# EXTENSIBLE EMISSION MODELING: A MEDIUM-CENTRIC APPROACH

**Mark L. Akey**
**Hughes Defense Communications Company**
**1010 Production Road 10-49, Ft. Wayne, IN 46808**

## ABSTRACT

The DIS Distributed Emission Regeneration (DER) protocol family, and specifically, the Electromagnetic Emission PDU, provides ample latitude and support to perform emitter, medium, and sensor interaction modeling in a symbolic frequency and time domains. However, the Emission PDU's complexity and format bias developers toward a platform-centric as compared to medium-centric approach. When new emissions, sensors, and media effects need to be added to an existing simulation, often this platform-centric approach forces modification of very tenured code.

This paper develops a coherent API (Application Programming Interface) using an object-oriented design that supports extensibility in emission modeling. Extensibility is achieved by coercing emission data into a medium-centric format. The design supports a client-server relationship between a given medium and its client sensors. Sensors are decoupled from direct association with an emission PDU; sensor modeling concentrates only on enumeration, detection, classification, and localization within beam patterns. Media, on the other hand, are directly associated with emission PDUs and carry the responsibility for target clustering and obscuration, line of sight and earth curvature, propagation loss, and emission jamming.

Extensibility of emission modeling is supported in three ways. First, modeling is extended via the typical class hierarchy -- new sensors are derived from base sensors, new media are derived from existing media. Second, the attachment of new sensors to media is straight-forward via the client-server (CS) relationship -- a natural extension of the CS paradigm. Third, and again using the CS relationship, processor load partitioning is spread across computer platforms without the modification of base code -- extending the life and utility of the existing code.

Finally, the API unifies within the medium the treatment of emission and dead reckoning (DR) based sensors. Regardless of the source, emission or DR, treatment of the medium effect is the same.

## BIOGRAPHY

Dr. Mark L. Akey received his BSEE, MSEE, and Ph.D. from Purdue University, West Lafayette, IN, in 1980, 1982, and 1985 respectively. He joined Hughes (formerly Magnavox) upon graduation helping to establish the Decision support systems Applied Center of Excellence (DACE) as a pivotal internal research and development group. To date, Dr. Akey has designed and implemented more than a dozen decision support systems and simulations, and has published dozens of technical articles and refereed papers.

# EXTENSIBLE EMISSION MODELING: A MEDIUM-CENTRIC APPROACH

**Mark L. Akey**
**Hughes Defense Communications Company**
**1010 Production Road 10-49, Ft. Wayne, IN 46808**

## 1.0 INTRODUCTION

The DIS Distributed Emission Regeneration (DER) protocol family, and specifically, the Electromagnetic Emission (EE) PDU, provides ample latitude and support to perform emitter, medium, and sensor interaction modeling in a symbolic frequency and time domains. Near-signal level information captured symbolically within the DER EE PDU describes frequency spectrum, bandwidth, and signature, pulse repetition rates, and beam patterns for an emitter system. To ease the computational burden of participating simulations, the sending (emitting) simulation computer determines which simulation entities are affected by the emission and packs their identities into the PDU. Finally, multiple emitter systems can be represented in a single DER EE PDU, effectively lowering the simulation bandwidth requirements between simulation systems. Thus, the DER EE PDU supports a functionally complete communication of emission information between simulation systems to support the regeneration of near-signal level emissions within the receiving simulation systems.

However, the DER EE PDU does not represent an emission-based simulation design on its own. It is simply a communication structure sufficient to convey emitter information. Too often, developers rush to embody PDU structures at the core of their simulation design without much forethought on the modeling extensibility of that decision. Further complicating a design's extensibility is the recognition that legacy code and simulation life-cycles carry a strong bias toward platform-centered development.

This last point can be seen in the manned flight simulator field. Many simulators are being upgraded to become DIS-compliant and to operate with other simulations/simulators. Beyond the entity state update PDU, how do these heterogeneous simulators interact with each other? Interaction is often accomplished through their respective sensors. But sensor simulation in the original simulators do not model the emissive environment sufficiently to perform this level of interaction. So developers create sensors that can, and directly employ the DER EE PDU as the basis for that interaction. The resulting emitter simulation design is typically not extensible nor scaleable across computing platforms due to this platform- or sensor-centric approach.

