Oracle Spatial: Narrowing the Gap Between CAD and GIS: A Topobase™ Implementation Case Study

Keath Long and Matt Seidl – Las Vegas Valley Water District

This paper covers the implementation of Topobase, used to enable facility management, as a replacement for the multiple in-house applications that previously enabled CAD-GIS integration. Various advanced features of Topobase will be explored, in addition to examples of how the open architecture behind Oracle Spatial and Spatial's native SQL capabilities allow for elegant interoperability solutions.

About the Authors:

Keath Long is a senior GIS analyst for the Las Vegas Valley Water District. His GIS career with the Las Vegas Valley Water District began in 1995 and he has worked in a professional capacity for the past 8 years. He is currently the technical lead for GIS database development and GIS application development for the AM/FM/GIS Division. Keath earned a degree in MIS from the University of Nevada, Las Vegas.

Matt Seidl is a registered Civil Engineer in the State of Nevada. Matt earned his B.S. and M.S. degrees in Agricultural Engineering from Iowa State University in 1998 and 2000, respectively. Matt currently works for the Las Vegas Valley Water District as a CAD Analyst responsible for maintaining and developing the Distict's CAD/GIS automation in support of its redlining, as-builting, and GIS data capture processes.

Introduction

The challenges of modernizing legacy CAD systems to more easily integrate with rapidly advancing GIS systems and to more closely support the business functions of today and tomorrow can be overwhelming for even the most progressive organization.

Not more than a decade ago the Las Vegas Valley Water District (LVVWD) maintained several standard facility maps using manual drafting protocols and redundant data entry practices. As CAD technology matured these maps were converted to vector format in CAD but technicians still produced each map independently. Eventually a GIS central database was constructed and map production was automated significantly reducing data entry tasks and improving the quality of data available to customers. Large efficiencies where gained by having CAD technicians manage most of the GIS content, due to CAD software's precision data entry and engineering design capability. The result was two teams, the CAD team maintaining as-builts and managing GIS content, and the GIS team, responsible for map production, data analysis and data operations and quality initiatives. Tools and processes were developed that supported a tightly integrated CAD/GIS environment. However, as GIS technology advanced, the capability of these tools fell short.

As an industry, GIS has matured outside of mainstream database concepts over the past 20 years. The rapid growth in GIS technology in the past few years can be attributed partly to the incorporation of geospatial information into mainstream database concepts. CAD technology has not advanced as rapidly and we recognize an increasing technological gap between CAD and GIS systems. That technological gap is costly to maintain and can only be bridged by continuously expanding our code base.

Topobase over Oracle Spatial is an appropriate modernization of CAD as a true GIS client. Oracle Spatial solves the centralized storage of spatial and attribute information problem, in a seamless network of information. Traditional database architecture and database development methods can be applied to transactions at the database level. This means data integrity can be validated at the time of a transaction. In effect the data is never disconnected from the business rules. The flexibility of Topobase in its configuration and customization allowed us to develop and deploy Topobase in only 5 months.

An Industry Perspective

Traditionally, when a Facilities Management (FM) group and a Geographic Information Systems (GIS) group co-exist, a duplication of effort often occurs. While the FM group is responsible for maintaining accurate as-built documentation, the GIS group is responsible for maintaining a geographic database. Standard operating procedure dictates that the FM group completes the as-built engineering drawings and subsequently hands them to the GIS staff for facility digitization and data attribution. These two functions, as-builting and GIS data acquisition, share much of the same look and feel, as well as the same data and geometry. For these reasons, it is better to enter data once, at the beginning of the process, and have all products flow from that singular data source. Because a singular data entry point exists, the potential for user errors is significantly reduced.

Legacy CAD – GIS Integration (Bridging the Gap)

In 2001 we deployed our first CAD-GIS integration tools. Efficiencies were gained as tools were created and robust features were developed. Over time we found ourselves trying to turn CAD into a GIS client. This was a huge success for us and allowed CAD users to manage corporate GIS data. However, this came at a cost which seemed to be increasing. Over the years as the GIS technology advanced and we took advantage of those advancements our CAD-GIS integration tools needed to support those changes. Also, a changing environment and customer needs drove new feature class development. The CAD-GIS integration code base expanded greatly with each new feature class. Changes like this would prompt

expansion in feature extraction, feature manipulation, and the delivery system back to the GIS. Even with our best efforts and a code base of approx. 42,000 lines of code, we still experienced little quirks in the system which created bad data. These data errors were costly to identify and correct. Some of these quirks also had an impact on the perception from the users of the tools.

Autodesk Topobase over Oracle Spatial (Narrowing the Gap)

Topobase is a product that truly narrows the gap between CAD and GIS. Topobase essentially turns CAD into a true GIS client. The centralized storage of asset data supplied by Oracle fits the corporate strategy of LVVWD. Oracle as a back-end data server also allows for traditional database architecture methodologies to be employed at design time. LVVWD business rules and QA/QC processes are housed in Oracle triggers (Feature Rules) running at the row level during an edit transaction. This equates to better data management and a much cleaner application layer.

Microsoft Visual Studio .Net is the main programming interface for application layer customization. The Topobase API is feature rich and well organized and does not swamp the programmer by being overly bloated with objects and options. In fact, prior to the Topobase deployment neither of the authors had previous experience writing .Net code. A 3-day training session from Autodesk at the beginning of our implementation project forged the path to application layer customization.

The skill set necessary to customize and implement Topobase matched our existing skill set. For LVVWD the time was right to adopt this new technology. Previous decisions regarding CAD-GIS integration and the lessons we learned doing so proved helpful to the successful deployment of Topobase.

