International Journal of Bio-Science and Bio-Technology
Vol.6, No.5 (2014), pp.33-42
http://dx.doi.org/10.14257/ijbsbt.2014.6.5.04

Analysis of Sensoring System Development Process

Sang-Young Lee

Department of Health Administration Namseoul University, 21 Maeju-ri, Seongwan-eup, Cheonan, South Korea
Sylee@nsu.ac.kr

Abstract

With the prevalence of the Global Positioning System, an increasing number of electronic devices and vehicles have been equipped with a GPS module for a variety of applications including navigation and location-based search. Early GPS software was developed as monolithic tool in which all functions packed in the same software. But, these GPS software have the problems of the high cost of constructing system. close system architecture and the reusability. And there is a lack of interoperability between them because most of them have their own unique data format according to their application fields. Therefore in the paper new extended for better modeling GPS application in the UML Diagram are proposed, and the Implementation of a program called StereotypeCreator, which is able to create iconic stereotypes used in one of the most popular visual modeling tools for software development, Rational Rose, will be also proposed.

Keywords: *Business, GPS, System, Reuse*

1. Introduction

GPS application domain is an especial environment that requires precise measurement and precision calculation of real-world geographical entities with the help of GPS (Global Position System) in both temporal and spatial factor[1]. The conventional modeling element of class in UML is not powerful enough to present the spatial feature and temporal feature that GNSS objects embody. Therefore, new modeling elements of class for GPS application with UML are needed [2]. This paper will propose several new iconic stereotypes formulating modeling elements for GPS application. Also in this paper the implementation of a program called StereotypeCreator, which is able to create iconic stereotypes used in Rational Rose, will be proposed.

The paper consists of five sections. The following section discusses issues for related works. Several iconic stereotypes presenting class meta-model element in the UML class diagram will be proposed in the third section. Section 4 describes the implementation of a tool called StereotypeCreator for Rational Rose Section 5 presents our conclusion.

2. Related Works

Currently a project of GPS component extraction and development process is being developed by our research team, 4S (GIS, ITS, SIIS, GNSS) research team of software engineering laboratory [3, 4]. In the modeling process we find it is not good enough to present the object features in GPS application domain by the conventional class model elements in UML [5, 6]. Because the conventional class model element cannot describe the geometry property of the class whose objects can be displayed in a monitoring system, and the conventional class model elements cannot embody the issue that the

ISSN: 2233-7849 IJBSBT

location of the object can be recorded in a time range and can be gotten in any time later[7, 8]. Therefore, the conventional stereotype for the modeling element of class is inappropriate for the modeling in GPS application with UML. In order to meet these specific requirements of GPS application domain, new extended iconic stereotypes for class modeling elements need to be brought forward and will be presented in the next section[9, 10].

3. Meta-model Element

In Figure 1, the visual notations are used to represent georeferenced classes which are distinguished from conventional classes. Main elements of georeferenced classes are:

- a graphical representation with a symbolistic icon,
- an iconic notation for geographic types (points, lines and polygons),
- the class name,
- attributes,
- operations.

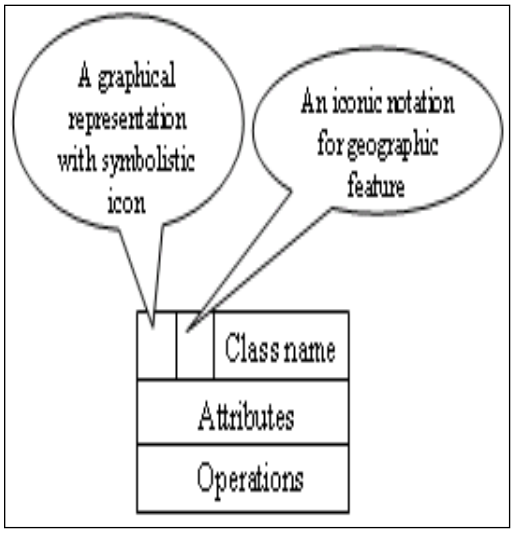

Figure 1. Graphical Representation of Class Meta-model Element

For example, the Figure 2(a) depicts the visual representation of a class named “Building” with a symbolistic icon and an iconic notation for geographic types (corresponding to a polygon) on the left side of the class name, Building. The polygon symbol means that each object "Building" is associated to a polygon. Attributes are "address" and "inhabiting_area". The only operation associated to the class is "build".

