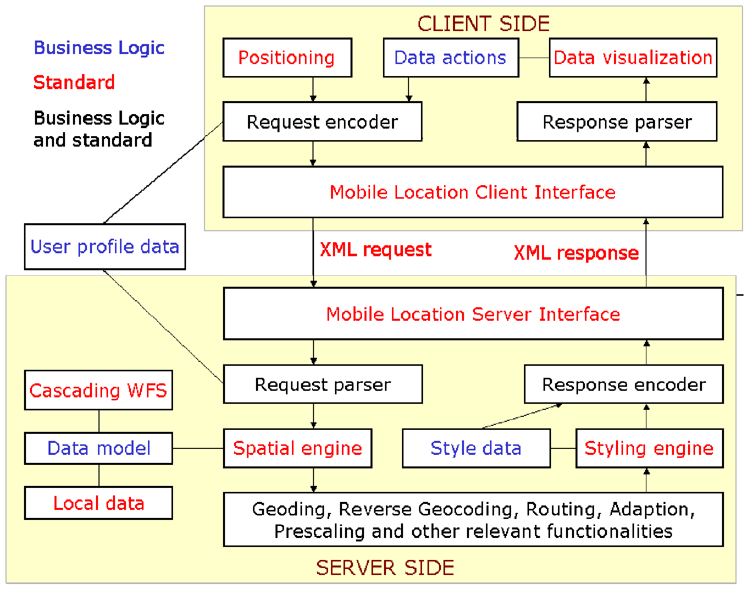


Fig. 1. Generic mobile LBS structure.

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 device” is referred to the data-adaptation issue. To improve the LBS performance, the number of functionalities processed in client side is increased but device constraints influence the service functionalities about data format specification, data transferring and its visualization on client.
In last year, standards based on XML language for LBS were defined. Examples are the specification of the GeoMobility Server and the XML for Location Service (XLS) proposed by OpenLS [3]. 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).

2.2 Mobile device constraints

Device constraints influence application design and development for mobile devices: practices and models should be refactored to achieve usability and performance goals on such equipment. This paragraph provides an overview of such constraints and describes some approaches common to mobile computing research field.
Application developers have to deal with hardware constraints in terms of small display (min. 96x54 pixel), reduced color palette (sometimes with one bit LCDs), CPU power, battery life, limited persistent storage (min. 8Kb), runtime heap and ROM (min. 512Kb).
Mobile device communication is based on narrowband wireless networks. Network congestion makes harder to get connected and the physical obstacles between terminal and base station brings to frequent “connection lost” status. Therefore, application designers try to limit the data size transferred in each session using compression and data splitting strategies.
Splitting data does not requires simple “byte packaging” but “information packaging” based on semantic analysis of content, in order to transform information in an aggregate of atomic (and small) 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”). The last 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.
The application programming interfaces (APIs) provided by runtime environments for mobile devices are poorer that their desktop counterpart and focused on the requirements for working on small displays and on the different strategies about GUI usability. Issues about GUI design and development are:

- Display size and resolution of PDAs and mobile phones reduced respectively to 1/4 and to 1/8 of PC screens (some recent mobile phones have high-resolution Quarter VGA screens).

- Reduced color palette (from 2 to 16 bits).
- For smartphones, the lack of a full-size QWERTY keyboard but also of a reduced, compact keyboard with essential keys. They only provide a numeric keypad, multi-tapping character selection, a few predefined buttons and usually two customizable soft buttons. Only few high-end models provide pen-based input system, very similar to PDA's.
- For smartphones, the pointing system must be simulated for giving the user the same feeling of the mouse. PDAs have touch-screens.
- Availability of programming libraries.
- The no-homogeneity of users' experience about information technology requires particular attention by the designer for defining very intuitive GUIs.
- The mobile user is "user on the road" (opposite to user sat in front of the monitor of his desktop), therefore the application must provide information easy to understand.

