한국사물인터넷학회논문지 Vol. 1, No. 1, pp. 11-19, 2015 http://dx.doi.org/10.20465/KIOTS.2015.1.1.011

Implementation of Tile Searching and Indexing Management Algorithms for Mobile GIS Performance Enhancement

Kang-Won Lee[1*], Jin-Young Choi[2]
[1,2]Dept. of Computer Science & Engineering, Korea University

Abstract The mobile and ubiquitous environment is experiencing a rapid development of information and communications technology as it provides an ever increasing flow of information. Particularly, GIS is now widely applied in daily life due to its high accuracy and functionality. GIS information is utilized through the tiling method, which divides and manages large-scale map information. The tiling method manages map information and additional information to allow overlay, so as to facilitate quick access to tiled data.
Unlike past studies, this paper proposes a new architecture and algorithms for tile searching and indexing management to optimize map information and additional information for GIS mobile applications. Since this involves the processing of large-scale information and continuous information changes, information is clustered for rapid processing. In addition, data size is minimized to overcome the constrained performance associated with mobile devices.
Our system has been implemented in actual services, leading to a twofold increase in performance in terms of processing speed and mobile bandwidth.

Key Words : GIS, tile, Map, rendering method

1. Introduction

The mobile and ubiquitous environment has created a smart workplace culture, in which users have access to information regardless of time and place. Furthermore, information and communications technology, including communication networks, terminals, and displays, have been developed to ensure functionality in mobile devices. In particular, the widespread use of technologies integrating lightweight mobile devices and communications has altered the lives of users.

Given the improved accuracy and wider coverage of Geographic Information Systems(GIS) and Global Positioning Systems(GPS), information and communications technology has been integrated with various multimedia for use in GIS services, offering functions that extend beyond simple voice calls[1-2]. Although GIS in the past only provided information to affiliated institutions, general users today are able to freely access information in real-time traffic, during travel time and for regional weather forecasts [3].

GIS functions by utilizing mutual information between geographical data sets and maps. Providing GIS information requires a method that facilitates the management of large-scale map information. The tiling method is commonly used to divide and manage maps. Map information and additional information are divided for easier management of the tiled data. While map information is less susceptible to change, additional information continuously changes. Thus, additional information is overlaid onto the basic map information

교신저자 : 최진영(choi@formal.korea.ac.kr)
접수일 : 2015년 12월 7일

after tiling. To provide map information and additional information through mobile devices, developing a new architecture that optimizes large-scale information in consideration of mobile characteristics instead of following conventional data processing is necessary.

Previous studies have provided location as additional information for the mobile environment. This paper goes a step further by proposing an architecture capable of processing additional large-scale information, similarly to fiber optics in communication networks. Since fiber optics involvethe processing of large-scale information and continuous information changes, information clustering is required for rapid processing. Minimization of data size also helps to overcome the constrained performance associated with mobile devices.

In this paper, we design and implement a new architecture and algorithms that effectively provides large-scale information for map services in mobile devices. Through actual implementation by a telecommunications company, we have verified the superior performance of our system.

This paper is organized as follows. Section 2 presents traditional methods of map information processing. The system design is described in section 3, followed by implementation results and analysis in section 4. Lastly, section 5 offers a conclusion and suggestions for future research.

2. Related Works

Rapid advancements in mobile network technology have stimulated improvements in GIS technology, which was previously limited to web services due to the issue of large-scale map information processing. However, rapid advancements in mobile devices and increased usage have prompted efforts to overcome resource weakness (memory capacity, CPU speed, network bandwidth, battery power, etc.), resulting in a diverse range of mobile applications[4].

In the early days, web-based GIS technology provided users with maps and spatial information using Active-X. Spatial information of maps were read as vectors, assessed for location information and converted to coordinates by indexing. The converted data was processed based on client device resources.

In the Web 2.0 environment, Java replaced Active-X for data processing. Google Maps is based on Asynchronous JavaScript and XML (AJAX), which is a simple JavaScript-based programming language. By utilizing server device resources instead of client device resources, users enjoy higher speeds and greater simplicity. However, with part of the map information provided by rendering, and location still reliant on the traditional vector method, the poor performance and lack of storage space when processing large-scale data must be addressed.