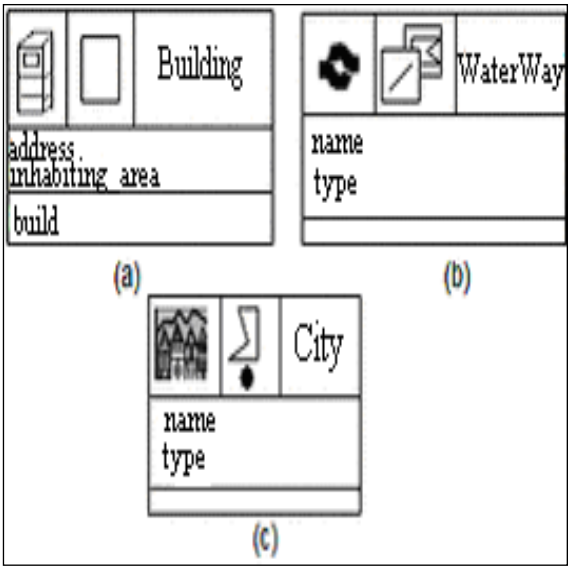

Figure 2. Examples of Georeferenced Class

The georeferenced classes allow the representation of complex objects and composed objects. Each complex object is related to several points, lines or polygons. For example, a waterway may be considered as a lake (a polygon) connected to a river (a line). Graphically speaking, overlapped iconic figures express complex objects. An example of a complex object class is given in Figure 2(b). Composed objects have several representations in function of scale's point of view. Figure 2(c) presents the graphical notation of a composed class "City". We do not consider composed-complex objects in this paper.

In many cases the value of an attribute of an object varies during all the life cycle of the object. It is possible to associate a temporality to an attribute x of an object a. In that case, during all the life cycle of a, the object is able to "know" all previous values of x. Graphically, an icon representing a clock is placed on the right side of the attribute in the class(Figure 3(a)). In the same way, a temporality may be associated to the geometry of an object. In that case, an icon representing a clock is placed on the right side of the iconic notation for geographic types (Figure 3(b)). The Figure 4 depicts the implementation of Class MobileStation presented in Figure 3(a).

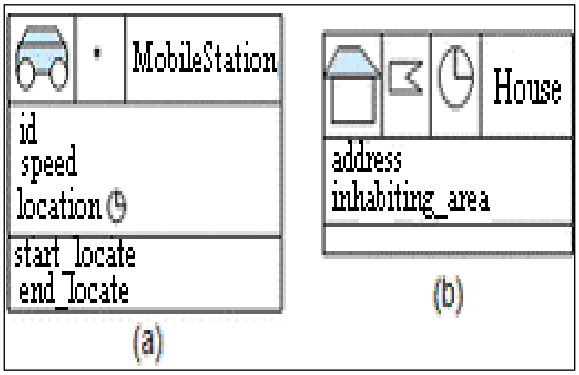

Figure 3. Examples of Georeferenced cCass with Temporality

```
using System;
using System.Data;
Class MobileStation
{//geographic features of object
 public Point the_point=null;
 public int ID;
 public int speed;
 public Point location;
 public MobileStation() {/*code of the constructor*/}
 public void start_locate();
{ the_point=new Point();
  /*...code of the operation,
    Coordinates are associated to the_point*/
}
public void end_locate()
{ if (the_point!=NULL)
   {/*code of the operation*/
    the_point=NULL;}
}
public Point get_location(double time)
{//return the value of location at a t time
  return search_value(set_of_location_values,time));
}
public void set_location(Point value)
{// get the current time of the system
  double current_time=getCurrentTime();
 //map the current value of location with
 // the current time of the system
 set_of_location_values.put(current_time,value)
 }
}
```

Figure 4. Example of Class MobileStation Implementation in C#

4. Implementation of Stereotype Creator

According to the specification of extended iconic stereotype of class meta-modeling element for a georeferenced class in UML class diagram, many kinds of the iconic stereotypes of class can be designed. Stereotype Creator is a software tool used to

automatically create all kinds of user-defined iconic stereotypes that can be used in Rational Rose. It can be considered as an extended tool for Rational Rose.