Topobase Implementation Methodology

Initially we had to decide how Topobase was going to co-exist with our existing SDE database. We decided that Topobase would stand beside SDE inside the corporate database and that both systems would be fully populated with our facilities data. This design requires that edits occurring in one system would have to propagate to the other. A method would have to be developed to support either pessimistic or optimistic locking between the two systems.

Next we had to look at where Topobase would logically fit into our business process and to identify where key automation was located on those processes. This allowed us to match the functionality of Topobase with our existing automation and exactly define the upstream and downstream hooks into the business process.

Because Topobase would replace our legacy CAD-GIS system, user acceptance was a key success factor. We wanted to provide a system with enhanced capabilities rather than just a duplicate system with a different name. At the same time we were careful not to fall into the “reinventing the wheel” trap. Existing business logic would have to be identified and carried forward into Topobase or abandoned in place. One of our main goals was code reduction, and we distinguished between critical tools for the initial deployment and efficiency tools that could come later, if at all.

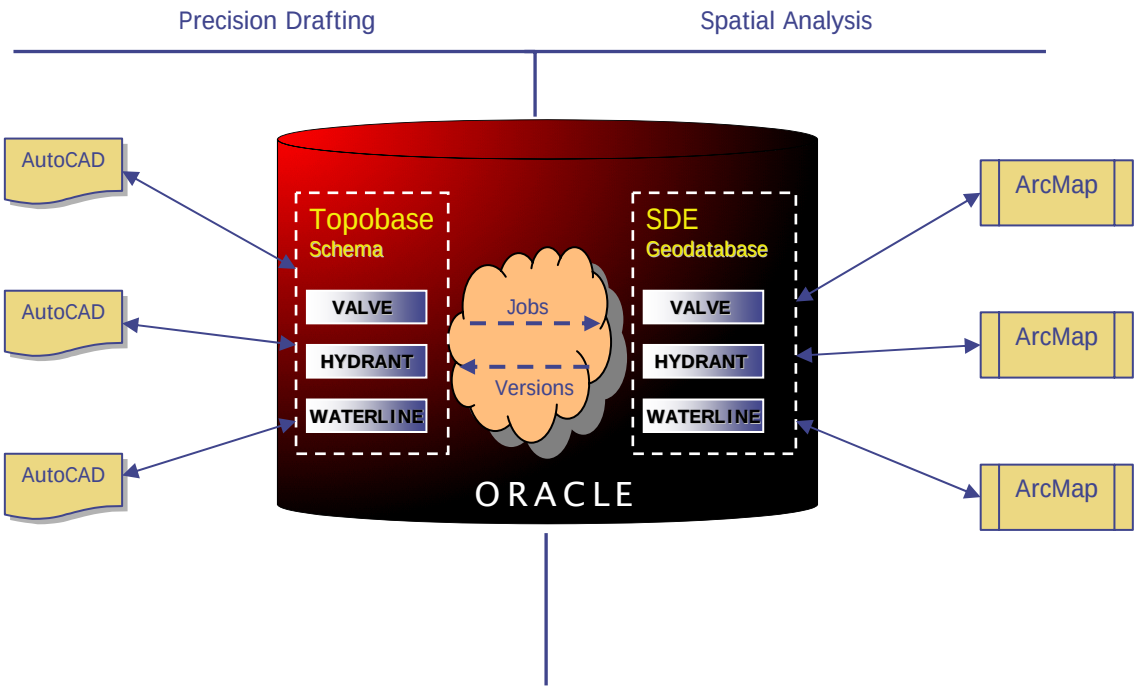


Figure 1. Topobase Implementation

Topobase Customization / Configuration

Data Design (Feature Class Creation)

Where else would we begin but good old database design? Having an existing SDE system that supported our business processes, it made sense to mimic the design of Topobase with that of SDE. This not only ensured Topobase would support our business processes but it made interoperability with SDE much less complex. The data definition between the two systems was modeled exactly – default values, data constraints and even data domains are synchronized at all times. This aids interoperability and ensures that if data gets into the database on one side, it flows without error to the other.

After creating a couple of feature classes using the Topobase Administrator we began to think there had to be a better way to do this. Using Topobase's SQL Spy, we were able to identify that the user interface was simply making calls to Oracle stored procedures. After researching the stored procedures we were able to extract the data design in SDE and script feature class creation in Topobase.

