Updating GIS Data using Personal Digital Assistants

Key words: GIS, OpenGIS, GML, PDAs, Data Synchronization.

Abstract: Geo-referenced data is usually acquired and then stored into an existing GIS. With the advent of Mobile Computing Devices (aka PDAs), the integration task can be avoided. We extended a PDA GIS visualization system (Mordomo) in order to allow the update of metadata. In this way the task of updating geo-referenced data can be done on-site. In order for the system to cope with different applications and file formats, we provide a transformation from/to GML, based upon the proposed OGC standard.

1 Introduction

Important human resources have to be involved in order to get formatted data into a Geographic Information System (GIS). It is required that professionals take several measures, and this most of the times requires physical presence. For example, they may annotate data on printed maps and, afterwards, this data is integrated into a GIS database. Thanks to the recent technological evolutions in personal digital assistants (PDA), the integration task may be made simpler or even avoided altogether. PDAs are regarded more and more as a major help for computational work. Nowadays, most professional (mainly Information Technologists) have some kind of PDA which they use daily at their work. Also, recently, the adoption of the OpenGIS standard caused a major advance in the interoperability of the GIS world.

Thus the means needed to use PDAs to update GIS data using OpenGIS standards (GML) are in place. We took an application, Mordomo, which allows Palm users to navigate over a map. This application, developed by ParadigmaXis, presents a vector map, that can be navigated. The need for further extending the application Mordomo so that users could annotate the map, caused the creation of the extension MetaGeo (for Geographic MetaData) that is the subject of this paper.

The remaining of this paper is structured as follows: section 2, the OpenGIS Consortium and the GML initiative, presents the proposed standard for GIS, Geographic Markup Language (GML); section 3, Data Synchronization, shows the issues we need to take care when trying to reconcile changed copies built upon a common source; section 4, Suggested Architecture, shows the steps we took in order to extend the Mordomo application; section 5, Synchronizing GIS Data, explains the process implemented, the tests made, and the results obtained; and finally, section 6, Conclusions, summarizes our findings and identifies future work.

2 The OpenGIS Consortium and the GML initiative

Maps are space abstractions which represent some reality. Maps do not represent all the characteristics of geographical information, they have just its representation in a restricted, identified system. It is the responsibility of the geographer to decide how to represent data. For example, the geographer decides how to group this data and what projection system to use. Generally this data is acquired, analysed, and integrated into a GIS. With the introduction of PDAs, the technology allows us to directly acquire analyse and insert data into a GIS, provided the PDA is connected to a Global Positioning System (GPS).

Geographic data can be represented, among others, in two main formats: raster and vectorial (Demers, 2000). The former quantifies space as points, repre-
sented each a piece of the earth surface, while the latter assumes a continuum space, where points, lines, and polygons are stored.

GIS are evaluated by their support of metadata manipulation, that is, to have more than the map, over a map (so, we need to associate phenomena to georeferenced locations) (Cox, 2001). In order to provide transfer capabilities among native formats, the OGC (Open GIS Consortium, 2000) developed a standard. This standard, widely accepted, is known as Geographic Markup Language (GML) (Cox et al., 2001).

GML is based entirely on XML Schema. The adoption of XML Schema (XSD) (W3C, 2000) is a major advance. It has support for type inheritance, distributed schema integration, and namespaces. GML uses XLink and XPointer Specifications to express relationships between geo-spatial entities. This means that such relationships can be expressed between features in the same database or between features across the Internet. Furthermore, GML 2.0 allows relationships to be constructed between GML feature elements in different databases without requiring any modification of the participating databases. No more than read access is required to establish a relationship (OGC Recommendation Paper, 2001).

3 Data Synchronization

By definition, mobile users are not always connected to a network and their stored data. Users retrieve data from the network and store it on the mobile device, where they access and manipulate the local copy of the data. Periodically, users reconnect with the network to send any local changes back to the networked data repository. Users also have the opportunity to be notified about updates made to the networked data while the device was disconnected and they may decide to integrate the changes into their copy. Occasionally, they need to resolve conflicts among the updates made to the networked data.

This reconciliation operation where updates are exchanged and conflicts are resolved – in other words, the process of making two sets of data look identical, is known as data synchronization. For a mobile device, synchronization applies to the data that the mobile device stores locally.

Applications for PDAs often generate or modify data shared with a server or with other mobile devices. Since a mobile device is disconnected from the network much of the time, or connected by a link that is too expensive or slow to use every time the data is modified, mobile applications typically maintain separate copies of the shared data on different devices. These copies are synchronized from time to time, so that all copies contain the same data.

Data synchronization is a complex process. However, in the Mobile Network Computing Reference Specification (MNCRS, 1999) an application programming interface (API) providing data-synchronization services, is presented. These services simplify the task of writing a mobile application.

Since GML is XML, we need a way to find the differences amongst XML documents. There are several proposals to perform such a comparison. X-Diff (Wang et al., 2000), HtmlDiff (Chen et al., 2000) and TopBlend (based on HtmlDiff) were evaluated. The main problem with the approaches taken by any of these tools is that, in order to perform blind XML comparison, they are not semantically-aware. As such, they mark minor changes on the XML data that don’t have any reflection on the final map (and could, as such, be ignored).

4 Suggested Architecture

Some PDAs use the PalmOS operating system. A conduit is a module of software which provides a translation bridge between a Palm Computing platform device application and a particular desktop application. To test a conduit, a Palm Computing platform device or a properly set up Palm OS Emulator (POSE), is needed.

