Towards RPR FOM 3: Revisiting the Datatypes

Björn Möller
Pitch Technologies
Repslagaregatan 25
58222 Linköping, Sweden
+46 13 4705503
bjorn.moller@pitch.se

Aaron Dubois
VT MÄK

150 Cambridge Park Drive
Camrbidge, MA, USA
+1 857-209-3451
adubois@mak.com

Patrice Le Leydour
Thales Training & Simulation
1, rue du Général de Gaulle – Osny
95523 Cergy-Pontoise, France
0033-1-34228252
patrice.leleydour@thalesgroup.com

Graham Shanks
BAE Systems
North Way, Hillend Industrial Park
Dunfermline, Fife, KY11 9HQ, United Kingdom
+44 1383 826450
graham.shanks@baesystems.com

René Verhage
CAE
Steinfurt 11
52222 Stolberg, Germany
+49 2402 106599
rene.verhage@cae.com

Fredrik Antelius
Pitch Technologies
Repslagaregatan 25
58222 Linköping, Sweden
+46 13 4705503
fredrik.antelius@pitch.se

Keywords: RPR FOM, HLA, Datatypes

ABSTRACT: *Version 2 of the Real-time Platform Reference FOM (RPR FOM) has recently been finalized. It is the most widely used FOM for defense simulations. The original purpose of the RPR FOM was to facilitate interoperability between the DIS protocol and HLA federations. Today it is often also used as a common basis for further adaptation and extensions in US and NATO federations.*

One of the main goals of the final phase of the RPR FOM 2 development was to maintain buffer compatibility with the widely used draft 17 of the RPR FOM 2. This in turn carries a lot of heritage from both the DIS protocol and the HLA version 1.3, including many convoluted data buffer layouts. Today these may not be seen as striking the best balance between low bandwidth utilization, simple encoding and decoding, flexibility and extensibility.

Now the time may have come to revisit the RPR FOM data representations for RPR FOM version 3. In addition to the reviewing the record data structures, a goal could be to remove the RPR FOM specific datatype encodings such as the length less array representations. Furthermore, an attempt to generate the Enumerations module from the SISO-REF-010 XML source showed that some enumerations may need to be reconsidered or moved to other modules. The RPR FOM 2 work has also revealed that some new datatypes may need to be added to the HLA standard, in particular to represent unsigned integers that are used in DIS.

This paper provides an analysis and recommendation for the RPR FOM 3 development and to some extent for the next version of HLA.

1. Introduction (All)

The Real-time Platform Reference FOM (RPR FOM) version 2.0 has been completed. During the final phase of this development (2012-2015), two decisions were agreed upon within the Drafting Group:

1) To keep the scope of the widely used draft 17 of the RPR FOM. No new classes, attributes, interactions or parameters were to be added, unless there is an obvious error.

2) Any information exchanged in attributes and parameters shall be backwards buffer-compatible with draft 17. This will eliminate the need to update federates and federations already supporting draft 17, facilitating and speeding up acceptance in the defense simulation community.

The second decision unfortunately resulted in keeping datatypes that may not be considered optimal today. Some contributing factors are:

1) Programming languages. In the days of SIMNET, the predecessor of DIS, languages like C and assembler were commonly used. Today high-level languages like C++ and Java are more common languages for simulation development. Data structures that are considered optimal in today's programming languages may be different from what was considered optimal in the early days of distributed simulation.

2) The HLA Datatypes of IEEE Std 1516™-2000 and onwards. The data representations developed up to RPR FOM draft 17 are not always natural or optimal for a FOM specified using an HLA 1516 OMT.

3) Evolution in hardware. CPUs of today move data in groups of four or eight bytes. Available network bandwidth has increased dramatically.

4) Cost of manpower and required time to market for simulation software. CPU power and bandwidth today is often abundant but less and less time and money is available for development. Data representation thus needs to be simple and easy to understand.

The purpose of this paper is to list and discuss a number of issues that have surfaced during the RPR FOM 2 completion. Pros and cons are described. In some cases no final resolution is proposed. Instead, this information is intended as input to the RPR FOM 3 effort.

1.1. Purpose of original RPR

The IEEE Std 1278.1™, known as the DIS standard, supports technical and semantic interoperability between compliant applications. It defines a set of PDUs with a predefined data model and binary representation. A standard DIS PDU assembled by one application can be understood by any other.

HLA introduced the capability for simulation developers to define their own data model. The data model that federates agree to share during an exercise is defined by a Federation Object Model (FOM). This allowed HLA federation developers a greater amount of flexibility to refine their object model as the requirements of their simulation changed or expanded, but they lost the a priori interoperability inherent in DIS.

The Guidance, Rationale, and Interoperability Modalities (GRIM) document that accompanies the RPR FOM outlines three primary goals for the standard.

The first goal was to support the transition of existing DIS systems to HLA. This was accomplished by modeling the RPR FOM on the existing data content of the DIS PDUs. As a result, converting existing software from DIS to HLA was considerably easier, as was mapping between the standards using a gateway application.