Sample Feature Class Creation Script:

```
var n number;
var v varchar2;
-- create a feature class (not all parameters are shown)
call TBADMIN.FeatureClass.CreateFeatureClass('VAULTPOLY',  , 'Vaults', 'WATERSTRUCTURES', … ) into :n;
--
-- Add column definitions
--
call TBADMIN.FeatureClass.AddColumnToFeatureClass('VAULTPOLY','IDFEATURE','       NUMBER(9) NOT NULL');
call TBADMIN.FeatureClass.AddColumnToFeatureClass('VAULTPOLY','IDQ','        VARCHAR2(6) default '999999' NOT NULL');
call TBADMIN.FeatureClass.AddColumnToFeatureClass('VAULTPOLY', 'INSTALLPROJID','          NUMBER(38) ');
call TBADMIN.FeatureClass.AddColumnToFeatureClass('VAULTPOLY', 'INSTALLPROJTYPE','         VARCHAR2(2) ');
call TBADMIN.FeatureClass.AddColumnToFeatureClass('VAULTPOLY', 'INSTALLDATE','    DATE');
call TBADMIN.FeatureClass.AddColumnToFeatureClass('VAULTPOLY', 'QAINSTALL',' NUMBER(4)    default 9        NOT NULL');
call TBADMIN.FeatureClass.AddColumnToFeatureClass('VAULTPOLY', 'QAMAP','        NUMBER(4)    default 6        NOT NULL');
call TBADMIN.FeatureClass.AddColumnToFeatureClass('VAULTPOLY', 'STATUS','       NUMBER(4)    default 1        NOT NULL');
call TBADMIN.FeatureClass.AddColumnToFeatureClass('VAULTPOLY', 'SUBTYPE','     NUMBER(4)    default 1        NOT NULL');
call TBADMIN.FeatureClass.AddColumnToFeatureClass('VAULTPOLY', 'WIDTH_FT','       NUMBER(38,8)                  ');
```

```
call TBADMIN.FeatureClass.AddColumnToFeatureClass('VAULTPOLY', 'LENGTH_FT',' NUMBER(38,8) ');
call TBADMIN.FeatureClass.AddColumnToFeatureClass('VAULTPOLY', 'DEPTH_FT',' NUMBER(38,8) ');
call TBADMIN.FeatureClass.AddColumnToFeatureClass('VAULTPOLY', 'ADDRESS',' VARCHAR2(75) ');
call TBADMIN.FeatureClass.AddColumnToFeatureClass('VAULTPOLY', 'ROTATION',' NUMBER(4) ');
call TBADMIN.FeatureClass.AddColumnToFeatureClass('VAULTPOLY', 'LASTUPDT',' DATE default sysdate ');
--
-- Add domain relations
--
call TBADMIN.Relation.InsertRelationDomain('AMFMDOMPROJTYPE_TBD' ,'VAULTPOLY' ,'INSTALLPROJTYPE', 'N');
call TBADMIN.Relation.InsertRelationDomain('AMFMDOMSTATUS_TBD' ,'VAULTPOLY' ,'STATUS', 'N');
call TBADMIN.Relation.InsertRelationDomain('AMFMDOMQAINSTALL_TBD' ,'VAULTPOLY' ,'QAINSTALL', 'N');
call TBADMIN.Relation.InsertRelationDomain('AMFMDOMQAMAP_TBD' ,'VAULTPOLY' ,'QAMAP', 'N');
call TBADMIN.Relation.InsertRelationDomain('AMFMDOMVAULTCONTENT_TBD','VAULTPOLY' ,'SUBTYPE', 'N');
--
-- Add additional relationships
--
call TBADMIN.Relation.InsertRelation('VAULTPOLY','IDFEATURE', 'AMFMRECORDDRAWING' ,'IDFEATURE', 'N');
call TBADMIN.Relation.InsertRelation('VAULTPOLY','IDFEATURE', 'FACWORKORDER' ,'IDFEATURE', 'N');
call TBADMIN.Relation.InsertRelation('VAULTPOLY','IDFEATURE', 'FACLOCK' ,'IDFEATURE', 'N');
call TBADMIN.Relation.InsertRelation('VAULTPOLY','JOB_VERSION','TB_JOB_VERSION' ,'JOB_VERSION', 'N');
call TBADMIN.Relation.InsertRelation('VAULTPOLY','IDFEATURE', 'FACGPSPTS' ,'IDFEATURE', 'N');
call TBADMIN.Relation.InsertRelation('FACLOCK' ,'IDFEATURE', 'VAULTPOLY' ,'IDFEATURE', 'N');
```

Dialogs Development

Development of data entry dialogs (forms) in Topobase is a straight-forward process because of Topobase Designer. The Designer automatically creates text and combo boxes for each feature class attribute. It is up to the developer to arrange the dialog controls and to create associations for one-to-many tabular relationships.

Our Topobase implementation required development of fifteen data entry dialogs representing fifteen feature classes. The waterline dialog is shown in Figure 2. Each dialog is composed of two main areas, the Master dialog (top), and the Detail dialog (bottom). The detail area can hold several detail dialogs representing multiple rows associated with the single row displayed in the main dialog. In this example, the detail dialogs include Record Drawings, Work Orders, Job Version, and Feature Locks. The master-detail relationship for Waterline to Record Drawing is highlighted in red. All one-to-many relationships are stored in the TB_RELATIONS system table.

In our implementation, many of the feature classes share similar attributes. For example, all fifteen feature classes require the IDFEATURE attribute. This primary key is not only used across the feature classes but also across multiple systems and departments to assign a unique identifier to each feature. A hidden benefit to utilizing Topobase dialog development is that all dialog and control information is stored in Topobase system tables. In the case of the IDFEATURE attribute, fifteen rows are stored in the TB_GN_CONTROL table representing this attribute across the fifteen feature classes. Changing the control's caption in all dialogs is easily accomplished in SQL by:

UPDATE TB_GN_CONTROL SET CAPTION = 'ID Feature:' WHERE NAME = 'IDFEATURE';

Using Topobase's built-in flexibility, it is easy to share detail dialogs between master dialogs. By default, Topobase names a detail dialog after the master and creates a new detail dialog for every master. The record drawing detail dialog was initially named WATERLINE#AMFMRECORDDRAWING. A second record drawing detail dialog was created for the VALVE feature class and was automatically named VALVE#AMFMRECORDDRAWING. To use the same detail dialog for both, one was renamed DETAIL#AMFMRECORDDRAWING and the other was deleted. Then, the TB_GN_DIALOG_DETAIL table was updated to point master dialogs to a single detail dialog.