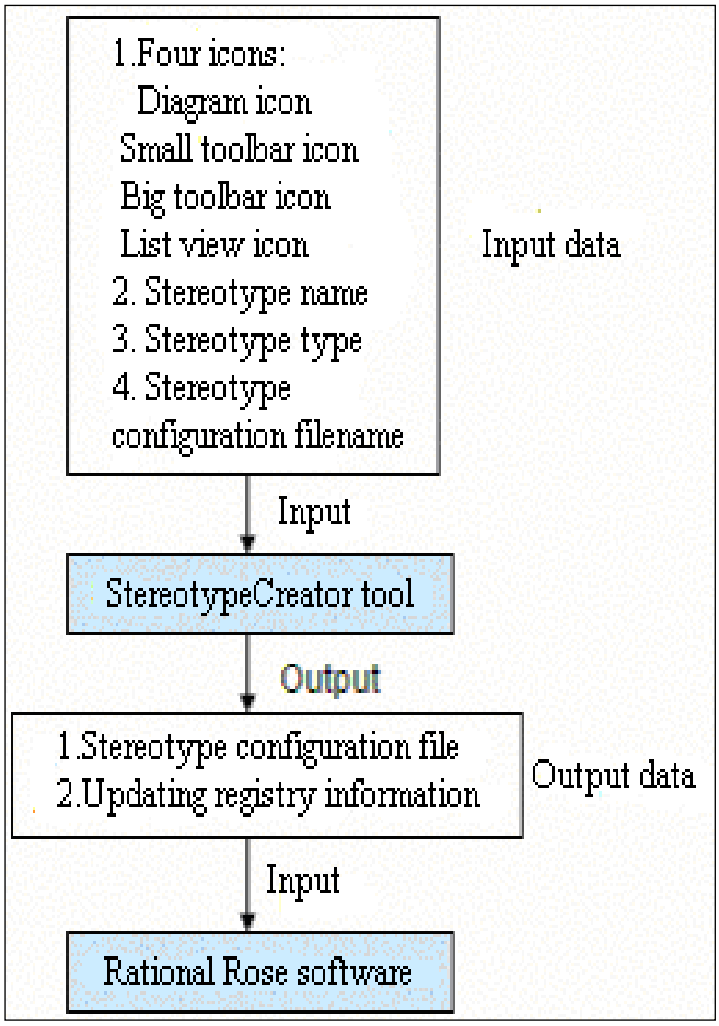

Figure 5. Whole Process of StereotypeCreator

4.1. Create Stereotype Configuration File

Stereotype configuration file is required by Rational Rose software. The stereotypes in Rational Rose must be defined in a stereotype configuration file. Rational Rose is delivered with a default stereotype configuration file, called DefaultStereotypes.ini. If possible, users may add own stereotypes to that file. If users do not want to use that file, a new stereotype configuration file can be created.