The purpose of this paper is to develop and present an emission-based simulation design that promotes extensibility of emission modeling. It is accomplished by first developing taxonomies for sensors and media, and then using these taxonomies to discriminate functionality between sensors and media. The design decouples platforms' sensors from direct response of emission PDUs on the DIS network. Coupling is accomplished through the media -- as it is done in the physical world. Extensibility is thus enhanced due to a natural partitioning of media and sensor functionality and APIs. In large part, new sensors and media must map to this partitioning and these taxonomies.

## 1.1 Modeling Requirements

Extensibility of emission modeling is the topic of this paper. Any discussion of this extensibility or how it is developed without a proper treatment of sensor modeling or the partitioning of sensor and medium modeling responsibilities would be remiss. This section presents such a partitioning of responsibilities, and in essence, provides taxonomies of both sensors and media modeling requirements.

**1.1.1 Sensor Modeling Effects.** Sensors do at least one of three gross activities well: detect, classify and locate (DCL)[1]. They perform these activities with the emissions presented to them via the medium in which they sense. Sensors perform complex N-dimensional analysis of these emissions over time and space to provide detection, classification, and localization[2].

**Detect.** Sensors have the ability to detect one or many emissions at a given time and in a given direction. This detection is accomplished in the midst of "noise" which can be comprised of other emissions or even the same emission. Often an emission is associated with a platform leading to a direct, but implied, relationship. For instance, detection of an acoustic subsurface signature implies the detection of a submarine. This implied association is important for a sensor to detect multiple entities or sources. For instance, does the receipt of two similar emissions imply single or multiple platforms? Without knowledge of the implied emission source, there can be no resolution of the possible source numbers.

**Classify.** Sensors have the ability to classify one or many emissions at a given time and in a given direction. Often the classification process subsumes the detection process when both processes use the same N-dimensional emission data. Detection and classification are instantaneous -- once an emission is detected it is also classified.

There are at least two cases where detection and classification exist as two separate processes. The first case involves insufficient computational resources to provide complete detection and classification in real-time. A coarse filter or process is applied to the N-dimensional emission data to detect a class of signals. A finer, more computationally intensive filter or process is then used to classify those emissions that pass the detection stage. The second case involves the use of "orthogonal" processes to discriminate the data -- the detection feature set does not overlap the classification feature set.

**Localize.** Sensors have the ability to localize one or many emissions at a given time. In all cases, localization is a function of a sensor receiver's kinematics relative to the emitting source, medium emission propagation delay times, and received emission power. Kinematics include position, aspect angle, and higher-order kinematics such as velocity (Doppler). Finally, localization is often improved by beam forming, finer range gating, time averaging of multiple signals and a knowledge of the emitting source.

To reiterate, sensors perform detection, classification, and/or localization (DCL) of emissions that are presented to them by their associated media at their receivers (antennae, arrays, etc.). Albeit gross, DCL provides a taxonomy of sensor functionality, and helps to guide the partitioning between sensor and medium modeling. The next subsection exposes medium modeling requirements.

**1.1.2 Medium Modeling Effects.** In its simplest form, a medium conveys information[3]. It may convey information from one or more sources to one or more receivers. In more precise terms, a medium is transmissive, reflective, and refractive. It has the ability to transmit 0% to 100% of the information emitted. In the case of 0% transmission, the propagation loss is complete; in the case of 100% transmission, the medium is lossless.

A medium is reflective -- emission power is reflected back toward the emitting source or scattered randomly (also viewed as propagation loss). Reflection is caused at the boundary of different media (air-earth, water-earth, air-water).

A medium is also refractive -- emission power incident upon a media boundary is bent as it enters the other medium. That is, some of the incident emission power is reflected back from the media boundary and some of the power is bent and transmitted through the other medium. With the exception of gravitational forces exerted by blackholes, all tactical medium modeling effects can be described in terms of a medium's transmissive, reflective, and refractive capabilities.

