A framework for dynamic updates of map data in mobile devices

Hae-Kyong Kang*

Department of Geographic Information Science
Office 406, C-26
Pusan National University
Pusan 609–735, South Korea
E-mail: hkang@pnu.edu
*Corresponding author

Ki-Joune Li

Department of Computer Science and Engineering
Office 406, C-26
Pusan National University
Pusan 609–735, South Korea
E-mail: lik@pnu.edu

Min-Soo Kim

Telematics. USN Research Division, ETRI
161, Kajeong-dong, Yuseong-gu
Daejeon, South Korea
E-mail: minsoo@etri.re.kr

Abstract: Stored map services in mobile devices, such as cellular phones, are being commercialised for the mobile and wireless environment. In order to ensure the quality and accuracy of the map, an update of the source map must be transferred automatically to mobile devices. While transferring the source map, at least two facts need to be considered, which are reduction of communication cost and lack of hardware capacity of mobile devices. As a compromise solution for these conflicting requirements, we propose a framework for updating spatial data on mobile devices. A strategy of our framework is to process all of the complicated transactions due to updates, such as the re-creation of new data set and consistency checking on the server side, not on mobile devices. As a result of the processing, update messages indicating spatial objects to be updated are created. The spatial objects will be replaced with original objects by an application on a mobile device. At the end of this paper, we show several snapshots via a simple prototype which we had been implementing.

Keywords: dynamic update of spatial data; mobile data service.

Reference to this paper should be made as follows: Kang, H-K., Li, K-J. and Kim, M-S. (2007) ‘A framework for dynamic updates of map data in mobile devices’, *Int. J. Web Engineering and Technology*, Vol. 3, No. 2, pp.176–195.

Biographical notes: Hae-Kyong Kang received her PhD in Geographic Information Science at the Pusan National University. Her research interests includes spatial data modelling, dynamic update of multi-scale spatial data and ego-centric mapping on a mobile environment.

Ki-Joune Li received his PhD in Computer Science from Institute National des Sciences Appliquees (INSA) de Lyon France. Professor Li has been lecturing in Pusan National University, South Korea from 1993. Professor Li was Programme Co-Chair of W2GIS 2005 and was Programme Chair of ACM-GIS 2000. He has served on the programme committees of four international conferences.

Min-Soo Kim is a Senior Researcher of the engineering staff of Electronics and Telecommunications Research (ETRI) Institute. His research interests include ubiquitous sensor networks, LBS, mobile database and telematics.

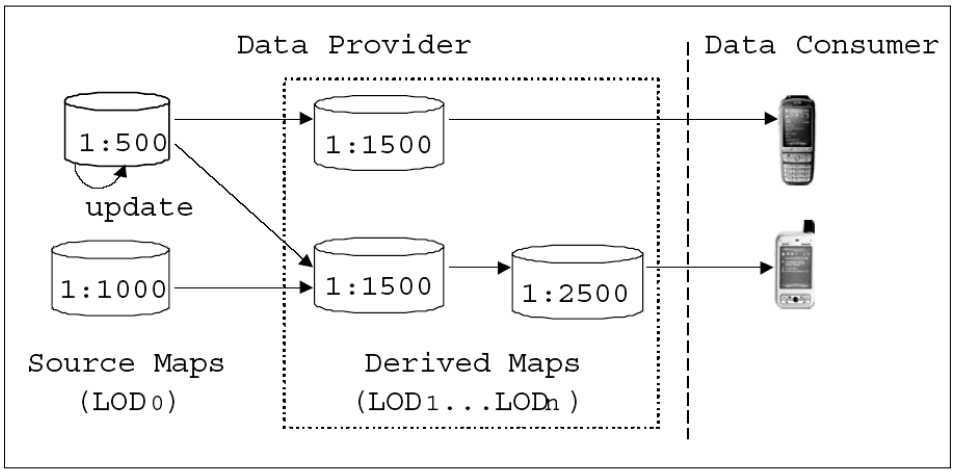
1 Introduction

With the recent progress in mobile devices, it becomes possible to store a certain volume of map data on mobile devices such as cellular phones or PDAs. For example, several commercial services are being provided for Location Based Service (LBS) and telematics applications with road map data stored on cellular phones. In order to maintain the accuracy and quality of map stored on mobile devices, the map data must be up to date. If updates take place on the source map databases, they must be transferred to every mobile device, where the source map databases are managed by a server. We should carefully consider several points to efficiently handle the update of map data on the server and mobile devices.

The map data distributed into a server and a large number of mobile devices has several important characteristics differing from the conventional distributed databases for several reasons. First, the updates for map services take place only at the server side, while they can occur at several sites in conventional distributed database systems. This difference makes the update management of map services simple. The second difference comes from the communication cost. For example, the cost of communication between the server and cellular phones is to be paid by subscribers, and becomes a crucial factor. Therefore, we should reduce the size of communication between the map data server and mobile devices. The third difference is the limited hardware capacity and battery of mobile devices. It means that a large amount of processing for updates in a mobile device may be critical.

A trade-off relationship is found between the communication cost and processing overhead in mobile devices. If we try to reduce the communication cost, it often results in an increase of computing on a mobile device, and vice versa. Suppose that an update occurs on the source map (LOD_0), and it must be propagated to correspondent objects on levels of details (LOD) derived from the source map. In addition, other objects, which are not even derived directly from the updated source objects, need to be modified because of the update (Kang *et al.*, 2004) to maintain spatial integrities such as topological consistency between several LODs (Egenhofer, 1994). In general, the process of update handling is much more complicated. Therefore, a mobile device is not capable of processing updates and their propagation because of its lack of hardware capacity. On the other hand, it is too expensive to transfer the entire map to mobile devices, after data providers have completely processed the updates and propagations. The price for transferring 1M bytes data to a cellular phone is approximately $US50, which is not negligible.

Figure 1 Map service to mobile devices



Therefore, we make a contribution to find a compromise solution between these contradicting requirements in this paper. That is, we propose the framework for the updating mechanism of a system under development, called MobiMap, which is a system consisting of a map server and mobile devices to provide mobile map services. A strategy of our framework is that all transactions caused by updates and their propagations, such as recreation of new dataset and consistency checking, are processed on the side of the map server, not on mobile devices. As a result of the processing, update messages representing spatial objects created owing to the updates of source maps and to be transferred to mobile devices are generated with an extended Simple Vector Graphics (SVG), which we will propose. In the extended SVG, new spatial objects, which will be replaced with original objects on mobile devices, are described with an SVG format (Andersson *et al*., 2003) and update operators based on XML. Details of the strategy will be described later in this paper.

We will develop our update mechanism under the following four constraints. As shown in Figure 1, we assume that maps on a mobile device are transferred from derived maps, not originally built ones on a data provider, even though both are applicable to maps of a mobile device. Another assumption is that the derived maps are created by only three operators, which are simplification, collapse and geo-aggregation (Kang *et al*., 2004b), even though many operators have been proposed in this research field (McMaster and Shea, 1992).

The next confine is about spatial consistency. The spatial consistency between original maps and derived maps can be evaluated by assessing their equivalences with several criteria such as object position, object existence, object dimension, object shape, object size, topological relationship and directional relationship (Abdelmoty and Jones, 1997). Choosing which criteria to apply is dependent on applications. In this paper, the topological relationships will only be considered to assess if an updated map is consistent with its original map or not. In future work, we will consider other criteria. The last constraint is about a data format for maps on mobile devices, and we will use Scalable Vector Graphics (SVG) (Andersson *et al.*, 2003) as the data format.

This paper is organised as follows. In Section 2, we will present the motivation of our research with an example and the related works. The framework for processing updates and their propagations in our MobiMap system will be presented in Section 3. The main components of our MobiMap system will be introduced. In the subsequent section, details of update handling on the side of a map server (provider) will be described based on the previous researches, including ours. As a result of the update handling, a set of update messages will be created. In Section 5, we will propose the update message format, namely, an extended SVG format. We will show several snapshots via a simple prototype, which we have implemented in Section 6. Finally, we conclude this paper in Section 7.

2 Motivations and related works

The map stored on a mobile device consists of several LODs. Thus, an update of the source map implies not only the modification of an object but also the propagation of updates to the corresponding objects of different LODs. Furthermore, its neighbouring objects may need to be modified to maintain the topological consistency between LODs.