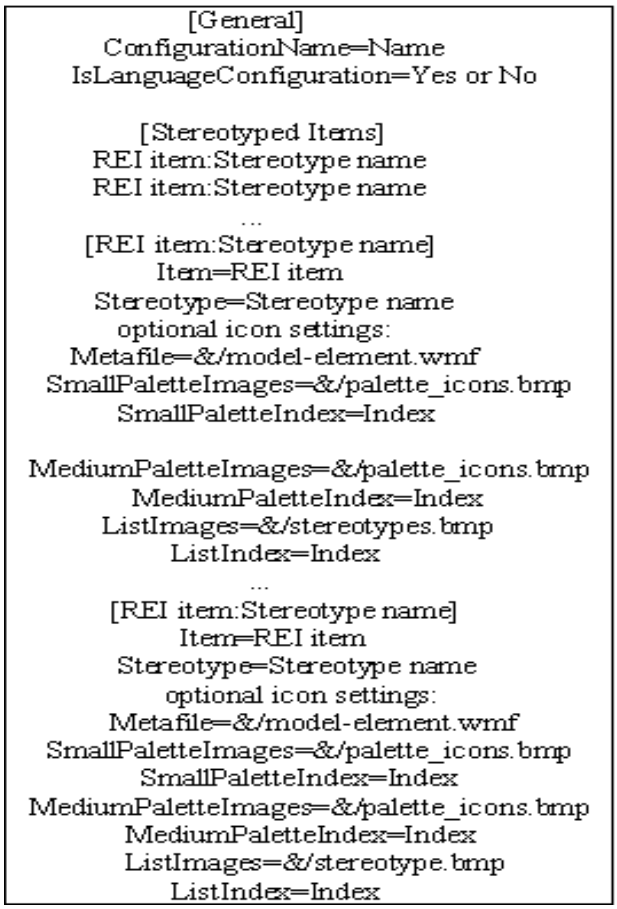

Figure 6. The Format of Stereotype Configuration File

The format of the stereotype configuration file must be known before the creation of stereotype configuration file. Stereotype configuration file is a text file with extension name of INI. A stereotype configuration file may include one or more stereotypes information. The Figure 6 presents the general format of stereotype configuration file. In the format of stereotype configuration file, the general section contains the information of specific setting. The section can be omitted when building the stereotype configuration file. The section of stereotypes item is like a table of contents for the stereotypes. It contains a list of stereotyped REI(Rational Rose Extensibility Interface) objects. For example, Class:Control, Component:DLL, Operation:Set. The section of REI item contains the settings for each stereotype, including any optional icon files and settings.

Considering the relationship between stereotype configuration file and stereotype that a stereotype configuration file may include one or more stereotypes information, two major classes are defined for implementation of StereotypeCreator: StereotypeSet class and IconicStereotype class. The definitions of some class methods are neglected. A StereotypeSet object is associated to a stereotype configuration file. A IconicStereotype object represents a kind of iconic stereotype. A StereotypeSet object may contain one or more IconicStereotype object. A StereotypeSet object may add or delete IconicStereotype objects. The relationship between StereotypeSet class and IconicStereotype class is presented in Figure 7.

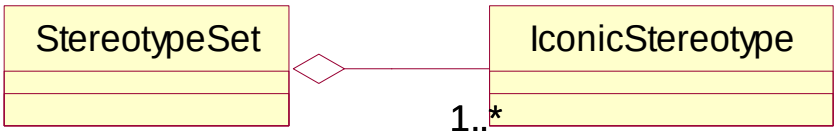

Figure 7. Relationship between StereotypeSet Class and IconicStereotype Class

The function of creation of stereotype configuration file is implemented by the method of WriteStereotypeCfgFile in the StereotypeSet class. The implementation code is the following (see Figure 9).The WriteStereotypeCfgFile method adds the stereotype sent by first parameter into the stereotype configuration file associated to current StereotypeSet object and determines whether to create new file or not before adding the stereotype according to the second parameter.

```
//create stereotype configuration file
public void
WriteStereotypeCfgFile(IconicStereotype
 styObj,bool bCreateOrAddFlag)
{   string str;
    string strBuffer=null;
    if(bCreateOrAddFlag)
    { //The section of[General]
      //The section of [Stereotyped Items]
      str="[Stereotyped Items]\r\n";
      strBuffer+=str;
      str= styObj.GetKindString()+":"+
      styObj.StereotypeName+"\r\n\r\n";
      strBuffer+=str;
      AddItemSection(ref strBuffer,styObj);
    }  else
      {//Read all information to strbuffer from file
        StreamReader InFile = File.OpenText
                              (StyCfgFileName);
        strBuffer=InFile.ReadToEnd();
        InFile.Close();
        //Insert one line about the new stereotype
        //information
        int iPos=strBuffer.IndexOf("\r\n\r\n");
        str="\r\n"+styObj.GetKindString()+
        ":"+styObj.StereotypeName;
        strBuffer=strBuffer.Insert(iPos,str);
        AddItemSection(ref strBuffer,styObj);
      }
      //Write stereotype configuration file
      StreamWriter OutFile;
      OutFile = File.CreateText(StyCfgFileName);
      OutFile.Write(strBuffer);
      OutFile.Close();
}
```

