Data Formats for GIS:
A Survey of Popular Systems

Daniel Ayers

October 1992

Contents

List of Tables

Chapter 1

Introduction

The field of *Geographic Information Systems (GIS)* is still quite young and there are many incompatible systems on the market. Due to the high cost of *data acquisition*, the process of collecting data, it is desirable for any system to be able to accept data ported from other systems or sources.

However, the conversion of data from one system to another is no simple task. Each system invariably uses its own set of graphic entities (points, lines, polygons, etc) as the building blocks for its model of the world. The requirement for rapid access to the data in the system's internal representation has resulted in each system having its own different (usually binary) file format. Sometimes the specifications for a system's internal file format are kept confidential.

While it is often possible to port a system's internal files from one platform to another (for example the VAX, UNIX and Prime versions of ARC/INFO store data internally in the same format) it is usually necessary to rely on one of the many *interchange formats* to move data between systems.

The general interfacing problem is very complex, and beyond the scope of this report.

The information we are concerned with may be divided into three broad categories (the term "object" is used to refer to any geographical feature being modelled):

- *spatial* information — information concerned with an object's location, coordinates.
- *topological information* — information concerned with the physical relationships between objects. Topological relationships might be "next to", "to the left of", "comes before" or "comes after".

- *attribute information* — other information that needs to be stored about an object. Examples of this could be "colour", "ownership" or "date of construction".

Any good CAD package should have no difficulty handling spatial information. Likewise, a database manager (such as Ingres, dBASE III or Oracle) is designed to handle attribute information. A GIS is a combination of the two, displaying spatial information as maps and holding attribute information in database tables. The ability to manipulate attribute information is what separates a true GIS from a CAD or desktop mapping system.

1.1 Aim

Richard Williams of Trimble Navigation (a co-supervisor of this project) has nominated four systems and interchange formats that should be researched. They are:

- Autocad DXF, Autodesk Inc.
- Microstation PC, Intergraph Corp.
- ARC/INFO, Environmental Systems Research Institute.
- MapInfo, Mapping Information Systems Corp.

The aim of this report is to present an investigation of the capabilities of the above systems. Each of chapters 2–5 deal with one of these system. In each chapter the terminology of the system in question has been adhered to as closely as possible. The original specification of this project, written by Richard Williams, is included in the appendices.

Also in the appendices are tables summarizing the equivalences between the different systems' spatial objects and terminology.

1.2 Spatial Data Transfer Standard

The (American) *Spatial Data Transfer Standard (SDTS)* is an attempt to standardize a format for data interchange between GIS systems. It is rich in features, and is intended to support a superset of the capabilities of the systems it interfaces between.

In this report, the SDTS is used as a reference point for a standard set of spatial objects. In each of the chapters dedicated to the individual systems, brief comments (labelled "**SDTS**") are made relating each spatial object to its nearest equivalent in the SDTS.

1.3 Acknowledgements

I would like to acknowledge the generous assistance of my supervisor, Professor J. P. Penny, for his guidance and proof-reading of the report. Thanks are also due to Richard Williams, Pip Forer, Derry Gordon, the Engineering CAD Centre and the Department of Forestry for supplying the documentation I needed.

Chapter 2

Autocad DXF

DXF is the data exchange format supported by the Autocad CAD package. It is also supported by a wide range of other CAD, GIS and analysis packages. There are different versions of the DXF specification, corresponding to the release of Autocad supporting it. In this report, I will describe DXF version 11.

2.1 Terminology

An Autocad *drawing* is composed of *layers* containing *drawing entities* (points, lines, etc).

A DXF file is an ASCII file describing a single Autocad drawing. The file is a series of groups, each occupying two lines in the DXF file. The first line is the *group code*, an integer identifying the type and/or meaning of the data to follow. The second line is the data in a format dependant on the group code preceeding it.

For example, a group code of 0 indicates the start of a drawing entity, table entry or file separator. The data value on the next line, a string, indicates which of these actually applies. A group code of 999 introduces the next line as a comment.

2.2 Drawing Entities

A list of Autocad's drawing entities follows. Other, non-graphic, entities may be found in DXF files (attribute values, for example) but these are not separately listed here.

- **Point.** *A zero-dimensional entity described by a location,* (x, y, z).

  Points may be drawn using a variety of *point markers.*

  SDTS: Similar to the (entity) point object.

- **Line.** *A straight line segment between two points* (x_1, y_1, z_1), *and* (x_2, y_2, z_2).

  Lines can be drawn using a range of line-types: solid, dashed, dotted, etc.

  SDTS: Similar to the line segment object.

- **Arc.** *A section (between angles* θ_1 *and* θ_2*) of a circle with centre* (x, y, z) *and radius* r.

  May be drawn using one of a number of line types.

  SDTS: A special case of the SDTS arc, which may be any general curve.

- **Circle.** *A circle centered at* (x, y, z) *with radius* r.

  Can be rendered in a range of line types.

  SDTS: Another special case of the SDTS arc.

- **Trace.** *Two-dimensional solid-filled lines of a specified width.*

  SDTS: No equivalent.

- **Solid.** *A two-dimensional quadrilateral or triangular object*

  "Solids" do not necessarily have to be filled in.

  SDTS: A special case of the SDTS G-polygon.

- **Shape.** *A small object defined outside Autocad. Inserted at location* (x, y, z) *with a specified orientation and scale factor..*

  Shapes are (in Autocad terminology) compound objects, like blocks (see below). They work in a way similar to blocks, but are less flexible and more efficient for Autocad to manipulate. They are suited to situations where speed is critical and a large number of instances of the same entity must be inserted into the drawing file. A variation of Autocad's shapes are used to define and draw text fonts.

  SDTS: Similar to complex objects.

- **Block.** *A compound entity defined as a grouping of primitive or other block entities.*

  Blocks are very important Autocad entities, from both a CAD and GIS point of view.

  Blocks allow the definition and repetitive insertion of similar objects in a drawing. They may be accumulated into libraries (separate from any drawing) from which they can be readily imported, customized (scaled, rotated, etc) and used.

  Blocks are defined in the `BLOCKS` section of the DXF file. A block definition begins with a block entity, is followed by a series of drawing entities comprising the block and is completed with an *endblk* marker. Instances of the block are then inserted into the `ENTITIES` section of the DXF file, using the *insert* entity.