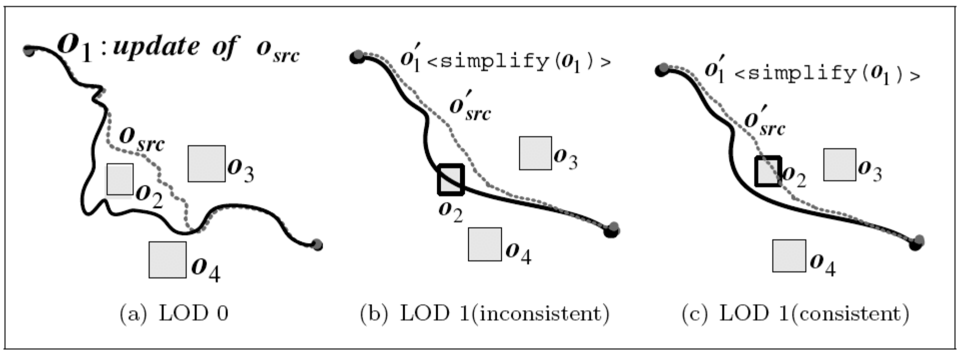
Figure 2 explains the processing of an update on the source map. In the application to areas, *e.g.*, navigation, the landmarks on the roads are extremely important and the topological relationships between roads and landmarks are crucial to drivers. An updated object o_1 does not intersect o_2 on the source map (LOD_0) as depicted in Figure 2(a), while they intersect on LOD_1 as shown in Figure 2(b), which is a generalised map from LOD_0. Because this topological inconsistency is critical, it must be corrected as shown in Figure 2(c) to ensure the topological consistency between roads and landmarks. Consequently, a simplification of an object on the source map results in not only a simplification of an object on LOD_1 but also in several modification of its neighbouring objects. In order to process an update and its propagation to other LODs, two approaches may be possible, as follows.

1 *Transfer of map data* – The processing for updates and their propagation to LODs are carried out at the server side and the map data modified is transferred to mobile devices through this approach. However, transferring a large amount of data results in expensive communication cost. The price for transferring 1M bytes data to a cellular phone is approximately $US50, which is not negligible.

2 *Transfer of update logs* – According to this approach, we transfer only update log data to mobile devices. It is evident that the communication cost is greatly reduced by this approach. However, each mobile device should process the update and its propagation to several LODs, which are large amounts of overhead to a small device like a cellular phone. This processing requires a lot of energy as well.

However, these two approaches are not feasible since each approach focuses on only one of two contradicting requirements, namely the reduction of communication cost versus the processing overhead of mobile devices. Therefore, both approaches fail to satisfy the two requirements. In this paper, we focus on a compromise between the two opposite requirements.

Figure 2 Topological consistency for line simplification



The update service for maps on mobile devices has been studied by several research groups in parallel. A few researches, such as ActMap (Otto *et al.*, 2004), have been done, and the results published. However, the ActMap used Geographic Data File (GDF) (Ertico, 2006) as its spatial model, and transferred them with an XML-based exchange format to mobile devices, such as a car navigation system. But the GDF was not considered the lack of hardware capacity of mobile devices unlike SVG which is the standard data format for maps of mobile devices.

Besides, derivation of new maps from a base map is necessary to provide various maps required for mobile devices. The ActMap assumed, however, that the maps on mobile devices provided from a data server were originally built maps (LOD_0), not derived maps. Thus, the ActMap has not considered update propagation caused by the update of source objects. Therefore, other approaches considering the update propagation of derived maps are needed as well.

The basic issue of this paper is related with how to process an update and its propagation to multiple LODs to ensure geometric and topological consistency. A number of works have been done to maintain the consistency between multi-scale or multiple LOD spatial databases, but few of them deal with the topological consistency except Sharma (1996); Tryfona and Egenhofer (1997); Kang *et al.* (2004b). Several methods have been proposed for maintaining topological consistency depending on the operation type of generalisation. In Tryfona and Egenhofer (1997), a method to assess the topological integrity has been proposed for aggregation operator. A similar work has been carried out for collapse operator in Kang *et al.* (2004a). Sharma has proposed a method to ensure topological integrity for line simplification operator in Sharma (1996). These works provide a theoretical background for our study and development of our MobiMAP manager.

3 Update framework of MobiMap Manager

In this section, we present the framework of update mechanism designed for the MobiMAP Manager, which is under development by our research team. The main focus in designing the framework is on how to reduce the communication cost as well as processing overhead of mobile devices because the hardware capacity of mobile devices is limited.

Our strategy is that all transactions caused by updates, such as re-creation of new dataset and consistency checking, will be processed on the side of the map provider, not on mobile devices. Generated through the processes are update messages representing spatial objects created as a result of the updates of source objects and to be updated on a mobile device. It reduces the communication cost because update messages include parts of objects to be updated on a mobile device, instead of entire objects.

Figure 3 shows the main components of the MobiMAP manager and their interactions to handle updates and their propagation. The handling processes are divided into five steps, and all of steps, except Step 5, are running on the server side.

1 Step 1 – Update on LOD_0

Updates of source objects take place on the source database (LOD_0). Then, update propagation is called to propagate the updates to its derived databases ($LOD_1 \ldots LOD_n$). We assumed that the derived LODs were created by only three operators: simplification, collapse and geo-aggregation (Kang *et al.*, 2001) in Section 1.

2 Step 2 – Geometric propagation on LOD_n

Update propagation is to update spatial objects derived directly as well as indirectly from the updated source objects. For this, the update propagation first searches derivation information (Kang *et al.*, 2004b) such as a generalisation operator used in a derivation process to retrieve correspondent spatial objects from the updated source objects. Then, the update propagation creates new spatial objects by calling generalisation operators. The newly created objects will replace the retrieved spatial objects, or will be inserted into the derived maps after a consistency assessment process on Step 3.

While processing the geometric propagation, contradiction on spatial integrities of derived maps, such as a topological relationship integrity, may occur. Therefore, the new spatial objects created must be verified before being stored. For this, the consistency handler is called.

3 Step 3 – Handling consistency

This step consists of two subtasks, which are assessment and correction of inconsistency.

- *Consistency assessment* – We assumed that topological relationship equivalence would only be considered as an assessment criterion of spatial consistency in Section 1. The *consistency handler* verifies whether or not the new spatial objects created in Step 2 are consistent with its source objects in terms of the topological relationship.

For each LOD derived from LOD_0, neighbouring objects in derived LODs are checked to assess topological consistency with LOD_0, because geometric propagation takes place during the changes of topological relationships between neighbouring objects (Kang and Li, 2005). The methods for assessment of topological consistency (Sharma, 1996; Tryfona and Egenhofer, 1997; Kang *et al*., 2004a; Kang and Li, 2005) will be explained in the next section.

- *Topological propagation on LOD_n* – If any topological inconsistency is found at an LOD, the neighbouring objects must be corrected to ensure the consistency with LOD0. A few researches (Ruas, 2002; Duchene, 2004) have been done in this field. However, our MobiMAP just reports inconsistent objects to users, and does not correct them. In the future, the MobiMAP will provide the correction method.

While inconsistent results are being reported, the derived LOD is updated with consistent objects and the consistent objects are transferred to mobile devices. For this transfer, the *update manager* is called.