Figure 8. WriteStereotypeCfgFile Method

4.2. Update Registry Information

After creating the stereotype configuration file, another task is to update the registry information for stereotype configuration file in order to make stereotype configuration file loaded correctly when Rational Rose software initiates.

```
//Update the registry for stereotype configuration
file
public bool UpdateRegistry()
{ string strStyCfgFileName=
  ExtractFileName(StyCfgFileName);
  try
  { Microsoft.Win32.RegistryKey
    RoseStereotypeCfgFileKey=
    Microsoft.Win32.Registry.LocalMachine.
    OpenSubKey("SOFTWARE\\Rational
    Software\\Rose\\StereotypeCfgFiles",true);
    if(RoseStereotypeCfgFileKey==null)
    { MessageBox.Show("Can't find the subkey
      of StereotypeCfgFiles", "Registry find error",
      MessageBoxButtons.OK,
      MessageBoxIcon.Error);
      return false;
    }
     string [] strValueNameArray=
     RoseStereotypeCfgFileKey.GetValueNames();
     string strMax=strValueNameArray[0];
     for (int i=1;i<strValueNameArray.Length;i++)
   if(strValueNameArray[i].CompareTo(strMax)>0)
         strMax=strValueNameArray[i];
     string str=null;
     string strNum=null;
     for (int i=strMax.Length-1;i>=0;i--)
     if (strMax[i]<'0' || strMax[i]>'9')
     {  str=strMax.Substring(0,i+1);
        strNum=strMax.Substring(i+1);
        break;
     }
     int num=1;
     if (strNum!=null)
         num=Int32.Parse(strNum)+1;
     RoseStereotypeCfgFileKey.
   SetValue(str+num.ToString(),strStyCfgFileName);
     RoseStereotypeCfgFileKey.Close();
    }
    catch(SecurityException)
    { MessageBox.Show("You do not have Registry
     Permission","Registry write acess error",
    MessageBoxButtons.OK,MessageBoxIcon.Error);
     return false;
    }
   return true;
}
```

Figure 9. Update Registry Method

The above figure presents the location where stereotype configuration information will be placed with the help of the registry editor tool. (In the figure registry key value corresponding to the key name, File 3 is the string of Mystereotypes.ini which is created by StereotypeCreator. The function of updating the registry is implemented by

the method of UpdateRegistry of StereotypeSet class. The implementation code is presented in figure. The UpdateRegistry method updates setting for stereotype configure file in registry associated to current StereotypeSet object. When stereotypes associated to a stereotype configuration file need to be unloaded from Rational Rose software, the setting corresponding to the stereotype configuration file should be removed from the registry.

```
public bool DeleteRegistryInformation()
{
 string strStyCfgFileName=
 ExtractFileName(StyCfgFileName);
 try
 {  Microsoft.Win32.RegistryKey
   RoseStereotypeCfgFileKey=
   Microsoft.Win32.Registry.LocalMachine.
       OpenSubKey("SOFTWARE\\Rational
     Software\\Rose\\StereotypeCfgFiles",true);
   if (RoseStereotypeCfgFileKey==null)
   { MessageBox.Show("Can't find the subkey of
       StereotypeCfgFiles","Registry find error",
   MessageBoxButtons.OK,MessageBoxIcon.Error);
      return false;
   }
   string [] strValueNameArray=
   RoseStereotypeCfgFileKey.GetValueNames();
   for (int i=0;i<strValueNameArray.Length;i++)
   if (((string)RoseStereotypeCfgFileKey.GetValue
    (strValueNameArray[i]))==strStyCfgFileName)
   {RoseStereotypeCfgFileKey.
     DeleteValue(strValueNameArray[i]);
     RoseStereotypeCfgFileKey.Close();
     return true;
   }
 }
 catch(SecurityException)
 {MessageBox.Show("You do not have Registry
   Permission","Registry write acess error",
   MessageBoxButtons.OK,MessageBoxIcon.Error);
 }
 return false;
}
```