  Blocks may have an associated set of attributes. The names and types of these attributes are defined using *attdef* entities immediately following the block's definition in the `BLOCKS` section of the DXF file. Instances of the block are followed by a corresponding list of *attribute* entities providing values for the block's attributes.

  Only blocks may have attributes attached in this fashion, although *extended entity data* may follow any entity — this is discussed later.

  SDTS: Similar to complex objects.

- **Polyline.** *A connected sequence of arcs and line segments.*

  Polylines provide an extremely flexible method for drawing both two and three dimensional shapes. They may have different line types and width, they can be tapered and they may form closed shapes.

  Polylines could be viewed as a means of representing very general lines and polygons in Autocad drawings.

  SDTS: A more general form of the string, polylines may contain both line segments and arcs.

- **Text.** *Textual information rendered using a range of fonts, sizes and other effects (known as* styles*).*

  Text can be used to display the values of attributes attached to instances of the block entity.

  SDTS: Related to the label point.

- **Dimensions.** *Compound entities used to attach measurements to a drawing.*

  SDTS: No equivalent.

Additionally, Autocad offers the following three-dimensional entities:

- **3D face.** *A three-dimensional entity constructed from triangular or rectangular plane sections.*
- **3D mesh.** *A three-dimensional mesh of triangular or rectangular polygons.*
- **Polyface mesh.** *A three-dimensional mesh of arbitrary topology.*

  The polyface mesh differs from the 3D mesh by removing the restriction that the polygons forming the mesh must be triangular or rectangular. The polygons forming a polyface mesh may have any number of sides, and may coincide in an arbitrarily complex manner — but not overlap.

2.3 Attribute Data

Autocad has no significant database features of its own, nor can it communicate directly with an external database manager. However, Autocad's DXF format can accommodate attribute data for exchange with other systems and Autocad can display attribute values as text in a drawing.

There are two methods of including attribute information in DXF files. The traditional method uses the block entity, and is described first.

2.3.1 Block attributes

Attribute values may be stored in a DXF file using "attrib" entities, which may only be attached to instances of block. When a block is defined in the blocks section of the file, it can be followed by a sequence of "attdef" entities. Each attdef entity describes the name, default value, text style and other characteristics of an attribute.

The list of attdef entities attached to a block definition form a *data dictionary* for all instances of that block.

When instances of that block are inserted (in the `ENTITIES` section of the DXF file) values may be specified for each of the attributes.

The problem with using this method to associate attributes with drawing entities is that all instances of the same block must have similar graphical

representations (except for position, orientation and scale). If you are handling objects with identical representations (power poles or distribution boxes, for example) then this is not a problem.

However, if each instance of the entity being modelled is different (pipes for example, each of which can be a different size and length) then it is clumsy, at best, to use this method.

Two possibilities for overcoming this problem are:

- Define a block for each dissimilar entity. If many blocks must be defined then this will greatly increase the size of the DXF file. Each block definition must include the list of attdef entities defining the attributes required.

- Define only one block for each set of attributes. This block would contain only a single point, which would be located on, within or near the drawing entity associated with the attributes. This solution is more efficient, but weakens the link between an entity and its attributes.

Attributes attached to blocks may be displayed in the drawing as text (which is the only data type available for attribute entities). There is also a simple data entry facility allowing the user to enter values for the attributes when a block is inserted.

Attribute data may be exported in the standard ASCII CDF (comma-delimited format) and SDF (space-delimited format). Additionally, attributes may be *extracted* into a DXF file containing only block references with attributes.

2.3.2 Extended entity data

Extended entity data (supported from DXF version 11 onwards) is intended to allow AutoLISP and ADS (two Autocad development environments — LISP-like and C-like, respectively) programs to attach attributes to drawing entities. Extended entity data may be attached to any entity type whatsoever, but is not used within Autocad itself.

Each application managing extended attribute data may use one or more *application names*, identifying the start of a particular sequence of attribute types understood by that application. Every application name appears in the APPID table in the `TABLES` section of the DXF file.

In this way, one entity may have attributes attached to it by several applications. The attributes are kept separate, being grouped by the application name.

Application names are necessary because there is no way of storing in the DXF file the data dictionary for the extended entity data. All applications must know the order and meanings of the attributes they manage under their application name(s).

Attribute data is stored in the DXF file immediately following the normal entity definition data (position, etc) using a special set of group codes starting at 1000. Programs not recognising these group codes will simply ignore them.

An application wishing to read the values of certain attributes from a DXF file would first search the file for entities with extended entity data. The application would then look for an occurrence of the application name(s) it recognises, and read the data — the meaning of which it will already know.

Obviously, care must be taken to avoid two independent applications making use of the same application name.

Autocad DXF supports the following data types for extended entity data:

- **String.** A character string containing at most 255 characters.
- **Application name.** A character string of up to 31 characters identifying the application to which the following data is meaningful.
- **Lists.** Attribute values may be grouped into lists. The characters "{" and "}" are used as delimiters.
- **Layer name.** Name of the layer associated with the attribute data.
- **Binary data.** "Chunks" of binary data, up to 127 bytes long. Multiple chunks may be inserted if required.
- **Entity handles.** Unique hexadecimal identifiers of entities in the Autocad drawing. Useful for establishing relationships between drawing entities.
- **Point/Vector.** Triplet of real values which may be interpreted either as a point or vector. Value is never altered by Autocad.
- **World space position.** A triplet of real values specifying a position in world space. May be moved, scaled, rotated, stretched and mirrored by Autocad as required.
- **World space displacement.** A 3D point subject to the same transformations (except stretching) as the world space position.
- **World space direction.** Identical to the world space displacement. The distinction between the two is application-defined.

- **Real.** A single real value. No transformations applied by Autocad.
- **Distance.** A real value that is scaled together with the parent entity.
- **Scale factor.** Same as distance. The distinction between them is defined by the application.
- **Integer.** 16-bit signed or unsigned integer.
- **Long.** 32-bit signed integer.

Autocad can assign to every entity in the drawing a unique *handle* (represented in the DXF file as a hexadecimal number). An entity's handle may be used to refer to it from attribute data elsewhere in the drawing, or from an external database.

2.4 Coordinate systems

Autocad DXF supports only one type of coordinate system – three-dimensional rectangular, (x, y, z). Naturally the z dimension may be ignored if the drawing is two-dimensional.