The second goal was to enhance a-priori interoperability among RPR FOM users. As a standard reference FOM, the RPR FOM was designed to provide a common base object model that would be immediately interoperable with any other RPR compliant applications, similar to the interoperability guaranteed by DIS. This common base could then be built upon to add new capabilities to support the individual requirements of each federation.

The final goal was to provide a pre-existing FOM that could be adopted by newly developed federates with similar requirements, eliminating the need for each new exercise to perform the task of defining a new FOM.

1.2. Mapping of DIS

The RPR FOM was designed to provide an intelligent mapping from DIS into HLA. Rather than produce an object model that blindly mapped DIS Protocol Data Units (PDUs) into FOM objects and interactions, the RPR FOM tried to exploit the benefits that HLA brought to distributed simulations. Some PDUs were broken up into multiple FOM objects whilst others were organized into object hierarchies. In general fields within the DIS PDUs mapped directly to object attributes and interaction parameters. One of the decisions taken was to try to preserve bit compatibility for datatypes, especially for structures and arrays. This meant that for most datatypes the encoding for DIS fields and RPR FOM attributes/parameters were identical. This eased the implementation of federates that were compatible with both DIS and RPR FOM, or were transitioning from DIS

to RPR FOM, as well as making the implementation of gateways between the DIS and RPR FOM protocols easier.

1.3. HLA 1.3 heritage

Development of RPR FOM 2 began in 1999, before IEEE Std 1516™-2000 was completed. As a result, initial versions of the RPR FOM 2 (until the ballot of draft 17) focused only on HLA 1.3. A number of IEEE Std 1516™-2000 and IEEE Std 1516™-2010 versions of the FOM were created before draft 19. However, they were generated from and in compliance with the official draft version in the 1.3 format. Before draft 19, none of the FOMs in the newer formats were included in the official distributions of the RPR FOM 2 drafts.

In several ways the HLA 1.3 OMT specification is not as unambiguous or complete as the later HLA standards in regards to how datatypes should be encoded. It does not specify whether numeric datatypes should be encoded in big or little endian format, it does not specify how arrays should be encoded, and it does not specify how proper byte alignment should be achieved. In addition, it chooses a different standard representation for strings than that chosen in the later IEEE Std 1516 standards. These discrepancies in FOM formats between HLA 1.3 and the HLA 1516 standards have directly led to RPR FOM datatypes and encodings that are considered non-standard according to the IEEE Std 1516™-2000 and IEEE Std 1516™-2010 OMT specifications. While sometimes cumbersome, these discrepancies have been maintained rather than break interoperability with previous drafts of RPR FOM 2.0 that have already been widely accepted and used by the distributed simulation community.

2. Datatype Issues

This section describes a number of topics related to datatypes. We have chosen to group them according to the technical character of the issue. This enables us to analyze the issues and their pros and cons in a consistent way. Nevertheless, decisions to change or retain a current specification for one topic may impact discussions on other topics.

2.1 Use of standard HLA encodings

The HLA OMT Specification includes a number of predefined encodings for constructed datatypes, but it allows object model developers the ability to define alternative encoding schemes as well. RPR FOM 2 uses several of its own encodings that pre-dates the HLA standard encodings, which were first defined in IEEE Std 1516™-2000. In some cases there may be a benefit to defining a new encoding type, but doing so may run the risk of adding unnecessary complexity to the FOM.

Array Encoding

The HLA OMT Specification defines an array encoding called HLAvariableArray to support arrays that may vary in length at run time. This encoding consists of a 32 bit integer indicating the number of elements in the array followed by each element in sequence.

The RPR FOM instead sometimes uses an alternative encoding called RPRlengthlessArray. A RPRlengthlessArray may also be of variable length; however the length of the array is not specified in the encoding. Instead, each element is simply encoded in sequence. It is left to the decoding federate to determine how many elements are in the array.

When array elements are of a fixed size datatype, determining the number of elements in a RPRlengthlessArray is simply a matter of dividing the total size of the data by the size of a single element. In such cases, decoding a RPRlengthlessArray is as simple as decoding a HLAvariableArray. It also has the small benefit of eliminating the unnecessary 4 bytes of data required to encode the array length.

For a RPRlengthlessArray containing elements of variable size, however, the length of the array cannot be pre-determined. In the best case, the size of each element can be determined during the decoding process. For example, this would be possible if each element in the array were a null terminated string. Each time a null terminating character was encountered, a new element could be decoded from the array. By fully traversing the data in the array, the number of elements could be determined. In other cases the size of each element may be more difficult to determine.

Another major drawback with RPRlengthlessArray is that it is very difficult to use within another datatype, for example as a field in a FixedRecord or as an element in another array. In such cases it is not trivial, and may be impossible, to determine when the RPRlengthlessArray ends and when the next data element begins.

