iTAX – Integrated Tax Administration

Integration of TatukGIS and Powerbuilder

By Dr. Dirk Lorenzen

CONTENTS

Abbreviations

Arc: A line in GIS
Attribute table: Data base for storing attribute information of map objects
ESRI: Todays leading GIS company (ArcINFO, ArcMAP, ArcVIEW)
GIS: Geographic Information System: An automated system composed of hardware, software, data and people used to create, store, disüplay and analyze spatial data and related attributes
Item: can be a column in an attribute file or a single theme in a collection of themes
Layer: Representation of spatial information, can consist points, lines or polygons
Node: Defines the starting point and the endpoint of a line segment
Shape: Single Object of a shapefile, can be a point, line or polygon feature
Shapefile: Common digital map format storing topology and attribute information. Consists at least three main files: *.shp, *.shx and *.dbf file.
TatukGIS: Polnish company deploying the TatukGIS Developer Kernel used in the iTAX application
Theme: Single layer in an GIS project

Abstract

ITAX is a computerized computing and accounting system for local revenue (levies, taxes). The iTAX software is developed using Power Builder 10.5 (Sybase), In the iTAX application tax mapping facilities are/have to be implemented. The connection of the map objects to the recordsets of the Ingres databank is done by linking via a unique ID, the PIN-ID. The software for the development of GIS functionalities within iTAX is TatukGIS developer Kernel (activeX version). The Major aim of this short term assignment was to help/teach the iTAX programmers to implement some basic GIS functionality and to develop some strategies for future production of thematic maps which are up to now produced by time-consuming manually work. These procedures should be automated in the future.

Furthermore a strategic concept was developed how to use tax mapping data and geographic algorithms for other purposes (see: 2.3).

Main Part

INTRODUCTION

iTax is a computerized computing and accounting system for local revenue (levies, taxes) which stores all relevant (credit/ debit) data of the project in individual accounts and a relational data bank and thus helps to monitor/control all tax transactions by its citizens and tax payers.(Fuchs, 2008). As an additional component GIS based digital maps should be implemented into the iTax application. For this purpose cadastral maps were digitized (using Mainfold GIS) and connected with data of the databank, so that selected data can be shown as thematic maps to visualize spatial distributions of selected topics. The implementation of GIS functionalities into the iTax application is done by using the developer kernel (activeX) of TatukGIS which can be done by ole embedding into the Powerbuilder development program, which is used for creating the iTax application. The major aims were:

Implementation of "basic" GIS functionalities such as: finding a lot by choosing a pin-id in the iTax database, zoom to it, view stored information of the selected lot, view information of the adjacent lots, subdivide a lot and assigning new pin-ids to the new polygons and combine several lots into one.

The creation of thematic maps is up to now done by time-consuming manually work, this procedure has to be automated by developing and programming algorithms that produce or update these thematic maps on "mouse-click".

Implementation of "basic" GIS functionalities into the iTAX Application

The iTAX- software is developed using Power Builder 10.5 licensed under Sybase Corporation. PowerBuilder has very good (and easy to use) capabilities in implementing Data banks (here: Ingres) into user developed applications. The GIS-functionalities are provided by the developer Kernel of TatukGIS (ActiveX-Version). These functionalities can be implemented into Power Builder applications by OLE - connections. The key questions for assessors and the graphical representation in the iTAX applications were:

Finding a lot
View information about the lot
Find and view information about the owner of the lot and other lots he is owning
Subdivide a lot and adding new PIN-IDs
Combine several lots into one lot (and changing the PIN-ID)

The primary aim was to connect the data of the iTAX database to the digital map (shape-file). Each lot can be identified by a unique PIN-ID. Using this field it is possible to link/join subsets of data of the Ingres-databank to polygons of the shapefile. By doing this has the advantage, that modifications in the databank are immediately represented in the corresponding map. Furthermore changes in attribute data only have to be done in the database and double storage of data is avoided. The following part of code shows how this is done in the Power Builder application.

