Interchanging simulation databases with third parties using SEDRIS

R. Harris
Thales
Crawley, UK

ABSTRACT

Successful, seamless interchange of simulation databases has long proved surprisingly difficult to achieve. Numerous technical difficulties, arising from the different environmental representations used by different simulation systems, have proved to be only one facet of this difficulty. Often such problems are in fact the result of more fundamental underlying issues, such as the mathematical relationships between different co-ordinate systems. Logistical issues, and collaborative aspects of database interchange between different groups or companies, also contribute to the problems.

Thales has encountered many of these issues over the years in generating a range of databases for its simulation systems. These databases are required to correlate closely with other sensor systems, in particular the visual, but it is often the case that these other systems are third party products, using databases modelled by companies other than Thales. In these circumstances, the strategy used by Thales to generate its databases has typically been to derive them directly from the visual database. This has involved directly processing the visual database, extracting relevant geometry and attribution and formatting it for use by the Thales simulation systems. Historically, such visual databases have been provided by third parties using the SIF/HDI interchange format and imported directly into the Thales database generation toolset. While generating such derived databases in this way has been achieved successfully, many interchange issues referred to above were encountered and needed to be addressed.

When the need arose to replace SIF/HDI, the opportunity was taken to seek a replacement that would not only provide better representational capabilities but also address many of the wider, non-technical issues as well. Analysis of a variety of formats was undertaken and SEDRIS emerged as by far the strongest contender. Not only did it provide the best all round support for existing data representation requirements, it also gave good support for addressing wider interchange issues and offered a variety of opportunities to enhance the database generation toolset, both during initial development and over time.

This paper will discuss experiences using SEDRIS in this context. It will examine the basic representational requirements that needed to be met and the interchange problems that were to be overcome. The ways in which SEDRIS was seen to address these problems will be considered, along with the other advantages SEDRIS offered. Experiences developing SEDRIS software and interchanging databases using SEDRIS will also be described, including some lessons learned concerning both the use of SEDRIS and database interchange in general.

Paper No. 3253. Manuscript received 5 November 2007, revised 7 February 2008, accepted 12 February 2007.

https://doi.org/10.1017/S0001924000002141 Published online by Cambridge University Press

1.0 INTRODUCTION

Achieving the high degree of interoperation required by an integrated simulator or networked simulation system is a challenging task, requiring high degrees of correlation between the databases used by the individual component systems (e.g. visual, infra red (IR), radar, computer generated forces (CGF), etc.). It is challenging even when all such components are homogeneous, proprietary systems, fully under the control of a single manufacturer. Increasingly, however, this is not the case, with simulations more and more the products of multi-company partnerships, employing increasing levels of commercial off the shelf (COTS) technologies and needing to interact with systems and databases, legacy or otherwise, produced by different manufacturers. These circumstances present the challenge of generating databases that correlate rigorously between such heterogeneous systems.

To address this situation in as flexible a manner as possible, and ensure correlation, Thales has historically based a number of its own databases on the visual database. Such visual databases have often come from third parties who used their own proprietary database formats. The SIF/HDI database interchange format[1] was therefore used as an intermediate interchange format for importing the databases into the Thales database generation toolset. SIF/HDI had the advantages of being a US military standard and of enabling the interchange of not only polygonal and texture data, but also vectors and elevation grids when necessary. A number of programmes were successfully completed, deriving correlated databases based on visual databases provided in SIF/HDI.

However, a number of issues conspired to require a change to the way that such databases were derived. SIF/HDI was being used less and less, reflecting an industry trend to move away from it and provide data in other forms. While broad in scope, the representational capabilities of SIF/HDI also lacked aspects that were becoming significant, such as animations. On the face of it, a simple change to the interchange format could have addressed these issues. However, it was perceived that there was an opportunity to address not only these technical requirements but also the wider technical and non-technical interchange issues encountered on various programmes. Though changes would need to be evolutionary rather than revolutionary, it was felt that the entire generation process should also be reconsidered.