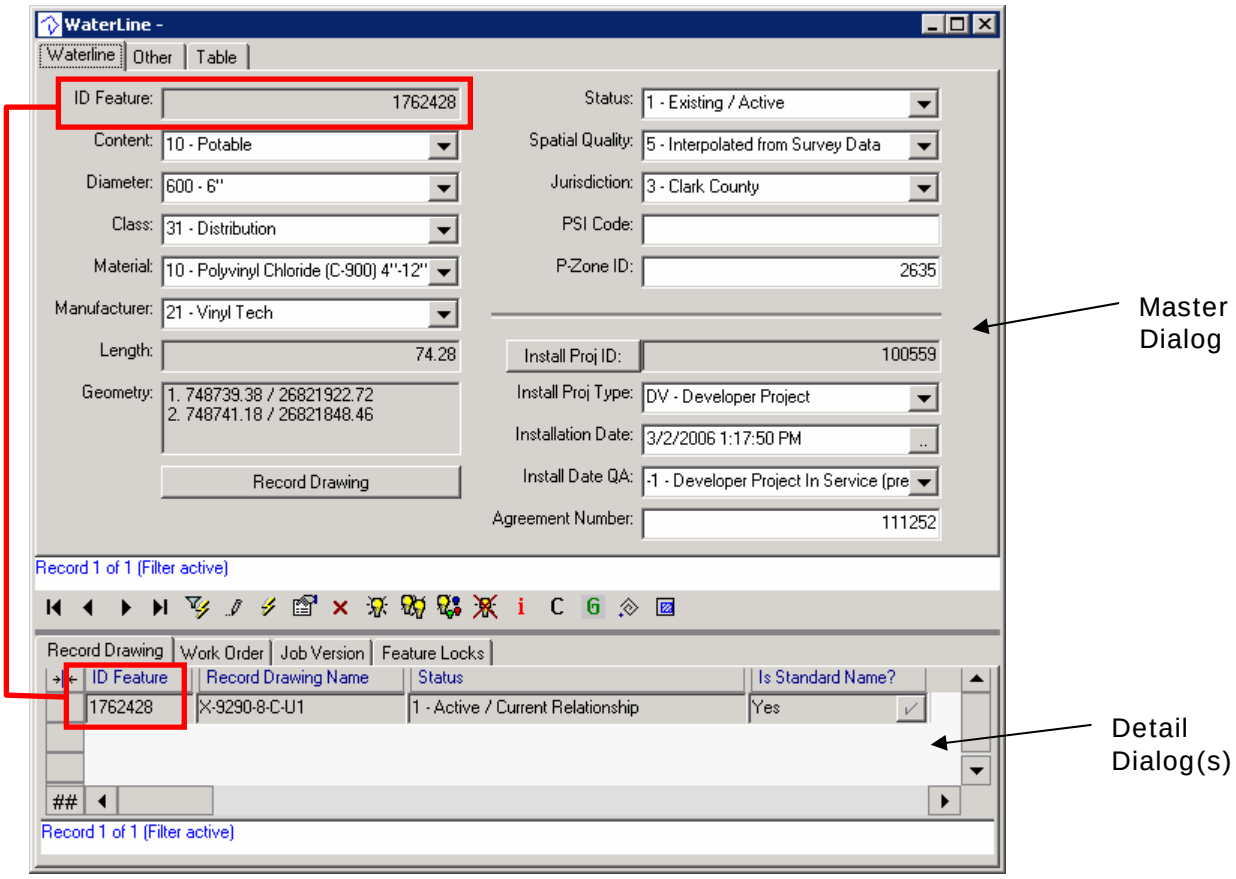


Figure 2. Waterline Data Entry Dialog

Display Model Development

The Display Model Editor (**DME**) is another very powerful feature of Topobase. This feature allows designers to control the symbology of each feature based on its attributes or the attributes of related features. This allowed us to create an editing environment rich with symbology that helps users capture data accurately.

In our implementation symbology denotes:

Feature state (open, pending, live)
Edit Operation (inserted, updated, deleted) **
Connectivity Status (connected, disconnected) **
Capture Completeness (incomplete, complete)
Lock Status (locked, unlocked) **

**display model details will be explained in this document

Edit Operation Display Model

The key to the edit operation display model is the TB_JOB_VERSION Topobase system table. Each edit that occurs on a job-enabled feature class is registered in TB_JOB_VERSION. The JOB_OPERATION_ID attribute indicates whether an edit represents an insert, update or delete operation. Once a relationship is established between a feature class and TB_JOB_VERSION the **DME** has access to the related attributes. The display model definition for Waterlines that have been updated in a job is shown below in Figure 3. Notice that the combination of Attribute_1,

Operator_1 and Value_1 essentially reads "tb_job_version.job_operation_id = 2". Perceivably this reads much like the "where clause" of a SQL query and that's exactly what it becomes. This was a critical discovery and allowed us to think of display models like SQL queries.

