FME Desktop Under the Hood

Table of Contents

FME Technical Background

This document provides an introduction to the core technology behind FME®. In modern usage, many of the concepts detailed here are hidden from the user by the Workbench graphical translation configuration environment. Nonetheless, these details may be of interest to someone looking to understand the core FME technology. Specific details on individual FME functions, factories, and transformers (which are the Workbench wrappers for FME functions and factories) can be found in the FME online documentation at:

- *www.safe.com/support/onlinelearning/documentation.php*

In particular, the FME Universal Translator help file has detailed information on the functions and factories of FME, and the FME Workbench documentation describes the transformers that most FME users now interact with.

1.0 Introduction

FME reads and writes spatial data from and to a variety of supported formats. However, FME is more than a simple data translator. In addition to format conversion, FME is capable of performing sophisticated processing during the translation process. FME may even be used as a configurable spatial and attribute processing utility without doing any format translation by reading from and writing to the same data format.

Historically, the task of moving data from one format to another has been difficult. As a result users with large data stores have been locked into a single vendor's format and thereby restricted to using the analysis and decision support tools available for that format. FME enables users to evaluate tools independent of the data format which they support. Using its sophisticated topological and attribute processing subsystems FME is able to resolve the geometric and attribute mapping problems between different systems.

1.1 Overview

All modules in FME output statistics, progress information, warnings, and errors to a translation log file.

FME provides a fully configurable geo-object-relational data processing environment with many capabilities. The true power of FME lies in its ability to combine these operations under user control. The result is that FME can easily accommodate both simple and complex translation and processing problems without the need for new software to be written. The capabilities of FME can be divided into the following five areas:

DATA SOURCE MERGING

Through the use of FME's Multi-Reader facility (described in section 15) data from several different data sources, possibly of varies formats, can be read, processed, and output to a single output data set. This can be used to:

- Combine several 1:50,000 maps together, generalize them, and create a single 1:250,000 map.
- Produce derived products for an area by overlaying input data stored in several different supported formats.
- Bulk load data into the new generation of "seamless" GIS products such as ESRI's SDE.

TOPOLOGY OPERATIONS

Through the use of a topology model FME is able to perform sophisticated topological operations during data translation. The topology operations can be used to:

- Form polygons, possibly with holes, from input linear geometry.
- Merge points to the polygons or donut polygons they are enclosed by. The attributes associated with the point are then associated with the enclosing polygon.
- Remove duplicate line segments from an input data set. This is useful when the input data set is a collection of polygons which share common boundaries and the desired output is the set of arcs which define the polygon boundaries.
- Connect line segments together. This is used to reduce the number of features by combining their geometry. In this scheme linework only breaks when the attribute values of the line pieces change or when a topologically significant node is encountered.
- Generate interior points polygons. This is used when the destination format requires an inside point for the output polygonal data.

GEOMETRIC OPERATIONS

In addition to the topological operations, FME also provides a number of geometric operations. These can be used to:

- Generalize features by removing points from lines, converting small polygons and short lines to points, and converting small polygons to lines. Two different line thinning algorithms are available, and each of the generalization operations is fully configurable on a feature by feature basis.
- Calculate feature area and length. These calculated values can then be stored in the destination format or used to control other aspects of the translation process.

ATTRIBUTE OPERATIONS

While FME is primarily a spatial data translation system, it also provides an extensive suite of attribute manipulation operations. In fact FME does not require that the data it handles to have any spatial component, and could be used to translate between relational or object-relational attribute databases. The attribute operations enable FME to perform the following operations:

- Convert data between a relational model and an object oriented model. This capability eases migration to systems with an object-oriented model.
- Read or write feature attributes from/to external tables during the data translation operation. The attribution may be stored in several different types of attribute files.
- Join multiple relational tables together to extract or output data related to a single feature. Join operations can even be performed across tables of different types.

EXTENSIBLE ARCHITECTURE

FME is designed and implemented using object oriented techniques. The design enables new functionality to be added to FME without impacting the rest of the system, allowing extension in several ways:

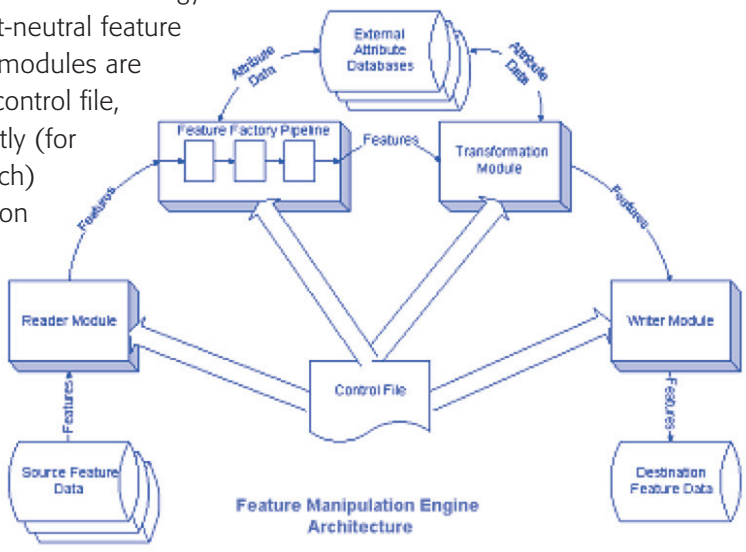
- Creation of new Feature Factories. Feature Factories are software units that features flow through. New factories can be added without a single code change to the remainder of the system. An API is planned which will allow FME sites to author their own factories.
- Creation of new Feature Functions. Feature Functions operate on a single feature at a time. Like the factories above, new functions can be added without a single code change to the remainder of the system. An API is planned which will allow FME sites to author their own functions.
- Creation of new reader and writer modules. This enables new formats to be added to FME. FME architecture is format neutral, so adding a new format to FME has no impact on any existing processing modules of the system.

1.2 Summary

In the past, new software was written for each translation situation. Over time, this introduced a maintenance problem, as it resulted in a large number of incompatible and inflexible translator programs. FME solves this problem by allowing the same translator program to be used to solve a variety of translation problems. End users need only use a text editor to adjust the configuration to handle completely different data models and transformations.

2.0 FME Architecture

FME is built using state of the art object oriented technology. The core of FME operates on a powerful format-neutral feature representation. Only the reader and writer modules are format specific. A script, originally called a control file, then called a mapping file, and most recently (for those created and edited by FME Workbench) called a workspace, governs the configuration and transformation of data from source to destination.



Feature Manipulation Engine Architecture

FME consists of software modules which operate on feature objects. The Reader Modules read features from external sources. The Factory Modules in the Feature Factory Pipeline join features together or split them apart in a variety of ways, all under the control of the user. The Transformation Module converts the feature from one format's representation into another's, perhaps relating the feature to data held in external databases. (Workbench generated mapping files/workspaces do not make use of the transformation module and instead do all the transformation within factories.) Writer Modules output the features in a supported format.

2.1 FME Functions

FME functions perform operations on features during the transformation process, and as they enter and leave feature factories. For example, the Generalize function performs point thinning on features, GeneratePoint generates internal points for polygonal features, and Count can be used to assign a unique number to each feature or group of features.

Feature functions may operate on entire features at a time, or they may be used to supply a value to an attribute. The Generalize is an example of the former, Count an example of the latter.

An open API is available to allow sites to author their own functions and use them within the FME environment. Tcl function definition in the mapping file is also allowed.

2.2 Feature Factories

Feature factories perform operations on groups of features during data translation. This is in contrast to feature functions, which operate on one feature or attribute value at a time.

Factories are used to solve processing problems involving one or more input features, and zero or more output features. For example, the PolygonFactory takes a number of linear features as input, and outputs zero or more closed polygons created by connecting the lines together.

Factories enable FME to perform very sophisticated processing tasks. Factories may be be combined with feature and attribute value functions, and chained together into a factory pipeline where the output from one factory becomes the input to the next. The result is that problems which traditionally required highly customized software can be solved by a single small FME control file.

2.3 FME Features

To transport features from one format to another, FME considers features to be collections of attribute names and values associated with two or three dimensional geometry (arbitrary dimensions will be supported in a future release). FME places no restrictions on the values, or types of attributes. Attribute names consist of one or more ASCII characters. FME features may contain any of the below types of geometry:

- Points
- Lines
- Polygons
- Donut Polygons
- Aggregates of one or more of the above

In addition to attributes and geometry, each FME feature has a feature type. Reader and writer modules interpret the feature type in ways specific to their own formats. For example, the IGDS reader and writer use the feature type to store the IGDS level of the feature, while the SAIF reader and writer use it to store the SAIF class of the object.

Feature attributes are usually a primitive type (integers, floats, characters). However, aggregates of attributes may also be stored in a single FME feature using an attribute list. Attribute lists behave just as primitive attributes, except that they contain an index enclosed in braces to identify an element of the list. Attribute list elements may contain primitives or other attribute lists. Attribute lists may be used by feature factories, feature functions, and the reader and writer modules.