2.0 INTERCHANGE PROCESS REQUIREMENTS

2.1 Representational requirements

The first requirement for any new approach had to be the capacity to interchange the necessary data types. Any new interchange method would need to match the required representational capabilities found in SIF/HDI, while also providing support for new types of data becoming increasingly significant to simulation databases. The former included:

- Polygonal geometry for terrain and models
- Textures for terrain and models
- Classification encoding such as FID codes
- Material attribution such as SMC codes
- Vector features
- Elevation data

New capabilities, largely unavailable in SIF/HDI, included:

- Classified (or 'sensor') texture, with classification and material attribution at the pixel level*
- Model state switching
- Articulations, such as rotor blades, tank turrets, etc
- Additional attribution such as extended material encoding, trafficability, physical characteristics, etc.

Further still in the future lay the need for facilities such as:

- Mapping data
- Dynamic terrain
- Real-time feature development

2.2 Interchange issues to overcome

Previous programmes deriving databases via visual database interchange had provided a wealth of experience in the practicalities of interchanging databases with third parties. The lessons learned from these programmes identified several broad categories of issue that needed to be addressed by any new process: collaborative, logistical and technical.

2.2.1 Collaborative issues

The first category has little to do with the technicalities of database interchange but relates to the necessity to establish an effective working partnership between the visual database producer and the database consumer.

To achieve correlation between databases, purely physical interchange of data is insufficient. The relationship between the visual database and the databases that are to be derived from it can sometimes be complex in terms of the interactions and dependencies between their respective requirements. This will often generate an additional set of requirements on the visual database to enable the derived databases to be successfully generated. These additional requirements must be defined and agreed between the database producer and consumer prior to the start of database production. There must also be ongoing collaboration between the teams responsible for each database to ensure, not only that these requirements are met, but also that the day-to-day decisions of one team do not undermine the development efforts of another.

For example, radar systems require details of material properties not typically present in a visual database. However, where the radar database is to be derived from the visual, this cannot be done efficiently unless such data is either explicitly present in the visual database or can be derived from it. While it is not the visual database producer's job to model a radar database, they have a responsibility to enable the development of related databases where necessary.

In short, no database producer is an island.

2.2.2 Logistical issues

Logistical issues arise largely from the shear size of modern databases, typically measured in gigabytes, if not terabytes. Such large data sizes make handling the data a cumbersome and slow process. Exporting, testing, transporting and importing of the data can take immense amounts of time and resources and careful planning is required to ensure that this does not impact either database interchange or related database development activities.

** SIF/HDI allows images with SMC values per pixel but not classification or other attributes.*

https://doi.org/10.1017/S0001924000002141 Published online by Cambridge University Press