As a result, it is recommended that RPR FOM 3 adopt the standard HLAvariableArray encoding scheme. This will provide a single, consistent encoding that will work in all cases. While it will introduce an additional 4 bytes per array, on modern systems this is a fairly minimal increase in size. In a few cases some other field implicitly provides information about the length of the array. In such cases it is recommended that the length specifying field be combined with the array in a standard HLAvariableArray. Depending on the size of the length specifying field there

may not be any overall increase in the number of bytes transmitted.

String Encoding

There are two predefined array datatypes specified in the HLA OMT Specification for encoding strings: HLAASCIIstring and HLAunicodeString. Both use the HLAvariableArray encoding. The RPR FOM defines a new datatype called NullTerminatedASCIIString that uses an alternative encoding called RPRnullTerminatedArray. Like the RPRlengthlessArray encoding, RPRnullTerminatedArray omits the 32 bit integer specifying the length of the array. However, unlike the RPRlengthlessArray it does provide a means of reliably determining the length of the array by requiring that the last element of the array be a null terminating character.

This is a case where the programming language a developer is using may dictate which encoding is preferable. In some languages, such as Java, strings are not null terminated. In order to construct a string from a null terminated array of characters, a Java application must traverse the array and count the number of characters prior to the null terminator in order to determine the size of the decoded string. In other languages, such as C++, strings are typically null terminated, and the language has built in support for easily constructing strings based on a null terminated array of characters. However, C++ does also include a simple way to construct a string given a predetermined size as well.

In this case, neither encoding is clearly better than the other. The additional 3 bytes required to encode a HLAASCIIstring seems negligible on modern systems. The recommendation is to use HLAASCIIstring in RPR FOM 3, simply as a means of reducing the complexity of the FOM by using a pre-existing and well understood encoding rather than defining an alternative without a clear advantage.

Variant Record Encoding

The HLA OMT Specification defines a single encoding for variant records called HLAvariantRecord. This encoding consists of the discriminant followed by the appropriate alternative that is associated with that discriminant.

The RPR FOM introduces an encoding called RPRextendedVariantRecord. This encoding adds an additional value after the discriminant that indicates the size of the variant portion of the record. The intent is to allow an application that may not support all of the alternatives to easily skip the rest of the variant record in cases where the variant record is part of a larger datatype. This can save development work when only a subset of the alternatives is supported by a given application.

In order to eliminate an unnecessary encoding, the RPRextendedVariantRecord could be removed and the standard HLAvariantRecord encoding could be used instead. Only one datatype in the standard RPR FOM, EnvironmentRecVariantStruct, currently uses the encoding, and it is unknown how many applications take advantage of the additional size field.

Another option would be to create a new fixed record datatype composed of a size field followed by a variant record encoded by the standard HLAvariantRecord encoding. This would eliminate the need for an additional encoding, but it would require the definition of an additional datatype for each use of the current RPRextendedVariantRecord encoding. While the standard RPR FOM only uses this encoding once, FOMs based on the RPR FOM may use this encoding in their own datatypes. As a result, it is unclear if this would truly reduce the overall complexity of the FOM.

In the short term it may be sensible to eliminate the encoding in RPR FOM 3 and use HLAvariantRecord instead. A better long term solution would be to include the RPRextendedVariantRecord as a standard encoding in a future version of the HLA standard. This has the advantage that it is better suited to the proposal to enable variant records (as well as enumerations) to be extended in other FOM modules since federates which do not load the module containing the extension will at least now the size of the variant part and can reliably decode datatypes containing such a variant record.

2.2 Missing HLA datatypes

RPR FOM 2 uses unsigned integers, for example for enumerations. In some cases all bits are necessary to express a particular enumerated value. The HLA OMT does not currently support unsigned integers. The addition of unsigned integers to the HLA standard is the preferred solution. Another option would be to not use unsigned integers in the RPR FOM. This would create some issues in the mapping to DIS.

2.3 Enumerated date type sizes

Boolean Encoding

The IEEE Std 1516™-2000 and IEEE Std 1516™-2010 OMT specifications define an enumerated datatype, HLAboolean, to represent a Boolean value. The data representation used by this datatype is HLAinteger32BE. The RPR FOM has always encoded Boolean values as a single byte. In order to remain compatible with older drafts of the FOM, RPR FOM 2 created a new enumerated datatype for Booleans, RPRboolean, that is

represented by an HLAoctet. The primary benefit of using RPRboolean is that it saves three bytes per usage. However, this savings seems negligible on modern systems, so it is recommended that RPR FOM 3 use the standard HLAboolean instead in order to eliminate the unnecessary duplication of datatypes.

Enumerated Datatype Encoding

RPR FOM 2 chooses the data representation for enumerated datatypes based on either the size reserved in DIS for possible enumeration values, or, for non-SISO-REF-010 enumerations, the smallest size integer required to encode all possible enumerator values. The latter helps to keep the size of the encoded data to a minimum. However, it also means that developers using the RPR FOM must consult the FOM for each enumerated datatype to ensure they are encoding and decoding the correct size integer. It also means that a RPR based FOM is more limited in the number of custom enumerator values they can add to an enumeration. By standardizing on a larger size integer, RPR FOM 3 would make developing RPR applications slightly less error prone and would allow FOM developers greater flexibility to add new enumerator values to existing datatypes. Nevertheless, to continue to support interoperability, it remains important to ensure compatibility with SISO-REF-010.