The following list summarizes the types of medium modeling effects required of tactical simulations.

**Line of Sight.** This effect is the most common requirement of tactical air simulations, and covers the RF, visual, and IR consequences of attempting to transmit power through rocks (terrain) and earth (earth curvature). Rocks have a very low RF transmission capability. Figure 1 illustrates line of sight effect with the earth's horizon.

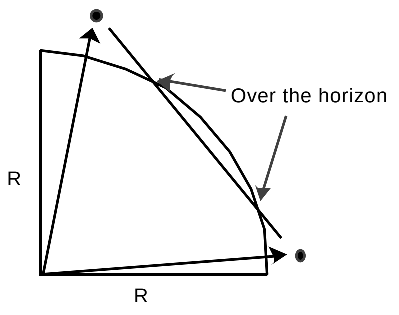

**Figure 1.** Line of sight example between two entities and the earth.

**Propagation Loss.** Second in the list of important medium model requirements is the ability to simulate propagation loss. Often this effect needs to consider slightly denser or less dense forms of the medium such as clouds (atmospheric) and ocean thermo-layers (acoustics). Here, a more nominal value of transmission is necessary.

**Clustering/Obscuration.** This effect is very similar to the line of sight requirement. For instance, an emitting source can be obscured by another platform from a receiver. Conversely, an emission reflection from an illuminated platform may be the emission that a receiver detects and not a direct line-of-sight emission from the source. Both transmissive and reflective properties can be important for emission simulation in these situations. Figure 2 illustrates the difficulty with discerning multiple entities within a sensor's beam.

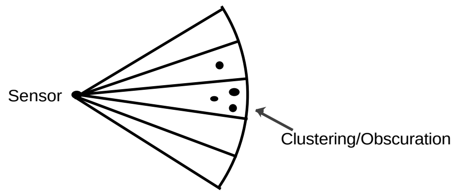

**Figure 2.** Difficulty with discerning more than one entity within a sensor's beam.

**Multipath.** Reflection of emissions from secondary surfaces cause both constructive and destructive interference at the receiver. Ghost reflections and misplaced primary reflections are a common sensor manifestation. Multipath reflections require the medium to understand all of its media boundaries. Transmissive, reflective, and refractive properties all come to play in modeling multipath. In general, this effect is very computationally intensive to model.

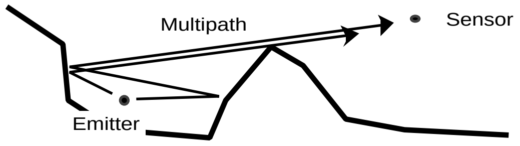

**Figure 3**. Multipath receptions at sensor caused by terrain or other media boundaries.

**Jamming.** Within a given medium, there is the potential for jamming. Jamming is the overloading of power at the receiver sufficient to obscure the detection, classification, and/or localization of desired emissions. The most commonly modeled jammers are in the RF domain. But jamming can occur between friendly platforms such as a wingman's radar interfering with ownship's RWR (much more common in earlier equipment). Another source of jamming is good ol' Sol in the IR and visual domains.

**Spectral Warping.** The previous effects deal with the presence or absence of an emission at a receiver and its total power. However, media have the potential to warp the relative frequency signature of an emission. In general, some frequencies pass through the medium with little power loss. Other frequencies sustain greater power losses. Water vapor is a great attenuator of microwave frequencies, for instance. Thus, it is possible for the medium to warp the spectral signature of an emission and do so in a non-linear fashion. These effects can be modeled within the current DIS architecture, however, as spectral resolution increases it becomes computationally expensive.

Summarizing, media effect the delivery of emissions to sensors. They can limit the receipt of emissions (transmissive), they can effectively re-position the source relative to the sensor (reflective and refractive), they can create multiple versions of a single emission with different incident angles upon the sensor (reflective/multipath), and they can non-uniformly attenuate an emission's spectral content. Again, the medium function taxonomy of transmissive, reflective, and refractive capabilities help to discriminate functionality between sensor and medium.