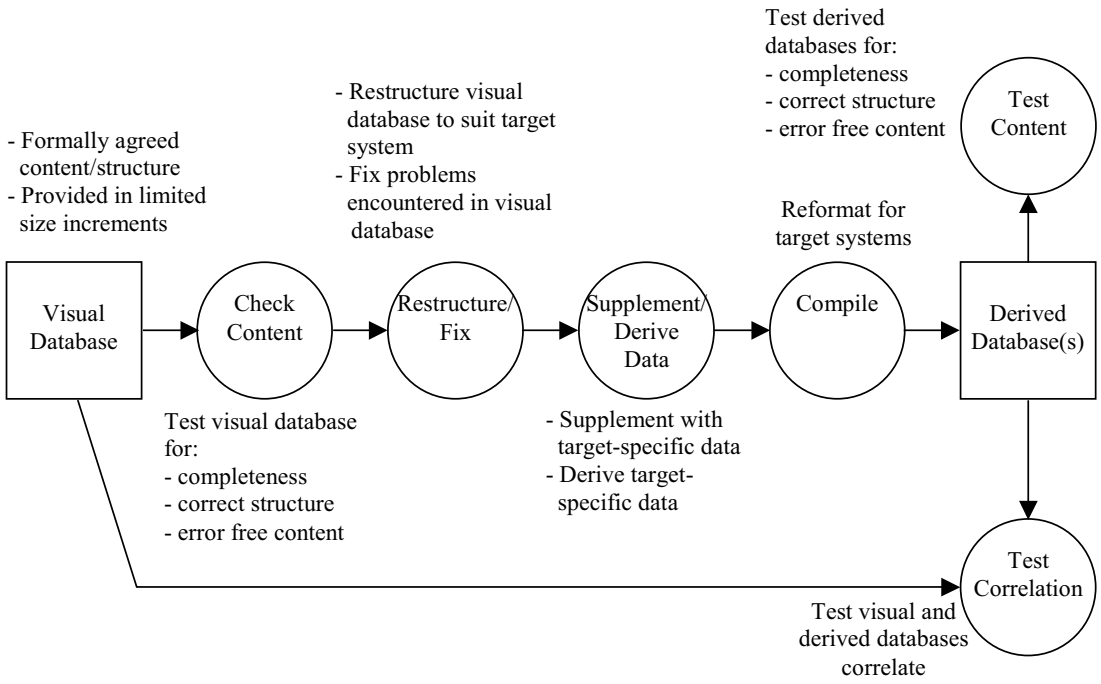


Figure 1. Database interchange process.

It was found that these issues were exacerbated in part by certain facets of the SIF/HDI format. SIF/HDI databases are represented as a single monolithic transmittal, made up of enormous numbers of interrelated files that are stored in the relatively large compressed ASCII format. This made it cumbersome to create, interchange and work with and meant that problems with one part imperilled the generation of the entire transmittal.

2.2.3 Technical issues

A surprisingly wide range of technical issues can affect the viability of database interchange. They can be broadly subdivided into those relating to data format, data content, tools and data processing.

Obviously the data format used must be capable of representing the necessary data and of doing so without distorting it. However, while essential, simple agreement over the database format to be used is completely insufficient to guarantee successful interchange. An interchange format may allow the same data to be stored in more than one way. While this can be useful in other contexts, it can undermine data interchange unless the way the interchange format is to be used is agreed between the database producer and consumer. Even then, producer and consumer may differ over what constitutes 'correct' usage of elements of the format, which may lead to misinterpretation of the data or even an inability to interchange some or all of it.

The fact that a format is capable of representing and providing the necessary data content does not mean that it will be used to do so. Database producers are concerned to represent their source database as accurately as possible but, even then, they may feel that it is inappropriate or too difficult to interchange some of the data. In contrast, database consumers are concerned that this same database contains what is necessary to convert it for use by another system, possibly of a different type. This may include data that is not required for the visual system alone. The requirements of the two parties may therefore be complementary in some ways and entirely contradictory in others.

The abilities of software tools to import, interpret and process an interchange database are only part of what is required to successfully generate a derivative database. Capabilities are also required to visualise, analyse, test and, if necessary, debug the interchange database. Since the interchange format is typically a secondary concern for the database producer, such tools may not be available or may be immature. As with all database generation tools, capabilities are also required for testing databases generated from the interchange data. However, in this context, there is the additional requirement to relate derived data elements directly back to the interchange database for purposes of debugging and testing issues such as correlation.

Data processing challenges revolve around the difficulties of reinterpreting a visual database such that it runs efficiently on a second simulation of a different type. Given the potentially different structural and content requirements of the two, this requires some capability to restructure or supplement the data. A powerful help in this area is the classification of data elements according to the type of features they represent, allowing additional attribution to be derived or associated based on this. Other issues associated with data processing can be complex algorithmically but have significant impact on the correlation of the databases, for example the conversion of co-ordinate systems.

In all cases, any bugs in the production or consumption of the interchange database will compromise the interchange process.

3.0 INTERCHANGE PROCESS DEFINITION

The issues considered above led to the definition of an abstract process for consuming and processing an interchange database, independent of the format or formats to be used. This process is depicted in Fig. 1.

https://doi.org/10.1017/S0001924000002141 Published online by Cambridge University Press

4.0 FORMAT SELECTION

Key to the implementation of the process described above is the underlying data format that is used to provide the visual database. Potentially, this same format could also be used to pass the data between the different steps, acting as a data backbone to the whole process. Several possibilities were considered.