Figure 10. Delete Registry Information Method

The function of updating the registry is implemented by the method of UpdateRegistry of StereotypeSet class. The implementation code is presented in figure 10. The UpdateRegistry method updates setting for stereotype configure file in registry associated to current StereotypeSet object. When stereotypes associated to a stereotype configuration file need to be unloaded from Rational Rose software, the setting

corresponding to the stereotype configuration file should be removed from the registry. The Figure 10 presents the function implementation by the DeleteRegistryInformation method of the StereotypeSet class.

5. Conclusion

This paper proposes extended iconic stereotypes of class meta-model element for GNSS application in the UML Diagram and provides its implementation as a tool of StereotypeCreator for Rational Rose. Current research is so little that a great deal of work remains and should be done in the future. Future research can be positioned in customizing iconic stereotypes for other modeling elements such as association, generalization, attribute and so on, in GPS application. Now the tool of StereotypeCreator just supports the creation of iconic stereotype for the modeling element of class in class diagram, but in the future StereotypeCreator for Rational Rose will support more kinds of iconic stereotype for different modeling element.

Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2013R1A1A2007598).

References

[1] B. W. Parkinson and P. Enge, "Global Positioning System: Theory and Applications Volume Ⅱ", **(1996)**.
[2] L. Lau and P. Cross, "Development and testing of a new ray-tracing approach to GNSS carrier-phase multipath modeling", Journal of Geodesy, vol. 81, no. 11, **(2007)**, pp. 713–732.
[3] M. Jacob, M. S. Schön, U. Weinbach and T. Kürner, "Ray Tracing Supported Precision Evaluation for GPS Indoor Positioning", Proc. 6th Workshop on Positioning, Navigation and Communication (WPNC), Hannover, **(2009)**.
[4] G. Del Duca, R. Perago, V. Paciucci, G. D. Bitonto and F. Principe, "Verification of GNSS Applications at Italian Regional Airports", Proc. of ENC-GNSS, Naples, Italy, **(2009)**.
[5] S. Guerrier, "Improving Accuracy with Multiple Sensors: Study of Redundant MEMS-IMU/GPS Configurations", Proceedings of the ION GNSS, Savannah, GE, USA, **(2009)**.
[6] R. L. Glass, "The Standish report: does it really describe a software crisis?", Communications of the ACM, vol. 49, no. 8, **(2006)**, pp. 15-16.
[7] F. Buschmann, K. Henney and D. C. Schmidt, "Pattern-oriented Software Architecture: On Patterns and Pattern Languages", John Wiley and Sons, **(2007)**.
[8] G. Booch, J. Rumbaugh and I. Jacobson, "The Unified Modeling Language User Guide", Addison Wesley Longman Publishing Co., Inc., USA, **(1999)**.
[9] S. Guerrier, "Improving Accuracy with Multiple Sensors: Study of Redundant MEMS-IMU/GPS Configurations", Proceedings of the ION GNSS 2009, **(2009)**; Savannah, GE, USA.
[10] A. Waegli, "Noise reduction and estimation in multiple electro mechanical inertial systems". Measurement Science and Technology, vol. 21, **(2012)**.

Author

Sang-Young Lee is a professor, Dept. of Health Administration, Namseoul University, South Korea.