2.4 Datatypes related to SISO-REF-010

Since the RPR FOM is based on the DIS standard, the enumerations as defined in Reference document SISO-REF-010 play an important role in establishing simulation interoperability. To highlight this dependency, and to enable the RPR FOM users working with HLA 1516-2010 to easily use an updated version of SISO-REF-010 or their own custom enumerations, these datatypes have been defined in a separate module.

However, as of today the Enumerations module does not fully reflect SISO-REF-010. Some of the enumerations in SISO-REF-010 are not used in RPR FOM 2. Instead a corresponding enumeration is defined directly in the RPR FOM. In some cases the semantics is the same in the RPR FOM and SISO-REF-010 but the actual enumeration differs. In some cases SISO-REF-010 bit fields have been split into several attributes, making updates hard to map.

Spread spectrum type enumeration

The easiest to solve is the one enumeration that moved from the SISO-REF-010 back into the DIS standard itself. IEEE Std 1278.1™-1995 refers to EBV-DOC section 9 for the information captured in the RPR FOM in SpreadSpectrumEnum16. As in IEEE Std 1278.1™-2012 the details on the Modulation Parameter Record are captured in Annex C. This enumeration should be moved to the Communication module in the RPR FOM.

Minefield sensor type & Camouflage type enumerations

More problematic are the CamouflageEnum32 and MinefieldSensorTypeEnum32 enumerations. These appear to be a combination of two or more individual enumerations in SISO-REF-010 ([UID 378, 384] and [UID 194-201] respectively). Although there is a clear relationship between these individual enumerations, and with the current enumerators it is indeed possible to merge them, it makes it difficult to align the Enumerations module with SISO-REF-010. It is the same kind of complex mapping that needs to be implemented in DIS/RPR FOM gateways. From this perspective it would be beneficial to create datatypes that match their definition in DIS closer and modify the classes that use them accordingly.

Bitfield enumerations

Another example of the consequences of the differences in datatypes between DIS and RPR FOM are the bitfield enumerations defined in SISO-REF-010. Many of these are not defined as distinct datatypes in the Enumerations module, but instead have been translated into individual attributes in the classes. In favor of interoperability it is indeed unlikely that existing bitfields, e.g. the various appearance definitions, will be modified. But in several bitfields there is space left for additional information to be exchanged. And this extensibility in DIS has already been used, as can be seen from e.g. the newer capabilities defined for the various platforms. As a consequence, there may be implementations compliant with the DIS 1998 standard, but due to these using the latest SISO-REF-010, their expressiveness cannot be fully mapped onto an HLA network using the RPR FOM 2.0. To revolve this in the next version of the RPR FOM, the bitfield structures from SISO-REF-010 could be translated into individual record datatypes and each bitfield row into a distinct enumeration. Another alternative, at least for the entity appearance, is discussed in section 2.5. This however requires a solution for the next topic.

Entity type enumerations

The largest part of SISO-REF-010 may be much harder to capture in the RPR FOM object model: the entity type definitions. The entity type definitions are not included the RPR FOM version 2. It would be preferable if they could be included as enumerated types as they contain useful information to the user. Many of the elements of the entity type structure are restricted in their values based on the contents of other elements. However the complex dependencies between the elements of the entity type

made it too difficult to easily represent in the HLA OMT format. Given that the move to an XML based format for SISO-REF-010 has given new insights in how to define these dependencies it may be that it is time to revisit this.

One could argue that the entity type enumerated values should remain separate from the object model, just as they are not defined in the DIS standard but listed in a reference document. Following this argument it could also be argued the all SISO-REF-010 based datatypes be removed from the RPR FOM, and, for example, opaque data structures used instead. Given the argument in paragraph 2.5 of this paper to do exactly the opposite, the authors call upon the readers to come up with ideas and discuss how to capture in HLA datatypes the information of the entity types commonly agreed upon.

The following are some thoughts to trigger the discussion. An entity type definition could be stored as one large integer value, effectively a union of the entity type record with a 64-bit integer. However, it would then not be possible to determine (easily) e.g. the kind or the country from the enumerator value. So maintaining the structure layout of the entity type record would be preferable. But defining an enumeration for each of these fields does not solve the issue; still it would then be possible to create an entity type that is valid according to the FOM, but not defined in SISO-REF-010. What seems to be needed is a capability to define an enumeration using a record as the representation datatype. Or are there other options that do not require the HLA standard to be updated?

2.5 Opaque data

The RPR FOM contains a number of opaque datatypes whose syntactic content is not contained in the RPR FOM. Instead the user has to consult other documentation to determine the structure of the opaque datatype. This goes against the principles of HLA where the FOM contains the format and syntax of the HLA object models. Automatic tools, such as data monitors, data loggers and code generators cannot properly process these opaque datatypes.