Direct import of native visual database formats would have had many advantages. Most significantly, the need for additional data conversion would have been avoided, saving considerable time and resources. This would also have avoided potential distortions of the data, whether due to the data format, the necessary conversion process or bugs in that processing. The data would also have been well understood by the data producer, who would have had more mature tools for processing it and a significantly higher interest in ensuring that the data was correct and bug free. However, at the time, many third party data producers used their own closed, proprietary visual formats, so this was not an option.

The use of a common, open interchange format, for use with all potential data producers, was therefore considered. Several formal or *de facto* standards existed at the time. SEDRIS[2], in the same way as SIF/HDI, aimed to address the full range of data representation used in simulation. Other formats, most notably Open Flight[10], had wider usage within the simulation community but covered only part of the requirement, in this case textured polygonal representation. Other de facto standards such as Shapefile[11] provided other aspects, such as vector data. These standards, or a combination of them, were therefore considered for use.

4.1 Format requirements

The format(s) selected needed to address the following requirements, based on the representational and process requirements discussed above:

1. Necessary representational ability
- As defined in Section 2.1

2. Space efficient
- Space efficient storage
- Self contained sub-division of data content

3. Well defined, to avoid incompatibility, misinterpretation, etc

4. Minimal data conversion
- Save time and disk space
- Avoid errors and distortions introduced by conversion

5. Good tools support for required process shown in Fig. 1
- Visualisation of data structure and geometry
- Verification of data format, content and correlation
- Restructuring of data for target system
- Defect elimination

6. A standard, efficient, well-documented application programmer's interface (API) to aid development

7. A common format able to interchange visual databases from any data producer

4.2 Reasons for choosing SEDRIS

To determine the format to use, a full scale analysis of the capabilities of the various options was carried out against the defined requirements shown above. The results clearly showed that SEDRIS not only best met the requirements; it also provided additional valuable capabilities:

- All parties likely to be interchanging data with Thales supported SEDRIS; they had been involved in developing SEDRIS and were known to have SEDRIS tools.
- SEDRIS had the widest breadth of representational capabilities, easily addressing the representational requirements that had been defined. It covered a range of representation types typically used in simulation: geometry; vectors; images; gridded data such as elevation data; animations; physical characteristics; etc. These were designed for specific application to a range of simulation types (visual, infra red (IR), radar, computer generated forces (CGF), etc). It also allowed these different representation and application types to be inter-related, potentially aiding data enhancement and derivation based on different data types. Additional capabilities of interest in this area included: multiple texture pixel formats, including materials; an extensive feature classification coding system in its Environmental Data Coding Specification (EDCS)[6]; and support for a wide range of co-ordinate systems in its Spatial Reference Model (SRM)[7]. The potential existed to extend these capabilities through interaction with the SEDRIS community.
- The files making up each SEDRIS transmittal were binary format and automatically zipped, thereby reducing the disk space required to store and transport them. Transmittals could also be sub-divided into multiple inter-related but independent transmittals using inter-transmittal referencing (ITR). This allowed for independent generation and processing of sub-sections of the overall data set.
- SEDRIS was deliberately designed to be semantically unambiguous. As such it clearly defined the usage and semantic meaning of all elements, something later to be defined even more robustly during the process of ISO standardisation.
- Use of SEDRIS did necessitate conversion of all data, something that it had been hoped to avoid. Unfortunately, the use of diverse proprietary visual formats by different producers left little choice at the time. Given this requirement, however, SEDRIS was designed to allow data producers to represent their data in the most accurate way possible. This attempted to reduce or eliminate data distortion introduced by the export of the data.
- Allied to the SEDRIS format were a varied set of tools that addressed many of the requirements of the defined process. Transmittal structure and attribution could be visualised easily, as could the majority of data types of interest, such as geometry, textures, features and elevation data. Some ability to visualise animations such as states and levels of detail were also available. Such visualisations could also be used to check correlation in certain cases.
- Other tools could be used to carry out formal verification of a transmittal's conformance to SEDRIS syntax and usage rules. The more advanced content checker tool, SEE-IT, allowed the content to be tested for a wide range of issues typically causing problems in a simulation context, such as holes in terrain or disconnected networks. There were no fully functioning SEDRIS data editing tools of the kind available for formats such as Open Flight, since SEDRIS was not designed to be a modelling format as such. However, low level editing of SEDRIS objects and attributes could be carried out and SEE-IT could correct certain conditions automatically.
- Tools to directly enable restructuring of data were not available at this point. However, the FOCUS tool, with its plug-in architecture, was already available as a potential basis for such work in the future. In addition, work was already underway to develop an object matching expression (OME) language,