4 Step 4 – Transfer of updated objects to each mobile device

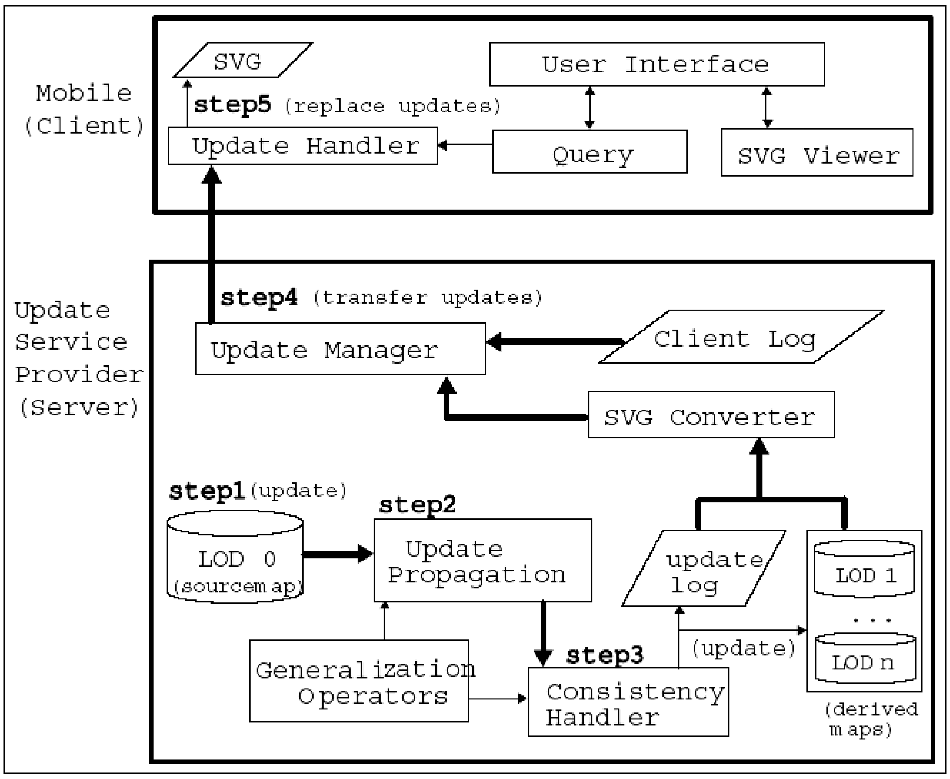
The *update manager* of the server transfers the updated objects, which are assessed to be consistent in Step 3, to each mobile device subscribed to get this update service. For this transfer service, the *update manager* searches the *client log* to retrieve clients subscribed in the update service, and the metadata of spatial data downloaded by the clients. Then, the consistent objects (updates) will be converted to an exchange format, an extended SVG, which will be proposed in Section 5.

5 Step 5 – Updates on mobile devices

The updates received from the server with an extended SVG format are to be reflected on a mobile device by the *update handler*, which is an interpreter of the extended SVG format. The *update handler* replaces the original objects in original SVG files with the updates in an extended SVG file using the ID (identifier) of spatial objects as a reference key. The updated SVG file can be shown by current applications supporting an SVG format after the replacement, because the structure of spatial objects in the extended SVG format is the same as the one in the SVG format. However, an extended SVG file is not interpreted by the current applications owing to several additional XML tags added for representing update information, thus, we implemented the *update handler*.

We will discuss in more detail the creation process of update messages (Step 2 and Step 3) in Section 4. Then, we will describe an exchange format, namely, an extended SVG format to transfer the update messages to a mobile device in Section 5. The replacement process in Step 5 by the *update handler* is simple, thus we omit its explanation.

Figure 3 Framework of update mechanism



4 Creation of update messages

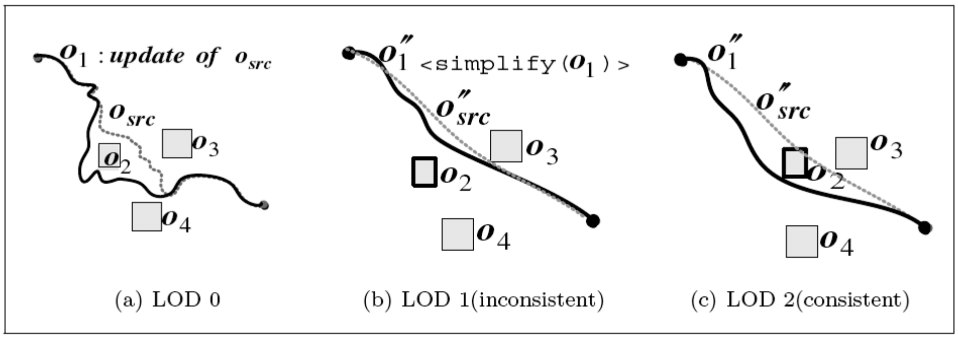
We will discuss a creation process of update messages consisting of two tasks: geometric propagation processed by *update propagation* and consistency handling by *consistency handler* in the MobiMAP. The processes of these two tasks will be explained based on our previous works and other researches for the three generalisation operators that we assumed in Section 1.

4.1 Creation of update messages for line simplification

A *line simplification* operator removes vertices of less importance from a line with a large number of vertices. The most popular algorithm of line simplification is proposed by Douglas and Peuker (McMaster and Shea, 1992). There is a one-to-one mapping between the original line and the simplified line, the difference between LOD_0 and derived $LOD_n(n \geq 0)$ is only found in geometric properties. It means that the topology in LOD_0 should be the same with the topology in derived LOD_n. If topology is different from each other, LOD_0 and LOD_n are considered as topologically inconsistent (Sharma, 1996).

In Figure 2, we already showed an example of topological inconsistency through line simplification. In addition, Figure 4 shows another example of directional inconsistency where LOD_0 is a source map, and LOD_1 and LOD_2 are derived maps from LOD_0. o_{src} in LOD_0 is an original line object and o_1 is the updated object from o_{src}. o''_{src} in LOD_1 and LOD_2 are the simplified objects from o_{src} in LOD_0. o''_1 in LOD_1 and LOD_2 are the simplified objects from o_1 in LOD_0.

Figure 4 Directional consistency for line simplification



We observe that o_2 is at the south-west of o''_1 in LOD_1 as shown in Figure 4(b), whereas o_2 is at the north-east of o_1 in LOD_0 in Figure 4(a). Because of the topological difference, LOD_1 is inconsistent with LOD_0. Thus, we must simplify o'' again or move o_2 or both, like Figure 4(c) so that the topologies between LOD_0 and LOD_1 are identical (Lonergan and Jones, 2001; Ruas, 2002).

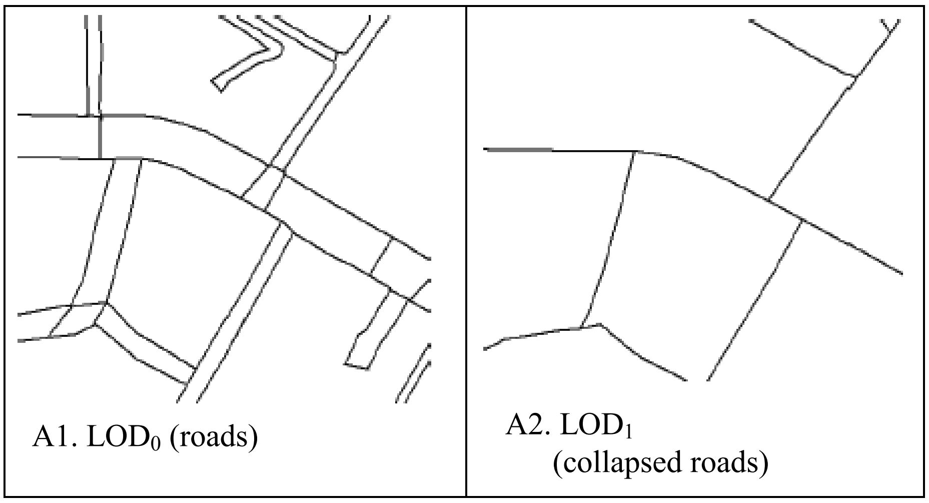
If o'' is simplified recursively without modifying any other objects such as o_2, o'' will be stored in an update message. And o''_{src} will be replaced with the o''. On the other hand, if o_2 is displaced and o'' is simplified again, then both o'' and o_2 will be written in an update message. And they will be replaced with the original ones.