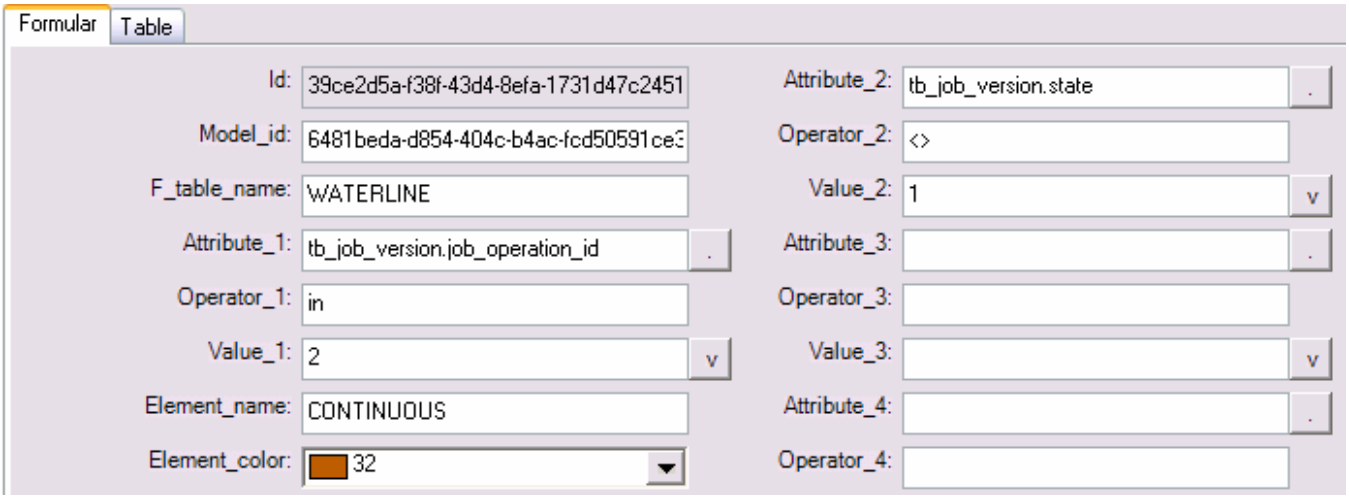


Figure 3. Edit Operation Display Model

Lock Status Display Model

We will discuss our feature locking model in detail later on. The key thing to understand is that feature locks are stored as rows in an external table. Again a relationship between the feature class and the FACLOCK table enables the **DME** to access the FACLOCK table. In this "where clause" we are looking for 2 things: the existence of a row based on the shared IDFEATURE key and a non-match between JOB_ID the current job ID. This query would read "faclock.idfeature = waterline.idfeature and faclock.lockversion <> tb_job_version.job_id".

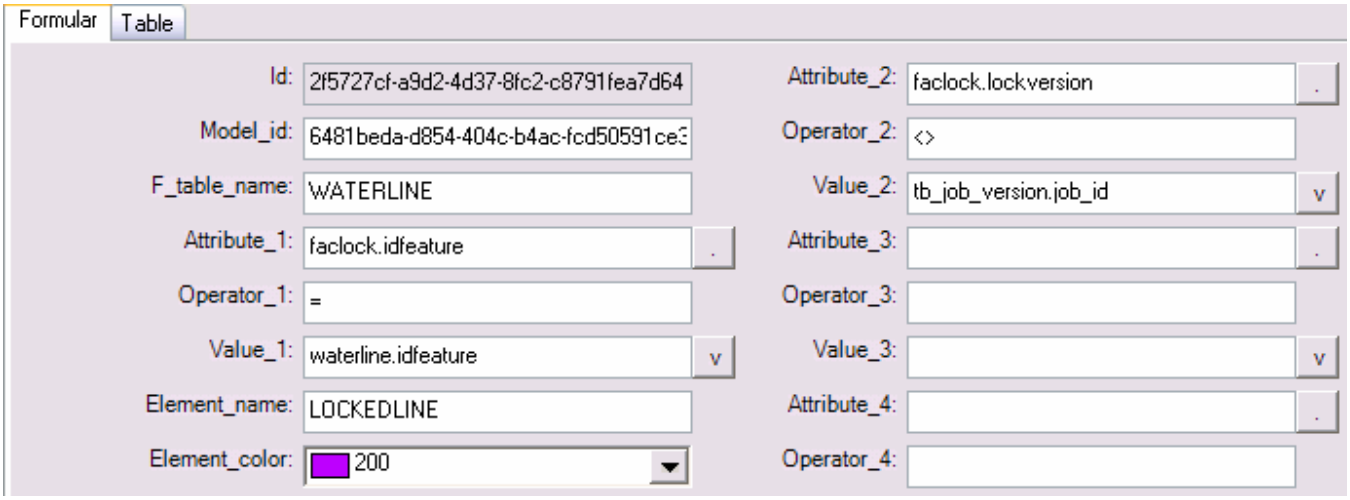


Figure 4. Lock Status Display Model

Connectivity Status Display Model

Connectivity is modeled through a connectivity indicator feature class we created. This feature class is not editable by the users and is managed behind the scenes via spatially enabled server-side feature rules. We will discuss this in depth later, but for now, just understand that the feature rules use the pseudo-attribute QUALITY on the connectivity indicator feature class to display connectivity problems. The feature rule detects an edit and adjusts the QUALITY of the connectivity indicator.