https://doi.org/10.1017/S0001924000002141 Published online by Cambridge University Press

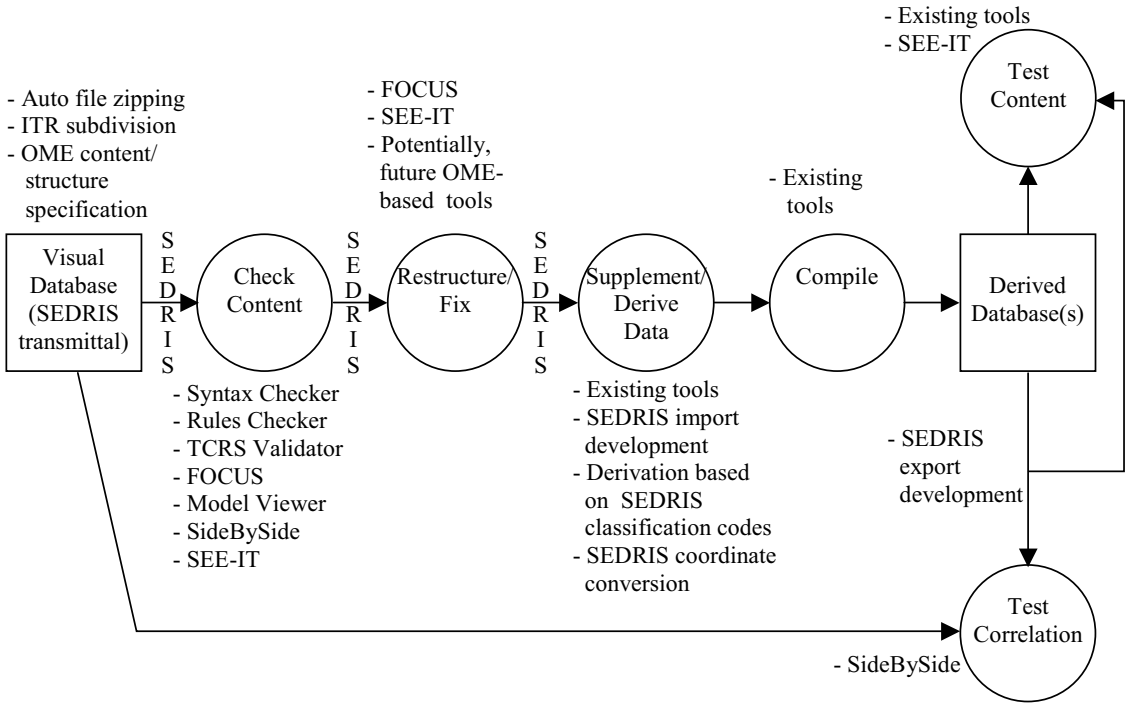


Figure 2. Database interchange Using SEDRIS.

allowing specific combinations of SEDRIS objects and attribution to be defined. The intention was already in place to use this as a basis for some level of automated content reorganisation, mapping transmittal content from one OME-specified arrangement to another.

- A well-documented C API had already been available for some time. A C++ API had recently been released which was easier to use and more efficient. In both, access times had been specifically minimised to address issues arising when working with large data sets. The API also provided extensive co-ordinate system conversion facilities helping to eliminate data distortion issues.
- Increasingly, SEDRIS was also appearing as a programme requirement in its own right.