While next generation devices will provide powerful hardware and rich software libraries, current technology has still to be exploited and will move to smallest devices (like watches) in next future.
About software, J2ME architecture is a robust environment for applications running on mobile devices: it relies on well-known Java platform, it is cross-platform, it does not require agreement to deploy applications and it is not related to a specific implementer (JVM have been developed by many independent software houses). It defines two profiles to fit the range of main device categories: Personal Profile for high-end PDAs and game consoles and Mobile Information Device Profile (MIDP) for mobile phones and entry-level PDAs. Major phone manufacturers bundle their own J2ME implementation in a rich set of their models, ranging from entry-level (games market) to high-end business communicator terminals (business market). The main MIDP's lacks are object introspection, Java Native Interface (JNI) and floating point mathematic. The last one can be performed by software but that could be expensive, in terms of power processing.
To date, the most important development environments alternatives to J2ME are In-Fusio, Mophun SymbianOS, Microsoft PocketPC Win32 and .NET Compact Framework. In-Fusio and Mophun keep some J2ME base characteristics and are mainly devoted to games market. They require the developer signs a distribution agreement with the technology providers and network operators. SymbianOS provides one the richest programming interfaces (based on C++ language) available in the mobile phone market and it is able to exploit all the hardware characteristics. Microsoft PocketPC Win32 and .NET Compact Framework are development environments for Windows Mobile platform. They address PDA market and recently the high-end mobile phones market, porting some of well-known programming concepts and paradigms of the Windows desktop platform.

2.3 Approaches

The early approaches have been thought for allowing users to view maps on their no-programmable phones. Mobile telecommunication companies and service providers realized applications based on messages: user sends a Short Message Service (SMS) containing the specification of the map and receives one Multimedia Messaging Service (MMS) with the image of the map.
To date, with the spread of Java-enabled phones and the increase of their capabilities, these approaches tend to be become obsolete and they are substituted by several kinds of commercial products. They can be grouped in three main approaches.
A common solution is based on native application tailored for one specific phone or PDA and requires information (geographic and generic) to be stored on client device. In this way, no Internet connection is required but information must be pre-selected and stored locally on the device before the trip.
Another approach is based on web applications: the user connects to a Web site through the browser bundled in device, specifies zoom factor, query keywords, and preferences about map details and the site returns the images. Some sites offer the possibility to receive images sized to the PDA display (see [4] and [5]). Nevertheless, each map is non-interactive and changing any parameter means the need to send another request to the site and to download the resulting page. Therefore, such an approach is efficient with reliable and cheap Internet connections.
The last one mixes the twos: the user browses the map service using a PC and transfers final map to the target device. During his trip, the user does not need an Internet connection but he could see only the pre-loaded map.
These three approaches share a common drawback: they are stand-alone applications. They neither are nor can be integrated with other application, like an agenda or an address book. This is not an issue for very specific applications, like professional cartographic programs, but often a large set of mobile application need a simple map-like representation of geographic data and maps are only an "accessory" functionality.
A new scenario emerges: simple maps should be retrieved remotely and described in a platform independent format. Such a solution can be easily implemented with web services technology and XML-based languages.

3 GML

The most known XML-based language for encoding geographic information is the GML (Geography Markup Language), defined by the Open GIS Consortium. It has been adopted as "de facto" standard and it is not related to any specific hardware or software platform: data encoded using it can be easily read and understood by any programming language and software system able to parse XML streams. GML encodes vector geographical information together with metadata on spatial and non-spatial resources.
The first two GML main releases, GML 1.0 [6] and GML 2 (GML 2.0 [7], GML 2.1.1 [8] and GML 2.1.2 [9]), have the objective to encode geographic information in XML,

allowing its modeling, storage and transport. They are based on the OpenGis® Abstract Specification [10] and, in particular, on the 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 a lot of new characteristics and concepts to improve and extend the GML capabilities. The language is also based on the ISO 19100 series [11] specification and not only on the OpenGIS® Abstract Specification. GML 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 and more expressive types of the real world and by the introduction of the three dimensional primitives. Curves and their segments, arcs, Bezier curves, clothoids, geodesic curves, triangles, rectangles, surfaces are some examples of new one and two-dimensional primitives while we find solid in the three dimensional ones. New sets of geometric elements allow defining aggregations of them whose interiors are disjoint and composite geometries.