The *world coordinate system (WCS)* is the default within Autocad, and is used to locate most points in a DXF file. The exceptions are points within planar entities (circles, arcs, text, etc) where an *entity coordinate system (ECS)* with appropriate origin and orientation is constructed so that the entire entity lies within its xy plane. This leads to a more compact and efficient representation since all points within the entity may be expressed as (x, y) pairs within the ECS – rather than (x, y, z) triples in the WCS.

The user may define a number of *user coordinate systems (UCS)* with any desired origin and orientation. These are used for data-entry only, no entities in the DXF file are located with respect to them. However, their definitions *are* recorded in the `TABLES` section of the DXF file.

The Autocad documentation mentions no restriction on the extent of any of these coordinate systems.

Autocad recognizes a range of units, which are defined in the file "acad.unt". The user may add definitions to this file to suit.

Unit definitions do not affect the DXF file. Coordinates are always expressed as decimal numbers (possibly in scientific notation), angles are always in decimal degrees with zero degrees being east of the origin.

Extension	Type of file
.adt	Audit report file
.bak	Drawing file backup
*.bk*n	Emergency backup file
.dwg	Drawing file
.dxb	Binary drawing interchange file
.dxf	ASCII drawing interchange file
.dxx	Attribute extract file (DXF format)
.flm	Filmroll file, for AutoShade
.igs	IGES interchange file
.old	Old version of converted drawing file
.shp	Shape/Font definition file
.txt	Attribute extract file (CDF, SDF)
.unt	Units file

Table 2.1: Extensions used by Autocad data files. Source: Autocad reference manual

2.5 File Format

Autocad drawings are stored in a single binary file with an extension (file-type) of ".dwg". A number of other filetypes are used by Autocad, these are summarized in table 2.1.

Autocad has several data exchange formats. The most widely known, and supported, and the primary focus of this chapter is *DXF - drawing interchange format*. DXF has become a de-facto industry standard for the sharing of drawing data.

Autocad also supports IGES (Initial Graphics Exchange Specification). Autocad generages IGES files compliant with version 4.0 of the specification, and can read files from versions 2.0 through 4.0.

DXF files are standard ASCII text, and may be edited with any text editor. A binary version of the interchange format, *DXB - binary drawing interchange file*, has been developed by Autodesk. The DXB format is a limited subset of DXF, supporting fewer entities. As a result of its binary storage format, DXB files are normally much smaller than corresponding DXF files and can be processed much more quickly.

Autocad is capable of reading DXB files, but has no direct method of writing them. They can only be generated as output from one of the plotter

drivers.

A DXF file consists of four sections, the first three of which are optional:

1. `HEADER` section. Contains values for any system variables which need to be set, or differ from their default values.

2. `TABLES` section. Contains a number of tables defining the line types, layers, text styles, views, user coordinate systems and application ID's used in the drawing.

3. `BLOCKS` section. Contains the definitions of all blocks used in the drawing.

4. `ENTITIES` section. Contains the actual drawing entities, including block insertions that make up the drawing. Attributes appear here following block insertions, and extended entity data following any type of entity.

2.6 Summary

Autocad DXF is the de-facto industry standard interchange format for graphical information. Its definition is stable, but easily extensible to accomodate new features.

Autocad's entity set is smaller than some, but very general — particularly the polyline. Translators from other formats to DXF translate a wide range of their own entities to polylines.

Until recently, block attributes were the only way to include attribute data in a DXF file. Since this type of attribute could only be attached to blocks, the association of attributes with a drawing entity was either clumsy (if the point-block method was used) or inefficient (if many different blocks were created).

This problem has been solved with the innovation of extended entity data, which may be attached to any entity whatsoever. However, there is currently no support for extended entity data in Autocad proper and it is ignored by current versions of translators to other formats. However, new software can safely generate DXF files containing extended entity data and be sure that programs who do not understand it will simply ignore it. This is one of the strengths of the "group code" structure of DXF.

Once support for extended entity data becomes more widespread, DXF will be a much more useful interchange format for GIS applications.

Chapter 3

Microstation

Microstation is a product of Intergraph Corporation. It is not a GIS, but a flexible CAD package with good support for data interchange and external databases. Implementations are available for a range of platforms, including the IBM PC and UNIX. This chapter describes the capabilities of Microstation PC version 4.0.

3.1 Terminology

Microstation operates on *design files*. Each design consists of a number of *levels*, which are analogous to the GIS conecpt of layers or coverages. In turn, each level contains the *elements* making up the design.

The extent of a Microstation design is called the *design plane* (two-dimensional designs) or the *design cube* (three-dimensional designs).

3.2 Design Elements

Microstation supports the following primitive (ie. indivisible) elements:

- **Line.** *A single line segment.*

  A line segment is specified by the coordinates of its endpoints. A point is considered to be a special case of a line, being a line of zero length. Any cell (see below) may be attached to either end of any linear element to form a terminator (eg. an arrowhead).

  SDTS: Equivalent to the SDTS line segment.

- **Line String.** *A connected series of line segments.*

  A line string is specified by the number and location of its data points. A simple line string may have up to 101 vertices (100 line segments). A complex string, also known as a complex chain, can be formed by concatenating lines, curves and arcs to overcome this restriction.

  SDTS: Equivalent to the SDTS string.

- **Point String.** *An ordered sequence of points with defined orientations.*

  A point string element consists of a number of vertices, each with a defined orientation. Point strings may be contiguous (linked with line segments) or disjoint (not linked together). There cannot be more than 48 vertices in a simple point string, but point strings may be combined into a complex chain if a longer sequence is required.

  SDTS: No direct equivalent. The contiguous point string is related to the SDTS string, as it is to the line string above.

- **Curve.** *A curve drawn through a sequence of points.*

  A curve is determined by the points it passes through. A simple curve may be drawn between a maximum of 97 points. Curves differ from line strings by being smoothed near the vertices. An extra (invisible) point is used at either end to define the curvature at the ends of the curve.

  SDTS: Related to the SDTS asrc, although defined in a different fashion.

- **Arc.** *A section of an ellipse or circle.*

  An arc is defined by it's semi-major and semi-minor axes, angle of orientation and start/stop angles.

  SDTS: A special case of the SDTS arc, which may be any general curve rather than a section of a circle or ellipse.