VariableDatumStruct

The VariableDatumStruct is defined in the RPR FOM as a HLA fixed record with three fields: the DatumID, the DataumLength and the DatumValue. The DatumValue field in the VariableDatumStruct is defined as an array of 64-bit unsigned integers. In reality the DatumValue does not always consist of 64-bit unsigned integers, for instance the DatumValue may well be an Entity Identitifer. The datatype of the DatumValue depends on the value contained in the DatumID field. The dependency on the DatumID is explicitly defined in the DIS Standard, referring to SISO-REF-010 for the definitions of the possible types, but is not explicitly included in the RPR FOM.

Given the dependency it is clear that the VariableDatumStruct should actually be encoded as a variant record. The preferred encoding would be the RPRextendedVariantRecord since it includes the length of the DatumValue. This allows for additional DatumID types to be defined in the FOM, e.g. from aligning with a newer SISO-REF-010, without requiring federates to be recompiled if they don't support it anyway. As argued in section 2.1, the VariableDatumStruct could be encoded with a combination of the standard HLAvariantRecord embedded in a fixed record including a field for the datatype size. There are, however, probably more simulations using the VariableDatumStruct than are using the EnvironmentRecVariantStruct and this needs to be taken into account when deciding whether to keep the RPRextendedVariantRecord encoding.

One of the issues with populating the variant record is that the datatypes of many of the DatumID values identified in SISO-REF-010 [UID 66] are unknown. There are over 750 distinct DatumID values identified in the latest version of SISO-REF-010 and probably for no more than 10% their datatypes can be determined from the description. As the DIS Product Support Group (PSG) and the Enumeration Working Group (EWG) are in the process of moving the datum record specification from SISO-REF-010 into the DIS standard, collaboration is required with the two groups to have the datatype for each of the DatumIDs explicitly defined, accomplishing unambiguous interoperability for both RPR FOM and DIS users. However, this should not stop turning the VariableDatumStruct into a variant record with the definition of a suitable default.

FixedDatumStruct

The FixedDatumStruct is defined in the RPR FOM as a HLA fixed record with two fields: the FixedDatumIdentifier and the FixedDatumValue. The FixedDatumValue field in the FixedDatumStruct is defined as a 32-bit unsigned integer. In reality the FixedDatumValue does not always consist of a 32-bit unsigned integer, instead the field can convey 8-bit, 16-bit or 32-bit data values including 32-bit floating point values. The datatype of the FixedDatumValue depends on the value contained in the FixedDatumIdentifier field. The dependency on the FixedDatumIdentifier is explicitly defined in the DIS Standard, referring to the same enumeration in SISO-REF-010 for the definitions of the possible types as for the Variable Datum Record, but is not explicitly included in the RPR FOM.

Hence the FixedDatumStruct should also be encoded as a variant record. In this case the HLAvariantRecord encoding would be suitable as the data length is at maximum 32 bits whereas the HLA byte alignment rules result in a minimum of 32 bits for the data in the alternative.

Since the discriminant of both the VariableDatumStruct and the FixedDatumStruct are the same (i.e. both DatumIdentifierEnum32) it is worth considering combining both the fixed and variable datum structures in a single datum structure. Thus far in the RPR FOM the separation into fixed and variable datum records followed the structures as defined in DIS, requiring less data to be transmitted by omission of the data length for datum values that fit into 32 bits. With increased network bandwidth this optimization may be regarded less important than reducing the object model complexity by defining only one datatype for exchanging data that is specified by the Variable Record Type [UID 66]. This perspective requires also looking into the next datatype, the RecordSetStruct.

RecordStruct and RecordSetStruct

The data field of the RecordStruct datatype (imaginatively, if not particularly obviously, named NumberOfBytes-A-RecordData) is defined as an array of octets. In reality this is yet another variable datum structure. At first impression, the proposed combined datum structure (replacing VariableDatumStruct and FixedDatumStruct) cannot be shared since the Record Set contains an array of datum structures rather than a single datum structure. A variant record with the same discriminant but with fields consisting of arrays of datatypes would match the current design. If there was a desire to keep the length of the data structure then this could be included in the RecordSetStruct (since all records in a Record Set are the same). However, as argued for the merger of the VariableDatumStruct and the FixedDatumStruct, inclusion of extra data in the transmission, in this case the repetition of the RecordSetIdentifier, would prevent repetition of similar data structures in the object model. Due to also another field being present in the RecordSetStruct, the RecordSetSerialNumber, this potential merger needs to be investigated in more detail, including verification that with new datatype structures data can be translated back and forth in DIS / RPR FOM gateways for all applicable PDUs / events.

AttributeValuePairStruct

The value field (NumberOfBytes-A-Value) in the AttributeValuePairStruct is defined as an array of octets. Although theoretically it is possible to turn this into a variant record it is unlikely that this would be possible in practice. The issue is that the value field can represent any attribute value in the FOM making the definition of the data structure tedious and hard to maintain. Even worse, the discriminant is an attribute handle which isn't a fixed value, rather it is allocated by the RTI (although it is recommended in section 2.6 that this be turned into a string representing the FOM name). It is unlikely that this can be made into a more descriptive structure. However the RPR FOM uses this structure in the AttributeChangeRequest interaction (and the associated AttributeChangeResult reply) – this functionality could be useful to other federations and it may well be worthwhile adding this to a future HLA revision.