4.2 Creation of update messages for collapse

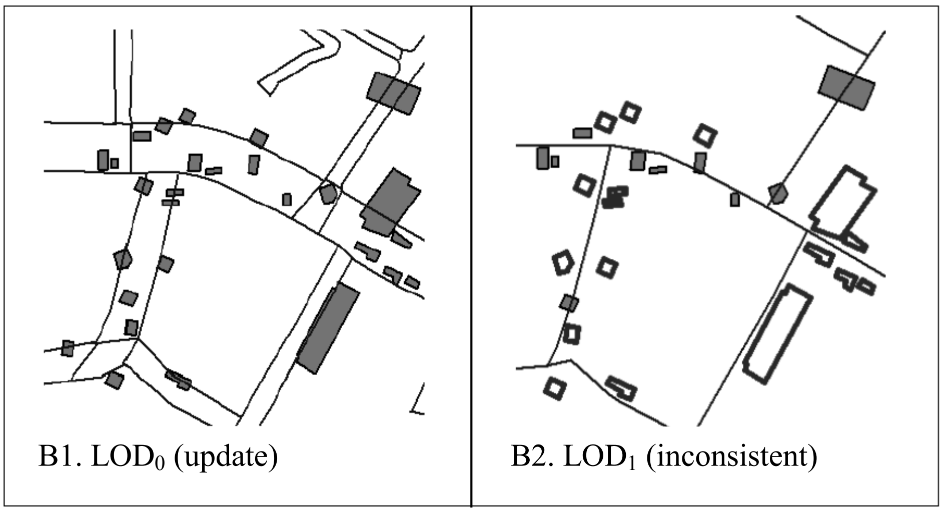
A *collapse* operator reduces the dimension of a spatial object, for example, from polygon to line or from polygon to point (Kang *et al.*, 2001). Because of the dimension change, topologies in original maps and their derived ones cannot be the same. For example, if a polygon *a* containing another polygon *b* is collapsed to a point *a'*, *a'* does no longer contains *b*. Consequently, the topological consistency for the collapse operator does not imply that the two topologies between original LOD_0 and its derived LOD_n are identical.

For this reason, the assessment of topological consistency for a *collapse* operation is a little more complicated than *line simplification*. A way to assess the topological relationship for a collapse operator has been proposed by one of our previous works (Kang and Li, 2005). In the work, consistent correspondence between polygon relationships and polygon-line relationships was proposed based on set properties, such as set-inclusion or exclusive-set. Consistent correspondence between polygon relationships and line relationships is being prepared for a publication.

Figure 5 Topological consistency for collapse



(a) original and derived road maps



(b) detection of inconsistent relations (white polygons) after update (insert) of new facilities into LOD_0

As an example of the consistent correspondence, if a polygon A is contained by a polygon B, and A is collapsed to a line A', then a consistent relationship between A' and B is PL_9 (Kang and Li, 2005). We omit the derivation process of consistent relationships in this paper.

Figure 5 shows an example of inconsistent topological relationships caused by update (insert). Initially, we have a roads map on LOD_0. LOD_1 is derived by collapsing the roads (polygons) on LOD_0 into lines as shown in Figure 5(a). Then, several road facilities are inserted into the map of LOD_0.

All facilities intersect with road polygons on LOD_0 as shown in Figure 5(b)-B1. But non-intersect facilities with road polygons on LOD_1 are observed in Figure 5(b)-B2. These inconsistent objects should be displaced by using a displacement method (Lonergan and Jones, 2001), which is our future work, so that they intersect with road polygons.

However, the current MobiMAP just reports the inconsistent facilities (white polygon) in Figure 5(b)-B2 to users, and writes consistent facilities in an update message. The objects in the update message will be inserted into correspondent-derived LOD_n, and transferred to mobile devices.

4.3 Creation of update messages for geo-aggregation

A geo-aggregation operator merges a set of objects satisfying predefined conditions, such as a distance given by a user into a new object. It reduces the total number of objects as well as the original shape of objects as a result of merging. Therefore, the topological consistency for the geo-aggregation operator does not imply that the two topologies between original LOD_0 and its derived LOD_n are identical.

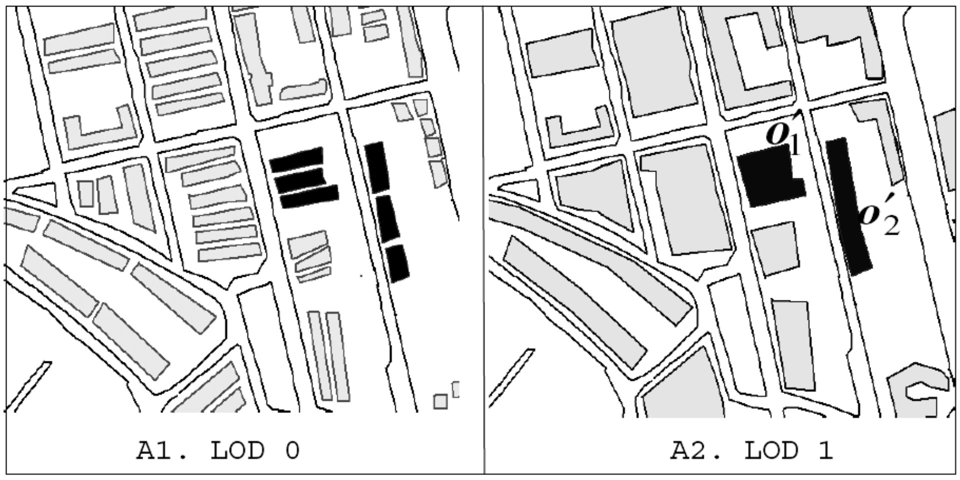
For this reason, the topologies of the original objects and their derived objects must be carefully considered for assessing their topological consistency. If an update occurs to an object on LOD_0, which is aggregated to an object in LOD_1, the aggregated object in LOD_1 must be re-aggregated taking the update on LOD_0 into account. Moreover, the topology on LOD_0 must be respected during the re-aggregation process as shown in Figure 6.

Figure 6 shows an example of inconsistent topological relationships owing to update (insert and geometry modification). Figure 6(a) shows the initial states of LOD_0. LOD_1 is created by aggregation of objects on LOD_0 in Figure 6(a)-A2. Then updates occur on LOD_0; three new objects o_1, o_2 and o_3 are inserted into LOD_0, and the shapes of o_4 and a road are changed as shown in Figure 6(b)-B1. Because of the modification of the road shape, o_1, o_2, o_3 and o_4 can be merged into one object. Consequently, a new aggregated object o' on LOD_1 is computed in Figure 6(b)-B2.

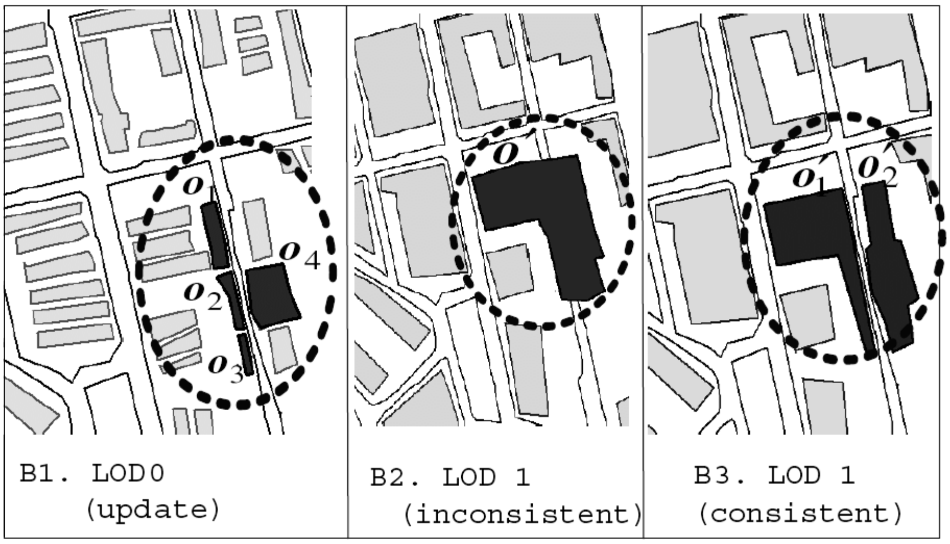
In Tryfona and Egenhofer (1997), a reasoning for consistent spatial relationship for the geo-aggregation operator was proposed by defining inference rules and a connected area among parts (original objects) based on the *meet* relationship between parts and the connected area. For an example of the reasoning, let *A*, *B* and *C* be a polygon. Let *X* be a connecting area between *A* and *B*. *A*, *B* and *X* are aggregated to a polygon *A**.