After doing this it is possible to set the map extent to the extent of the selected lot:

Fig. 1: Joining databank data to a shapefile by using a unique id (here: "prop_id")

//Please note the ioo_layer is an instance variable for the window:

```
//This example generates a result set in a ResultSet object from an existing DataStore object.
//The ResultSet object is used to populate a new ADOResultSet object.
//The GetRecordSet function on the ADOResultSet object is used to return an
//ADO Recordset as an OLEObject that can be used with ADO Recordset methods.
resultset lrs_resultset
ADOresultset lrs_ADOresultset
OLEObject loo_ADOrecordset
datastore ds_source
ds_source = create datastore
ds_source.dataobject = 'd_property_data_map'
ds_source.settransobject( sqlca )
ds_source.retrieve()
// Generate a result set from an existing DataStore
ds_source.GenerateResultSet(lrs_resultset)
// Create a new ADOResultSet object and populate it from the generated result set
lrs_ADOresultset = CREATE ADOResultSet
lrs_ADOresultset.SetResultSet(lrs_resultset)
// Pass the data in the ADOResultSet object to an OLEObject you can use as an ADO Recordset
loo_ADOrecordset = CREATE OLEObject
lrs_ADOresultset.GetRecordSet(loo_ADOrecordset)
ioo_layer.JoinADO = loo_ADOrecordset
ioo_layer.JoinPrimary = "prop_id"
ioo_layer.JoinForeign = "prop_id"
```

Because the viewer window is connected to the attributes control, attribute information of the selected lot is shown in the attribute control window.

Fig. 2: Zooming to selected lot and make it flashing 4 times

```
ls_layername = 'valencia'
ls_pinno = "046-20-013-02-046"
loo_layershape = ole_gis.Object.Get( ls_layername )
//same code with above
//loo_layershape = ole_gis.Object.Items.item[0]
ll_rowcount = loo_layershape.GetLastUid()
For ll_row = 1 to ll_rowcount
        ls_fieldpinno = loo_layershape.GetField( ll_row , 'pin_no' )
        If ls_fieldpinno = ls_pinno Then
                ioo_shape = loo_layershape.GetShape( ll_row )
                If Not IsNull( ioo_shape ) Then
                        ioo_shape = ioo_shape.MakeEditable()
                        ole_attrib.show()
                        ole_attrib.Object.ShowShape( ioo_shape )
                        ole_GIS.object.VisibleExtent = ioo_shape.Extent
                        ioo_shape.flash( )
                        il_currentselected = ll_row
                End If
        End If
Next
```

Fig. 3: Showing attributes of current selected shape

```
ole_GISAtt.ShowShape(GIS.Editor.CurrentShape)
```

Putting the select function on a “mouse-down event” makes it possible to select a lot just by clicking with the mouse an view attribute information.

