MANAGING DATA LAYER UPDATES

I. PURPOSE AND CONTENTS

Data maintenance is often an expensive yet neglected aspect of managing GIS data. Each data layer is unique and will have its own system for making and tracking updates. By consequence, tools for data maintenance are generally not built into GIS software.

This guideline contains a collection of techniques and procedures for managing GIS data layer updates. Appropriate techniques can be adopted (with modification as need be) for a given data layer. Some data layers change very little over time, and may only require simple procedures for reporting errors and making rare revisions. Other layers change often and require procedures for recording updates in a manner that supports quality control and tracking of changes over time. A full range of techniques is discussed here, including methods which enable an audit trail of modifications over time for a given feature. The text is primarily intended for those familiar with the Arc/Info GIS software.

Data layers with complex maintenance needs should have individualized update standards. For layers with no update standard, the user must choose the procedures that will work best for the given layer. It is hoped that individual update programs will be able to draw techniques from these guidelines.

The first version of this guideline is primarily based on the author's experience and a brief review of conference proceedings. Any additional techniques, resources and general comments would be much appreciated and will be incorporated in future versions as appropriate.

II. DEFINITIONS

These terms are defined as used in this document, but are not necessarily based on common usage.

Update	Any modification made to the data; each update is either a *correction* or a *revision*.
Correction	An update which rectifies an error (feature location or attribute values) in the data.
Revision	An update which reflects a change of state for the represented entity. A parcel subdivision requires a data *revision*, although no error has been corrected.
Feature	An Arc/Info data element: point, arc (or line), polygon, polygon label point, annotation, and several others.
Coverage	An Arc/Info data set with features representing a certain theme (e.g., roads or wetlands)

Layer A GIS layer of one or more coverages on a common theme (a layer may may be broken up geographically into multiple *tiles*).

III. UPDATE DOCUMENTATION FILES

The VGIS standard on *Data Layer Documentation*, in the *VGIS Handbook*, describes two files for tracking updates. The <layer>.UPD file should be used to describe any major update of the data layer, at which time a new 'version' of the layer is established (the date of the major update is used as the layer 'version'). The <cover>.HST file is specific to each coverage, and can be used to record notes on minor modifications to the data layer being developed or updated. Neither of these files is suitable for tracking individual feature updates over time. Examples of such files are given in the section on tracking update transactions, below.

IV. REPORTING UPDATES TO THE DATA MANAGER

Data available through VCGI (and elsewhere) are often refined at a more local level. Any errors found are often corrected in the local data set without being reported back to the data manager for correction in the original data. Such corrections are easily forgotten if not recorded. At a later date an improved data set may be available in which these corrections may not have been made, presenting a lose-lose situation for the data developer and the data users.

When refining data acquired elsewhere, it is imperative that any errors found be reported back to the data manager (listed in the documentation), or to VCGI. Several procedures for reporting errors (or needed revisions) are listed below. If you are uncertain of the best procedure to use, please contact the data manager.

1. **Phone Call**. If there are only a few needed updates to report, a phone call may be adequate. However, if features need to be moved then it may be necessary to provide the corrected data (see below).
2. **Data Problem Report Sheet.** A *Data Problem Report Sheet* can be found in Appendix E of the *VGIS Data Catalog*. This includes space for describing the problem, where it's located, the feature's unique ID, corrections made and so on. Where occasional errors are being corrected, this may be the easiest way to record them for future submission to the data manager.
3. **Submission of Updated Data**. Corrected or revised data can be submitted to the data manager for incorporation into the original data. However, the data manager will need to quality control the incoming data, and may want to check the submitted corrections one-by-one before incorporating them. The

corrections should be described (with locations and/or identifiers) to enable the data manager to isolate the needed corrections from the data layer as a whole.