### 1.2 A Medium-Centric Approach

Most simulations cut their DIS sensor teeth on the Entity State (ES) PDU. The ES PDU is the mainstay of the DIS community; it brings a great deal of integration in a single fell swoop. Diverse simulated objects have the ability to interact with each other across the network. On-board platform sensors use local dead reckoning information to model DCL. For scenarios with continuous emissions, there is little need to implement and use the DER EE PDU. As far as first generation simulations go, most would deem "This is good".

However, there are a number of reasons this first generation interaction becomes inadequate. First, realistic scenarios include transient emissions and dynamically changing emissions. Sensor models that use only dead reckoning information quickly lose their realism. Second, there exists a dichotomy between the treatment of dead reckoning based and emission PDU based media effects. This dichotomy arises from the separate code modules from where the emissions are pulled. Third and due to this dichotomy, emission model extensibility is hindered.

A medium-centric approach mitigates these problems. A medium acts as a true medium by delivering emissions to the sensors with transmissive, reflective, and refractive effects. It also unifies the treatment of dead reckoning and EE PDU based emissions. With the taxonomies developed, extending the base treatment of emissions both by the media and sensors is simplified and manageable -- "This is better".

## 2.0 EXTENSIBLE ARCHITECTURE

### 2.1 Architecture

Figure 4 illustrates a high-level instantiation of a medium-centric architecture. The architecture maintains a modeling separation between medium and sensor. Media are responsible for transmissive, reflective, and refractive effects on emissions as they are presented to given sensors. Sensors are responsible for the DCL portion of the emissions that are presented to them.

Following a client-server (CS) paradigm, sensors subscribe to appropriate media. This subscription activity informs a given medium what sensors need emission information and the type of emission information they require: emission PDU based, dead reckoning based, or both. The medium maintains a coherent model of the emissive world using both emission PDUs and dead reckoning "emissive" data. Thus, there is no dichotomy in the handling of medium effects relative to "source" and extensibility is enhanced.

During simulation execution, medium processes interact with the DIS stream to filter out incoming emission PDU packets. Coupled with local dead reckoning and ground truth information, the medium model determines the resultant medium effect to the emissions. Subscribed clients to the medium process are passed the emission data either when they ask for it (synchronously) or when an appropriate emission PDU is received by the medium (asynchronously).

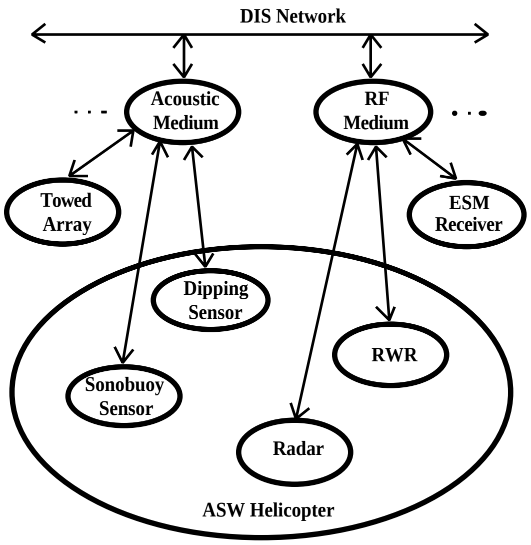

**Figure 4.** An instantiation of the medium-centric architecture

Finally, this architecture allows for distributed processing. There is no explicit requirement forcing the medium model to reside on the sensor model or platform model computer host. The architecture obviates the need to rewrite modeling code as the simulation models evolve and mature. This is especially important as medium and sensor modeling become more resolved; the processing requirements of these models often consume their computer hosts.

## 2.2 One-To-Many Mapping

When created, the emission PDU is a compendium of emissive information by the emitting entity platform. The PDU may contain information about multiple on-board emitters, where they are emitting (beam information), and who is being illuminated (track/jam entity identities). In short, the emission PDU is centered around the emitting entity -- a one-to-many mapping. It is the responsibility of receiving entities to unwrap the emission PDU and determine if they individually are being illuminated. Coining a soon-to-be overused term, the emission data must be made receiver-friendly -- a many-to-one mapping.