According to the Tryfona and Egenhofer (1997), if *A* and *B* are disjointed with *C* and *X* meets *C*, *A** meets *C*. In another example of reasoning (Tryfona and Egenhofer, 1997), if *A* and *B* are disjointed from *C* and *X* contains *C*, then *A** contains *C*.

Figure 6 Topological consistency for aggregation



(a) original and derived road maps



(b) update of the original map and its propagation results

In Figure 6, according to Tryfona and Egenhofer (1997), it can be assessed that the derived relationship between *o'* and the road in Figure 6(b)-B2 is consistent, and this result ignores the disjoint relationship between the roads and facilities. To complement this weakness, in our MobiMAP manager, we extended the Tryfona and Egenhofer (1997) by constraining relationships of the connected area to consider semantic of spatial objects as follows:

When a connected area C is created, R' has to be a subset of $R(R \subset R')$ where R' is a set of relationships between C and a set of the same type of spatial objects. S, S_i and S_j represent a different type of spatial objects, R is a set of relationships between S_i and S_j. For example, facilities o_1, o_2, o_3 and o_4 are disjointed with the road, then their connected areas have to disjointed with the road. Consequently, their aggregation has to be disjointed with the road.

According to our extension, the result of update propagation in Figure 6(b)-B2 is inconsistent, whereas the one in Figure 6(b)-B3 is consistent. o_1' and o_2' will be written in an update message, and transferred to mobile devices using LOD_1. The update message will be described in an exchange format, which we will propose in the subsequent section.

5 Extended SVG format to transfer update messages to mobile devices

We will propose an extended SVG format as an exchange format of the update messages explained in the previous section to mobile devices. Spatial objects described in the extended SVG format are interoperable with the ones of SVG format by using an interpreter such as the *update handler* of the MobiMAP, but the extended SVG format itself is not supported by current applications supporting SVG files, such as Microsoft Internet Explorer or Adobe Acrobat.

In the extended SVG format, only spatial objects in update messages will be described with the SVG format since we assumed SVG as a data format for maps on mobile device in Section 1, and update operators will be described based on XML. Figure 7 shows the conceptual model of the extended SVG format. In the class diagram, a map consists of several update operators, because different update can take place on a map several times. Each update operator is described with XML tags and consists of updated objects, which are spatial objects in the SVG format, such as path or polygon.

Figure 7 Conceptual class diagram of an extended SVG

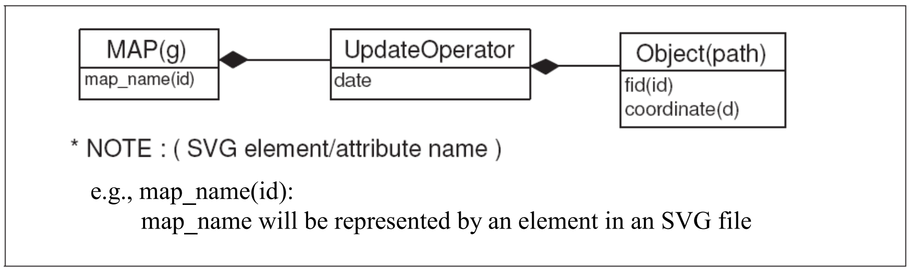
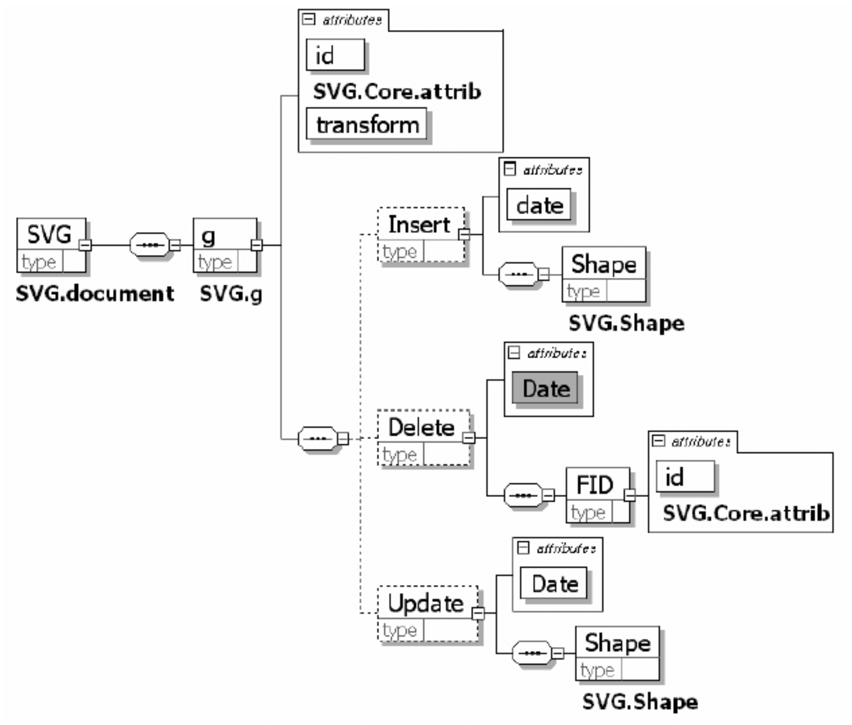
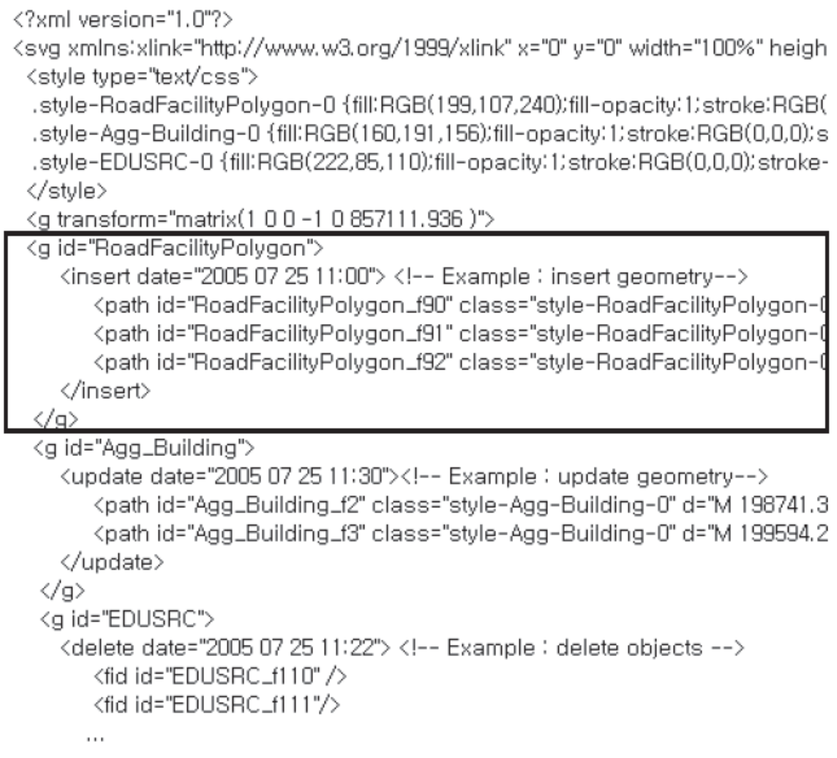

MAP(g)
map_name(id)
UpdateOperator
date
Object(path)
fid(id)
coordinate(d)
* NOTE : (SVG element/attribute name)
e.g., map_name(id):
map_name will be represented by an element in an SVG file

Figure 8 shows the update schema of this conceptual model generated by the *update manager* on the server in Figure 3. The update schema consists of several tags to describe update operators, an identifier of an updated object and its new geometry. In Figure 8, the SVG element indicates the beginning of an extended SVG document, and consists of *g* elements representing maps. Each *g* element consists of a set of maps with transform conditions *transform* for coordinate systems and *id* corresponding to the name of a map, for instance *RoadFacilityPolygon,* and a set of update operators such as insert, delete and update.