- **Ellipse.** *A closed circular or elliptical shape.*

  An ellipse is defined by the centre coordinate, major/minor axes and orientation angle. A circles is a special cases of an ellipse.

  SDTS: A special cose of the SDTS arc.

- **Shape.** *A closed element bounded by line segments.*

  A shapes is described by the line segments forming its perimeter. There may be a maximum of 100 line segments (and 100 vertices, since the shape is closed) forming the perimeter of a shape.

SDTS: Similar to the SDTS G-ring, with the restriction that the perimeter must consist of straight line segments.

- **Text.** *A single line of textual information.*

  While the text element may contain only one line of text, a *text node* can contain multiple lines of information. Text elements may be inserted into the drawing with a specified size and typeface (font). They may contain *enter-data fields* which are placeholders for future user input.

  SDTS: Related to the SDTS label point.

Closed two dimensional elements have an associated flag recording whether the interior area is included.

Additionally, *complex elements* may be constructed as a combination of the above primitive elements (B-spline curves are the exception, they are considered to be complex elements but are not constructed from primitive ones).

The complex element types supported by Microstation are:

- **B-spline curve.** *A mathematically-defined curve whose shape follows a sequence of* poles.

  A B-spline curve consists of a header element and up to 101 unique *pole elements* (each pole element being a vertex having localized effect on the shape of the curve).

  SDTS: Although they are defined differently, an SDTS arc may be constructed with the same shape as a B-spline curve.

- **Cell.** *A reusable grouping of elements. Cell definitions are grouped into cell libraries (separate from any design file) and may be imported into a design as required.*

  Cells may be *shared*, where there is only one definition of the structure of the cell in the design file and each cell instance is represented by a reference to that definition. If a change is made to any shared cell, the definition will change and all other cell instances sharing that definition will conform to the change.

  If cells are *unshared* then a new copy of the cell definition will be inserted into the design file for each cell instance. Any changes made will only be reflected in those cells directly edited by the user.

  Cell types can be further subdivided into *graphic cells* (which have their own colour and line style characteristics stored as part of the cell

definition, and will therefore be consistent across shared instances), *point cells* (which use the currently-active colour and line settings, and may differ across shared instances), *menu cells* and *tutorial cells.*

A *group* is a cell that was not imported from a cell library, but defined within the current drawing.

SDTS: Related to the SDTS complex object.

- **Complex chain (complex string).** *A series of connected primitive elements forming an open figure.*

  Complex chains may be formed by joining lines, line strings, arcs and curves. They are used to form complicated or overlong single-dimensional shapes.

  A common use of complex chains is to circumvent the limitation on the number of vertices in line strings and curves. A line string or curve longer than the allowable limit may be constructed by concatenating a number of them together as a complex chain.

  SDTS: Related to the SDTS chain, which is also a sequence of line segments and arcs/curves. Unlike the SDTS chain the Microstation equivalent has no topological information — it has no start/end nodes nor direction.

- **Complex shape.** *A closed figure bounded by primitive single dimensional elements.*

  Complex shapes are identical to complex chains, except that the first and last elements in the chain are joined to form a closed figure.

  SDTS: Very similar to the SDTS G-ring, also a closed figure bounded by line segments and arcs/curves.

- **Text node.** *A multi-line text element comprised of single-line primitive text elements.*

  SDTS: No direct equivalent, the most closely-related SDTS object would be the label point.

- **Raster.** *One or more scan lines of bit-image data.*

  Raster images are prefixed with a raster header element, followed by a series of raster data elements each containing up to one scan line of the image.

  SDTS: A complete Microstation raster would be equivalent to an SDTS image.

Some of Microstation's elements are specific to three-dimensional design files. The only primitive element in this category is the cone, which may be either *right* or *skewed.* Cylinders are also supported, but as a special case of the cone.

Complex three-dimensional elements include spheres, slabs (a volume with a rectangular cross-section), solids of projection (ie. extruded solids) and solids of revolution.

3.3 Coordinate systems

Microstation allows the use of a number of auxilary coordinate systems (ACS) which may have a different origin and/or orientation than the standard (world) coordinate system (WCS).

In two-dimensional designs, the coordinate systems available are:

- **Rectangular.** The standard cartesian coordinate system, (x, y).
- **Polar.** Points specified using a direction and radius vector, (r, θ).

In three-dimensional design files, there are three available coordinate systems:

- **Rectangular.** Points specified as multiples of the three mutually orthogonal unit vectors, (x, y, z).
- **Cylindrical.** Points specified using a direction and radius vector in a plane, and a height above the plane (r, θ, z).
- **Spherical.** Points specified using a radius vector and two angles, (r, θ, ϕ).

3.4 Three-dimensional designs

Microstation has two distinct design file formats, for two and three dimensional designs. It is possible to convert interactively between 2D and 3D files within the program.

Some elements and coordinate systems (see above) are only available (and meaningful) in three-dimensional design files.

A maximum of 4,294,967,296 (2^{32}) distinct *units of resolution* are supported in each direction in both two and three dimensional design files.

A number of different units may be used in the same design. The largest unit is known as the *master unit.*

3.5 Attribute data

Microstation (PC) is able to establish relationships (called *linkages*) between design elements and rows in database tables managed by external databases such as Oracle or dBASE III/IV. The external database manager functions as a database *server*, and Microstation operates as a *client* requesting services from the database software.

The Oracle interface is the more powerful of the two. In fact, the Microstation documentation does not describe the operation of the dBASE interface much beyond listing the features it does not support. Implementations of Microstation on other platforms are able to interface to different database systems (the UNIX version is able to communicate with both Oracle and Informix).

Microstation can add and remove linkages between design elements and database rows, produce reports based upon the values of database attributes, display attributes from the database as text nodes in the design and perform graphical operations based upon database search criteria.

Microstation is capable of deleting rows in the external database when corresponding elements in the design are deleted.

As a result of its abilities in this area, Microstation may be considered to be a low-end GIS package rather than a CAD package alone. It certainly distinguishes itself from Autocad in this area.

Database attributes may be displayed as text nodes in a design. Screen forms are used to control the format of the displayed data. Any or all attribute columns may be displayed in the design.

Each database with linkages must contain a table named `MSCATALOG` (also called the *control table*) containing one row for each database table with linkages. One of the columns of `MSCATALOG`, `DATABASE`, must contain the name of a *displayable attribute table* listing mappings of displayable attribute types to screen forms.