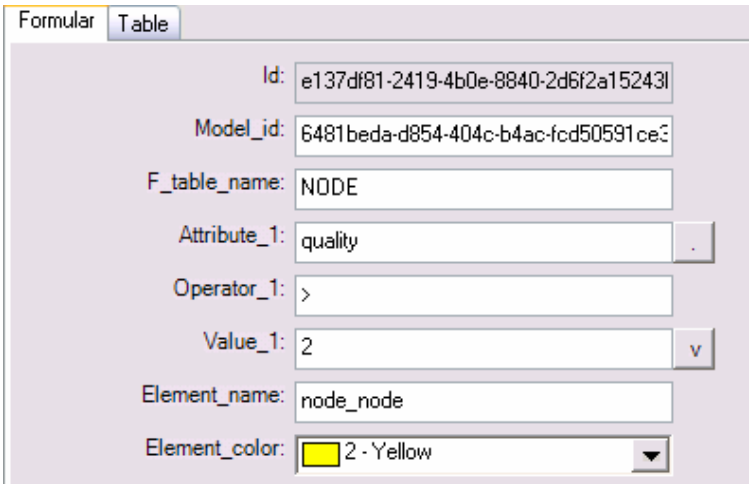


Figure 5. Connectivity Status Display Model

Connectivity Model

We decided not to implement a geometric network because it wasn't supported by our existing database design. Also, implementing a geometric network would have caused significant changes to our drafting methodologies which would have presented a greater challenge for our users. A geometric network enforces connectivity inherently behind the scenes. We decided to utilize Topobase display models to notify users of connectivity problems vs. enforcing connectivity while users are drafting. This presents an environment that allows users to draft freely while introducing them to the concept of network connectivity. Spatially enabled server-side feature rules detect change during the database transaction (any edit). These feature rules manage the QUALITY attribute on each feature class causing symbology changes via the display models.

Connectivity can be broken down into two categories, points moving to or away from lines and lines moving to or away from points. In the case of line features, because one line feature could be connected to multiple point features, the key is that a feature rule has access to the new and old geometry while still inside the transaction. But what does that mean?

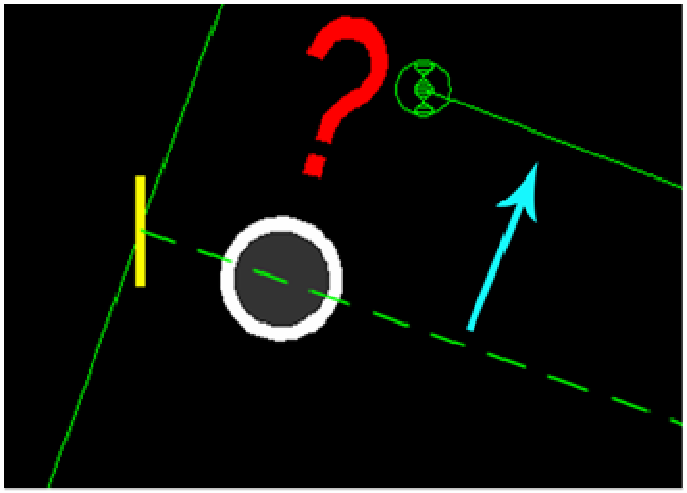


Figure 6. Anatomy of a Spatial Edit

To the right we see the anatomy of a spatial edit. The green dashed line represents the old geometry and the solid green line with the cyan arrow pointing to it represents the new geometry. Here the feature rule uses the old geometry to identify the valve that will be symbolized as disconnected and updates the QUALITY attribute of that valve.

In our environment WATERLINE is our base feature class, in that, everything must be touching a waterline to be considered connected. As such we developed a feature rule that checks to see if the new geometry of a point feature is touching a waterline, and then applied it to each feature class. The server-side feature rule you see below is applied to every point feature class. When the

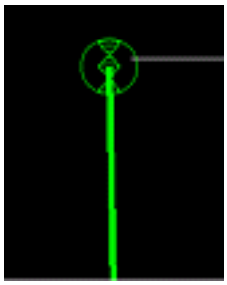
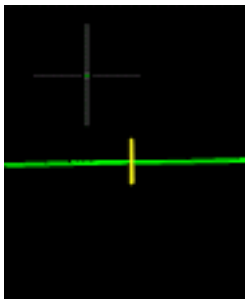
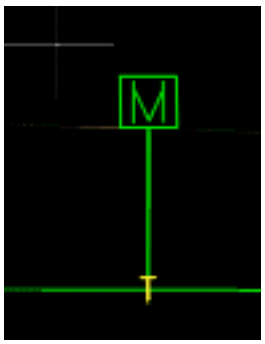
geometry of a point feature changes this feature rule determines connectivity and manages QUALITY.

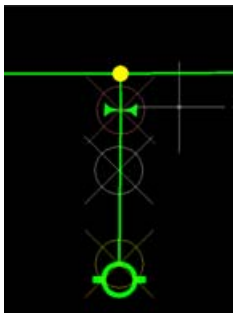
```
PROCEDURE LVVWD_DEVICE_TOUCHES_WL
    (dev_geom in mdsys.sdo_geometry, nQuality out number) AS
    hits number;
    myJob varchar2(6);
BEGIN
  --Only allow quality change to occur if the waterline is not locked in another job
  --Only allow quality change to occur if the feature is not deleted
  --Do not check connectivity if waterline.status > 1

  select to_char(job3.getjob()) into myJob from dual; --get current job
  -- do the spatial query
  select count(1) VAL into hits
    from TBAMFM.WATERLINE f where
    MDSYS.sdo_within_distance(f.geom,dev_geom,'distance = 0.001') = 'TRUE' and
    (select count(1) from tbamfm.faclock l where l.lockversion <> myJob and l.idfeature = f.idfeature) = 0 and
    (select jv.job_operation_id from tb_job_version jv where jv.job_version = f.job_version) < 3 and
    f.status = 1;
  --return the proper quality
  IF HITS = 0 THEN
    nQuality := 0;
  ELSIF HITS > 0 THEN
    nQuality := 1;
  END IF;

END;
```

This model forces all waterlines to be terminated by either a water device (blowoff, meter, hydrant, etc.) or a connectivity indicator. This helps to ensure network connectivity and also shows connectivity problems to the users. Figure 7A shows a connectivity indicator at the endpoint of a waterline. Figure 7B shows the connectivity indicator where two waterlines are touching. Figure 7C shows three waterlines coming together at a tee with no break at the mainline; this occurs when a service lateral taps a main. Figure 7D shows the indicator when 3 or more waterlines are connected at their end points.