Figure 8 Update schema and example





```
<?xml version="1.0"?>
<svg xmlns:xlink="http://www.w3.org/1999/xlink" x="0" y="0" width="100%" heigh
 <style type="text/css">
  .style-RoadFacilityPolygon-0 {fill:RGB(199,107,240);fill-opacity:1;stroke:RGB(
  .style-Agg-Building-0 {fill:RGB(160,191,156);fill-opacity:1;stroke:RGB(0,0,0);s
  .style-EDUSRC-0 {fill:RGB(222,85,110);fill-opacity:1;stroke:RGB(0,0,0);stroke-
 </style>
 <g transform="matrix(1 0 0 -1 0 857111.936 )">
 <g id="RoadFacilityPolygon">
     <insert date="2005 07 25 11:00"> <!-- Example : insert geometry-->
         <path id="RoadFacilityPolygon_f90" class="style-RoadFacilityPolygon-
         <path id="RoadFacilityPolygon_f91" class="style-RoadFacilityPolygon-
         <path id="RoadFacilityPolygon_f92" class="style-RoadFacilityPolygon-
     </insert>
  </g>
  <g id="Agg_Building">
     <update date="2005 07 25 11:30"><!-- Example : update geometry-->
         <path id="Agg_Building_f2" class="style-Agg-Building-0" d="M 198741.3
         <path id="Agg_Building_f3" class="style-Agg-Building-0" d="M 199594.2
     </update>
   </g>
   <g id="EDUSRC">
     <delete date="2005 07 25 11:22"> <!-- Example : delete objects -->
         <fid id="EDUSRC_f110" />
         <fid id="EDUSRC_f111"/>
        ...
```

Update operators are mandatory when the *id* is specified in the *g* element. Depending on the types of update operators, different attributes are included in the update schema. For example, updated geometries are needed for *insert* and *update* types of update, but are not needed for *delete*. Each update type has attribute *Date* describing the time stamp of the update transfer. Updated geometry is specified by the core attributes of an SVG document, such as *path*. However, we assumed that the geometry *id*, for instance *pathid* = "*RoadFacilityPolygon_f90*" is unique. Figure 8 shows an example of the extended SVG for the three update cases: insertion of new objects into *RoadFacilityPolygon*, modification of two objects on *AggBuilding*, and deletion of several objects on *EDUSRC*.

This SVG document is to be interpreted by an *update handler* in a mobile device in Figure 3 rather than directly displayed by an SVG viewer, since it contains only the updated parts of the entire map with the SVG-extended format based on XML. The *update handler* of a mobile device must replace the old data with the new ones transferred to build new map data by comparing the extended SVG file to an original SVG file. But this processing does not result in an overhead of mobile device, since the replacement work only includes searching *id* and simple replacement operations.

6 Simple example

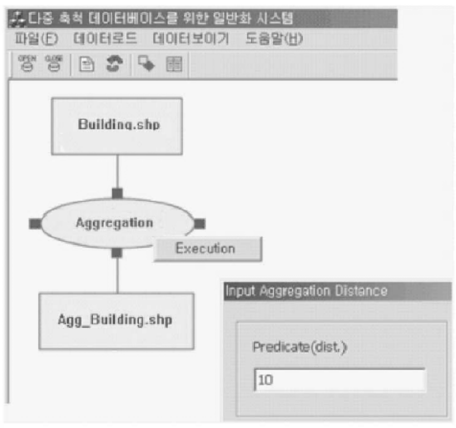
Through several snapshots of our MobiMAP manager, we will show a simple example of dynamic update. The MobiMAP manager has been developed on Windows 2000 with VC++, Java, and ArcObjects. We used syntactic data because real data do not reflect all of the cases that we have to consider.

In Figure 9, *Agg_Building* is derived from *Building* by geo-aggregation operator merging objects within distance. As a result, ten objects are created in *Agg_Building*. *SVG converter of MobiMAP* manager converts the *Agg_Building* to SVG format in Figure 9-(Agg_Building.SVG).

After initiating a mobile device with *Agg_Building*, its original data *Building* is updated by inserting several objects in Figure 10. With the update of *Building*, *Agg_Building* must be updated. *Update Propagation of MobiMAP* manager handles the update of derived classes such as *Agg_Building*. The process of update propagation was proposed in Kang *et al*. (2004b). After the update process, one object is updated on *Agg_Building*, three objects are inserted, and nine objects are deleted.

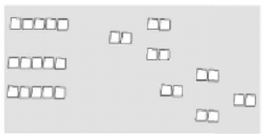
For transferring the updates to clients, *Update Manager* generates a document with the extended SVG format in Figure 11. The *update handler* of the mobile client replaces the original objects with updated ones written in the extended SVG document shown in Figure 12.

Figure 9 Generating initial data for mobile device

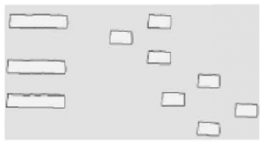

다중 축척 데이터베이스를 위한 일반화 시스템
파일(F) 데이터로드 데이터보이기 도움말(H)
Building.shp
Aggregation
Execution
Agg_Building.shp
Input Aggregation Distance

Predicate(dist.)
10

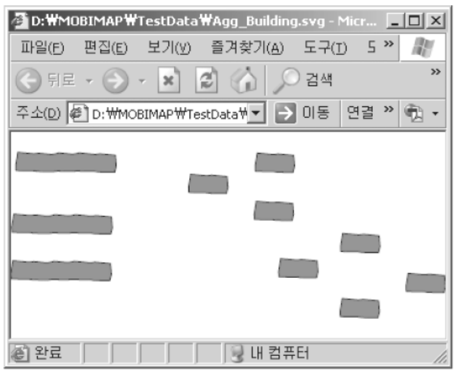
Modelling Tool

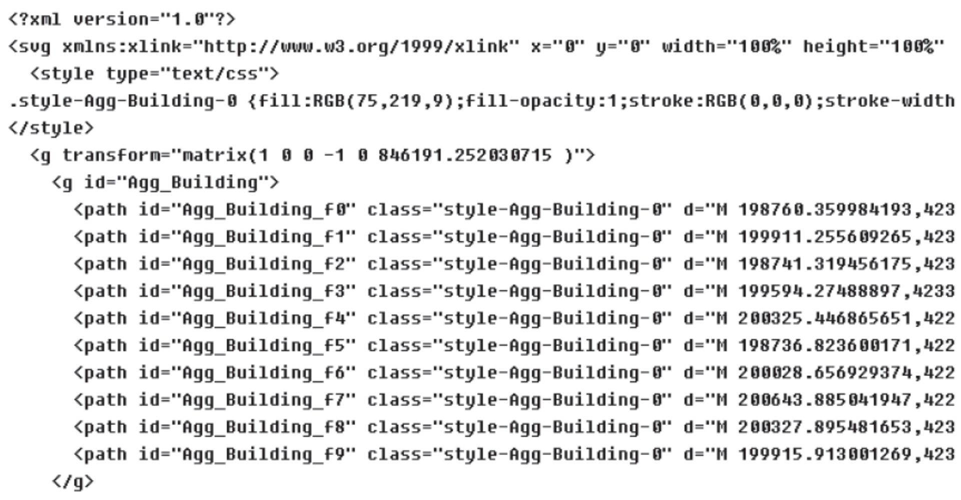


BasedDB(Building.shp)



LOD_1(Agg_Building.shp)


D:₩MOBIMAP₩TestData₩Agg_Building.svg - Micr...
파일(F) 편집(E) 보기(V) 즐겨찾기(A) 도구(T)
뒤로
검색
주소(D) D:₩MOBIMAP₩TestData₩
이동 연결
완료
내 컴퓨터