Any database table being linked to a design must include a column called `MSLINK`. The contents of this field are written by Microstation, and must be able to accomodate values as high as 9,999,999,999. `MSLINK` serves as the primary key for the table. Linkages between design elements and database rows may be of any cardinality.

3.6 File Format

Microstation design files are stored in a binary format, the details of which are published in the appendices of the MDL Reference Manual.

Design files can be used by the same version of Microstation running on different platforms, and may also be usable by different versions of Microstation and VAX IGDS.

3.7 Data Exchange

Microstation is able to input data from Autocad version 10 DXF files. DXF is the interchange format supported by Autocad, and is the subject of another chapter of this report. At the time of writing, a new Microstation translator supporting Autocad version 11 (the current release) is expected in the near future.

Drawing files may be output in the following formats:

- Encapsulated PostScript
- RIB *(RenderMan Image Binary)* files, for dedicated image rendering software
- Autocad DXF version 10

When exchanging data to/from DXF, the following should be considered:

- Microstation supports 63 levels (equivalent to DXF layers) and eight line styles, both of which are assigned numbers. Cell names are 6 characters (A-Z, 0-9, $ and .) long. However, DXF uses character names up to 31 characters long to identify drawing layers, line types and blocks.
- The Microstation design plane or cube (depending on whether the design is two or three dimensional) is limited to 4,294,967,296 (2^{32}) unique units of resolution in all directions.
- Differences in the elements/entities supported by Microstation and DXF will inevitably result in inexact translations, particularly when translating design files to DXF. DXF supports fewer but more general elements, many Microstation elements are simply special cases of a DXF entity (eg. Microstation ellipses, B-splines and curves are special cases of the DXF polyline).

  Tables 3.1 and 3.2 detail the mappings between Microstation elements and DXF entities performed by the translator.
- DXF supports different line thicknesses for polylines only.

- DXF does not support reference files (data included in the design as a backdrop or template). Reference files must either be included as part of the design or excluded entirely when translating to DXF.

A set of ASCII files called *translation tables* are maintained to define the desired mappings between Microstation names and codes and the equivalent DXF names/codes used in both directions during the translation process. For example, the translator might be required to convert the Microstation cell `CELL1` to the DXF block `BLOCK1`. To accomplish this, one line in the block/cell translation table would read:

```
BLOCK1 CELL1
```

The translation tables contain two entries per line, separated by whitespace. The first item is the name of the item (eg. line style, cell, etc) in the DXF file and the second is the name of the corresponding item in Microstation. The names of the translation files are shown in table 3.3.

The font translation table "font.tbl" has a third column, the scaling factor that should be applied to font widths during translation from DXF to design format.

3.8 Summary

While Microstation is basically still a CAD package, it is a very capable one. The ability to interact with external databases while the package is running lends it some of the flavour of a true GIS.

The built-in DXF to design translator makes importing data from other formats quite straightforward. One possible problem would be the handling of attributes (attribute entities attached to block instances, extended entity data is not supported by the current translator — refer to the DXF chapter for clarification on this point) embedded in DXF files. The translator would convert these into text attached to the appropriate design element. It would be more desirable for these attributes to be inserted into an external database during the translation process.

The DXF translator is implemented as an MDL *(Microstation Development Language)* application. There is no reason why an improved translator couldn't be written to provide better support for attributes by recognising extended entity data in DXF files (which may be attached to any entity) and inserting it into the external database during the process of translating from DXF.

Attributes could be extracted from the external database and included as extended entity data when a design file is exported to DXF format.

DXF entity	Microstation element
line	line
point	line (zero-length)
arc	arc
circle	ellipse
block	cell
text	text
attributes	text
trace	shape
solid	shape
3D face	shape
3D line	line
extruded entity	surface
polyline with arcs	complex element
polyline without arcs	line string

Table 3.1: Conversion of DXF entities to Microstation elements. Source: Microstation user guide

Microstation element	DXF entity
line	line
circular arc	arc
circular ellipse	circle
cell	block
text	text
complex chain	polyline
complex shape	polyline
multi-line	multiple lines
non-circular arc	polyline
non-circular ellipse	polyline
shape	polyline
line string	polyline
curve	polyline
surface	polyline mesh
solid	polyline mesh
cone	polyline mesh
cylinder	polyline mesh
point string	*not supported*
text node	*not supported*
raster element	*not supported*

Table 3.2: Conversion of Microstation elements to DXF entities. Source: Microstation user guide

Filename	Translation table for:
CELL.TBL	Blocks and Cells
COLOR.TBL	Colours
FONT.TBL	Text styles
LEVEL.TBL	Layers and Levels
LINECODE.TBL	Line types and styles
WEIGHT.TBL	Line widths

Table 3.3: Microstation translation table filenames

Chapter 4

ARC/INFO

ARC/INFO is a GIS system developed by the Environmental Systems Research Institute (ESRI). The system is modular in design, but includes two major components: ARC, the geographic input and analysis tool and INFO the database manager.

This chapter details the capabilities of ARC/INFO version 6.0.

4.1 Terminology

An ARC/INFO *workspace* is a set of *coverages* (analogous to layers). Each coverage contains a set of spatial *features* modelling real-world occurrences..

4.2 Feature Classes

ARC/INFO supports the following entities, which it terms *feature classes:*.

- **Arc**: *A continuous string of points,* (x, y) *pairs.*

  The arc is the primary linear feature supported by ARC/INFO. An arc is a string of points connected by line segments An arc between two points is a straight line, with more points it could be any arbitrary one-dimensional shape.

  One arc may contain at most 500 points. Arcs longer than this can be simulated by concatenating a number of smaller arcs together.

  The ARC file contains the spatial locations of the arcs in a coverage. The *arc attribute table (AAT)* contains the topological information associated with the arcs in a coverage:

- **From node.** Internal identifier of the node where the arc starts.
- **To node.** Internal identifier of the node where the arc ends.
- **Left polygon.** Internal identifier of the polygon to the arcs left.
- **Right polygon.** Internal identifier of the polygon to the arcs right.
- **Length.** Length of the arc in coverage units.
- **Sequence number.** Internal sequence number of this arc in the ARC file.
- **User ID.** User-assigned identifier (feature-ID) of this arc.