Figures 7A. 7B. 7C. 7D.
Connectivity Indicators

The logic for connectivity indicator management is unfortunately much too complicated to explain in this brief paper. Hopefully, the content of this document in its entirety is enough to lead you down the right path.

.Net Customization

The Topobase Application Programming Interface (API) is very accommodating and allows the developer to customize the application through .Net dlls. We were able to create tools for data entry validation, external child table management, feature manipulation, and data import/export.

To control the user editing environment we created tools using VB.Net. Our .Net customizations came about as a result of our survey of our environment and our identification of critical business tools. This automation fell into 3 categories. First, forms development is mainly used to enforce child table validation rules. Second, workflows are algorithms applied to multiple features at once. Finally, feature editing tools are aimed at increasing efficiency during feature editing and enforcing business rules. The following tools were developed:

- Forms Development
  - Record Drawing Linking
  - Valve-Hydrant Linking
  - GPS Linking
  - Callout Generator
  - Project Information Update
- Workflows
  - AFL to Topobase
  - Remove SDE Versions
- Feature Editing Development
  - Split/Join Linear Feature
  - Digitize with Rotation
  - Rollback Edits
  - Grow Perimeter
  - Publish Job
  - AutoMeter

Locking Model

In the Geospatial industry much attention is paid to the idea of multi-user concurrent editing or allowing users to edit the same features with the software detecting conflicts and generating conflict resolution events. As far as we know, no business process is willing to absorb the cost of conflict resolution, and what typically happens is that one edit overrides another with the complete loss of the first edit.

In our environment allowing conflicts to occur is just not practical. Besides it simply doesn't happen often enough to warrant the cost of conflict resolution. We opted for conflict avoidance with a pessimistic locking model, which guarantees our users a lossless editing environment. Pessimistic locking also aids in our interoperability between SDE and Topobase because a lock from one system ensures the directional flow of data.

External System Requirements (SDE)

Prior to the Topobase implementation we already had a GIS database design that met the needs of our business processes. We chose to adopt that same design in Topobase for the previously stated reasons, but also because system equivalency is the key to interoperability. The idea is that a feature class in SDE named Valve is synchronized with a feature class in Topobase named Valve.

We identified many child tables that needed to logically sit between the two systems. For example, we have a GPS points table that we use to link with facilities. Rather than duplicating this table of

600,000 rows in SDE and Topobase and also dealing with synchronization, the two systems share the table.

Topobase – SDE Interoperability

What is the business case for having 2 GIS databases, other than the principle that if one is good two is better? We live in the fastest growing city in the United States. Our CAD team as-builts over 800 projects a year, while our GIS produces over 7,000 maps per year. To keep up with that growth we look to automation to reduce manual and redundant processes.

Over 90% of all GIS data input is performed by our CAD staff. The CAD team understands the engineering logic behind the water distribution system and are able to best use the CAD engineering drafting tools to perform as-builting. In our data driven process the data is entered into the GIS once and all products flow from that single data source. Topobase is a natural fit for this activity. Topobase elevates AutoCAD to the status of a true GIS client.

The natural model for true CAD-GIS integration would be to have both Topobase and ESRI ArcGIS systems capable of supporting one database. This was problematic due to the methods by which these software support long transactions or versioned edits. A simplier approach was developed where both systems would be fully populated with data and edits made in either system would propagate across to the other side. This would allow GIS users to use the ESRI software they are accustomed to and allow CAD drafters to use Autodesk tools of which they are familiar. But how did we achieve this task?

There are 2 key factors to synchronization. First, share a key between the two systems. SDE requires its feature classes to have OBJECTID and Topobase requires a field called FID. Each system manages these primary keys but we, for the most part, ignore them. We maintain our own primary key called IDFEATURE, which allows us to identify features across systems. Second, we developed a locking model. We only allow a feature to be updated in one system at a time. This ensures that edits will only flow in one direction.

What are the actual mechanics? We tried utilizing existing functionality in each system, but ran into numerous problems. We looked at using a request broker system to perform synchronization activities, but wanted the users to get an immediate response. Our final implementation is a push button solution where the Topobase .Net API interacts with the ArcObjects .Net API in a single dll.

SDE versions and Topobase Jobs allow us to identify what edits have occurred. Similar functionality is available on both sides of the API to perform those edits in the other system. In each API you drive down to the feature level and use one feature's attribute values to populate the other. This is the most granular level of compatibility available as the two APIs pass integers, strings and dates between features (see Figure 8).