The architecture formalizes this transformation by moving the mapping responsibility to the medium. Incoming emission PDUs are received by a medium. The medium dissects the emission PDU, transforms the information into a receiver-friendly form, and passes this information on to the sensor. The vehicle for the emission data is called a Received Emission or RecEmission for short. The RecEmission contains all of the information relevant for a sensor to perform its DCL (detect, classify, located) functionality.

The RecEmission also contains information on the type of emission that it represents: emission PDU or dead reckoning. Dead reckoning based emissions are typically generated as a result of continuous or cyclic emission events. They do not appear on the DIS network as emission PDU; they must be regenerated from shared emission databases and entity state information. For example, submarine propeller and cavitation noise in the acoustic domain can be derived from entity velocity, perspective to "illuminated" entity, position, and positions of articulated parts. Here, there is no need to explicitly and cyclically emit emission PDUs. The advantage with this type of regeneration is lower consumption of DIS bandwidth; the disadvantage is that this regeneration falls into a grey area of the DIS standard. Aside from these concerns, the RecEmission contains much of the same information regardless of how it is created, via an emission PDU or dead reckoning. The obvious advantage is that medium effects are applied coherently to either form and the sensor model maintains the same interface to the medium. Thus, design and code extensibility is enhanced.

## 2.3 Media Functionality

It is the responsibility of the medium to determine what emissions are deliverable to its client sensors and how their content may be affected. Again, media capabilities are categorized as transmissive, reflective, and refractive. These capabilities, the emissions in the medium and other objects in the medium produce the major tactical media effects at the sensor:

- Line of Sight,
- Propagation Loss,
- Clustering / Obscuration,
- Multipath,
- Jamming, and
- Spectral Warping.

Minimal modeling requires positions of the sensor and the emitter. For instance, to model line of sight obscuration due to the earth's horizon, the model needs the locations of the emitter and the sensor as well as earth physical data. Propagation loss is usually handled as a uniform loss of power in dBm per meter; again the locations of the sensor and emitter must be known to determine effective power at the sensor receiver. Clustering and obscuration is also concerned with the beam pattern and/or resolution of the sensor. In this case, the medium must deal with a number of entities. Multipath is conceptually easy to model, but requires greater amounts of computational capabilities. Here again, knowledge of other elements in the environment is needed. Jamming is an overpowering of sensors' sensitivities or an increase in the background emission noise sufficient to lose emissions of interest. Tallying emission power at the point of reception requires a knowledge of all emissions within a sensor's beam. Spectral warping is simply a product of that summation spread over different frequencies with different attenuations.

The important point is that for medium modeling to perform realistically, for it to evolve and for the design and coding structures to evolve with it, the medium must be the focal point for all emissions before they filter down to the medium's client sensors. This architecture provides that necessary focus.

One final point: it is not necessary for an emitter to issue an emission PDU with the track/jam fields set. In this case, the emitter sets the high density flag within the PDU and sends it on its way. It then becomes the receiving model's responsibility to ascertain the simulation worth of this emission PDU. This architecture provides that mechanism via the medium model. At the highest fidelity levels, the medium model should ignore the track/jam entity identification and determine which client sensors should receive the emission PDU. A higher fidelity regeneration model, thus could operate with lower fidelity emitter models and produce simulation effects compliant with the higher fidelity.

## 2.4 Emitter Functionality

Up to now, little has been said about the emitter portion of the emitter-medium-sensor relationship. This is for good reason. DIS emission regeneration and this architecture require little of the emitter other than to submit a DIS emission PDU into the DIS stream. Obviously, the PDU must be constructed as mentioned in section 2.2, a non-trivial task. But the bulk of regeneration functionality is accountable at the receiving end by the medium and sensor.

## 3.0 APPLICATION PROCESS INTERFACE

Focusing on the interaction between the medium and the sensor, this section details an application programming interface. The term API is used loosely here. The definitions are versed in terms of C++ objects. Public methods are defined for the medium (Medium), sensor (Sensor), received emissions (RecEmission), and emitter (Emitter) classes. This is by no means a complete API between the sensor and the medium; it contains only the major functional interfaces.