The user may append attributes to the above schema for the AAT and use them for non-spatial information.

SDTS: ARC/INFO arcs and nodes together are equivalent to the SDTS complete chain, which has both node and polygon topology.

- **Node:** *The endpoints of an arc, the intersections of arcs in a network.*

  Each arc has a *from node*, the node where it starts, and a *to node*, the node where it ends. This gives arcs the concept of direction. A maximum of 100 arcs may connect at the same node.

  There are two special types of nodes:

  - Pseudo nodes, where only two arcs intersect (or an arc connects with itself forming an island).
  - Dangling nodes, which are part of only one arc.

  The following information may be associated with each node in the coverage, and contained in the *node attribute table (NAT)* file:

  - **Arc number.** Internal identifier of one of the arcs connecting at this node.
  - **Sequence number.** Internal identifier of this node.
  - **User ID.** User-assigned identifier (feature-ID) of this node.

  The user may add attributes to the schema of the NAT file for non-spatial data.

  SDTS: Identical to the SDTS node.

- **Label Points**: *Points, (x, y) locations, having no area or length.*

  Points are used to locate (and name) features that have no extent on the scale of the coverage (eg. telephone poles, wells, etc). They are also used to assign User-IDs to polygons by attaching the User-ID to the point and positioning the point inside the polygon.

  Within a particular coverage, label points can be used for *either* naming features *or* assigning User-IDs – but not both.

  The *point attribute table (PAT)* contains the following information about each label point in the coverage:

  - **Polygon area.** The area of the polygon containing the point.
  - **Polygon perimeter.** The perimeter of the polygon containing the point.
  - **Sequence number.** Internal identifier of this label point.
  - **User ID.** User-assigned identifier (feature-ID) of this label point.

  Attributes may be added to the PAT file by the user.

  A single coverage cannot have both point and polygon features.

  SDTS: When used to identify a point feature, the ARC/INFO label point is equivalent to the SDTS entity point. Otherwise, when used to attach identifiers and attributes to polygons, it is equivalent to the SDTS area point.

- **Polygons**: *A two-dimensional feature defined by the arcs forming it's boundary. A Label Point is positioned inside it to assign a unique User-ID to the polygon.*

  Polygons may have holes ("islands") in them. It is possible for a polygon to have more than one Label Point inside it, but this practice is not recommended.

  Two separate data files are maintained to store arc/polygon information. The first is the *Polygon-Arc Topology* (in the PAL file) which relates a polygon's (internal) sequence number and User-ID with the arcs and nodes used to define it.

  The second file (the AAT file, discussed earlier) contains the *Left-Right Topology*, storing the "from" and "to" nodes and the left and right polygons for each arc in the coverage.

  A PAT file for each polygon coverage contains the following information about each polygon:

- **Area.** Area of the polygon.
- **Perimeter.** Perimeter of the polygon.
- **Sequence number.** Internal identifier of this polygon.
- **User ID.** User-assigned identifier (feature-ID) of this polygon.

The user may add attributes to the PAT schema.

A single coverage cannot have both point and polygon features. The boundary of a polygon, including islands within it, may not contain more than 10,000 arcs.

SDTS: Related to the SDTS GT-ring.

- **Tics**: *Points of known geographical location, used for calibration purposes during digitising and editing.*

  Tics are stored in a TIC file, containing the user-id and (x, y) coordinate of the tic.

  SDTS: No equivalent.

- **Annotations**: *Textual labels, used for display purposes only.*

  Labels can be placed horizontally, at any angle or along an arbitrary spline curve (of up to 500 points).

  It is possible to control the offset and justification of annotations.

  SDTS: Related to the SDTS label point.

- **Links**: *Vectors used for rubber-sheeting and coverage adjustments.*

- **Route Systems**: *Ordered collection of arcs or parts thereof used to represent linear features.*

  SDTS: No equivalent.

4.3 Attribute Data

Non-Spatial attribute data may be stored in either *Feature Attribute Tables*, which are always INFO data files, or *Related Tables* which may be managed either by INFO or an external DBMS such as INGRES, SYBASE, ORACLE, INFORMIX, etc.

Feature-Attribute Tables are managed by INFO, and their structure is controlled by INFO. There is one record in the F.A.T. for each feature in the

coverage. The feature attribute tables contain mandatory attributes used by ARC/INFO, these were described in the previous section.

Related Tables are much more flexible. There can be any number (0 ... many) of records for each feature in the ARC/INFO coverage. Using this, and the foreign-key approach, it is easy to create 1-many relationships (eg. a list of heating types used in a particular house identified as a feature in the coverage).

Records stored in any INFO table may not exceed 4kb in length.

The following "item" (data) types are available for use in INFO data files:

- Binary integers, 2 or 4 bytes long, signed.
- Character strings up to a length of 320 characters.
- Dates, displayed either in DD/MM/YY or DD/MM/YYYY format.
- Floating-point numbers. Single precision, 4 bytes giving 7 significant figures; Double precision, 8 bytes giving 14 significant figures.
- Integer, one byte per digit. 1-16 digits.
- Decimal numbers, 1-16 characters wide.

These data types apply to both user and system information in the Feature Attribute Tables and Related Tables.

All features have an internal sequence number, used to uniquely identify that feature within ARC/INFO. They also can have a corresponding user-id, assigned by the user. This may be used to identify the feature in attribute data — both in INFO and external database tables.

There are 2^{32} unique user-id's available.

4.4 Projections and Coordinate Systems

ARC/INFO supports 41 distinct map projections, and 26 spheroids (models of the Earth's irregular shape).

ARC/INFO coverages are strictly two-dimensional, all points are stored as (x, y) locations. A third dimension, if required, may be included as a user-defined attribute — but there is limited support within ARC/INFO for recognising this.

Additional packages for modelling in three-dimensions, such as TIN, are available for ARC/INFO.

4.5 Data Dictionaries

Data Dictionaries (the ARC/INFO documentation calls them *Item Definitions*) record the following information about each column in the INFO tables:

- Name
- Data Type (see above)
- Size (Number of bytes of storage required)
- Display Width
- Decimal Places (floating-point numbers only)

INFO tables can contain any number of columns, but each record must be less than 4kb in length. The first few attributes in a Feature Attribute Table are fixed - although new attributes can be added, the name/type/size of these initial attributes must not be changed.