Fig. 4: Showing attributes of a shape by clicking the shape with the mouse

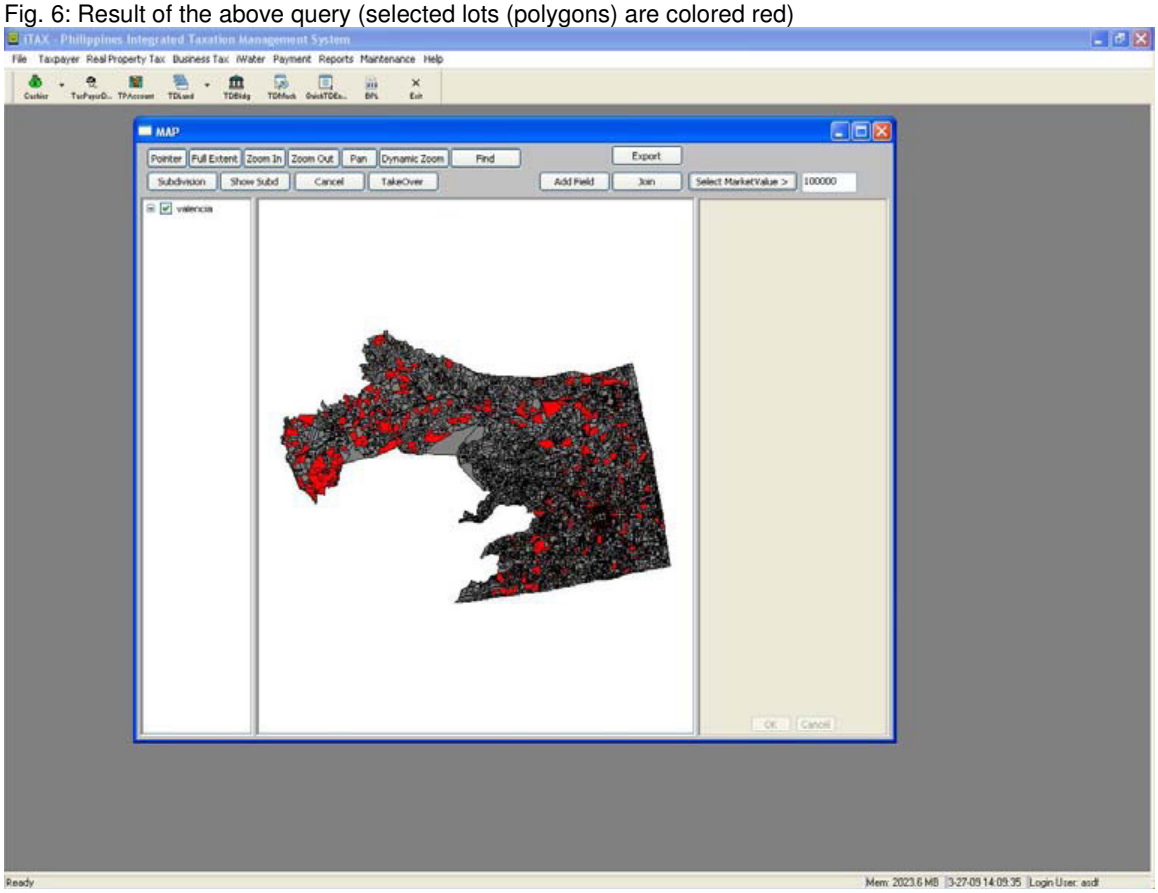
```
ole_gis.object.mode = 0
loo_shape = ole_gis.object.locate( ole_gis.object.screentomap( loo_utils.point( ocx_x, ocx_y ) ), 5 / ole_gis.object.zoom )
loo_shape.flash( )
long ll_rtu
ole_attrib.show()
ole_attrib.Object.ShowShape( loo_shape )
ole_GIS.object.VisibleExtent = loo_shape.Extent
```

Another possiblity which was implemented is to select polygons (lots) by doing a query on the attribute data, e. g. owner name, and market value.

Fig. 5: Selecting lots by doing a query

```
Boolean wahl
String layer_name, ls_layername , ls_pinno
Integer z = 0
Long ll_row, ll_rowcount, ll_rtm
long ll_fieldmarketvalue, ll_marketvalue
OleObject loo_layershape
loo_layershape = Create OleObject
ll_rtm = loo_layershape.ConnectToNewObject("TatukGIS_DK.XGIS_LayerSHP")
If ll_rtm <> 0 Then Messagebox("Connect... to: XGIS_LayerSHP", ll_rtm)
ole_gis.object.mode = gisSELECT
ll_rowcount = ole_gis.object.Items.Count - 1
ls_layername = 'valencia'
ll_marketvalue = long(sle_1.text)
loo_layershape = ole_gis.Object.Get( ls_layername )
//Reset MAP to orignal state.
loo_layershape.RevertAll()
ll_rowcount = loo_layershape.GetLastUid()

For ll_row = 1 to ll_rowcount
        ll_fieldmarketvalue = long( loo_layershape.GetField( ll_row , 'marketvalue' ) )
        If ll_fieldmarketvalue > ll_marketvalue Then
        ioo_shape = loo_layershape.GetShape( ll_row )
                If Not IsNull( ioo_shape ) Then
                        ioo_shape = ioo_shape.MakeEditable()
                        ioo_shape.Params.Area.Color =RGB( 255,0,0) //Rand(255) * 256 * 256 + Rand(255) * 256
+ Rand(256) //65536 * Blue+ 256 * Green+ Red
                End If
        End If

End For

ole_gis.object.update()
```

Fig. 6: Result of the above query (selected lots (polygons) are colored red)



Sometimes it is required to combine several lots to one or to subdivide a lot. This can be done by using the topology functions of TatukGIS.