The list of new features in GML 3.0 [12] and GML 3.1 [13] comprises: 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 associated to a capture event and the value for the result of the observation, coverage model and representations, styles definition.

The style information may be used for styling but may also 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 SVG [14], Mobile SVG [15], VML [16] or X3D [17] using standard XML transformation facilities (XSLT [18]).

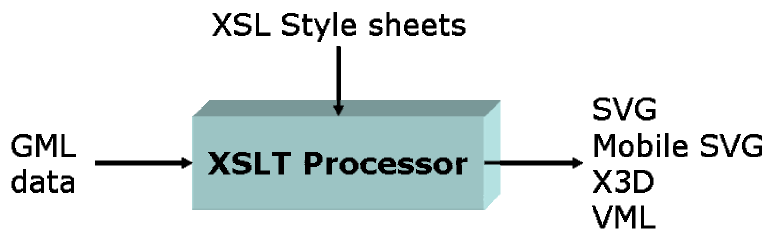


Fig. 2. Transforming GML into graphical format.

Usually, the language resulting by the XSLT processing is SVG for desktop applications (see [19] e [20]) and Mobile SVG for mobile devices ones. Mobile SVG consists of subsets of SVG's elements, attributes and events selected for being used in mobile devices and considering the variety of mobile devices, two profiles are defined: SVG

Tiny (SVGT), suitable for restricted mobile devices like smartphones; SVG Basic (SVGB), targeted for higher-level mobile devices, like the PDAs.
There are commercial and open source tools to perform such a translation and SVG/Mobile SVG rendering. Some renders are natively embedded into web browser (i.e. Mozilla [21]), some distributed as plug-ins for many browsers (i.e. Adobe [22] and Corel [23] SVG Viewer) and some available as stand-alone viewers (i.e. Apache Squiggle [24]). The overview on SVG implementations is completed by native SVG editors (W3C Amaya [25] and Jasc WebDraw [26]), SVG-exporting editors (Adobe Illustrator [27] and Corel CorelDraw [28]) and toolkits (Batik SVG Toolkit [29], CSIRO SVG Toolkit [30] and TinyLine [31]).

4 The Compact Geographic Markup Language (cGML)

There are some drawbacks in direct use of GML for mobile devices. Processing and saving GML data require a considerable memory space which cannot be provided on mobile devices (see section 2.2). Furthermore, the transfer of GML documents of hundreds of kilobytes through unreliable and narrowband wireless connection is expensive and boring for users. The GML document could be compressed before the transfer but the decompressing process in the mobile device requires considerable processing power and extra memory capabilities.
The GML 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.
These considerations and the wish to eliminate the GML translation in SVG in favour to draw directly the geometric data, have driven the definition of compact Geographic Markup Language (cGML).

4.1 Language features

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 upper versions are not among the objectives of the current cGML release.
The word “compact” in the cGML definition summarizes one of its key aspects. We can 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 and does not provide support to XLink [32]. 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).
A cGML document would be a stand-alone XML document so we introduced the `cGML` root element and defined concrete elements based on the 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.
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 EPSG coordinate reference system code [33] for the UTM (WGS84) Zone 32N 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), the real box area as (bx1, by1, bx2, by2), the real coordinates Preal(x, y) of a point Pview(x_v, y_v) in the view are defined by the rules:

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

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

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(25, 187) of the view are:

$$x = \text{round}(0.2667 * 25) + 510775 = 510782$$

$$y = \text{round}(0.2667 * (400 - 187)) + 4339616 = 4339673$$

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. 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 Shape1 (rows 9 – 22), containing two features, with the id attributes are a1 (rows 10 – 15) and b6 (rows 16 – 21), and another feature, with the id attribute is p_o (rows 23 – 27).
The feature element contains 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.

Example of cGML document.

```
 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="Shape1" name="areas">
10:      <Ft id="a1" name="mainstreet">
11:        <LnSt>
12:          <cds>267,39 276,42</cds>
13:        </LnSt>
14:        <info>info for feature a1 in Shape1</info>
15:      </Ft>
16:      <Ft id="b6">
17:        <LnSt>
18:          <cds>326,64 353,78 396,102</cds>
19:        </LnSt>
20:        <info>info for feature b6 in Shape1</info>
21:      </Ft>
22:    </FtCl>
23:    <Ft id="p_o" name="postal office">
24:     <Point>
25:        <cds>79,10</cds>
26:     </Point>
27:    </Ft>
28: </cGML>
```

Rows are numbered to simplify references to parts of the document.