Silent Entity Appearance

The EntityAppearance field in the SilentEntityStruct is defined as an array of 32-bit unsigned integers. In reality the appearance is not a 32-bit integer, instead it is a bitfield, whose contents depend on the EntityType (sometimes in quite complex ways). The DIS appearance field has been split out into separate attributes in the RPR FOM (although the RPR FOM has not kept pace with additions introduced in SISO-REF-010). As proposed in section 2.4, different fixed records should be created for the possible appearance structures, matching their definition in SISO-REF-010. Then ideally the EntityType and EntityAppearance fields in the SilentEntityStruct should be combined in to a single variant record. However, currently HLA has no easy way of using a fixed record structure like the EntityTypeStruct as discriminant. An alternative would be to introduce a new enumeration to define the different possible appearance structures and use this as a discriminent.

DIS Version 7 Structures

On the fairly safe assumption that RPR 3.0 will contain structures representing functionality introduced in IEEE 1278.1™-2012 it will be important that no new opaque datatypes are introduced in the process. The most obvious candidate datatype that could be defined as an opaque datatype is the DIS Standard Variable Specification record. Inspection of this type shows that the proposed datum structure could be used for this field as well, since it shares the same enumeration as the discriminant (i.e. DatumIdentifierEnum32).

2.6 Representation of RTI data

In a few cases a reference to an HLA attribute is given, for example in the AttributeValuePairStruct used by the interaction AttributeChangeRequest. This is specified using four bytes representing the federation-wide encoded attribute handle. While this is the commonly used encoding, HLA allows for any size of attribute handles. A

better option would be to use the HLAhandle datatype that is already part of the standard MIM in IEEE 1516™-2010. This datatype is specifically designed for encoded handle values. The downside to this approach is that this value is only valid during a federation execution and cannot be easily decoded, for example, from a data recording. Another option is to use the FOM name of the attribute.

2.7 Complexity vs. Understandability

There are quite a few complex datatypes within the RPR FOM; the actual data represented by a basic, simple, or enumerated datatype may be found as deep as six levels down. For example, the category of an attached part is a field of the EntityTypeStruct, this store type is part of the AttachedPartsStruct, which is an alternative of the ParameterValueVariantStruct, in turn contained in the ArticulatedParameterStruct, and one or more elements of the latter structure are transmitted in an ArticulatedParameterStructLengthlessArray. This is an example of the structures of the datatypes closely matching the DIS standard, and likewise enabling a large flexibility in exchanging simulation data.

Spatial data structures

That other solutions can be found to representing the same data is obvious, and can also been seen in the changes from RPR FOM 1.0 to RPR FOM 2.0 with respect to representing entity spatial information. The seven individual attributes in RPR FOM 1.0, some of which are not needed for certain types of dead reckoning, have been replaced in RPR FOM 2.0 by one attribute using a variant record datatype. In this SpatialVariantStruct, the type of dead reckoning algorithm determines which of the nine alternatives applies, which use one of the five different fixed records defined for holding spatial data.

Is this too complex? Given that all five fixed records contain the WorldLocation, IsFrozen, and Orientation fields, the following approach could be used:

A fixed record is used that contains

- the fields WorldLocation (FixedRecord), IsFrozen (Boolean) and Orientation (FixedRecord);
- a variant record with the dead reckoning algorithm and optional velocity and acceleration information.

This may help first time users in starting to understand the RPR object model, as it makes explicit that basically ‘spatial’ is all about the location and orientation. It would also enable applications more convenient access to this basic spatial information.

On the other hand, such a structure bears the risk of initially leading to an incorrect understanding of one of the basic architecture concepts of DIS, and therefore also of the RPR FOM: the reduction of communications processing through dead reckoning. For the fields WorldLocation and Orientation are not supposed to be processed without inspecting the dead reckoning algorithm. With the exception of the static algorithm, the location and orientation are only valid at transmission time. Depending on the algorithm, the velocity and acceleration values must also be used to dead-reckon the location and orientation for the times between the transmissions of the spatial information.

With the currently defined dead reckoning algorithms the alternative structure does provide an optimization in terms of removing the repetition of three fields; in object-orientation terminology, through generalization into a parent structure. However, due to the characteristics of what is represented, it is still required to also process the variant record; in object orientation terminology, the parent structure is abstract, it cannot be used without its children. Unfortunately these objected orientation concepts can currently not be captured in an HLA object model. Hence the relationship between the spatial fields and the chosen dead reckoning model can only be captured in the semantics of or notes to the structures.

What is perceived as complex or assisting understandability depends on perspective and personal preference. The authors therefore call upon the readers to contribute to the discussion and provide their arguments and opinions as to whether the spatial structures should be kept as they are, or are candidate for improvement in RPR FOM 3.