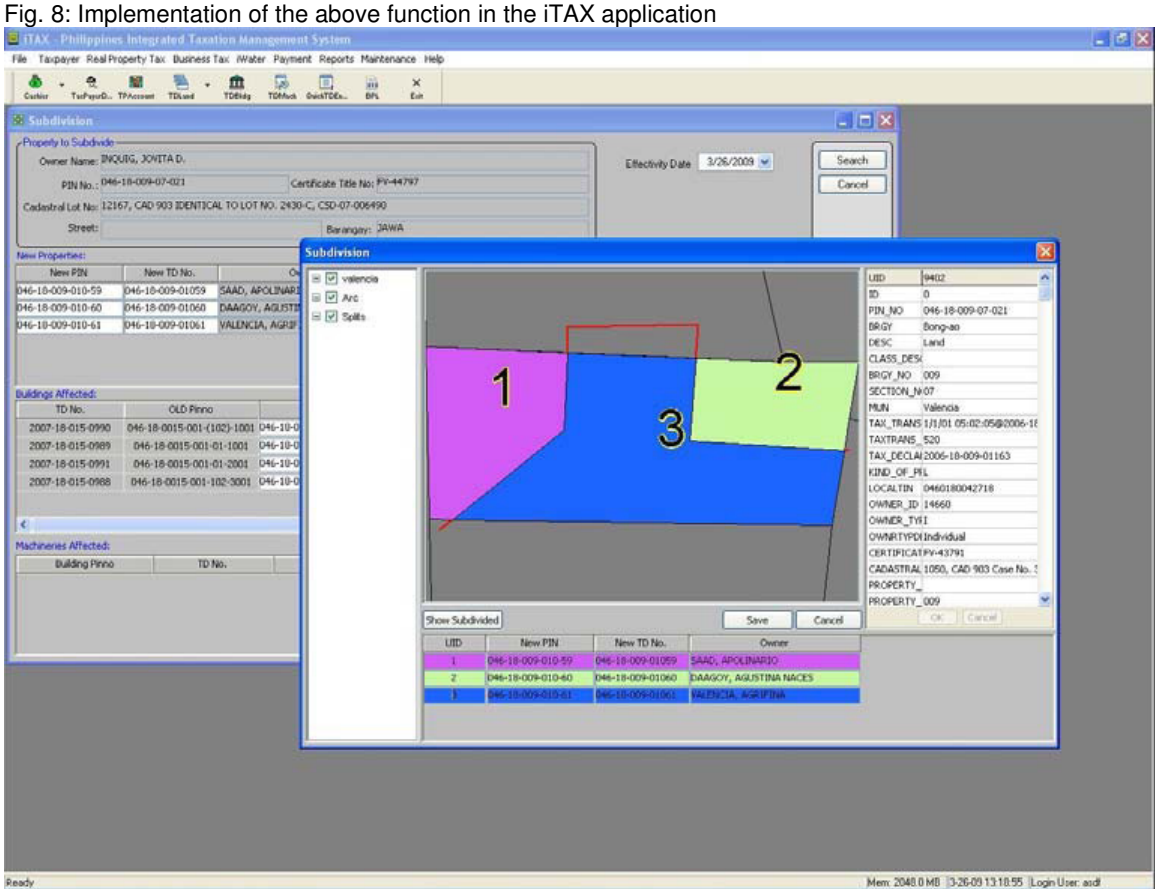
The following excample shows how the subdividing function is implemented in the iTAX application. Because new PIN-IDs have to be assigned this has been implemented in the subdividing function.

```
ole_gis.object.mode = gisSELECT
ll_rowcount = ole_gis.object.Items.Count - 1
ls_layername = 'valencia'
loo_layershape = ole_gis.Object.Get( ls_layername )
ll_rowcount = loo_layershape.GetLastUid()
For ll_row = 1 to ll_rowcount
        ls_fieldpinno = loo_layershape.GetField( ll_row , 'pin_no' )
        If ls_fieldpinno = as_pinno Then
                ioo_shape = loo_layershape.GetShape( ll_row )
                If Not IsNull( ioo_shape ) Then
                        ioo_shape = ioo_shape.MakeEditable()
                        ole_attrib.show()
                        ole_attrib.Object.ShowShape( ioo_shape )
                        ole_GIS.object.VisibleExtent = ioo_shape.Extent
                        ioo_shape.flash( )
                        ioo_shape.Params.Area.Color = Rand(255) * 256 * 256 + Rand(255) * 256 + Rand(256)
                        //65536 * Blue+ 256 * Green+ Red
                        il_currentselected = ll_row
                End If
        End If
End For

If il_currentselected < 1 Then
        Return FAILURE
End If

Return SUCCESS
```

