

SEDRIS As An Interchange Medium

Michael R. Welch
Orion Development Group, Inc.

1. Introduction

1.1 Purpose

“Correlated initial environments is the critical precondition to achieving environment interoperability.” [1] Since existing synthetic environments are maintained as databases (mostly in proprietary formats), the essential pre-exercise interchanging of data between partnered simulators preparing to perform a combined exercise is awkward at best and frustratingly unsuccessful at worst. This paper documents the standardized interchange component of SEDRIS with respect to accommodating diverse clients with a variety of representational requirements.

This document is intended to coalesce existing ideas already presented by the SEDRIS team. When completed, this document will become Chapter 2 of the SEDRIS Overview.

1.2 References

1. Transcript of Paul Birkel’s Spring 1996 Presentation
2. Synthetic Environment data representation and Interchange Specification (SEDRIS) Rationale Document, Version Draft 0.2, 20 June 1996
3. Birkel, Paul A., *SEDRIS Geospatial Reference Model*, Draft 0.5, 10 July 1997
4. FAQest Transcript, STRICOM, 23-24 July 1997
5. SEDRIS Presentation made on 2 July 1997

2. Interchange Concept

2.1 Definition of Interchange

There is only one ground truth! [1] There is a single natural environment that all simulation applications are trying to represent. The natural environment is large and diverse. As a consequence, it takes the efforts of many producers to gather, analyze, define, describe and finally catalog the data elements of the environment. The consumers of this cataloged data have diverse system-unique requirements for fidelity, maintainability and run-time presentation. Consequently, each organizes their interpretation of the ground truth data to suit their specific needs.

Training and games are more engaging and effective when the wits of many are matched in a common scenario. But, if separate simulation applications (represented by their individual training needs and implementations) are to co-join in battle, they must share a common ground

DRAFT

truth. To share implies the two-way interchange of information and perception. No problem – I will give you my environmental data and you will give me yours. But, interchange is more than the physical swapping of data. To use the data, we must both speak the same language – so that we can truly share information. It is the common interpretation or perception of the environmental data, resulting from the use of a common language, that defines a successful data interchange.

Interchange, therefore, comprises consensual sharing of data as well as acceptance of a common definition of the interchanged information. We must either agree to convert one format to another or adopt a common (and standardized) transmittal medium which can serve as the basis of our interchange. The former concept has been tried in a variety of ways and has never achieved meaningful success. To support the implementation of the latter concept, an effort must be made to “capture the practical data types and their relationships, and to generate the basis for a robust and efficient synthetic environment interchange mechanism that meets the stringent needs of virtual and constructive simulation systems.” [2]

2.2 Data Representation and Applications [2]

The type of data found in a synthetic environment covers a wide range: information in a variety of forms describing the terrain surface itself, the complex features placed on the terrain, the dynamic objects with special 3-D model attributes and characteristics, the atmospheric and oceanographic features and much more.

There are three driving factors that affect representation of this varied data set from the view point of the application:

1. **Representational Polymorphism:** Applications, despite sharing a common geographical area requirement, often only need a subset of the entire data set. This means that these applications find the remaining representations of the data irrelevant to their needs and would prefer not to be burdened by these during data access.
2. **Representational Completeness:** The method by which the same exact data type is viewed can be radically different across applications. The type itself is not the issue, but the form in which it is expected becomes important (*e.g.*, a 3-D CAD model, a 2-D building, a 1-D symbol) and if not found can lead to serious divergence among synthetic environments.
3. **Representational Efficiency:** The format of a representation is often just as critical as the capabilities of the methods used for its access and the efficiency of those methods. While the individual elements of the representation may be complete, they may not be concise or their relationships may not be explicitly maintained.

The combination of these three factors results in a potential multiplicity of data representations and implementations, all of which are important to some consuming application.

DRAFT

In simple terms, the problem is that information (data elements, relationships, encodings) as viewed by one user could be interpreted as noise by another!

Therefore, the interchange issue is not only the specific data content (*i.e.*, the specific features and their attributes) but the way the data is represented and accessed. Furthermore, when the representation has to be accessed by a wide range of applications, each expecting to see the data element(s) in a “native” application-specific context, it makes efficiency and elegance in data interchange difficult to achieve. Fortunately, there is a bound to both the types of data elements and the accepted methods by which applications represent and access data elements. This makes the interchange challenge more manageable in both describing a representational model and accessing its data through software.

2.3 Database Interchange Before SEDRIS

Prior to the development of SEDRIS, synthetic environment data interchange was accomplished by point-to-point unique conversions between two specific systems. Conversion of one system’s data to another format was based upon rigidly defined formats for the source and target databases.