Update attributes added directly to the layer (to the AAT and/or PAT) are a very useful, easy way to describe any updates made. Some or all of the attributes below (or variations thereof) may be appropriate:

Field	Type	Content
ADD_DATE	Date	Date the feature was originally added to the layer
UPD_DATE	Date	Date the feature was updated (corrected, added, etc.)
UPD_ACT	1 Char	Update action: 'A' = Added feature 'D' = Deleted feature 'M' = Moved feature (including reshaped arc) 'S' = Split an arc into two new arcs 'U' = Unsplit two arcs, leaving one arc 'C' = Changed attribute value(s) 'N' = Not updated (original version)
UPD_SRC	10-40 Char	The source map or other information used to make the update
UPD_TYPE	1 Char	Distinguishes, if desired, between: 'C' = Correction, 'R' = Revision
UPD_WHO	10-30 Char	Who made the update (organization and/or name or initials, e.g., 'VCGI: NDD')
UPD_NOTE	30-60 Char	Note describing the update made

Note that if there are multiple updates to a feature, only the last update will be recorded. Also, there will be no record for deleted features unless they are coded as deleted (UPD_ACT = 'D') and left in the data. (For the submission of updated data, the data manager must be aware of any deleted features.) The section below on **Tracking Update Transactions** describes procedures for recording individual edits to a feature, and for storing deleted and other spatially altered features in a 'transaction' coverage.

The ADD_DATE is only needed if there is a desire to record when the feature was originally added to the data (in addition to the update information). If the ADD_DATE attribute is added to an existing layer, an initial date would be set for all current features, described in the documentation for the ADD_DATE item.

Original features that have not been updated would have the UPD_DATE set to the original version date (the 'current version' date), and the UPD_ACT set to 'N'.

V. MANAGING LAYER ARCHIVES

Periodic archives of a data layer are the easiest method for recording changes over time. A data archive system should allow the user to:

- list available previous versions of the data layer,
- restore previous versions, and
- understand what is new with each update.

Several useful procedures for managing data archives are given below.

1. <u>Distinguish between the *current version* and *working copy* for a data set</u>.

Current version: a static data set, fully documented with an update record (in the <layer>.UPD file) describing the actions of the last update and providing the date of update (the layer date serves as the layer version).

Working copy: a copy of the last version to which edits are made. Periodically, or when there are sufficient edits, the working copy becomes the current version, documented with a new update record and date stamp.

2. <u>Use a systematic naming convention and/or directory structure to distinguish between the current version and working copy</u>. At VCGI, all current data for distribution are kept in a separate directory (actually a separate network drive, `J:`). The directory structure is based on the various tile structures for VGIS data. All data under development (or revision) are stored in a separate directory on a separate network drive (`h:\ddev`), under the same directory structure as the current versions. Thus there is no confusion between current versions and working copies.

A layer naming convention can also be used to distinguish between current versions and working copies. For example,

> "The working copy will have the underscore character appended to the coverage name. If necessary, the last character of the coverage name will be replaced with an underscore."

Therefore, for datalayer OPENLAND the working copy would

be named OPENLAN_.

Beware: Where the current version and working copy are stored in the same directory there is a risk that they will be confused, either by accident or by an inexperienced operator unfamiliar with the naming convention. Therefore, use of a separate drive or directory for data under development is preferable to use of a coverage naming scheme.

3. **Name archives for easy future reference.** For a data layer that is annually archived, it may be easiest to append the year to the coverage name (e.g., PARCEL91, PARCEL92, etc.).

   For archives not on an annual update cycle, a naming convention which includes the archive date may be used. For example, the data to be archived is placed under a directory 'arYYMMDD' where YYMMDD is the year, month and day. The YYMMDD order is preferred so that dated directories will sort chronologically, and to avoid confusion between MMDDYY and DDMMYY.

   It is also useful to put all archives under the same parent directory, so that they may all be restored from tape by restoring the single directory. For example, archives for data layer RDS might be organized as follows (DOS directory structure):

   ```
   \rds\ar920815
   \rds\ar930207
   \rds\ar930521
   etc.
   ```

4. **Maintain an archive listing.** A separate archive listing should be maintained as a text document describing the contents of the tape library (or other media). At VCGI, the archive listing includes columns for tape name, archive date, directory backed up, parent directory (from which the backup was made), backup command used, data format and data contents.