4.6 File Format

In ARC/INFO terminology, a "workspace" is a set of coverages and a set of INFO tables associated with them. A workspace is stored in a directory subtree with one subdirectory per coverage and one more for all the INFO files.

Each coverage directory contains a set of files describing the spatial features present in the coverage. The INFO directory contains INFO database files holding all the non-spatial information for all entities in the workspace.

The ARC/INFO manuals state "as a general rule, only the records in feature attribute tables are directly accessible to the user. Other files are stored in binary format and are automatically maintained by ARC/INFO."

ARC/INFO imposes no limit on the number of arcs, points, polygons, annotations, tics, etc in a coverage.

ARC/INFO is capable of importing data from a variety of formats, including Autocad DXF and IGDS *(Interactive Graphics Design Software)* files. The latter is the same binary format as that used by Microstation.

4.7 Summary

ARC/INFO is a very powerful GIS. Naturally it has extensive support for spatial, topological and attribute information. While the INFO database manager

is not widely recognised as "excellent", it is adequete for the requirements of a GIS. None of the other systems investiaged have quite the same analytical capabilities of ARC/INFO.

ARC/INFO can read a number of interchange formats, simplifying the task of getting data into the system.

Chapter 5

Map-Info

MapInfo is a desktop mapping package available for the Macintosh, Sun, HP and IBM PC (both DOS and Windows). It is designed to handle two-dimensional maps and can exchange data with a wide variety of formats.

In this chapter I will describe the capabilities of MapInfo version 1.1.3 for Windows, with particular regard to its MIF/MID format exchange files.

5.1 Terminology

MapInfo *maps* may consist of multiple *layers*. Each layer contains *graphic objects* representing the features on a map.

5.2 Objects

MapInfo's entity set may be broadly partitioned into the following three groups:

1. Points – zero-dimensional locations
2. Lines – one-dimensional lines and curves
3. Boundaries – two-dimensional closed polygons

The actual entity set is as follows:

- **Point.** *A dimensionless location specified by a coordinate pair* (x, y).
  Points may be displayed using one of a range of symbols.
  SDTS: Equivalent to an entity point.

- **Line.** *A straight line between two specified endpoints,* (x_1, y_1) *and* (x_2, y_2).

  Lines may be drawn using a particular *pen type* (line style).

  SDTS: Equivalent to a line segment.

- **Polyline.** *A line described by a sequence of points* (x_i, y_i).

  Polylines may be *smoothed* or *unsmoothed.* An unsmoothed polyline is simply a number of connected line segments, possibly with sharp corners at the vertices. A smoothed polyline has these corners removed by "bending" the lines near the corners to make them smooth and continuous.

  SDTS: An unsmoothed polyline is equivalent to an SDTS string. A smoothed polyline is equivalent to an SDTS arc.

- **Region.** *An area bounded by one or more polygons.*

  Regions may be used to represent related areas fragmented into a number of polygons (a country consisting of a number of islands, for example).

  The boundaries of the polygon(s) can be drawn using a particular pen type, and filled with a chosen pattern (the *brush type*).

  SDTS: Similar to a G-polygon, which can have islands within itself (holes) but not outside itself.

- **Arc.** *Part of the perimeter of an ellipse or circle, located within the rectangle whose opposite corners are* (x_1, y_1) *and* (x_2, y_2)*, extending from angle* θ_1 *to* θ_2.

  An arc can be drawn in a specified pen type.

  SDTS: Special case of the SDTS arc, which is a general curve.

- **Text.** *Textual information enclosed within the rectangle between points* (x_1, y_1) *and* (x_2, y_2).

  Text may be drawn in any one of a range of fonts.

  SDTS: Related to the label point.

- **Rectangle.** *A rectangular region described by diagonally opposite points* (x_1, y_1) *and* (x_2, y_2).

  The perimeter of the rectangle may be drawn using any pen type. The rectangle may be filled with a pattern drawn using any desired brush type.

SDTS: Special case of the SDTS G-polygon.

- **Rounded rectangle.** *A rectangular region with rounded corners, described by diagonally opposite points* (x_1, y_1) *and* (x_2, y_2).

  The amount by which the corners are rounded is controlled by the *degree of rounding*, expressed in coordinate units.

  The rounded rectangle may be drawn using the same range of pen and brush types described for the ordinary rectangle.

  SDTS: No equivalent.

- **Ellipse.** *An ellipse inscribed within the rectangle specified by diagonally opposite corners* (x_1, y_1) *and* (x_2, y_2).

  An ellipse may be drawn using a specified pen type and filled with any one of the available patterns (brush types).

  SDTS: Special case of the SDTS arc.

5.3 Coordinate systems

MapInfo operates entirely with two-dimensional coordinate systems.

The most useful coordinate system is what they call the *spherical* coordinate system. Points are specified using latitude and longitude (with no "radius" like the conventional spherical coordinate system).

Alternatively, a *rectangular* coordinate system may be used, where points are located using (x, y) coordinate pairs.

MapInfo maps may be overlaid, but only if the same coordinate system is used in each.

When importing or exporting data from/to an interchange format, a transformation may be specified between the two coordinate systems. The transformation is determined by specifying the coordinates of two points in each of the two coordinate systems.

By default, coordinate systems in MapInfo have a "northwest" orientation (the y coordinate increases to the north, and the x coordinate increases to the west). This is appropriate for the latitudes/longitudes in North America since it is west of the Greenwich meridian. Most likely, this will need to be altered if a rectangular coordinate system is being used.

The units used in the spherical coordinate system are degrees. In the rectangular system any sort of units may be chosen: miles, kilometres, feet, etc.

At the time of creating or importing a map, the resolution of the coordinate system (in terms of decimal places of accuracy) must be specified. The default value is 6 decimal places.

5.4 Attribute data

Attribute data (stored in MID files) may be associated with graphical objects (stored in MIF files) by their ordering in the respective files. The first attribute record (line) in the MID file is associated with the first graphic object in the MIF file, and so on.

The format of the attributes (MID) file is determined by the header in the data (MIF) file. It contains the data dictionary (order, type and names of attributes) and the definition of the delimiter character.

The data types available for attributes are:

- **String.** Character string of specified maximum length (at most 255 characters).
- **Integer.** 32-bit signed integer.
- **Smallint** 16-bit signed integer.
- **Decimal.** Decimal number with specified field width and decimal places.
- **Date.** Expressed in the US format, `MM/DD/YY`.
- **Logical.** A boolean true/false value.