These problems were resolved through a caching tiling map server in [5], and verified with Google Maps. In [5], map images are stored in the form of a simple matrix to allow quick access and maximization of space. Map information is stored in tiles with a matrix notation. The tiles are then given attributes for longitude, latitude and size before storing in the file system of the cache server. The stored map tiles are numbered according to zoom level and stored in separate directories. However, as this approach leads to an increase in directory depth, more access time is required and the number of tile files grows exponentially as zoom level increases. The increase in directory paths causes problems in the operating system, thus imposing a limit on the storage of map tiles.

To improve the directory organization mechanism of [5], a tile feed architecture using tile keys was proposed in [6]. The proposed method processes data through direct access of directories in tile keys. Although rapid processing is possible, it proceeds in the order of information access and does not consider map size. When applied to mobile devices, performance eventually deteriorates as bandwidth occupancy increases.

In [7], performance was improved by using Web GIS based on Oracle MapViewer, addressing the complexity of GIS and high costs. The package-based method of [7] is difficult to customize and a disadvantage of Flash is its low security.

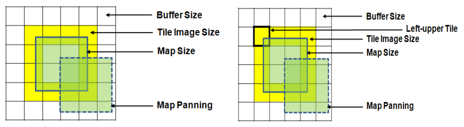
The architecture suggested in this paper processes information using Java API to ensure performance and maximize space in mobile devices. Instead of AJAX used in [5], we improve the tile key access method of [6]. AJAX was not considered because greater security is needed to protect information on business processes contained in GIS, in addition to map information. Also, more delays are experienced with AJAX when changing map scale. Even if the improved rendering method is used, a change in map scale takes up tremendous resources and consumes mobile bandwidth [8-11].

3. Design

Presently, GIS information provided through mobile devices contains not only map information, but also additional information such as road traffic and weather forecasts. Required data must be provided within a limited time frame due to space and time constraints. Nevertheless, services available on mobile devices have advanced from providing one-way information to offering added value despite restrictions on time and space. This paper proposes a new architecture to utilize large-scale map information and additional information in mobile devices.

First, we utilize the map tile rendering method, in which the view size is comprised of the changed portion. As presented in [Fig. 1] (a), the traditional tiling method generates grids based on the central value of the map and makes a tile request. When the zoom level changes, the number of tile requests resent is equivalent to the total number of tiles. By setting a buffer under options, the grid may be larger than the map size. Only neighboring tiles are requested during map panning.

Instead of sending as many requests as the number of tiles, however, we first obtain the tile index value and difference (dx, dy) for the upper left corner, as shown in [Fig. 1] (b). Based on the tile index requested to the server, a tile of the same width and height is formed and sent as an image. If a change in map level or map panning results in a different map area, a new request is made for the upper left tile. Usingtile rendering, tile images are stored in the server after being generated with XML style settings through the DB and interface. The relevant tiles are sent out upon request or generated from the rendering process, if unavailable.



(a) Traditional method (b) Proposed method

[Fig. 1] Tiling Method

Secondly, for efficient management of tile information, we propose the tile searching method, which allows quick access to directories even in mobile devices. For improved performance, the background map is processed using the raster method, while facilities and other variable objects are subject to the vector method.

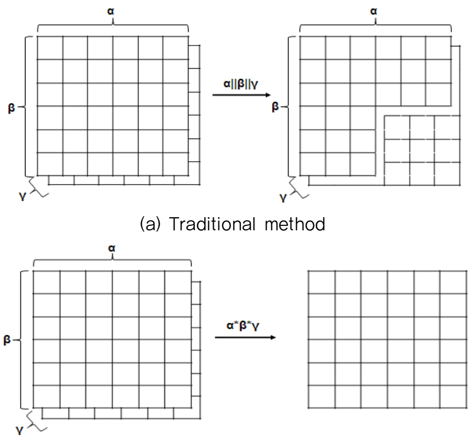
1. Set the center at level 0 as the basic coordinates.- Initial basic coordinates + coordinates during map panning = basic display (extract central values from coordinates)
2. Calculate the latitude/longitude of the 4 corners of the rectangle using function. Four values are derived from one calculation. - Extract left and upper location of the screen -> measure width and height of screen -> estimate required number of tiles (map calculation) -> extract image paths according to number of tiles (map extraction: base, line calculation)
3. Search for the relevant tile (Lan_lat_2_index(α,β)). α:Latitude/longitude(index), β: scale (1's complement), index: logic for conversion to filename - Extract image if available in the file cache- Image download from server / image extraction / map overlay (base, line, etc.)
4. Decompress the encrypted image.
5. Generate an image for the requested area.
6. Send the generated image to the client.