GML allows to define the coordinates of the vertexes of the geometries either as a sequence of `coord` elements (that encapsulate the `X`, `Y` and `Z` elements) or as 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 with each tuple composed from one to four integers separated by commas. Furthermore, the cGML coordinates are related to the screen device. In rows 5, 12, 18, 25 of previous cGML document, examples of `cds` values.

Table 1. Comparison between some GML and cGML tags.

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%

4.2 The XML Schemas

The cGML is defined by three XML Schema ([34] and [35]) 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 (http://www.crs4.it/nda/cgml).

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. Even if, schema design lets developers to write XML documents, not cGML instances, based only the cGML geometries or the cGML features and geometries.

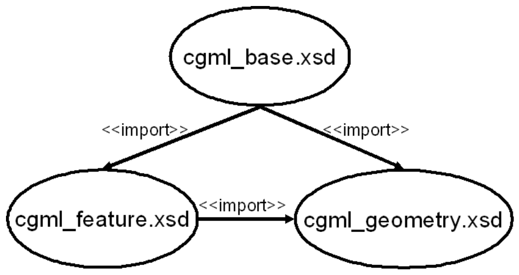


Fig. 3. The XML Schema files of the cGML.

4.3 Advantages of short tags

The introduction of the compact notation and the deletion of some elements produce immediate advantages in the manipulation of data on mobile devices. The XML documents, instances of cGML, are an average of about 64% shorter then GML ones, so:

- 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. In this way, cGML parser reduces the memory space for their processing and it can work in entry-level smartphones too.

Although the short tag names, the cGML document preserves the human-readable feature of XML documents.

5 An architecture based on cGML

We have tested cGML on a web service that exports geographic data provided by a GIS service to a Java client. In next subsections, we describe the server and the client implementations.

5.1 The server side

The server gathers various kinds of data divided in geographical feature and their no-geographical attributes. All of these data are referenced by name and contain planar coordinates.
Data processing is performed by means of a predefined cGML model. The geographic information is retrieved from topology GIS data files and the non-geographic data from items collected in tables. The SRS is EPSG:32632 and coordinates are multiplied by 1000 to have one pixel per millimeter. The server application is able to create the model using either local data files (i.e., ESRI shape files) and external data (i.e., GML from WFS) or it can use data model of a wide AOI defined directly in cGML. A geographic feature is related one by one to a cGML feature element by the value of their `id` attribute.

Example of `Head` block of a cGML data model used by server side application. It refers to a specific AOI where the scale factor of the data is defined by the `zoom` attribute.

```
<Head>
   <RealBox srsName="EPSG:32632">
         <cds>507000,4339000 518000,4348000</cds>
   </RealBox>
   <View zoom=”1000.0”>11000000,9000000</View>
</Head>
```

In this example, one meter is represented by one thousand of points in the viewport.

The server waits client requests on port 3010. Each request contains the specification of the new features context based on device screen size and on the AOI (the `RealBox` element). It can also request one single particular features collection (one layer) through the specification of the `id` attribute of the `Layer` element.

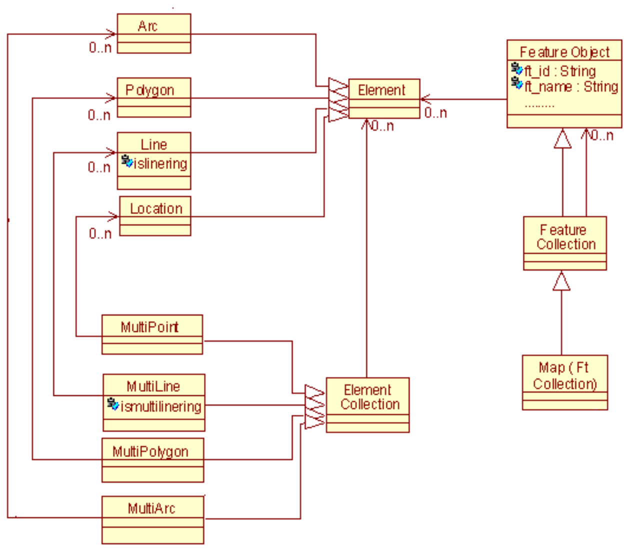


Fig. 4. The features data model in the server side application.