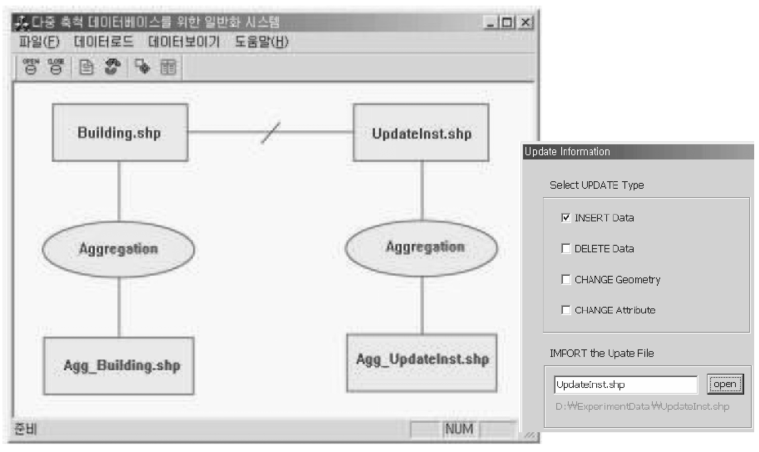


```
<?xml version="1.0"?>
<svg xmlns:xlink="http://www.w3.org/1999/xlink" x="0" y="0" width="100%" height="100%"
  <style type="text/css">
.style-Agg-Building-0 {fill:RGB(75,219,9);fill-opacity:1;stroke:RGB(0,0,0);stroke-width
</style>
  <g transform="matrix(1 0 0 -1 0 846191.252030715 )">
    <g id="Agg_Building">
      <path id="Agg_Building_f0" class="style-Agg-Building-0" d="M 198760.359984193,423
      <path id="Agg_Building_f1" class="style-Agg-Building-0" d="M 199911.255609265,423
      <path id="Agg_Building_f2" class="style-Agg-Building-0" d="M 198741.319456175,423
      <path id="Agg_Building_f3" class="style-Agg-Building-0" d="M 199594.27488897,4233
      <path id="Agg_Building_f4" class="style-Agg-Building-0" d="M 200325.446865651,422
      <path id="Agg_Building_f5" class="style-Agg-Building-0" d="M 198736.823600171,422
      <path id="Agg_Building_f6" class="style-Agg-Building-0" d="M 200028.656929374,422
      <path id="Agg_Building_f7" class="style-Agg-Building-0" d="M 200643.885041947,422
      <path id="Agg_Building_f8" class="style-Agg-Building-0" d="M 200327.895481653,423
      <path id="Agg_Building_f9" class="style-Agg-Building-0" d="M 199915.913001269,423
    </g>
```

Agg Building.SVG

(The right part of the SVG source is omitted.)

Figure 10 Generating updated data on server side



Update Information

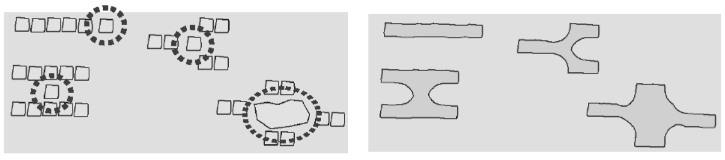
Select UPDATE Type

- [✓ INSERT Data]
- ☐ DELETE Data
- ☐ CHANGE Geometry
- ☐ CHANGE Attribute

IMPORT the Upate File

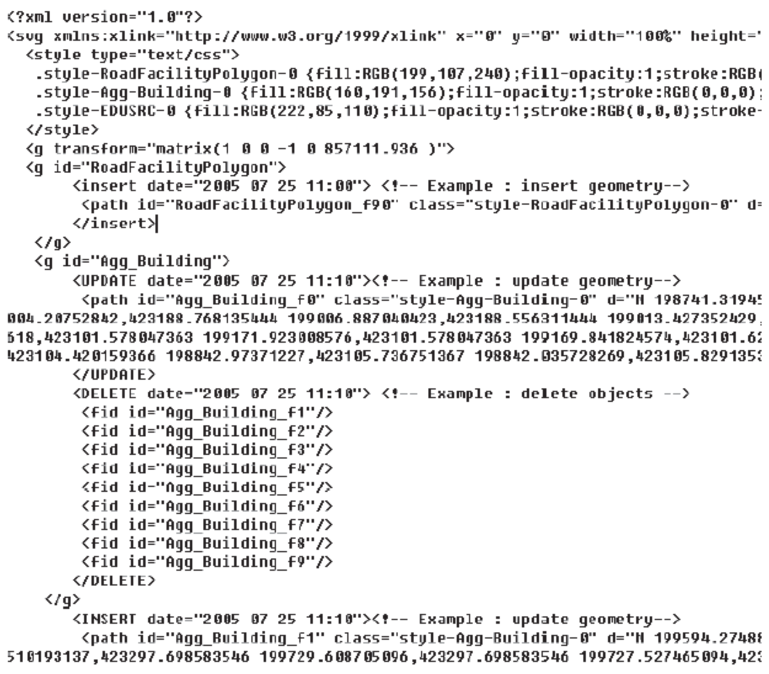
UpdateInst.shp [open]

D:₩ExperimentData₩UpdateInst.shp



BaseDB Updated LOD$_1$ Updated

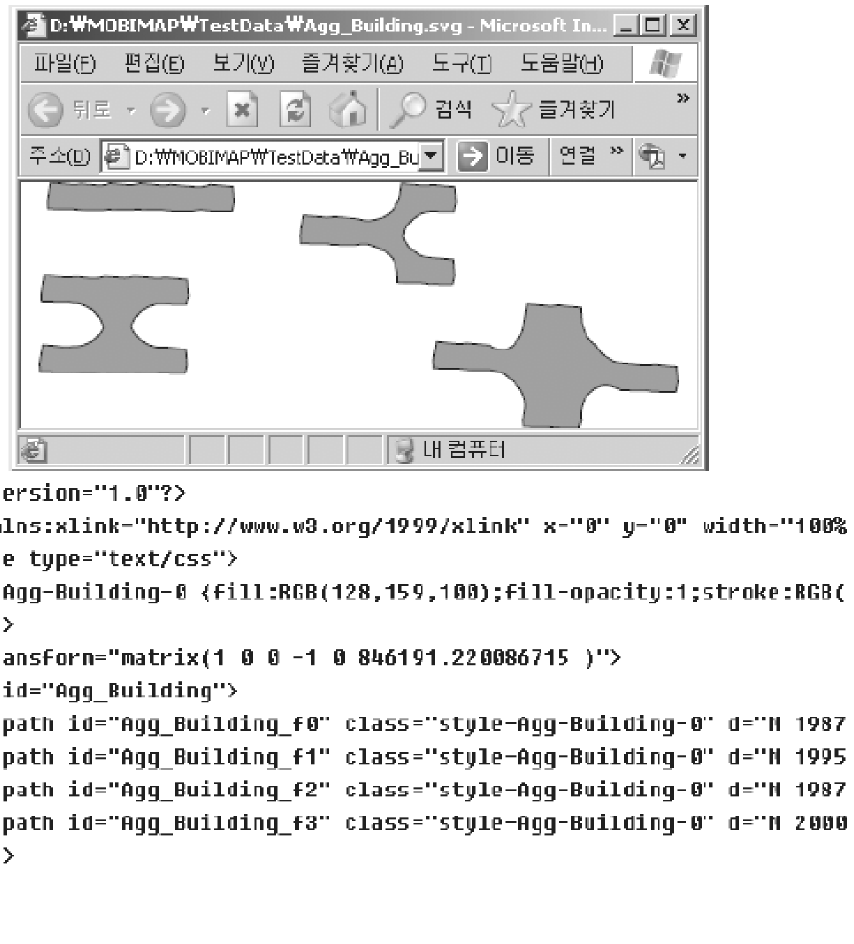
Figure 11 Update schema with the extended SVG format



```
<?xml version="1.0"?>
<svg xmlns:xlink="http://www.w3.org/1999/xlink" x="0" y="0" width="100%" height='
  <style type="text/css">
   .style-RoadFacilityPolygon-0 {fill:RGB(199,107,240);fill-opacity:1;stroke:RGB(
   .style-Agg-Building-0 {fill:RGB(160,191,156);fill-opacity:1;stroke:RGB(0,0,0);
   .style-EDUSRC-0 {fill:RGB(222,85,110);fill-opacity:1;stroke:RGB(0,0,0);stroke-
  </style>
  <g transform="matrix(1 0 0 -1 0 857111.936 )">
  <g id="RoadFacilityPolygon">
       <insert date="2005 07 25 11:00"> <!-- Example : insert geometry-->
        <path id="RoadFacilityPolygon_f90" class="style-RoadFacilityPolygon-0" d=
       </insert>
   </g>
   <g id="Agg_Building">
       <UPDATE date="2005 07 25 11:10"><!-- Example : update geometry-->
        <path id="Agg_Building_f0" class="style-Agg-Building-0" d="M 198741.3194
004.20752842,423188.768135444 199006.887040423,423188.556311444 199013.427352429
518,423101.578047363 199171.923008576,423101.578047363 199169.841824574,423101.6
423104.420159366 198842.97371227,423105.736751367 198842.035728269,423105.829135
       </UPDATE>
       <DELETE date="2005 07 25 11:10"> <!-- Example : delete objects -->
        <fid id="Agg_Building_f1"/>
        <fid id="Agg_Building_f2"/>
        <fid id="Agg_Building_f3"/>
        <fid id="Agg_Building_f4"/>
        <fid id="Agg_Building_f5"/>
        <fid id="Agg_Building_f6"/>
        <fid id="Agg_Building_f7"/>
        <fid id="Agg_Building_f8"/>
        <fid id="Agg_Building_f9"/>
       </DELETE>
    </g>
       <INSERT date="2005 07 25 11:10"><!-- Example : update geometry-->
        <path id="Agg_Building_f1" class="style-Agg-Building-0" d="M 199594.2748
510193137,423297.698583546 199729.608705096,423297.698583546 199727.527465094,42
```

(The right part of the SVG source is omitted.)