5. **Make multiple archives when appropriate**. Tapes and diskettes may become corrupted or damaged. When appropriate, make two archives and store one off site.

6. **Maintaining 'CLEAN' and 'unCLEAN' data**. Some users are concerned that running the CLEAN command multiple times will shift features around somewhat randomly, resulting in

'fuzzy creep'. Generally, this concern is unfounded (for an article dispelling the myth of 'fuzzy creep', see ArcNews, Summer 1991, p. 6). Once a coverage is CLEAN, successive CLEANs with the *same* fuzzy tolerance will only move new features added *within* that fuzzy tolerance. Under very unusual and unlikely circumstances, when features are successively added *within the fuzzy tolerance* of an original feature, there may some shifting of a feature beyond the fuzzy tolerance.

Concerns over 'fuzzy creep' persuade some data developers to maintain two versions of their data, a CLEAN and unCLEAN version. All edits are made to the unCLEAN version (the working copy), which is CLEANed when needed to establish a new current version. This procedure may be useful in very special situations but is generally not needed. Often it is best to establish a CLEAN coverage as the first version, then require that all successive updates maintain proper topology so that the coverage need not be CLEANed again. (Maintaining proper topology means no intersections, all polygons closed and one point per polygon).

VI. TRACKING UPDATE TRANSACTIONS

For certain highly sensitive data layers it will be desirable to track updates on an edit-by-edit, or transaction, basis. Although tracking edit transactions introduces overhead, the process can be automated to minimize the time to keep records.

The need for recording transaction records and the level of complexity implemented will depend on:

- the frequency of updates,
- the types of updates being made,
- the desired level of quality control checks for updated data, and
- the need to be able to trace the history (an audit trail) of a given feature.

Goals for Recording Edit Transactions

Each layer should have its own update procedures. Several potential goals of procedures for recording edit transactions include:

- To enable, for each feature, generation of an audit trail showing when the feature was added and the updates that have occurred over time.
- To distinguish between and identify corrections and revisions.
- To enable quality control checks on updated data.
- To display the state of the data at any given time, including between archives.

- To minimize the burden of record keeping imposed on those maintaining the data.
- To minimize data layer complexity.

It is also assumed that topology is maintained as the working copy of the data is edited, so that it will not be necessary to CLEAN the coverage. If CLEAN is to be used, care must be taken to assure the transaction file is properly maintained.

Data layers are occasionally subject to global updates, as by overlays with other layers, removing all unnecessary pseudo nodes or similar global operations. For such global operations it probably will not be practical to record all edit transactions. Ideally, all such operations would be completed during initial data development, before transaction recording is begun.

In the world of relational data base management systems (RDBMS), there are many tools for on-line transaction processing (OLTP) and a vast literature. Those tools and techniques could likely be transferred to the GIS environment. Users wanting more detailed discussion on transaction processing are encouraged to investigate those sources.

GIS edit transactions can be tracked to varying levels of detail. Several techniques are described below in order of increasing complexity (and functionality).

1. **Record the 'add' and 'update' dates.** The simplest method of recording updates is to do so in the coverage feature attribute table(s). The attributes described in section IV.3. are recommended for this case. Note that these attributes record the most recent update for a feature, but do not record all updates over time.

2. **Record transactions in an 'update transaction' database file.** If a database file (INFO or dBase) is used for tracking edits, then there must be a unique identifier (here called uniqID) for each feature that links the feature to the database file. A uniqID cannot be reused when a feature is deleted (unless the UPD_DATE is used to distinguish different features with the same uniqIDs at different times, which increases complexity).

   It is recommended that transaction database files be named <layer>.PATUPD or <layer>.AATUPD, since they are intimately linked to the unique identifiers in the PAT and/or AAT. (In pcArc/Info the DOS files would be named

PATUPD.DBF or AATUPD.DBF.) For tiled data layers, it may be sufficient to have a single transaction file, rather than one for each coverage. The transaction file should be documented under 'Associated files' for the data layer in the <layer>.DOC file.