As illustrated on Figure 1, the proposed architecture allows us to Analyse and move the different GML elements amongst the platforms (PDA and server). The (GML2PDB) module separates elements according to their category, into three Palm databases which are created. These databases are then used by the Palm applications, mainly by Mordomo, which uses the data to present the cartography and the interest points. Thus, these databases are used by Mordomo only in read-only mode. The module designed and developed in this work (MetaGeo), presents and allows data updates on metageo.pdb, which is the only one with read-write access on the Palm. Whenever a sync task is performed, the data is processed and converted into a DOM tree. This DOM tree is matched against the original GML resulting DOM tree. The module PDB2GML that performs this task is called by the conduit developed in this work.

The first step in order to perform geo-referenced data maintenance is to transform existing data into a recognizable format. Since the selected platform doesn’t provide a database system, we need to store records in a file (the PDB file format). We have developed a data model able to perform a conversion from GML to the PDB format. This developed model has been based on Codd’s relational model (Codd, 1970) and presents us with an Interface similar to the Microsoft’s DAO Model.
This Model presents us with a few Collections, mainly Recordsets which represent data in a table or a query. Detailed information may be found at Microsoft’s Data Access Web Page http://data.microsoft.com/.

5 Synchronizing GIS Data

In order to synchronise GIS data in the PDA and in the GIS database, we developed GML2PDB. This module parses one (or several) GML files, generating the corresponding Palm databases.

We first tried to store the GML file itself, using the XMill algorithm (Liefke and Suciu, 2000). It is indeed optimized to achieve great compression on XML documents. The problem here is the decompression phase, because when it is needed we most certainly will not have enough memory (since we are using a Mobile device) mainly because the algorithm works with the full file, and not with parts of it. The algorithm uses blocks of fixed length (which may be parameterized), but on the decompression phase we can have very different results (one block could have a greater compression ratio than another). Another problem using the algorithm is that it is not semantically-aware, therefore it does not organize data in a good fashion for geographic processing, such as incorporating nearby data in the same block. These blocks could then be processed as compressed tiles, with several zoom levels (Yu and DeWitt, 1996) (see Figure 2).

The darker tile on Figure 2 represents the visualized fragment of the map, while the light gray tiles represent data already decompressed on memory (they’re more likely to be seen next). Tiles are then brought in memory and discarded as the user moves along.

We defined smaller compression fragments in order to cope with the device memory constraints, but the algorithm became less efficient. We also tried to pre-parse the XML file so that nearby data was tightly together, but the results were not as good as expected. We didn’t try to modify the algorithm so that we could give a decompression limit (which could allow the reading buffer on compression to be flushed even without being full). The fixed blocks compression makes the algorithm lose even more efficiency. Besides the memory issue, the main problem with having the GML file compressed on the PDA was that, when data was updated, we had to recompress the block, and that slowed down the system.

We decided to use the GML file only as a means to get standard formatted data, and to provide means to import and export data into the Palm in a transparent way.

GML4Java, developed by Galdos Inc parses a GML document, providing a GML-aware DOM-compatible Tree. The classes provided closely map the OGC Abstract Model, mainly with GML Features. Thanks to this tool, we were able to develop a meta-parser that created the Palm databases, by traversing the DOM tree nodes and, according to the Feature type, transform the data to the expected Palm format. The three generated databases (geometric data...
mordomo.pdb, interest points – interestPoints.pdb, and all the related metadata – metageo.pdb) are then sent to the Palm device (see Figure 1).

In order to be able to perform data manipulation in the PDA an application was then developed with the interface shown in Figure 3.

Figure 3: MetaGeo Main Screen

In order to get the data modified on the PDA into the GIS database, we can consider two ways:

**Updates performed only on the PDA** the simpler case, where we just need to compare the corresponding DOM trees and act according to the identified differences;

**Updates performed on both systems** we need to have access to the DOM tree that created the Palm database, so we can merge the differences amongst the three files.

In order to detect the differences, we could:

1. Generate the GML from the Palm databases, comparing documents (Wang et al., 2000; Chen et al., 2000);
2. Only get the changes done in the Palm, generating the new GML over the original (it means, adding, deleting or updating the GML tags) (Gossett, 2000).

While the first approach has the advantage of being able to identify all changes performed over the original GML, most of the time this isn’t supposed to happen, because the kind of use we were focused was only on updating data on interest points, and not the map itself.

In order to evaluate the proposed architecture, a sample application was made. We took the Mordomo OPorto city map, and begun marking the interest points needed by the Percursos Agustinianos project in the context of the Porto 2001 European Capital of Culture. The project dealt with places in the city which were described in the Agustina Bessa Luis books. We did some annotations, using a Palm connected to a GPS, followed by a synchronization on the data server so that the annotations were inserted into a new GML file.

### 6 Conclusion

Since data was transferred from GML into the Palm device, and vice-versa, we have shown how to use GML in order to provide the means to update geographic data using PDAs.

This is an evolving project. We are trying to extend it in order to make a common library that can be called back from other applications, now it is closely integrated with Mordomo, as such it does not work well with other applications. To do so, we are developing an Application Programming Interface (API) that can be easily incorporated in other systems (instead of providing special support for any system, we want it to be as parameterized as possible).

This work was developed in order to satisfy a requirement of a large scale project. As the project is, metadata can be modified in a Palm device and synchronized to the server. Support for XML Schemas is guaranteed (as long as it is a GML-compatible schema), thus it is not tied to a specific solution or a single application.

### REFERENCES