Fig. 7: Subdividing a lot using the topology function of TatukGIS

```
is_flag = 'subdvision'
Long ll_rtu ioo_layerarc.ConnectToNewObject("TatukGIS_DK.XGIS_LayerVector")
ole_gis.object.mode = gisEDIT
ioo_layerPolygon = ole_gis.Object.Items.Item[0]
ioo_shpPolygon = ioo_layerPolygon.GetShape( il_currentselected ).MakeEditable()
If isNull(ioo_shpPolygon) Then Return
ioo_layerarc.Params.Line.Color =RGB(255, 0, 0 ) //Red
ioo_layerarc.Params.Line.Width = 25
If isNull( ioo_layerarc ) Then Return
ioo_layerarc.Name = "Arc"
ole_GIS.Object.Add( ioo_layerarc )
ioo_shpArc = ole_GIS.Object.Items.Item(1).CreateShape( gisSHAPETYPEARC )
If isNull( ioo_shpArc) Then Return
ioo_shpArc.AddPart()
ioo_layerObj.Name = "Splits"
ole_GIS.Object.Add(ioo_layerObj)
Ole_GIS.Object.RestrictedExtent = Ole_GIS.Object.Extent
ioo_layerObj.RevertAll() //'clear layer with new polygons
ioo_shpArc.Reset() //'clear line
ioo_shpArc.AddPart() //'initiate for new points
Ole_GIS.Object.Update()


//cancel this action: event ue_cancel()
ioo_layerObj.RevertAll()
ole_GIS.Object.Delete("Splits")
ole_GIS.Object.Delete("Arc")
```

Fig. 8: Implementation of the above function in the iTAX application



Automatic creation of thematic maps

Up to now the creation of thematic iTAX-maps is done by time-consuming manual work, this procedure has to be automated by developing and programming algorithms that produce or update these thematic maps on “mouse-click”. The aim was to develop a strategic concept for future production of thematic maps using the database of iTAX. Up to now the data is selected and added (joined) to the map-features manually.

The TatukGIS developer kernel provides all functionalities to produce layouts of maps by programming all needed steps (including legend, north arrow, scale bar, text, etc.). So the idea is to implement predefined or very easy to use queries on the data of the databank, join the results to the attribute table of the shapefiles and layout the whole map just “on mouseclick”.

Besides the time saving effect, this will minimate failure production and will ensure that always the most actual data is used for map production. The figure 9 shows the schematic concept of this idea. Another advantage of this approach is that the layout of the final maps can be defined by the programmers, so that all “result-maps” have a unique design and due to this can be easily identified a “iTAX-maps” by the contemplators.

Fig. 9: Creating thematic maps automatically by using query-functions and joining the results to the shapefiles

3 LITERATURE

Fuchs, Ernst-Dieter (2008): iTAX – A Solution for Revenue Generating, GTZ Fiscal Decentralisation Project, Dumaguete City,