Because of differing proprietary database formats, each conversion required the development of a custom data converter software application. These point-to-point solutions were expensive, time consuming and often unreliable. To meet the specific implementation of the target system, the converted datasets usually had to undergo several additional conversions before a useable run-time format was obtained. Each conversion added to the risk of data loss or fidelity diminution. And, the number of unique conversions increases geometrically as additional data sources are added.

Figure 1 illustrates the many duplicative paths of a point-to-point conversion approach to data sharing and interchange. [5]

The Air Force began Project 2851 as an attempt to establish a means of interchanging representational environmental data between training systems. The project was intended to capitalize on the costs of gathering specific geographic data and formatting that data for inclusion in environmental databases. The user interface for Project 2851

was encapsulated in the Generic Transformed Database (GTB). The GTB was the central part of the repository and data was inserted and retrieved through its standard mechanism, the Standard Simulator Data Base (SSDB) Interchange Format (SIF). Both producers and consumers filled out a form that indicated:

- What type of data was produced/consumed,
- The associated set of data structures,
- The number of sides for polygon surfaces,
- The feature density.

The central library accessed the particular database that was of the desired geographic region and formatted the data into the structure that was requested. [4]

Unfortunately, Project 2851 did not solve the data interchange problem for environmental databases. The SIF only included representational data types used in visual scene generators. The SIF model did not account for other uses of the environmental data such as sensors and CGF. Also, SIF was focused on database re-use, not interoperability between heterogeneous training systems.

2.4 Interchange Based on a Data Model

A data model is a description of the data elements, including their attributes, and the logical relationships between these data elements. In object-oriented terminology, this is viewed as a class hierarchy. Each major element with important or explicit relationships is captured to show

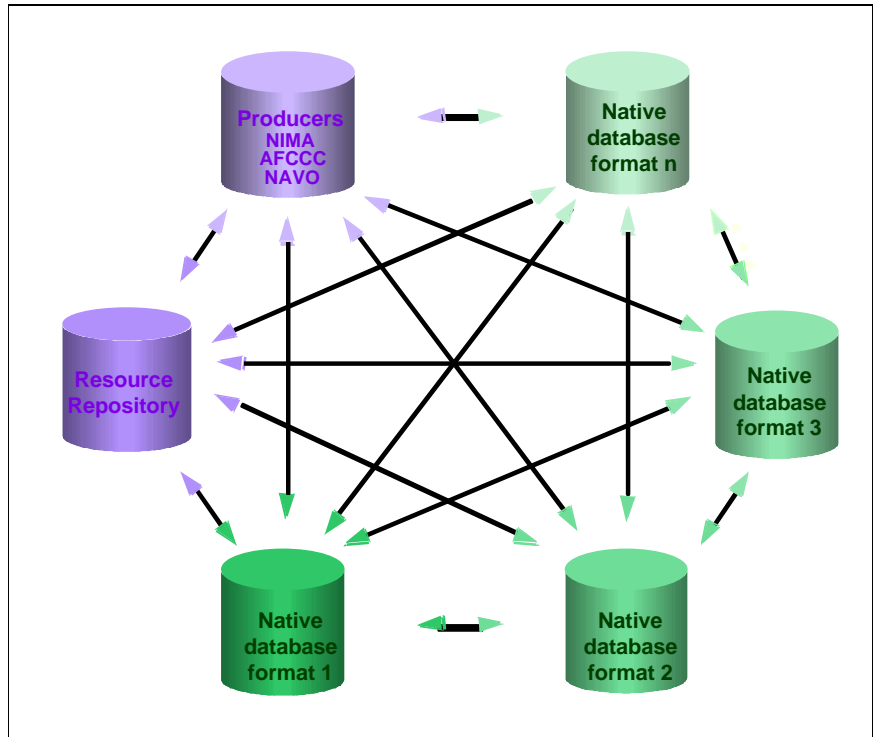


Figure 1 Before SEDRIS

DRAFT

its logical relationship to other data elements. [2] The strength of a data model is its capability to unambiguously define these data relationships.

It is easier to communicate with a data model (*i.e.*, a meta-model) of the data than with the actual data. The meta-model describes the data through its attributes rather than through its storage format. [4] The model removes ambiguity by ensuring that all types of environmental data are captured and relationships between alternate representations (feature vs. geometry) are defined. Using a polymorphic approach, multiple representational views can be supported:

- A 3-D visual view which includes all forms of visual and non-visual sensors (including communications),
- A 2-D overhead view for maps and charts,
- A computational view for use by computer generated forces algorithms. [4]

A data model supports development of Application Programmer Interfaces (APIs) to be used for accessing the data. It therefore enables reuse and interchange of synthetic environment data since data producers and consumers can readily develop software tools, utilizing the APIs, to convert their native data to and from the data model.