Figure 12 Replacing original data with update data


D:₩MOBIMAP₩TestData₩Agg_Building.svg - Microsoft In...
파일(F) 편집(E) 보기(V) 즐겨찾기(A) 도구(T) 도움말(H)
뒤로
검색
즐겨찾기
주소(D) D:₩MOBIMAP₩TestData₩Agg_Bu
이동
연결
내 컴퓨터

```
<?xml version="1.0"?>
<svg xnlns:xlink-"http://www.w3.org/1999/xlink" x-"0" y-"0" width-"100%
  <style type="text/css">
.style-Agg-Building-0 {fill:RGB(128,159,100);fill-opacity:1;stroke:RGB(
</style>
  <g transforn="matrix(1 0 0 -1 0 846191.220086715 )">
    <g id="Agg_Building">
      <path id="Agg_Building_f0" class="style-Agg-Building-0" d="M 1987
      <path id="Agg_Building_f1" class="style-Agg-Building-0" d="M 1995
      <path id="Agg_Building_f2" class="style-Agg-Building-0" d="M 1987
      <path id="Agg_Building_f3" class="style-Agg-Building-0" d="M 2000
    </g>
  </g>
</svg>
```

(The right part of the SVG source is omitted.)

7 Conclusion

With the recent progress in mobile devices, stored map services has begun to be commercialised for cellular phones and PDAs. In order to ensure the quality and accuracy of maps, the updates on the source map must be automatically reflected to mobile devices. However, the expensive communication cost and lack of hardware capacity of mobile devices are important constraints in dealing with the updates. It implies that we should find a compromise solution between the transmission of the entire map and the transmission of only update logs to mobile devices. The transmission of completely processed map results in an expensive communication cost, while simple transfer of update logs requires a large amount of processing to maintain geometric and topological consistency between multiple LODs. Therefore, we proposed a way to transfer updated objects allowed to be interoperable with SVG documents, but neither with the entire map nor update logs.

This paper has three contributions. First, we propose a framework of update mechanism in mobile environments where the map data is stored on each mobile device. This framework provides an efficient strategy for processing an update and its propagation to multiple LODs without expensive communication costs and large amounts of processing in mobile devices. Second, several methods are introduced to maintain topological consistency between LODs, depending on the type of generalisation operators. The third contribution is the extended SVG to be used as a transfer format of updated message to mobile devices.

Acknowledgements

This research was supported by the Internet Information Retrieval Research Center (IRC) in Hankuk Aviation University. IRC is a Regional Research Center of Kyounggi Province, designated by ITEP and Ministry of Commerce, Industry and Energy, and was supported by the Programme for the Training of Graduate Students in Regional Innovation, which was conducted by the Ministry of Commerce, Industry and Energy of the Korean Government.

References

Abdelmoty, I. and Jones, C.B. (1997) 'Towards maintaining consistency of spatial databases', *Proc. of the 6th International Conference on Information and Knowledge Management (CIKM)*, pp.297–300.

Andersson, O., *et al.* (2003) 'Mobile SVG profiles: SVG tiny and SVG basic', *Technical Report of W3C (World Wide Web Consortium)*, http://www.w3.org/TR/2003/REC–SVGMobile –20030114/.

Duchene, C. (2004) 'The CartACom model: a generalisation model for taking relational constraints into account', *Proc. 6th ICA Workshop on Progress in Automated Map Generalisation*, Leicester, http://aci.ign.fr/Leicester/paper/duchene-v2-ICAWorkshop.pdf.

Egenhofer, M. (1994) 'Evaluating inconsistencies among multiple representations', *Proc. of 6th International Symposium on Spatial Data Handling*, pp.902–920.

Ertico (2006) 'Geographic Data Files (GDF)', *ERTICO*, http://www.ertico.com/en/links/links/gdf _—geographicdatafiles.htm.

Kang, H. and Li, K. (2005) 'Assessing topological consistency for collapse operation in generalization of spatial databases', *Proc. of 2nd Workshop on Conceptual Modeling for Geographic Information Systems in Conjunction with ER2005, Lecture Notes in Computer Science 3833*, Springer-Verlag, pp.66–77.

Kang, H., Do, S. and Li, K. (2001) 'Model-oriented generalization rules', *Proc. (in CD) ESRI User Conference*, San Diego, USA, July.

Kang, H., Kim, T. and Li, K. (2004a) 'Topological consistency for collapse operation in multi-scale databases', *Proc. of 1st Workshop on Conceptual Modeling for Geographic Information Systems in Conjunction with ER2004, Lecture Notes in Computer Science 3289*, Springer-Verlag, pp.91–102.

Kang, H., Moon, J. and Li, K. (2004b) 'Data update across multi-scale databases', *Proc. of the 12th International Conference on Geoinformatics*, pp.749–756.

Lonergan, M. and Jones, C.B. (2001) ‘An interactive displacement method for conflict resolution in map generalization’, *Algorithmica*, Vol. 30, pp.287–301.

McMaster, R.B. and Shea, K.S. (1992) *Generalization in Digital Cartography*, Association of American Geographers, pp.72–28.

Otto, H., Beik, L., Aleksic, M., Meier, J., *et al*. (2004) ‘ActMAP specification version 1.0’, *Report of Information Society Techonologies*, IST-2001-34141 (Deliverable D3.2).

Ruas (2002) ‘Generalization of updated data (in the context of multiple representation)’, *Joint ISPRS/ICA Workshop on Multi-Scale Representations of Spatial Data*, Ottawa, Canada, July.

Sharma, J. (1996) ‘Integrated spatial reasoning in geographic information systems: combining topology and direction’, *PhD Thesis*, The Graduate School of Spatial Information Science and Engineering, University of Maine.

Tryfona, N. and Egenhofer, M.J. (1997) ‘Consistency among parts and aggregates: a computational model’, *Transactions in GIS*, Vol. 1, No. 3, pp.189–206.

Bibliography

Balley, S., Parent, C. and Spaccapietra, S. (2004) ‘Modelling geographic data with multiple representation’, *International Journal of Geographical Information Science*, Vol. 18, No. 4, pp.327–352.

Belussi, M.N. and Pelagatti, G. (2000) ‘An integrity constraints-driven system for updating spatial databases’, *ACM-GIS*, pp.121–128.

Egenhofer, M.J. and Herring, H. (1990) ‘Categorizing binary topological relations between regions, lines, and points in geographic databases’, *Technical Report*, Department of Surveying Engineering, University of Maine.

Guo, Z., Zhou, S., Xu, Z. and Zhou, A. (2004) ‘G2ST: a novel method to transform GML to SVG’, *ACM-GIS*, pp.161–168.

Kilpelainen, T. and Saarjakoski, T. (1995) ‘Incremental generalization for multiple representations of geographical objects’, in J.P. Muller and R.W. Lagrange (Eds.) *GIS and Generalization: Methodology and Practice*, Taylor & Francis.

Paiva, J.A.C. (1998) ‘Topological consistency in geographic databases with multiple representations’, *PhD Thesis*, University of Maine, http://library.umaine.edu/theses/pdf/paiva.pdf.