When the server receives a request, it performs these transforming operations on data model items:

- Clipping: it selects the geographic data of the specified AOI.
- Projection: the server processes different data with different projection and merge them.
- Scaling: the real coordinate values are adapted for plotting in the viewport without 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 result of such a transformation is used to define cGML geometric elements and their coordinates. Final cGML document is completed by adding the feature attributes, the `info` elements and the `Head` block.

The server side is developed using the Java programming language.

An example of client request.

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

5.2 The cGML Viewers

J2ME has been chosen as reference implementation platform, since it addresses most of cellular phones and PDAs market and provides platform independence, thanks to its MIDP and PP profiles (see section 2.2). We developed a cGML Viewer for each J2ME profile (both PDA and smartphone). The two clients have the same core and the same functionalities but very different GUIs because they are adapted to the device constraints.

The software routines of the viewers keep particular cure into limit the memory usage, for both static object allocation and dynamic heap for methods invocation, and into reduce the number of time-consuming operations. Well known design patterns have been slightly modified (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). This approach limits the flexibility but, on the other side, saves memory and allows having an interactive mapping system even on a device with very constrained resources.

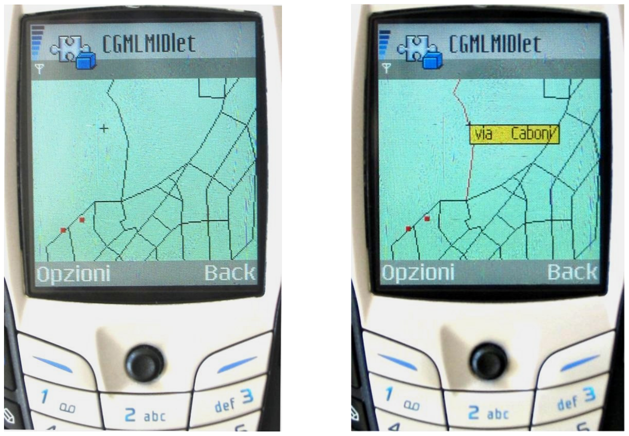
The viewer obtains a cGML map file establishing an HTTP connection to the server port 3010 and sending a request formatted like the previous example. The size of the returned document is compatible with the RMS (Record Management System) persistence facility of the MIDP profile, therefore it is possible to store the map in phone memory as well and to recover it without a new connection to the server.

The viewer parses the downloaded cGML document using the kXML SAX parser [36] and it instances objects for managing the `Header`, `Ft` and `FtCl` elements and geometry information.

These objects are the input of the plotting routines: they can be displayed without extra processing since data has been pre-processed on the server side. Plotter has been optimized to reduce allocation of new objects: double buffering and flyweight pattern to share a common set of elementary components.

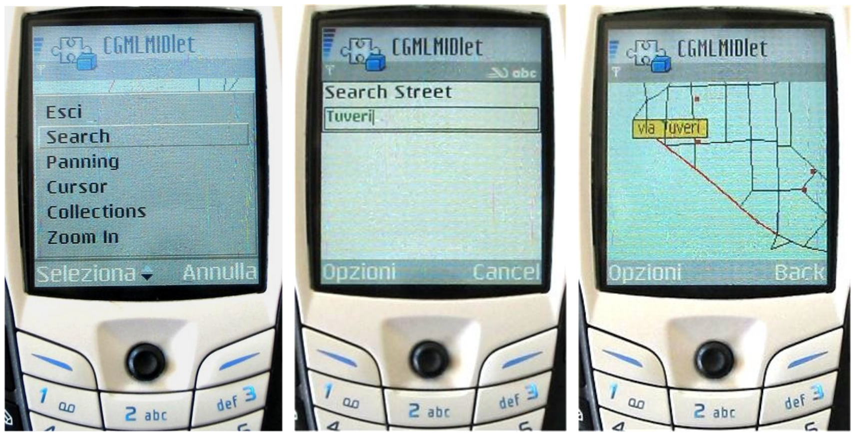
The resulting map is shown by the cGML Viewer. It also includes the possibility of process the common basic operations on the map and has some little but important features. The basic operations are: map saving/loading in/from device memory, zooming in – out, panning, searching a point of interest by name, choosing a point of interest by means of cursor and selecting the order of layers visualization. All these operations are processed in the device without establishing new connection to the server and downloading other cGML documents. The server connections are required only if the user asks for data outside initial AOI.