2.5 Database Interchange With SEDRIS

Right off, it should be understood that currently SEDRIS is a “transmittal medium” not a storage medium. SEDRIS serves as a centralized intermediary between differing formats thereby providing a conduit for interchange. The intent of SEDRIS is to create one standard method to interchange environmental databases “in a consistent fashion across the widest possible range of heterogeneous simulation systems which incorporate synthetic environments.” [2]

Figure 2 illustrates the central commonality of SEDRIS when interchanging environmental data between a variety of producer and consumer applications. [5]

From the beginning, SEDRIS set out to explicitly accommodate the superset of the system-specific data elements required by an interchange mechanism rather than merely the minimal set of shared data elements. [2] In its current form, SEDRIS supports the full range of synthetic environmental data. The SEDRIS data model discipline provides unambiguous and loss-less transfer of all data.

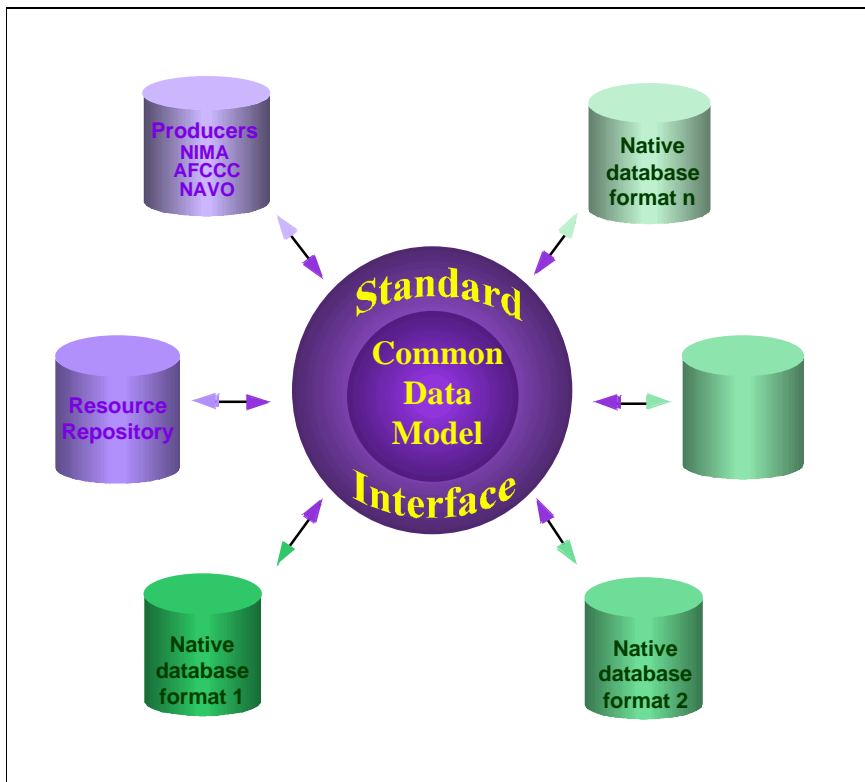


Figure 2 Using SEDRIS

The structure of SEDRIS data is transparent to both producers and consumers. The SEDRIS API efficiently enables M&S system and producer conversion software to pass data through the SEDRIS data model. The design of the SEDRIS API and transmittal files does not mandate any specific hardware platform dependency. At this time, software interfaces must be created in the standard C language.

Common tools and software are shared and reused by all SEDRIS users, thereby achieving a reduction in conversion/interchange costs. Ad hoc conversions are eliminated; therefore, there is less opportunity for loss of significant data leading to correlation problems. Location and color conversions are performed by “centralized” API routines to ensure that everyone gets the same results. The SEDRIS data model based interchange of synthetic environment data provides a practical solution that captures more than 80 percent of the environmental representational data needs of the M&S community. [4]

3. SEDRIS Data Interchange Capabilities

3.1 Built-in Access Support

To facilitate the straight forward use of a SEDRIS transmittal, a common set of tools and software are provided for shared reuse. An API provides a software interface to the transmittal data while a set of stand-alone utilities supply direct viewable access of the transmittal's contents.

3.1.1 Application Programmer Interface

The SEDRIS API provides a consistent interface between a software application designed to store and process the synthetic environment data and the underlying SEDRIS transmittal medium. The API decouples the database generation application from the interchange medium's data structures, allowing the format, its data structures and the application to evolve independently of each other.

The SEDRIS API has been implemented as a layered software model on two levels: Level 0 and Level 1. The routines contained in each level are intended to be linked into a consumer's native application to provide direct program access to a SEDRIS transmittal.

Level 0 is designed to perform data access/transfer with no derivation of data. What you provide is what you get back! To accomplish this strictly enforced interface, Level 0 provides a minimal set of fundamental access routines. These routines provide the capability to:

- Open and close a synthetic environment transmittal,
- Find and retrieve an object in the transmittal,
- Manipulate objects.