One-field fixed records

There are a few fixed records in the RPR FOM which seem to have an unnecessary complexity as they only contain one field, or a second one only for padding: BreachStruct, GridValueType0Struct, and GridValueType2Struct; and the UniformGeomRecStruct even contains only a padding field. The conversion tool used to convert from HLA 1.3 to HLA 1516-2000 created a lot of additional fixed records with only one field. They were removed as part of the RPR FOM version 2 finalisation.

From an implementation perspective there is indeed no need to hide the data another level deeper. However, when considering the understandability of the RPR object model and a consequent naming scheme, it is preferable to keep these intermediate fixed records.

For example the BreachStruct can be understood from looking at its context. There are four kinds of linear environment objects: BreachableLinearObject, BreachObject, ExhaustSmokeObject, and OtherLinearObject. Apart from the latter, which has no attributes defined, these all have one array holding the data. The datatypes used for the other arrays, BreachableSegmentStruct and ExhaustSmokeStruct, do contain multiple fields.

Another example is the UniformGeomRecStruct, used in a variant record. The other alternatives in this variant record have a corresponding fixed record, albeit with multiple fields. If all explicit padding fields are removed for the HLA 1516 versions of the RPR FOM 3 (see section 2.8), and depending on the interpretation of the definition of this record in SISO-REF-010, this fixed record could even become empty. Since the HLA OMT allows to include alternatives for a variant record without defining name and datatype, it would then be advised to remove the UniformGeomRecStruct datatype, but do list the alternative UniformGeometryRecordType to prevent users from wondering whether this environmental process geometry record type has been forgotten in the RPR FOM.

2.8 Handling different HLA OMT versions

The HLA 1.3 OMT specification contains no rules about how proper byte alignment should be achieved in complex datatypes. As a result, FOM developers would explicitly define padding fields in their datatypes. The RPR FOM has always included such explicit padding fields where necessary.

In IEEE Std 1516™-2000, standard rules for handling byte alignment were defined. It was no longer necessary to explicitly define padding fields in your FOM, as the appropriate padding was implicitly required by the rules of the standard. This reduced clutter in FOM datatype definitions. In RPR FOM 2, however, all explicit padding fields have been maintained. This helped in generating the HLA 1.3 format of the FOM from the IEEE Std 1516™-2010 format, as it provided explicit instructions for the conversion tool to insert padding fields where necessary. In order to support this, two new datatypes were defined: RPRpaddingTo32Array and RPRpaddingTo64Array.

In RPR FOM 3, it is recommended that these padding fields be removed from the IEEE Std 1516™-2000 and IEEE Std 1516™-2010 formats of the FOM since they are redundant to rules defined by the standard. This will declutter the FOM, making it easier to read. It is proposed that any conversion tool used to generate the HLA 1.3 format of the FOM be updated to automatically insert explicit padding fields as necessary by following the alignment rules outlined in the IEEE Std 1516™-2000 and IEEE Std 1516™-2010 standards.

3. Impacts on Federates, Federations and future RPR FOM Development

The proposed modifications do not affect the semantics of the RPR FOM. However, any federate that uses these new datatypes will not be data buffer compatible with older federates. Since other parts of the RPR FOM are expected to change for RPR FOM 3, this may not necessarily be a major issue.

For new (and to some degree existing) developers of RPR FOM, these changes will make it considerably easier and less error-prone to develop federations, in particular with respect to data encoding.

It will be easier to learn the RPR FOM for developers with a basic understanding of HLA when the RPR FOM is better aligned with HLA datatypes.

3.1 Impact on future RPR FOM development

It is expected that several new concepts will be added to the RPR FOM in version three, for example directed energy weapons and information operations. It is beneficial to add any improved approach for representing data before these additions are developed.

4. Conclusions

This paper proposes a number of improvements to the RPR FOM datatypes for version 3. Some changes for future HLA versions are also discussed.

This would be a good time to revise the datatypes, since RPR FOM 2 development has been completed and RPR FOM 3 development is expected to start shortly.

It is expected that these changes will make the RPR FOM easier to use, extend and maintain.

It will also reduce risk for RPR FOM federation developers by reducing the use of non-standard datatypes and encodings.

References

[1] IEEE: "IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)", IEEE Std 1516-2010, IEEE Std 1516.1-2010, and IEEE Std 1516.2-2010, www.ieee.org, August 2010.

[2] SISO: "Real-time Platform Reference Federation Object Model (RPR FOM) Version 2.0D17", www.sisostds.org, September/October 2003.

[3] IEEE: "IEEE standard for, Distributed Interactive Simulation – Application Protocols", IEEE Std 1278.1-1995 and IEEE Std 1278.1a-1998, www.ieee.org, September 1995 and March 1998.

[4] SISO: "Reference for: Enumerations for Simulation Interoperability", SISO-REF-010-2015, www.sisostds.org, 17 March 2015.

[5] Björn Möller et al.: "RPR FOM 2.0: A Federation Object Model for Defense Simulations", 2014 Fall Simulation Interoperability Workshop, (paper 14F-SIW-039), Orlando, FL, 2014.