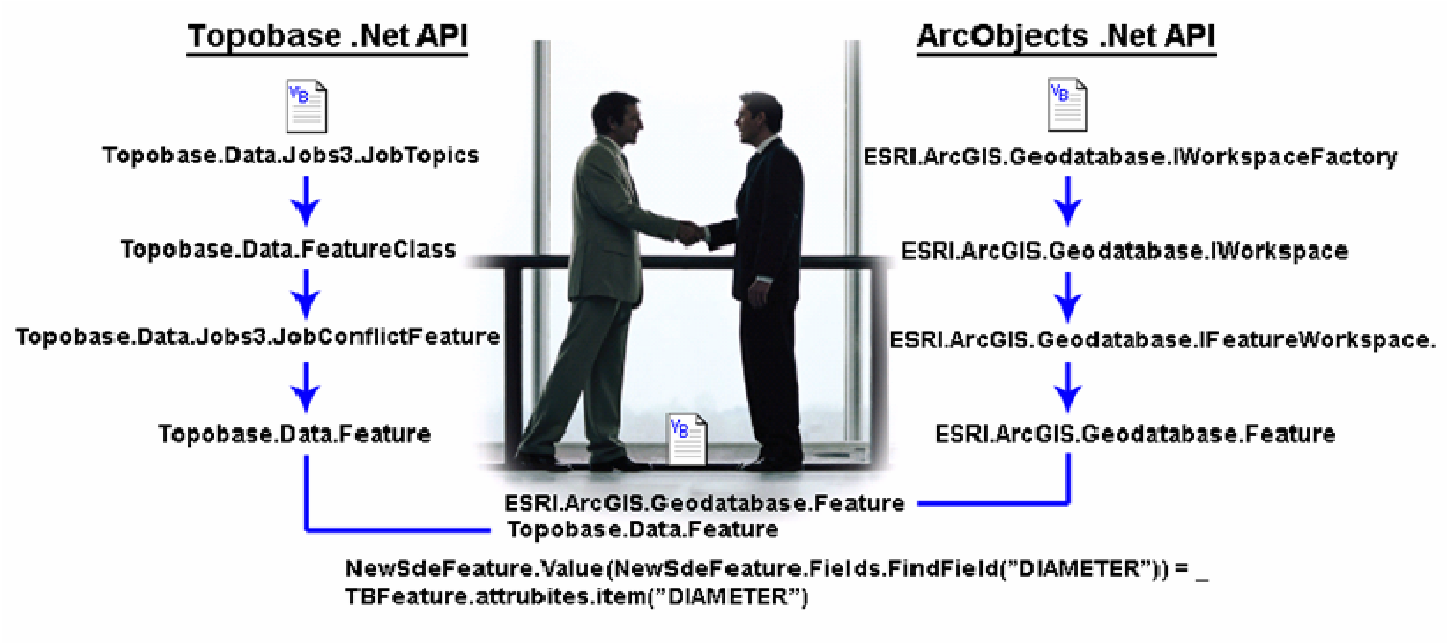


Figure 8. ArcSDE-Topobase Push Button Synchronization

Where Are We Now?

After spending approximately 5 months developing 35 feature classes and associated customizations, we are running Topobase in our production environment. In the first month, we've completed 57 gis capture and as-builting projects installing 13.5 miles of waterline, 249 valves, 87 fire hydrants, and 1,037 service connections. Those same projects updated 4.1 miles of existing waterline, 106 valves, 29 fire hydrants, and 64 service connections. Needless to say, the system is functioning well. Our project completion rate is expected to improve by at least 20% and the time spent by GIS personnel identifying and fixing errors associated with project work is expected to decline by at least 15%. The cost savings achieved through more efficient work processes and reduced data oversight is conservatively estimated at $250,000 annually.

Code management has been significantly reduced. We tabulated the lines of actual code (spaces and comments not included) required for CAD to GIS integration in our old system versus the lines of code developed for the Topobase system. Astonishingly, our code base has been reduced by 82% from 41,900 to 7,600 lines. Previously, we used multiple programming languages including AML, Java, Lisp, AutoCAD VBA, and ArcObjects VBA. The new system is based primarily upon VB.net and PL/SQL, a welcome consolidation. The time savings associated with system operation and maintenance is at least 60%. The cost savings achieved through a much more efficient and powerful application interface is estimated at $210,000 annually.

In addition to the modest monetary benefits listed above, we now have better editing tools and two tightly-coupled GIS systems with built-in redundancy. Our editing environment assists drafters via on-the-fly spatial and attribute validation. Real-time data availability insures that asset information is readily available to employees company-wide.

What's Next?

Topobase Web (TB Web) has the potential to change our current business process, and will be investigated in our next phase of implementation. The biggest advantage of TB Web is that it allows us to create data entry forms which can be used by both desktop and web client users. Essentially, web users including external customers could be granted the ability to not only view, but to digitize

features into our system. This could provide benefits to local land developers including faster project approval times and possibly on-the-fly engineering analysis. Imagine the developer having the ability to compute fire-flow rates, analyze costs, and determine water pressure requirements directly from our information system, while sitting at their desk!

Autodesk has released the next version of the software, Topobase 2007. Currently, the District's Oracle 9i database is incompatible with Topobase 2007. Specifically, updates to Oracle Spatial's networking modules in Oracle 10g address some of the limitations in Oracle 9i. Because we are not utilizing Topobase's network analysis, we may not be affected and may be able to implement Topobase 2007 in Oracle 9i. Testing and implementation of Topobase 2007 is to begin immediately.

Summary

Implementation of Autodesk Topobase at the Las Vegas Valley Water District has been a huge success for the following reasons:

- The Topobase software allows AutoCAD to act as a true GIS client.
- Open GIS standards utilized by Oracle Spatial and Topobase allow for extreme flexibility in system design and integration – Topobase easily adapts to the existing business process.
- Standard Topobase functionality including the ability to create data entry dialogs for external data sources allows for rapid integration with external systems like ArcSDE.
- Topobase display models provide immediate feedback to users at the time of data entry thereby reducing errors and creating efficiencies.

Benefits to the AM/FM/GIS division include:

- A code base reduction of 82%,
- Annual cost savings projected at approximately $½ million, and
- A better set of tools for data capture and as-built processes.