[Fig. 2] Pseudo-code of the proposed method

3.1 Map Tile rendering Method

In general, map information can be viewed by processing tile requests sequentially. If α, β is the number of maps and ɣ is the number of layers, the traditional method processes α, β, ɣseparately as shown in [Fig. 3]. However, considering the limited bandwidth of mobile devices, the number of tiles must be minimized.

To manage the increase in the number of tiles providing information in mobile devices, we make use of coordinates to process the layers as a single tile, thus optimizing performance. To provide large-scale information on facilities including communication optical maps, simplifying map tiles and other information is essential. Our method minimizes the number of tiles and simplifies information by merging α, β, ɣ using server resources.



(a) Traditional method

(b) Proposed method

[Fig. 3] Rendering Method

Tiles are stored separately according to Base Map and additional information (facilities information, optical, etc.). Like the majority of Web GIS, our initial settings are 256 x 256 at level 0.

While existing methods send rectangle values of each tile to the map server, our proposed method executes batch processing at the server. Since a ɣ number of requests are made depending on the number of layers, we avoid unnecessary use of bandwidth in mobile devices.

3.2 Map Tile rendering Method

To search for tiles requested by the user, the current location area of the client and scale level information is sent to the map API. The image URL of the indexed value is retrieved through the map server.

The map control of the map API connects to the relevant URL, from which the map tile image is downloaded and displayed on the user screen. A buffer search is conducted followed by map rendering, which facilitates mobile movement. Tile overlay is enabled by generating user-defined layers in the web application. Coordinate systems are matched for overlay based on map API functions such as coordinate definition and coordinate conversion.

```
public static const MISSING_TILE_URL:String="";
public static const DEFAULT_MAX_RESOLUTION:Number =
156543.03392804097;
public static const DEFAULT_NUM_ZOOM_LEVELS:uint = 21;
public var singleTileWidth:Number = 256;             //tile
size(Width)
public var singleTileHeight:Number = 256;    //tile size(Height)

Function GetURL(bounds:Bounds):String
var res:Number = this.map.resolution;
var x:Number = Math.round((bounds.left - this.maxExtent.left)/(res
* this.singleTileWidth));    //x tile index
var y:Number = Math.round((this.maxExtent.top - bounds.top) /
(res * this.singleTileHeight))              //y tile index
var z:Number = this.map.zoom;
var limit:Number = Math.pow(2, z);
var dx:Number = this.dx;
var dy:Number = this.dy;

if (y < 0 || y >= limit ||x < 0 || x >= limit) {
            return HMOSM.MISSING_TILE_URL;
} else {
   x = ((x % limit) + limit) % limit;
   y = ((y % limit) + limit) % limit;
   var url:String = this.url;
   var path:String = "z=" +z + "&y=" +y + "&x=" +x + "&dx=" +
dx + "&dy=" + dy + "&xml=" + his._xmlNm;
    if (this.altUrls != null) {
       url = this.selectUrl(this.url + path, this.getUrls());
        }
    return url + path;
}
```

[Fig. 4] Algorithms of tile searching method

3.3 Map Tile Indexing

A complement is proposed to resolve the problem of directory depth increase and to enable efficient tile management. Like Google Maps, most tile-based map information is based on Quad Tree to provide scalability. When 20 zoom levels are considered, $\sum_{i=1}^{20} 4^i$ tiles are produced. However, as the number of tiles increases, the directory depth increases as well. Instead of adopting the usual Quad Tree method, we index the main keys for conversion into filenames before processing the required images. Map information is further broken down after area division.

The traditional method makes separate requests to the server for rectangle values but the proposed method reduces I/O time by retrieving all the files from a single request. If nine tiles are required for the screen, Google Maps processes 9*tile images, whereas our proposed method provides 1*the number of tile images.

The proposed tile indexing method first obtains the central coordinates, and then calculates the latitude/longitude of the four corners. Using the coordinate system for directory management, the values areindexed and converted to directory names.

At this point, directories exist for each layer (base map layer, information layer, etc.). During information processing, layers are overlaid and compressed into a single image at the map server. UTM coordinates are directly obtained from the URL and information is retrieved from the directory if the relevant coordinates are included in the directory name. The coordinate system proposed in this paper allows more efficient directory organization, resolving the problem of directory name length caused by an increase in tile depth, as in the case in Google Maps. Moreover, it is no longer necessary to divide into separate maps and create tile keys.