The following fields are recommended for transaction files:

Field	Type	Content
UPD_NUM	Integer	A unique update (transaction) number (if needed to relate to a transaction coverage, described below)
old_uniqID	Same as in coverage	The original uniqID linking to the feature attribute table
new_uniqID	Same as in coverage	The updated uniqID
UPD_DATE	Date	Date the feature was updated (corrected, added, etc.)
UPD_ACT	1 Char	Update action: 'A' = Added feature 'D' = Deleted feature 'M' = Moved feature (including reshaped arc) 'S' = Split arc 'U' = Unsplit arcs 'C' = Changed attribute value(s)
UPD_TYPE	1 Char	Distinguishes, if desired, between: 'C' = Correction, 'R' = Revision
UPD_SRC	10-40 Char	The source map or other information used to make the update
UPD_WHO	10-30 Char	Who made the update (organization and/or name or initials, e.g., 'VCGI: NDD')
UPD_NOTE	30-60 Char	Note describing the update, as needed

These fields allow for complete audit trails for arcs and points beginning with when each feature is added to the coverage. The old_uniqID and new_uniqID are only needed for splitting and unsplitting arcs (as described below); point data can have a single uniqID field.

Several edits require special handling in the arc transaction file (<layer>.AATUPD):

a) **Add, Delete, Move or Change an attribute value**. The old_uniqID equals the new_uniqID. Both are filled in (no null values allowed).

b) **Reshape an arc**. A reshaped arc is considered to be moved.

c) **Move a node**. If a node is moved, each arc attached to the node has been moved and is entered into the transaction table as a moved arc.

d) **Split an arc**. A split arc begins with one uniqID but ends with two uniqIDs. Usually, the original uniqID is carried over to one of the resulting arcs. The old and new uniqIDs are recorded for each new arc, both with update type 'split' (UPD_ACT = 'S'). In this way the audit trail can be traced back to when the original arc was added.

e) **Unsplit two arcs**. Two arcs may be unsplit if their attributes are identical, in which case one of the unique_ID's is lost. In this case two transaction records are entered, one for each original arc, both with the new uniqID and update type 'unsplit' (UPD_ACT = 'U').

f) **Move a pseudo node**. A pseudo node is moved along two connecting arcs by splitting in a new location, transferring attributes to the middle arc, then unsplitting at the old node. This operation is equivalent to reshaping the two original arcs, thus it is recorded in the transactions file as two moved arcs (i.e., two records).

Here is a sample of transaction records in <layer>.AATUPD for a line coverage (UPD_DATE, UPD_TYPE, and UPD_SRC not shown):

old_uniqID	new_uniqID	UPD_ACT	Action (could be UPD_NOTE)
105	105	A	Feature added, first time
577	577	D	Feature deleted
299	299	M	Arc reshaped, or node moved
432	488	S	Arc split, new uniqID assigned
432	432	S	Arc split, uniqID carried over
301	301	C	Attribute changed
301	306	U	Unsplit, uniqID changed
306	306	U	Unsplit, old uniqID retained

Transaction records must be added in chronological order; otherwise, a full time stamp is needed to order updates made on a feature on the same day. To create a transaction table for an existing data layer, it may be desired to initially populate the table with one record for each feature, and to assign an initial 'add' date for all current features.

3. **Use of transaction records to record changes in attribute values**. If desired, transaction records can record individual changes in attribute values by including all feature attributes in the transaction file, and copying the original attribute values to the transaction file when a change is made. However, in most cases it will be easier to simply compare the attributes in the original data with the attributes in the updated data, feature by feature.

4. **Use of transaction records for quality control.** For data being updated by different people or at multiple sites, transaction records allow the edit history of a given feature to be traced. Any questionable edits can be traced to their source (by the UPD_WHO field) for an explanation as needed. If the quality of an operator's work is in question, his or her edits can be spot checked or individually verified.