Other features have been implemented to improve user accessibility. The map can emphasize some descriptive labels for selected objects and they are always drawn with horizontal text. In this way, the user does not need to rotate the device to read texts and the display emphasizes only the objects the user has required, providing good experience even in bad light conditions. Furthermore, the cGML Viewer for smartphones uses context commands accessible through the custom soft buttons as user command inputs and supports a virtual cursor, which can be used to select objects on the map, because it has not a pointing system such as mouse or a touch-screen.



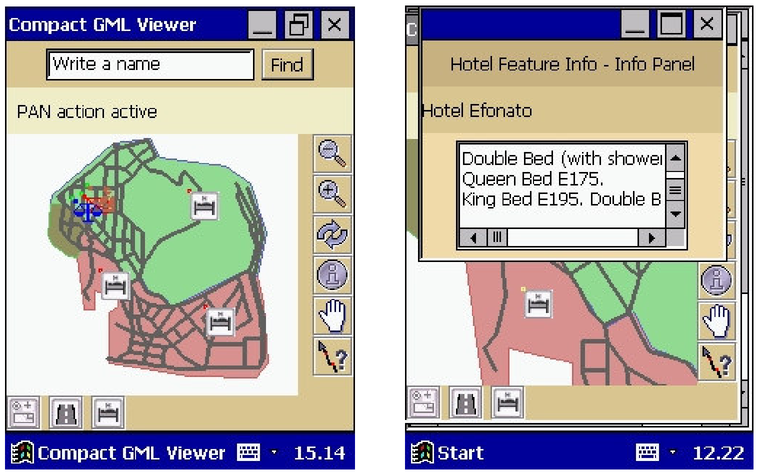
(a) **(b)**

Fig. 5. Two images about the cGML Viewer for smart-phones. The image (a) shows a section n of a map and contains the virtual cursor. Image (b) shows a section of the map and emphasizes the name and the line of the selected item.



(a) **(b)** **(c)**

Fig. 6. The three images show the way to find a street by name using the cGML Viewer for smart-phones. The image (a) shows the list of available functionalities. Image (b) displays the GUI where inserts the name of the street. While image (c) shows the map with a highlighted street.

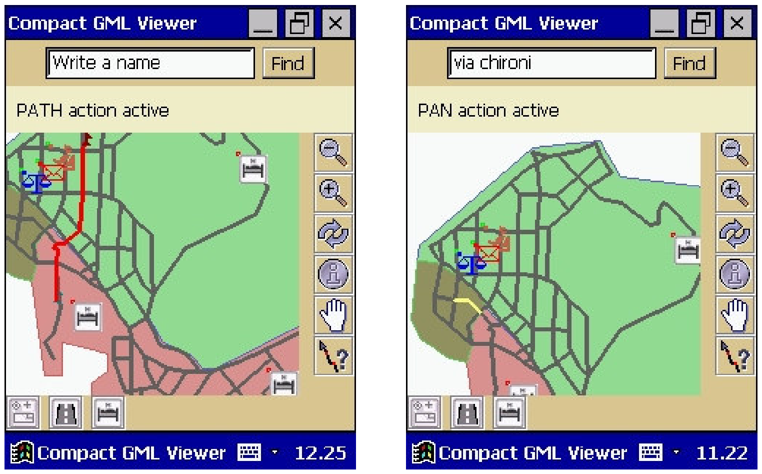


Hotel Efonato

Double Bed (with showe
Queen Bed E175.
King Bed E195. Double B

(a) (b)

Fig. 7. Two images about the cGML Viewer for PDA. The image (a) depicts the whole map, emphasizes the different areas and shows some POIs. Image (b) displays the information related to the selected POI.