With the coordinates generated under the proposed method, we can convertany coordinate system. The number of tiles to be retrieved is calculated based on the upper left coordinates and applied to the map size. For example, if the information obtained is as follows,http://sample.test:7002/tileserv/GetTile?z=17&y=406369&x=894379&dx=5&dy=5&xml=basemap

We get 900125 from calculating index x and y, and one's complement of 900125 is 900120. We applied a hash by performing a bitwise operation on this value with 0x01, resulting in -120 for the first value, 31 for the second, -60 for the third, and -4 for the fourth. A bitwise operation is performed with 0xff on the hash data, thus creating a folder structure. We then search for the image meta file. In the above case, we select the 136.meta file in the 214/181/196/31 directory at zoom level 17.

During overlay, the raster and vector methods are applied to varying information and fixed information, respectively. Since the base map is lesssubject to change, we consider changes for information in the information layers and user-defined layers.

3.4 Map Cache Method

The map cache generates a pre-rendered image set within the defined scale and stores the images in a specific map server directory. The client retrieves the map images of the corresponding scale through the index of the web application. Tiles are provided in the form of images for the location and scale requested from the client.

Since the map cache displays pre-rendered images based on the image index, it has a fast rendering speed and reduces load on the GIS server when requesting areas with multiple users and many map objects.

In this paper, we acquire cache storage space and reduce rendering time through pre-rendering of tiles containing facilities information at level 16 or higher. Also, we apply a differentiated rendering technique for varying objects stored in the cache. Instead of changing the raster information for all tiles, we apply the re-rendering technique only for tiles containing altered information, thus reducing consumed request time and load.

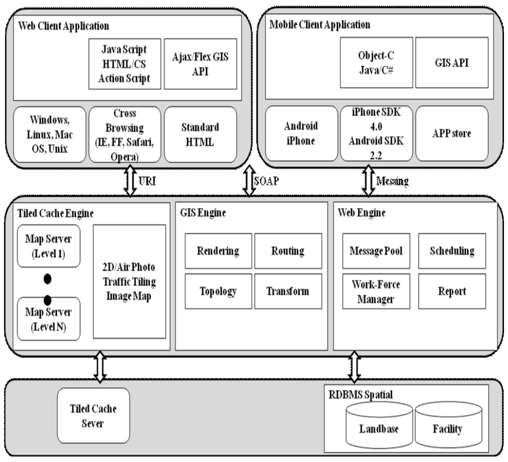
```
Function : setTileServLayerData

clearTileServImage()
tileServDownloader.panToTileServImage(tileServMapOffsetX,
tileServMapOffsetY);
tileServDownloader.downloadBitmap(Tile list, information, length,
count, width, zoom level)
tileServUrl = X index, Y index, Zoom level, number of tile X,
number of tile Y
final byte[] data = dataStream.toByteArray();
bmp = BitmapFactory.decodeByteArray(data,0,data.length);
            //convert byte to image
setWMSImage(bmp);

Function : setTileServLayerData
tileBitmap =
this.tileProvider.getBaseMapTileFromCache(tile.tileCoords,
tile.zoomLevel, i);
tile.tileBitmap.add(new TileBitmap(tileBitmap));
tileProvider.requestMapTile(requestList);
urlString=layer.urlMaker_.makeRequestURL(tile x position, tile y
position, zoom level, layer name);
bmp=BitmapFactory.decodeByteArray(data, 0,data.length);
saveFile(tileFile, data);
```

[Fig. 5] Algorithms of map cache method

3.5 System Design

The system architecture is designed for the mobile environment by separating tiling and map information. The map server is utilized for tile generation, while the rendering server transmits map areas (tile images of scale levels) requested by the Flex mobile environment.