Updated data (the 'working copy') can be compared with the original data ('current version') feature-for-feature by relating their attribute tables by the uniqID. Several sets of features can be identified, as described in the following table:

Feature set	Method for selecting features
Features with new uniqIDs	Relate new attribute table to old table, select records with no match
Deleted uniqIDs	Relate old attribute table to new table, select records with no match
Arcs with modified locations/shapes	Select matched records that have different LENGTH values - virtually all moved arcs will have different lengths
Points with modified locations	Select matched records with different X,Y coordinates (use ADDXY)
Features with modified attributes	Select matched records with different attributes

These sets of features correspond to the update actions (UPD_ACT) recorded in the transactions file as follows:

QC check	Transaction type (UPD_ACT)
New uniqIDs	Added or Split ('A' or 'S')
Deleted uniqIDs	Deleted or Unsplit ('D' or 'U')
Modified locations/shapes	Moved, Split or Unsplit ('M', 'S' or 'U')
Modified attributes	Changed attributes ('C')

A complete record of edits made since the last update can be provided by listing all transaction records back to the date of that data version. For each modification revealed between the original and updated data, there should be a corresponding transaction record.

As well, the edits recorded in the transaction file can be compared with the types of edits described (in the <layer>.UPD file) to have been made. For example, if a set of edits was made to "correct road locations," then there should be moved roads but no deleted roads. Any deleted roads noted in the transaction record can be reviewed to be sure they are valid.

As another example, suppose a specific attribute should not have been edited. Using transaction records, it can be verified that the values for this attribute are unchanged for all original features. However, if edits have involved splitting and unsplitting arcs, then the transactions may have altered the uniqID of an arc several times. Consider transaction records for an arc that is split twice:

old_uniqID	new_uniqID	UPD_ACT	Action (could be UPD_NOTE)
105	105	S	Arc split, uniqID carried over
105	402	S	Arc split, new uniqID assigned
402	402	S	Attribute changed
402	402	S	Split, uniqID carried over
402	403	S	Split, new uniqID assigned

A portion of the arc starting with uniqID 105 became number 402, had a change in attribute value, then became number 403. In this case, the complete audit trail would have to be examined to determine whether the attributes had changed, and if the change was valid.

5. **Use of an edit transaction coverage.** Transaction files record updates but do not explicitly record changes in feature locations. For example, a transaction file may record that a feature was deleted or moved (or reshaped), but does not show the state of the feature before being modified. The audit trail can be followed back to the previous version of the layer to see the feature in its original state, but it may be preferable to store modified features in a separate 'transaction' coverage.

A transaction coverage could be used without a transaction database file (.AATUPD and/or .PATUPD), but is likely to be most effective when used together with a transaction file (or files).

The transaction coverage can be given the same name as the original coverage with a 'UPD' suffix (TB24UPD for coverage TB24, or PARCUPD for coverage PARCEL). For tiled data layers, a single transaction coverage may be sufficient. The transaction coverage should be documented under 'Associated files' for the data layer.

Features to be updated must first be copied ('put') into the update coverage; that is, the transaction coverage records the spatial state of each feature before it was updated. Features are only put into the transaction coverage if the action modifies the

location or shape of a feature. An attribute in the transaction coverage attribute table (AAT and/or PAT) is used to relate to the file of transaction records:

Field	Type	Content
UPD_NUM	Integer	Transaction file update number

Feature attributes can be copied to the transaction coverage (or to the transaction database) as needed.

Proper use of the transaction coverage is described in the table below for several specific edits:

Edit	Feature(s) copied to transaction coverage
Delete a feature	The feature to be deleted
Reshape an arc	The original arc
Move a node	Each original arc attached to the node
Split an arc	The original, unsplit arc
Unsplit two arcs	The original two arcs
Move a pseudo node	The original two arcs

The transaction coverage should never be CLEANed, as this would potentially create undesired intersections or eliminate duplicate features.

6. Recording update transactions for polygon coverages. Polygons are typically made up of one or more arcs bounding the polygon, and a single label point. Any edit transactions on a polygon can be accounted for by recording the edits made to individual arcs and label points. If a transaction coverage is used (for spatially modified features), then moved or deleted label points will be copied to the transaction coverage and stored there as independent points (the transaction coverage cannot have polygon topology).