(a) (b)

Fig. 8. The cGML Viewer enables to define itineraries, image (a), and search streets by name, image (b).

6 Future works

The cGML 1.0 implements some basic ideas about mobile cartography on very constrained devices. We plan the features to introduce in next version, in way to apply some characteristics of the wide set of specifications of GML 3. The set of geometric primitives will be extended with new two dimensional and three dimensional items. Direction and coverage information will enable to enrich the cGML, while other objects will be studied and evaluated. Next research activities will focus on styling information and multilevel structure.

About styling information, GML 2 does not provide indications and GML 3 presents a default style that may be completely ignored. We would like to define one styling system considering the CSS [37] – HTML [38] model. In the cGML `Head` block, the optional `style` element will be added. It will have a reference to a separate file containing the style information and styles will be applied following the same cascading mechanism of CSS. The feature and geometric elements will have the optional `style` attribute and it will have the same role of the HTML `class` attribute. The style information will be encoded in XML and the use of available XML styling languages must be evaluated. The style information will enable the user: to define the color of geometric elements and features, the border of lines (single, double, dash, dotted, etc.); to link images related to features or geometric entities; to insert detailed descriptions, specify their visual presentation and structure them in tables or lists; to refer web pages.

Together with new features, the existing schemas will bear a refactoring process to make the cGML a GML application schema. In GML 2, the application schemas were restricted to the development of dictionaries for geographic features and based only on the `feature.xsd` file of GML 2. Therefore, we could not develop the cGML as an application schema and we have chosen to define simple schemas inspired by the ones of GML 2.1.2 but without neither importing nor referencing them. In GML 3, the application schema concept is extended to almost of GML schemas and next cGML version could use them.

7 Conclusions

The GML is the “de facto” standard to exchange geographical data but it is not suitable for low-end mobile devices. In this paper, we have proposed a compact version of GML 2.1.2, based on short tags and encoded with pre-projected and pre-scaled coordinates: the cGML.

The cGML design and development follow the Dynamic Systems Development Method (DSDM) principals [39] and they have been characterized by a continuous designing, coding and testing loop. Design phase concerns design specifications of XML elements and attributes. Development phase concerns actual XML Schemas definition, their example instances, the parser, the server side and the client viewers. During integration and test phase, the different parts are put together and tested. The test results become the input for restarting the cycle. Our attention has been focused

on to find a compromise between the geographic information, their visual representation and the mobile device power processing capabilities of the J2ME enabled devices. Since target platform has stronger limits than SymbianOS or WindowsCE/PocketPC operating systems, design phase has required to investigate the device limits and sometimes application models have been trashed since they were not appliable on the commercial devices. The result of this iterative process is cGML 1.0 and application prototype.
cGML acts as model and view at the same time. 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 [40] 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 [41], it has been exposed that it is not enough to focus on adaptations to technical parameters (device characteristics, quality of service, location) but maps need to be dynamically generated according to lots of variables. cGML enables to be on-the-fly generated according to device profile adapting the view to the screen size of the device. It can also be tailored according to any other user (or device) property specified in the request to the server.
Over to the GML, the XML language closest to cGML is Mobile SVG. They 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 their "father" languages: cGML encodes geographic information, even if it could be directly showed, and Mobile SVG encodes vector graphics.