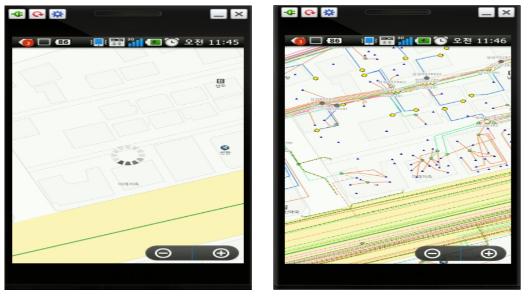
Two methods are used in the design of large-scale map information for better performance. The raster method is applied to facilities to facilitate quick access. Since optical line location is often newly created and deleted, data is stored using the vector method and grouped according to layers (base map layer, information layer and user-defined layer). The information layer executes batch processing, and the user-defined layer provides information changes in real-time

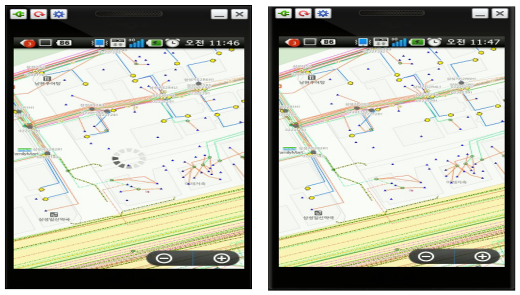


[Fig. 6] System Component Design

4. Implementation

We have proposed a map processing method to efficiently provide map information in the mobile environment, and improved processing speed by minimizing the bandwidth of map information. Our proposed methods are currently being implemented in commercial services. The web server is HP rp3440 (1GHz*4, 20GB mem), and the rendering server is HP DL570 (1GHz*4, 8GB mem). We are using LTEphones as mobile devices and Oracle 10g (for itanum) as the spatial DB. SMALLWORLD GIS data was rendered during implementation, and logical information pertaining to paths synchronized. Comparisons have been made for the measurement target based on the tile key access method proposed in [6]. 1) Use a remote object to call a java class from Flex. 2) In Flex, obtain the area to be displayed based on the base map. 3)When a request is made for the obtained area, the server conducts a search and submits results to Flex. 4) Feature items are generated in the form of polygon, line, point, etc. and styles are set. 5)Feature layers are individually generated and added on top of the map. Layers are sent to the bottom of the hierarchical structure in the order of addition. For instance, if a polygon is generated after point feature, PolygonFeature is overlaid onto PointFeature.


(a)Traditional method


(b) Proposed method

[Fig. 7] Comparison of results in mobile devices

Delays occur by phase if tiles are individually retrieved as shown in [Fig. 7]. However, the proposed method enables viewing under a single frame.

In general, tiles take up space that is four times greater for a scale of x2 according to zoom level, resolution, and range. As such, they are saved as images or in compressed format depending on the storing of tile images.

As shown in <Table 1>, the proposed method is able to store data 1/2 times the original size if basic information is applied to a 21-level scale. The traditional method also uses compression, but the method proposed in this paper increases the efficiency of compression with separate processing of images and data.

〈Table 1〉 Comparison of data size (byte) after rendering

Original data size (Byte) (Zoom level)	Traditional rendering method[6] (Byte) (a)	Proposed method (Byte) (b)	Enhancement rate (%)(a/b*100)
5TB (21 Level)	4TB	2TB	B200.0
4TB (18 Level)	3TB	1.8TB	166.7
3TB (16 Level)	2.5TB	1.5TB	166.7
2TB (14 Level)	2TB	1.3TB	153.8
1TB (12 Level)	1TB	0.9T	11.1

To efficiently provide map information through mobile devices, information can be divided into layers, or grouped at the server as we have proposed. For a specific area based on a zoom level of 16, we compared bandwidth and view speed by varying the level from 0 to 21. <Table 2> illustrates that the proposed method improves bandwidth occupancy and view speed from level 12 onwards. The view speed was measured with consideration of the appropriateness of directory organization for searches.

<Table 2> shows that the traditional method has low bandwidth occupancy but is not as fast in view speed. Since data is received and merged at the client, delays occur in data viewing depending on the performance of the mobile device. On the other hand, our proposed method provides data by merging layers at the server beforehand, resulting in high bandwidth but improved speed. Because a fast view speed is essential for mobile tasks, our method is clearly more effective.

〈Table 2〉 Comparison of bandwidth occupancy (MB) and view speed (sec)

Zoom level (z)	Bandwidth occupancy (MB)		
	Traditional method[6] (a)	Proposed method (b)	Enhancement rate (%)(a/b*100)
0~5	60	55	109.1
6~10	75	70	107.1
11~15	100	85	117.6
16~21	150	102	147.5

Zoom level (z)	Bandwidth occupancy (MB)		
	Traditional method[6] (a)	Proposed method (b)	Enhancement rate (%)(a/b*100)
0~5	0.2	0.3	-34.0
6~10	0.3	0.5	-40.0
11~15	2.0	1.2	166.7
16~21	4.0	2.0	200.0

<Table 3> summarizes the proposed method in terms of tile rendering, searching, map cache, and tile indexing. With the exception of tile searching, the traditional method and proposed method differ in the use of servers for tile merging and data processing.