4.3 Other advantages of SEDRIS

SEDRIS also provided other benefits that had not been specified in the requirements for a new format but which nevertheless addressed certain aspects of the interchange issues identified.

The OME format, used for defining combinations of SEDRIS objects and attribute values, had originally been developed to define the contents of specific SEDRIS transmittals. It was a formal XML-based language allowing producers and consumers to jointly specify the structure and content of transmittals to be interchanged. Such definitions had originally been informal documents called transmittal content requirements specifications (TCRSs). The use of OME not only allowed these TCRSs to be formalised but meant that a tool was available to verify that transmittals conformed to such a formalised TCRS.

Further advantages of using SEDRIS were that the core technologies and the majority of the tools were freely available from the SEDRIS website[13] at no charge. Most of them ran on multiple platforms.

5.0 INTERCHANGE PROCESS USING SEDRIS

Using SEDRIS, therefore, it was envisioned that the process of interchanging databases and using them to generate derived databases would be as shown in Fig. 2. This diagram shows the SEDRIS technologies used to enable or support each step.

6.0 EXPERIENCES WORKING WITH SEDRIS

6.1 Developing with SEDRIS

SEDRIS has proved easy to work with and fast to develop. This has been greatly helped by the introduction of a C++ implementation of the API. The C API is very powerful but consequently complex. The C++ API is able to use the advantages of object orientation to provide a more intuitive interface, doing more of the work for the user. The application code can therefore be simpler and thereby more reliable and faster to develop.

SEDRIS documentation is generally good, particularly that for the core Data Representation Model (DRM)[14] and the API[15]. This is now supplemented by the detail of the various SEDRIS ISO standards[2-9], available from the SEDRIS standards website[12]. Supporting papers and tutorials are also available at the SEDRIS website[13]. Finally, the SEDRIS Core Team have provided excellent support, including advice and code patches, etc.

SEDRIS itself has proven to have powerful representational capabilities directly applicable to this application, but also exceeding those currently required. As such it has not only been possible to use SEDRIS to represent all current data requirements but to start to use

https://doi.org/10.1017/S0001924000002141 Published online by Cambridge University Press

it to support new capabilities such as sensor texture and animations of various kinds. This breadth and depth makes for a large and complex entity that can be daunting to new users. However, it is very scaleable and only the elements required for a particular application need be used. In practice, this makes it no more complex to use than other standard formats.

The well-defined DRM provides a very solid means of representation in itself, but it is greatly enhanced by the environmental data coding specification (EDCS) component of SEDRIS. This includes a comprehensive set of classification codes, far in excess of the FID or FACC encodings. It also includes comprehensive encodings for a variety of physical attributions. Both have been vital for this application, since classification data is an important tool for deriving databases from dissimilar visual databases. Material attribution is, of course, essential, for radar and other sensor databases. The semantic power of the EDCS encodings became evident in processing these aspects of the visual databases. However, it is offering even greater potential support for the implementation of sensor texture where each texel requires classification and/or material attribution. There is great potential to use EDCS in other contexts requiring semantic encodings of these kinds.

The SEDRIS spatial representation model (SRM) has also been a powerful help, enabling relatively easy transformation between different spatial reference frames (SRFs) or, colloquially, co-ordinate systems. This has allowed easy use to be made of several co-ordinate systems and the SRM software has already been used in other contexts within Thales.

As discussed in Section 6.0, the various SEDRIS tools have proven to be very helpful, both in support of software development and in carrying out database analysis and debugging.

6.2 Interchanging databases using SEDRIS

To date, radar, CGF and digital map databases have been successfully derived from SEDRIS that has been generated by Thales and third-party suppliers.

SEDRIS has shown itself more than capable of representing the necessary data types and allowing them to be interchanged. With the initial implementation in place, enabling the interchange of currently used data types, further work was undertaken to exploit more advanced SEDRIS representational capabilities such as sensor texture. The representational capabilities of SEDRIS not only allowed interchange of such data but helped enable the necessary processing of it too.