### 3.1 Media Interfaces

The following public methods are part of the medium class.

**AddSensor(Sensor *s, Boolean async)**

The AddSensor method adds a sensor to the called medium. The medium checks the sensor for compatibility with the medium frequencies using the 'IsCompatible' method. It returns false if there is an incompatibility, else true. If successful, the AddSensor method places the sensor in its client list and returns true. If the 'async' parameter is true, then the medium will notify the sensor immediately of newly received emission PDUs.

**RemoveSensor(Sensor *s)**

The RemoveSensor method removes a sensor from the called medium. If emissions are pending, they are discarded.

**NewEmission(PDU *e)**

The NewEmission method is called by a local DIS engine which dispatches emission PDUs to the individual local mediums. From the medium's sensor client list, the medium checks each sensor for detectability with the new emission. If the emission is detectable, it creates a RecEmission and saves in it the pertinent information from the emission PDU for each sensor. If the sensor requests asynchronous update, then the medium notifies the sensor via the sensor's 'NewEmission' method.

**ScanMedium(Sensor *s)**

The ScanMedium method effectively scans the medium for contacts that the passed sensor can detect, classify, and localize. The method returns a list of RecEmissions that the sensor can "see" with its current beam(s). This method determines the tactical transforms on the currently stored RecEmissions for a given sensor. These transforms include line of sight, obscuration, propagation loss, multipath, jamming, and spectral warping. Note that the medium doesn't truly limit their arrival at the sensor. The RecEmissions are tagged according to the transpired effects. This permits the modeling of clustered entities at the sensor or composite burn-through in the midst of broadband jamming.

**IsCompatible(Sensor *s)**

The IsCompatible method determines if the passed sensor can sense within this medium via an intersection of the sensor's and medium's frequencies. If there is overlap, it returns true.

**IsEmitterCompatible(Emitter *e)**

This method determines if the passed emitter can emit with this medium via an intersection of the emitter's and medium's frequencies. If there is overlap, it returns true.

### 3.2 Sensor Interfaces

The following methods isolate the major interfaces with the medium. They do not address the more general problem of what to do with RecEmissions when the sensor receives them, i.e., DCL.

**CanTrack(RecEmission *r)**

The sensor's 'CanTrack' method is called by the medium to determine if this RecEmission can be tracked by the sensor. Note that this is a coarse filtering; frequency compatibility between the sensor and RecEmission is checked. The method return true if compatible, false otherwise.

**CanBeJammed(RecEmission *r)**

The sensor's 'CanbeJammed' method is called by the medium to determine if this RecEmission can jam the sensor. Note that this is a coarse filtering; frequency compatibility between the sensor and the RecEmission is checked. The method returns true if compatible, false otherwise.

**EmissionKind(void)**

The 'EmissionKind' method is called by the medium to determine what type of emission regeneration is suitable for this sensor (emission PDU or dead reckoning based). Returns an enumerated type.

**Scan(void)**

The 'Scan' method is called by the sensor with its update routine or as part of medium notification of a new emission. In either case, the method calls the medium's 'ScanMedium' method to get a list of RecEmissions that are available to the sensor. It is within this base method that a sensor must perform the tasks of detection, classification, and localization.

**NewEmission(void)**

The 'NewEmission' method is called by the medium when a new relevant emission PDU is received by the medium. This method prompts the sensor to perform a scan.

**CueSensor(void)**

This method is called by the medium to determine the sensor's current beam patterns and directions.

### 3.3 RecEmissions

The following methods pertain to the RecEmission class and are simply accessor methods. All information is set by the medium in the 'ScanMedium' method.

**BeamIndex(void)**

The 'BeamIndex' method returns the index of the sensor's beam that this RecEmission was detected.

**IsColinear(void)**

The 'IsColinear' method returns true if this RecEmission exists behind another RecEmission that is closer to the sensor and within the same beam.

**IsJammed(void)**

The 'IsJammed' method returns true if this RecEmission is being jammed via an emitter.

**ReceivedPower(void)**

The 'ReceivedPower' method returns the raw emission power of the RecEmission available at the sensor.