Attribute names may be up to 12 characters long, and contain the characters A-Z, a-z and underscore.

5.5 Interchange Files

This chapter has been primarily concerned with the MIF/MID interchange format supported by MapInfo. Additionally, the following formats may be imported by MapInfo:

- **Autocad DXF.** Graphics (and attributes in MapInfo versions 1.1.3 and later) may be imported from DXF files. Coordinate system conversions need to be specified by the user.

  The translation of DXF entities to MapInfo objects is summarized in table 5.1.

DXF entity	MapInfo object
Block insertion	Exploded into components
Line	Line
Point	Point
Circle	Ellipse
Arc	Arc
Trace	Line
Text	Text
Shape	*ignored*
Attributes	*Tabular data*
Polyline	Polyline, limited
3D Face	*ignored*
Viewport	*ignored*
Dimension	*ignored*
Solid	Region

Table 5.1: Translation of DXF entities to MapInfo objects. Source: MapInfo users' guide

- **DOS MapInfo.** External utilities `MAP2MIF` and `BDY2MIF` are supplied to perform the conversion from DOS to Windows MapInfo files.
- **MBI/MMI.** These MapInfo-DOS interchange files may be imported directly by the Windows version.

The following formats (in addition to MIF/MID) may be exported to:

- **Autocad DXF.** Graphics and tabular data may be exported to Autocad DXF files.

  The translation of MapInfo objects to DXF entities is summarized in table 5.2.
- **dBASE DBF.** Tabular data may be exported to a dBASE .DBF file.
- **ASCII.** Tabular data may also be exported to delimited ASCII files.
- **WKS and XLS.** These two spreadsheet formats, used by Lotus and Excel respectively, may be directly written by MapInfo.

MapInfo object	DXF entity
Region (single polygon)	Closed polyline
Region (multi-polygon)	Block of closed polylines
Circular arcs	Arc
Non-Circular arcs	Scaled block containing the arc
Rectangle	Closed polyline
Rounded rectangle	Closed polyline, no rounding
Circular Ellipse	Circle
Non-Circular Ellpise	Scaled block with ellipse
Text	Text
Line	Line
Point	Point
Polyline	Polyline

Table 5.2: Translation of MapInfo objects to DXF entities. Source: MapInfo users' guide

Appendix A

Bibliography

1. GIS World, Inc. *The GIS Sourcebook.* ©1989.
2. Environmental Systems Research Institute. *ARC/INFO version 6.0* documentation. ©1991.
3. Autodesk, Inc. *Autocad R11 Reference Manual.* ©, 1990.
4. Bentley Systems, Inc. and INtergraph Corporation. *Microstation PC Users' Guide, Reference Guide and MDL Manual.* ©1991.
5. Mapping Information Systems Corporation. *MapInfo User's Guide (version 1.1) and Release Notes (version 1.1.3).* ©1990.
6. US Geological Survey, National Mapping Division. *Spatial Data Transfer Standard, version 12/90.* ©1990

Appendix B

Project Specification

Computer Science Honours Project
Richard Williams
Trimble Navigation
30 January 1992

- **Title**

  Data exchange formats for GIS.

- **Introduction**

  Each GIS in the market at the moment tends to use its own internal format for data storage, but data can be exchanged between systems through a number of GIS exchange formats. Each GIS will typically support the import and export of a range of these exchange formats.

  Understanding these exchange formats, and being able to process them in the correct manner, are important issues for anyone involved in software development for the GIS industry.

- **Description**

  The aim of this project is to produce a report detailing the most common GIS exchange formats and their essential elements and differences.

  This should include discussions on

  - how different entity types (eg. points, lines, regions, etc) are handled
  - how coordinate and attribute data are handled
  - how non-spatial data is handled

- 2D versus 3D data
- updating existing databases versus creating new databases
- handling of data dictionaries
- handling of units
- handling of coordinate systems and projections
- what data types (integers, reals, characters, etc) are supported
- the format of the data (ASCII/binary, one/many files, etc)
- what limitations are imposed on the data

Also of interest would be table showing which formats are imported and exported by the various GIS and any other practical concerns that are of relevance in GIS data exchange.

Appendix C

Terminology Cross-Reference

The table below lists equivalent terms from the vocabularies of the four systems under consideration.

Autocad DXF	ARC/INFO	Microstation	MapInfo
layer	coverage	level	layer
drawing entity	feature (class)	design element	object
line type	line symbol	line style	pen type
attribute	attribute (column)	column (field)	column (field)
drawing extent	coverage extent	design plane/cube	map extent
drawing file	workspace	design file	map
user coordinate system (UCS)		auxiliary coordinate system (ACS)	
world coordinate system (WCS)		design plane coordinates (units of resolution)	
entity handle	feature ID (user-id)	`MSLINK` column	unique identifier
	table	non-graphic database	table

Table C.1: Equivalent terms from each of the four systems

Appendix D

Entity Cross-Reference

The following table lists (nearly) equivalent entities in each of the four systems.

Autocad	Microstation	ARC/INFO	MapInfo
point	line	label point	point
line	line	arc	line
	line string	arc	
arc	arc		arc
		polygon	region
circle	ellipse	arc	ellipse
polyline	curve		polyline
text	text	label point	text

Table D.1: Entity equivalences in each of the four systems

Honours Projects 1992

As far as possible I have allocated tasks in accordance with your preference although some minor shuffling was necessary.

Peter Smith
Investigation of Load-balancing for a Network of Suns:

Supervisor: Paul
Examiner: Paddy
Oracle: Wolfgang

Mark Emberson:
Textual Image Compression and Faxes:

Supervisor: Tim
Examiner: Bruce
Oracle: Krys

Corey Yeatman
Error Correction in LR Parsers:

Supervisor: Bruce
Examiner: Rod
Oracle: Krys

Mark Cox
An EDI Graphical Design Tool for Small Businesses

Supervisor: Ray
Examiner: Tim
Oracle: Neville

Mike Hayton
Triangular Irregular Networks

Supervisor: John
Examiner: Wolfgang
Oracle: Paddy

Daniel Ayers
Data Exchange Formats for GIS

Supervisor: John
Examiner: Neville
Oracle: Paul

Ray Hunt
for COSC460