Author Biographies

BJÖRN MÖLLER is the Vice President and co-founder of Pitch Technologies. He leads the development of Pitch's products. He has more than twenty-five years of experience in high-tech R&D companies, with an international profile in areas such as modeling and simulation, artificial intelligence and Web-based collaboration. Björn Möller holds a M.Sc. in Computer Science and Technology after studies at Linköping University, Sweden, and Imperial College, London. He is currently serving as the vice chairman of the SISO HLA Evolved Product Support Group and the chairman of the SISO RPR FOM Product Development Group.

FREDRIK ANTELIUS is a Senior Software Architect at Pitch and is a major contributor to several commercial HLA products, including Pitch Developer Studio, Pitch Recorder, Pitch Commander and Pitch Visual OMT. He holds an M.Sc. in Computer Science and Technology from Linköping University, Sweden.

AARON DUBOIS is a principal software engineer at VT MÄK and is currently working on VR-Forces, MÄK's Computer Generated Forces application. Prior to that, he was the lead engineer for MÄK's interoperability products, including the MÄK RTI, VR-Link, VR-Exchange, and the MÄK Data Logger. He has been an editor of the GRIM since joining the latest RPR FOM drafting group in Fall SIW 2012.

PATRICE LE LEYDOUR is a system architect in the field of M&S for support to bids and projects at TTS (Thales Training & Simulation). His main interests are simulation interoperability using HLA (High Level Architecture), and C2-Simulation interoperability. During Fall SIW 2012 he volunteered as one of the FOM editors in the Drafting Group.

GRAHAM SHANKS is a technical manager for synthetic environments in BAE Systems. He has over 35 years' experience of simulation and has been involved in the development of DIS, HLA, RPR FOM and other IEEE and SISO standards. He was the FOM editor of the RPR FOM through until RPR FOM 2.0 draft 17 and most recently served as the editor for the updates to the IEEE 1278.1 and 1278.2 standards.

RENÉ VERHAGE is a software architect at CAE's office in Germany, with a focus on synthetic environment and simulator networking. Since his introduction to DIS and HLA in 1999, the topic distributed simulation and interoperability returned on his desk in the execution of many projects. Since 2011 he contributed to the finalization of the RPR FOM 2.0, acting as one of the FOM editors in the Drafting Group.

Appendix A: RPR FOM 2 Modular Structure

This appendix provides a graph of the RPR FOM 2 modules

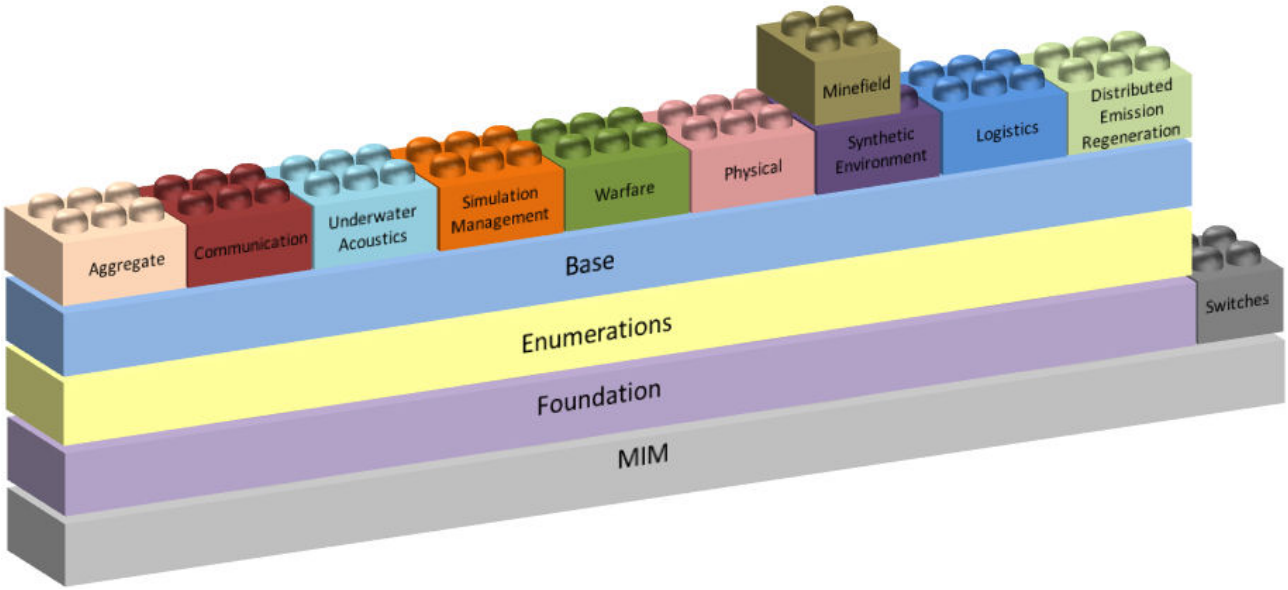

Figure 1: RPR FOM 2.0