Level 0 is technically sufficient to allow complete bi-directional data access with a SEDRIS transmittal.

Level 1 utilizes the Level 0 routines and provides additional routines of a higher level of abstraction. Level 1 routines attempt to supply missing data by derivation if existing data makes derivation possible. Level 1 can be thought of as a small library of specialized applications that provide derived information: *e.g.*, "Get me the footprint of a specific 3-D model".

Access to SEDRIS data is context sensitive to support representational polymorphism. To support the user's particular data needs, only the data that fits the context is provided as output or required as input. Contextual retrieval is supported through the capability to define a geographic search boundary within the total area covered by the database, to create a search filter for limiting the type of objects which are examined, and to generate an iterated list of the located objects. The iterated object list represents the limited context specified by the user.

The SEDRIS API Level 0 supports 12 forms of location representation and 3 forms of color representation. To ensure that conversions are done consistently, the API provides conversion routines to all providers and consumers. Level 0 routines let a user select which

DRAFT

location representation and color representation is to be used for all of his accesses. Additionally, Level 1 routines are provided to perform coordinate and color derivations for discrete data elements.

The 12 supported Location 3-D coordinate conversions are: [3]

- GDC - Geodetic,
- GCC - Geocentric,
- GEI - Geocentric Equatorial Inertial,
- GSE - Geocentric Solar Ecliptic,
- GSM - Geocentric Solar Magnetospheric,
- SM - Solar Magnetic,
- GCS - Global Coordinate System,
- PS - Polar Stereographic PCS,
- LCC - Lambert Conformal Conic PCS,
- TM - Transverse Mercator,
- UTM - Universal Transverse Mercator,
- LST - Local Space Rectangular.

The three color representation formats are:

- RGB - Red, Green, Blue,
- HSV - Hue, Saturation, Value,
- CMY - Cyan, Magenta, Yellow.

3.1.2 Utilities [4]

A set of software tools, based on the SEDRIS API, have been developed to aid in examining, validating and viewing elements of a SEDRIS transmittal. The currently available tools are identified and briefly described below:

1. Depth - provides a plain text printout of the entire transmittal. Depth is primarily intended for use by Producers to debug data relationships. Depth will operate on any platform that supports a C compiler and the SEDRIS Read API.
2. Checker - a plain text program which validates the syntactical correctness of all entries in a SEDRIS transmittal in accordance with the Data Model. Checker is intended to aid Producers in developing syntactically correct SEDRIS transmittals by reporting and explaining encountered errors. Checker will operate on any platform that supports a C compiler and the SEDRIS Read API.
3. Browser - an X-windows based, point-and-click GUI browser, which presents a hierarchical view of the elements of the data model within the transmittal. Browser works in the same manner as a Windows File browser. The initial element is a synthetic environment object. Subsequent examination reveals the make-up of its sub-components: libraries, base, transmittal encoding and accuracy & lineage. Each of these components can be examined in further detail.
4. Model Viewer - provides an X-windows based viewer for 3-D polygonal models. It can also be used to view 2-D textures. The Model Viewer currently supports:
 - Level of detail,

DRAFT

- Articulated components,
- Textures, color, light.

Model Viewer runs only on UNIX platforms using Open GL and X windows.

1. Side by Side Terrain Viewer - supports the side-by-side viewing of two SEDRIS transmittals to visually compare their terrain elements. Identified differences can be made to conform by replacing the data in one transmittal with data from the other transmittal. Side by Side Terrain Viewer only runs on an SGI platform and requires Performa Loader for SEDRIS.
2. SEDRIS Vector Product Format Draw - provides a side-by-side 2-D map presentation of VPF features. This X-windows based program works with VPF-based SEDRIS transmittals.

3.2 Supported Data Interchange Users [5]

At this time, several SEDRIS team members have demonstrated the interchange of data using the SEDRIS prototype data model as the intermediary. Where as these preliminary efforts by no means represent the planned totality of the SEDRIS capability, they do indicate the ability to support a diverse community or users.

3.2.1 Polygon-based 3-D Models

The following table lists the data interchanges currently performed with a variety of 3-D models.

Source	Target	Tested On
S1000	SEDRIS	SGI, Sun
E&S - GDF(IDF)	SEDRIS	Sun, NT
CSI - DWB	SEDRIS	Sun
LMTDS - DBGS	SEDRIS	Sun
LMIS - TARGET	SEDRIS	Sun
SEDRIS	LMTDS - DBGS	Sun
SEDRIS	LMIS - TARGET	Sun

3.2.2 Source Data Products

The following table lists source data successfully inserted into a SEDRIS transmittal.

Source	Target	Tested On
VPF based data	SEDRIS	SGI, Macintosh
DTED	SEDRIS	SGI