7. Automating the update process to record edit transactions. With an established process for tracking updates, macros (AMLs and SMLs) can be used to automate the process of keeping transaction records for specific operations. For example, consider the case of a line coverage for which transactions are recorded in a database file (<layer>.AATUPD) and in a transaction coverage (for moved and deleted features). Automation methods are described below for several specific edits:

Startup. Before editing starts, set up the edit environment, determine the next unique feature ID, and determine the current maximum transaction update number (UPD_NUM).

Delete an arc. Select the arc (by clicking on the arc or by logical expression). Increment the update number (from a global variable). Put the arc into the transaction coverage, and assign it the update number (this could be passed as feature ID by temporarily changing the $ID to the update number). Write a record to the transaction file with the UPD_NUM, UPD_ACT = 'D' and any other pertinent information. (For pcArc/Info, the record could be written to a text file which can later be appended to the transaction database file). Delete the arc.

Split an arc. Point to where the arc is to be split, and capture the X,Y location. Increment the update number. Select the arc at X,Y and put it into the transaction coverage with the UPD_NUM. Split the arc at X,Y. Query the user to point at the two new arcs. Assign a new, unique feature ID to one of the arcs, and write two update records to the transaction file with old_uniqID, new_uniqID, UPD_ACT = 'S' and any other appropriate information.

Add an arc. First split any existing arcs, if needed, to generate nodes at the new arc's endpoints. Set the new arc's unique ID. Increment the update number. Add the arc with its new ID, and write a record to the transaction database.

VII. TEMPORALLY REFERENCED DATA LAYERS

Some of the techniques described above use a simple date stamp to record when features were updated. However, a complete incorporation of time as an integrated aspect of GIS data is a current and growing field of research, with a large body of literature. Techniques have been described which enable a single GIS layer to store varying states of the layer over time, but these are beyond the scope of this guideline. Users needing this functionality should review the current literature.

Also, users should be aware that ArcStorm, slated to be part of Arc/Info release 7, will include tools for managing archives and updates over time. According to an ArcNews article (spring, 1993), ArcStorm will include "maintenance of feature history (cartography and attributes)," allowing a user to "retreive historical versions of the

database for a given point in time." The article states that ArcStorm will "interface to Arc/Info Rev. 7.0, ArcView Rev. 2, and ArcCAD rev. 2." The full extent of ArcView tools has not been investigated for the current version of this guideline.

VIII. BIBLIOGRAPHY

___, *ArcStorm - a strategic product for GIS data management*, ArcNews, Spring 1993.

Ferber, Don, *Tracking TIGER: the use, verification and updating of TIGER data*, in GIS/LIS '91 Proceedings, 28 October-1 November 1991, Atlanta, Georgia, Vol 1., pp 230-239.

Hazelton, N. W. J., F. J. Leahy and I. P. Williamson, *On the design of temporally-referenced 3-D geographic information systems: development of four-dimensional GIS*, in GIS/LIS '90 Proceedings, 7-10 November 1990, Anaheim California, Vol. 1, pp. 357-372.

Li, Manna and J. B. Hokansos, *Property map update and maintenance on Arc/Info system*, in Proceedings of the Twelfth Annual ESRI User Conference, Redlands, CA, 1992, Vol. II, pp. 359-371.

Luers, L. and W. Bamberger, *GIS and computer dispatch: a transactional geofile maintenance system*, in Proceedings, URISA 1993 Annual Conference Proceedings, July 25-July 29, Atlanta, Georgia, Vol. 2, pp. 63-74.

Rice, S. and J. Jackson, *Now that the data is converted, how do we maintain it?*, in Proceedings, URISA 1993 Annual Conference Proceedings, July 25-July 29, Atlanta, Georgia, Vol. 1, pp. 168-179.

Young, Larry, *Using Arc/Info for parcel maintenance*, ArcNews, Fall 1992, pp. 3-5.