〈Table 3〉 Differences between proposed and traditional methods

Type	Traditional method [6]	Proposed method
Tile rendering	Transmission by tile (in order)	Transmission after tile merge
Tile searching	Tile key (Tile feed architecture)	Tile key
Tile indexing	Quad tree	Key indexing (using complement)
Map cache	Entire cache Recaching of changes	Pre-rendering cache for tiles containing additional information Cache changes only

5. Future Works and Conclusions

With mobile devices becoming more widespread, we, in consideration of web-based services expanding to the mobile environment, have proposed a new architecture and implemented functions to provide information more efficiently Web-based services have provided information with guaranteed performance by the client. However, RIA (Rich Input Application) -based optimized data must be utilized to overcome the limited performance of mobile devices.

By processing key management using the proposed image index method, we were able to retrieve and provide rendered images at a higher speed. Also, the GIS server load was reduced even when making requests to areas with multiple users and many map objects. The cache map technique was modified to provide large-scale data while reflecting mobile characteristics, thus allowing the efficient use of mobile devices. However, further research is needed to expand functions for processing of changes in rendering information in order to enable mutual changes in large volumes of information.

References

[1] Bin Jiang, Xiaobai Yao, “Location Based Services and GIS in Perspective,” Location Based Services and TeleCartography, Lecture Notes in Geoinformation and Cartography, Section I, pp.27-45, 2007.

[2] Carl-Fredrik Sørensen, Alf Inge Wang, Reidar Conradi, “Support of Smart Work Processes in Context Rich Environments,” International Federation for Information Processing(IFIP), 191, pp.15-30, 2005.

[3] Daoxun Xia, Xiaoyao Xie, Yang Xu, “Web GIS server solutions using open-source software,” Open-source Software for Scientific Computation (OSSC), 2009 IEEE International Workshop on IEEE Conferences, pp.135-138, 2009.

[4] Tim Verbelen, Raf Hens, Tim Stevens, Filip De Turck, Bart Dhoedt. “Adaptive Online Deployment for Resource Constrained Mobile Smart Clients,” Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, MobileWireless Middleware, Operating Systems, and Applications, Part 3, vol.48, pp.115-128, 2010.

[5] Zao Liu,Pierce, M E,Fox G C, Devadasan, N.“Implementing a caching and tiling map server: a Web 2.0 case study,” Collaborative Technologies and Systems, pp.247-256, May, 2007.

[6] Yiming Liu, Erik Wilde. Scalable and Mashable, “Location-Oriented Web Services,” Web Engineering Lecture Notes in Computer Science, vol.6189, pp.307-321, 2010.

[7] Wu Kehe, Bai Xuewei and Wang Xiaohui, “Power Web GIS Based on Oracle MapViewer,” Internet Technology and Applications, 2010 International Conference on IEEE Conferences, pp.14, 2010.

[8] Quan Sh,i Hengshan Wang, Hui Wang, Jianping Chen, “Design of WebGIS Rendering Engine Based on Silverlight-based RIA,” Intelligent Computation Technology and Automation (ICICTA), 2011 International Conference on IEEE Conferences, vol.1, pp.1050-1053, 2011.

[9] Hoseok Ryu, Insuk Park, Soon J Hyun, Dongman Lee, “A Task Decomposition Scheme for Context Aggregation in Personal Smart Space,” Lecture Notes in Computer Science, vol.4761, pp.20-29, 2007.

[10] René Meier, Anthony Harrington, Thomas Termin, Vinny Cahill, “A Spatial Programming Model for Real Global

Smart Space Applications," Distributed Applications and Interoperable Systems Lecture Notes in Computer Science, vol.4025, pp.16-31, 2006.

[11] Shah Rukh Humayoun, Tiziana Catarci, Massimiliano de Leoni, Andrea Marrella, Massimo Mecella, Manfred Bortenschlager, Renate Steinmann, "Universal Access in Human-Computer Interaction. Applications and Services," Lecture Notes in Computer Science, vol.5616, pp.343-352, 2009.

Kang-Won Lee [정회원]

- 1997. 2 : BSc and MSc from Korea University in Computer Science
- 1997.2~ : Software Programmer in Network Company

<관심분야>

mobile communication, network management systems