8 References

[1] Williams, D. H.: It's the (LBS) applications, stupid! http://www.wirelessdevnet.com/features/williams_lbs01/ (2003)
[2] Karimi, H.A., Liu, X.: A Predictive Location Model for Loation-Based Services. In Proceedings GIS'03 (2003) 126 – 133
[3] OpenGIS Location Services (OpenLS): Core Services. Version 1.0. http://portal.opengis.org/files/?artifact_id=3418 (2004)
[4] Maporama, http://www.maporama.com
[5] Mapquest, http://www.mapquest.com
[6] Geography Markup Language (GML) v1.0. http://www.opengis.org/docs/00-029.pdf (2000)
[7] Geography Markup Language, v2.0. http://www.opengis.org/docs/01-029.pdf (2001)
[8] Geography Markup Language, v2.1.1. http://www.opengis.org/docs/02-009.pdf (2002)
[9] OpenGIS® Geography Markup Language (GML) Implementation Specification, version 2.1.2. http://www.opengis.org/docs/02-069.pdf (2002)
[10] OpenGis® Abstract Specification, http://www.opengis.org/specs/?page=abstract

[11] International Standard Orgnization, Technical Committee 211 - Geographic information/Geomatics. http://www.iso.org/iso/en/stdsdevelopment/tc/tclist/TechnicalCommitteeDetailPage.TechnicalCommitteeDetail?COMMID=4637
[12] OpenGIS® Geography Markup Language (GML) Implementation Specification – Version 3.0. http://www.opengis.org/docs/02-023r4.pdf (2003)
[13] Geography Markup Language (GML), version 3.1. http://portal.opengis.org/files/?artifact_id=4700 (2004)
[14] Scalable Vector Graphics (SVG) 1.1 Specification. http://www.w3.org/TR/SVG/ (2003)
[15] Mobile SVG Profiles: SVG Tiny and SVG Basic, Version 1.2. http://www.w3.org/TR/SVGMobile12/ (2004)
[16] Vector Markup Language (VML). http://www.w3.org/TR/NOTE-VML (1998)
[17] X3D. http://www.web3d.org/x3d/
[18] XSL Transformations (XSLT) Version 1.0. http://www.w3.org/TR/xslt (1999)
[19] Lake, R.: Making Maps With Geography Markup Language (GML). Galdos Systems Inc. (2000)
[20] Guo, Z., Zhou, S., Xu, Z., Zhou, A.: G2ST: A Novel Method to Transform GML to SVG. In Proceedings 11th International Symposium of ACM. ACM Press (2003) 161 – 168
[21] Mozilla SVG Project. http://www.mozilla.org/projects/svg/
[22] Adobe SVG Zone. http://www.adobe.com/svg/
[23] Corel® SVG Viewer. http://www.smartgraphics.com/Viewer_prod_info.shtml
[24] Apache Squiggle – the SVG Browser. http://xml.apache.org/batik/svgviewer.html
[25] W3C Amaya. http://www.w3.org/Amaya/Amaya.html
[26] Jasc WebDraw. http://www.jasc.com/products/webdraw/
[27] Adobe Illustrator. http://www.adobe.com/products/illustrator/main.html
[28] CorelDraw. http://www.corel.com/servlet/Satellite?pagename=Corel2/Products/Home&pid=1047022690654
[29] Batik SVG Toolkit. http://xml.apache.org/batik/
[30] CSIRO SVG Toolkit. http://sis.cmis.csiro.au/svg/
[31] TinyLine. http://www.tinyline.com/
[32] XML Linking Language (XLink) Version 1.0. http://www.w3.org./TR/xlink/ (2001)
[33] EPSG Geodesy Parameters database of geodetic parameters and Coordinate Reference Systems. http://ocean.csl.co.uk/
[34] XML Schema Part 1: Structures. http://www.w3.org/TR/xmlschema-1/ (2001)
[35] XML Schema Part 2: Datatypes. http://www.w3.org/TR/xmlschema-2/ (2001)
[36] kXML Project. http://kxml.enhydra.org/
[37] Cascading Style Sheets, level 2 CSS 2 Specification. http://www.w3.org/TR/REC-CSS2/ (1998)
[38] HTML 4.01 Specification. http://www.w3.org/TR/html4/ (1999)
[39] DSDM Consortium. The Underlying Principles. http://www.dsdm.org/en/about/principles.asp
[40] Reichenbacher, T.: The world in your pocket towards a mobile cartography. In Proceeding of the 20th International Cartographic Conference (2001)
[41] Zipf, A.: User-adaptive maps for location-based services (lbs) for tourism. In Proceeding of ENTER 2002 (2002)