Using SEDRIS' ITR capabilities, source databases have been processed as a set of interrelated transmittals. This has simplified working with large data sets and allowed source databases to be provided incrementally when necessary. It has also de-risked the generation of the interchange data since the whole data set is no longer imperilled by problems with one part. The automatic file zipping of the SEDRIS transmittal format (STF) files has also helped when working with large data sets by reducing necessary disk space, file copying times, etc.

With SEDRIS in place, good use has been made of a number of the SEDRIS tools, available from the SEDRIS tools website[16]. These have proven useful during software development as well as for analysing and debugging source visual databases. The FOCUS tool has been of particular use, with its built in hierarchy browser and low level editing capabilities. It is also a convenient platform from which to run the SEDRIS syntax and rules checkers and the Model Viewer applications on part or all of a transmittal. The Checker tools have allowed structural problems with SEDRIS transmittals to be identified before starting to process the data. The Model Viewer has enabled the visualisation of models, including their level of detail, state and articulation animations, enabling problems with these to be identified. The SideBySide viewer has provided similar support for polygonal terrain. All these tools have provided a useful common frame of reference between data producer and consumer that have provided a basis for discussing problems and their necessary resolutions.

SideBySide and SEE-IT have also been used to visualise, analyse, verify and debug derived databases. Exporting the derived databases back out to SEDRIS has enabled this. Use of these tools has allowed a variety of problems to be identified, analysed and addressed.

Use of SEDRIS has addressed many, though not all, of the interchange issues identified in Section 2.2. In particular, many of the technical issues have been addressed and some of the logistical issues have been eased. The majority of the collaborative issues identified remain, but this is to be expected; by their nature, such collaborative issues cannot be addressed directly through the application of an interchange technology of any kind.

7.0 CONCLUSIONS

SEDRIS has shown itself to be a powerful method for representing and interchanging simulation databases, providing many valuable capabilities that often go beyond those of other database interchange methods. These capabilities address wider interchange issues and also have applicability to database processing in general. Individual SEDRIS components, such as EDCS and SRM are valuable even used independently of other SEDRIS elements. Software to use SEDRIS can be developed easily using the well-documented APIs. The C++ API is particularly easy to use. Tools support is very good and some tools, such as SEE-IT, have application to database development in general, not just to SEDRIS transmittals.

SEDRIS is a valuable resource for database development, not only for database interchange, in its role as a standard, vendor-neutral, intermediate format, but for database processing in general.

REFERENCES

1. Standard simulator data base (SSDB) interchange format (SIF), 17 June 1993, Design Standard, MIL-STD-1821.
2. SEDRIS — Part 1: Functional specification, May 2006, ISO/IEC 18023-1.
3. SEDRIS — Part 2: Abstract transmittal format, July 2006, ISO/IEC 18023-2.
4. SEDRIS — Part 3: Transmittal format binary encoding, July 2006, ISO/IEC 18023-3.
5. SEDRIS Language bindings — Part 4: C, May 2006, ISO/IEC 18024-4.
6. Environmental data coding specification (EDCS), June 2005, ISO/IEC 18025.
7. Spatial reference model (SRM), June 2006, ISO/IEC 18026.
8. EDCS language bindings — Part 4: C, July 2005, ISO/IEC 18041-4.
9. SRM language bindings — Part 4: C, August 2006, ISO/IEC 18042-4.
10. OpenFlight scene description database specification, November 2007, Version 16.3, Revision A.
11. ESRI Shapefile technical description, July 1998, ESRI White Paper.
12. SEDRIS standards website, http:// standards.sedris.org.
13. SEDRIS website, http://www.sedris.org.
14. The SEDRIS data representation model, SEDRIS technology documentation set, **4**, (2).
15. SEDRIS transmittal access C++ API, Version 4.0.0.
16. SEDRIS tools, website, http://tools.sedris.org.

https://doi.org/10.1017/S0001924000002141 Published online by Cambridge University Press