**SNR(void)**

The 'SNR' method returns the signal-to-noise ratio of the RecEmission in the midst of other emissions within the sensor's beam. This value is an average across all RecEmission's frequencies within the beam.

## 4.0 USING THE API

### 4.1 Walk-Through Example

A walk-through of the functionality using the API is often insightful. For discussion, assume a radar sensor and an RF medium. Radars are active participants in the RF environment; this one also has an emitter. The radar is part of a ground site; there are numerous aircraft in the area.

To begin, the radar requests service from the RF medium (AddSensor). The medium replies that frequencies are compatible and the sensor is made a client. Next, the radar scans (Scan) the horizon. A list of aircraft contacts are returned to the sensor. The contacts or RecEmissions are created by the RF medium. Using the dead reckoning (DR) database, a candidate RecEmission is created for each DR entity. Within the ScanMedium method, the medium uses these candidate RecEmissions to probe the sensor's detectability of the entity via the sensor's CanTrack method. The medium also calls the sensor's CueSensor method to determine the sensor's direction and if the candidate RecEmission is physically detectable (line of sight, etc.). The sensor uses the returned contact list to further model the DCL portion of the sensor.

Up in one of the aircraft, an RWR (radar warning receiver) is notified by an RF medium (though not necessarily the same RF medium object that the radar uses) of a new emission (NewEmission). The RWR scans (Scan) and receives a single RecEmission originating from the ground-based radar. Using the same protocol as the radar, this contact is filtered against the RWR's CanTrack and CueSensor methods.

Both sensors use the same medium; each sensor is supplied emissions from that medium. The radar sensor receives internally generated RecEmissions; the RWR receives emission PDU supported RecEmissions. The protocol is identical; the logic is clear. The ability to extend the design is straight-forward.

### 4.2 Extensions

Now consider a simple sensor extension. A new class of radars is created called ISAR (inverted synthetic aperture radar). It has the ability to image ground and surface targets from the air, but it has a much smaller beam pattern. To create this new functionality, the ISAR class has two new methods, CanTrack and CueSensor, which override the functionality defined in a common radar. When the medium calls the ISAR's CanTrack on ground-based targets, it will return true. On air-based, it will return false. When the medium determines if the candidate RecEmissions can be physically seen (line of sight, beam patterns, etc.), it will call the ISAR's CueSensor method. As a result, the ISAR illuminates much less volume than a common radar, but provides a greater amount of resolution detail for that smaller volume.

The same approach to extending base functionality can also be applied to the media. Assume a frequency range of RF that supports over-the-horizon travel. To create this new medium model, a new class is created that includes a new ScanMedium method. This method leverages from the base ScanMedium and augments it with over-the-horizon modeling. All other methods remain the same; they simply perform with a potentially greater list of contacts.

The use of an object-oriented language aids this extensibility. But it is the architecture and design that makes this level of extensibility possible. Without overriding method capability, these extensions could still be achieved via a number of approaches: table look-up, function pointers, etc.

## 5.0 SUMMARY

This extensible architecture supports emission modeling within the DIS framework. It provides extensibility via 1) classic object-oriented design of the real world elements; 2) the client-server relationship between the sensor and medium; and 3) model partitioning that allows distributed processing. The approach is based on coercing the emission data into a medium-centric format as compared to a platform-centric format. And finally, the isolation of medium and sensor taxonomies provides a base functionality which new sensors and mediums can use.

This architecture has been used extensively in Hughes over the past 4 years. As a testament to its extensibility, the base sensor and medium code has not been modified since 1994. Since that time, medium and sensor code extensions account for three times the size of this base code.

## 6.0 REFERENCES

[1] M. Schwartz and L. Shaw, Signal Processing: Discrete Spectral Analysis, Detection, and Estimation, McGraw-Hill, New York, 1975, pp. 148-264.

[2] G. Walter, Wavelets and Other Orthogonal Systems with Applications, CRC Press, 1994, pp. 93-114.

[3] D. Halliday and R. Resnick, Physics, John Wiley and Sons, New York, 1966, pp. 1013-1146.