Simulating Information Warfare
Using the HLA Management Object Model

Roger Smith
BTG Inc.
Orlando, Florida 32765
smithr@modelbenders.com

Keywords:
Information Operations, High Level Architecture, Management Object Model, Military Simulation

ABSTRACT: *This paper describes concepts and techniques for implementing Information Operations within a distributed simulation environment supported by the High Level Architecture. The HLA Management Object Model provides services that are specifically applicable to the types of operations that IO can have on combat objects. The paper provides a functional blueprint for creating an IO federate that can operate in any HLA federation.*

1. Introduction to Information Operations

The terms "information operations" and "information warfare" are relatively new descriptions for military actions. But, the concepts behind the terms are very ancient. These operations refer to various techniques for controlling or influencing the knowledge, perception, judgement, and decision making of an enemy force. Prior to the electronic age "information operations" may have been accomplished through rumors, torture, and intercepting couriers. More modern applications of this type of warfare include public affairs, civil affairs, physical destruction, electronic warfare, and computer network defense and attack [4].

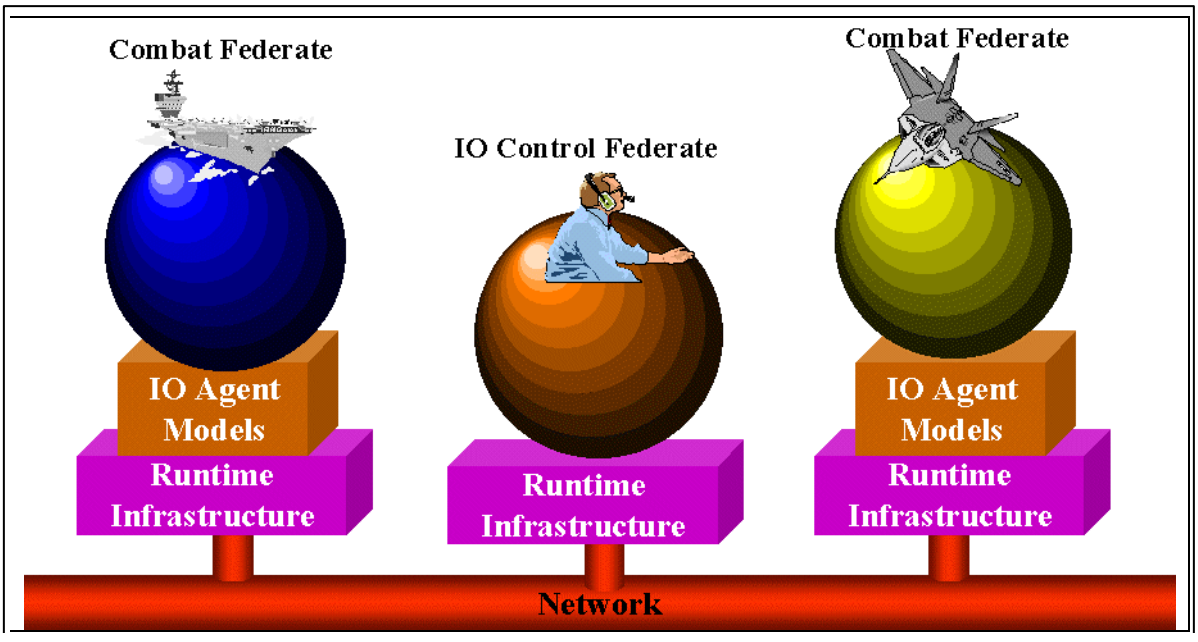
Information operations vary from the very obvious and unclassified to the very protected. The concepts presented in this paper are based on unclassified operations and the abstraction of techniques that are very well known in military operations. The mechanisms used to implement these operations may also be useful for more modeling more exotic forms of IO.

2. Applications in Interoperable Simulations

Simulating IO presents very different challenges in a single system and than it does within a distributed system. In a single system, is possible to place IO models in the software such that they access and interact with all objects and events in the simulation. In a distributed simulation it is much more difficult for any IO model to access and impact objects and interactions generated and consumed by different simulations all over a network.

When adding an IO simulation to an existing federation of simulations it is very difficult, or impossible, to modify all of the federates to enable IO impacts on their operations. Previous papers have presented techniques that attempt to accomplish this with minimal impacts on the receiving federates [3]. Those techniques were very intrusive to the existing simulations or they did not guarantee that IO operations would be recognized by the target system.

Since all DOD simulations are becoming HLA-compliant it is desirable to design an IO modeling technique that can



be applied to all simulations that are using the HLA Run-time Infrastructure (RTI). The HLA interface specification defines a wide variety of services that are available to support distributed, interoperable simulations [5]. The Object Management services are used to exchange information among federates. These assume that each federate has been programmed to respond to the combat effects delivered from remote simulations through these services. For most simulations, the actions supported fall into categories like Dynamics (physical movement), Exchange (combat), Electronic Interaction (Communication), and other similar interactions. However, most simulations were constructed without an appreciation of the impacts that IO could have on the simulated objects and, therefore, omitted any mechanisms to respond to IO intervention. These models are impervious to effects of IO.

One recourse is to cast IO in terms of one of the known combat interaction types. It is possible for an IO deception to be modeled as the delivery of misleading communications messages. It is also possible to represent electronic warfare as the delivery of RF noise through the services and FOM objects used for standard combat.

However, some IO operations are more subtle and can not accurately be cast as combat interactions. It is desirable to expand the number of options available for modeling these IO effects. Additional, and hopefully more flexible, ways of representing IO in distributed simulation should support more modern forms of combat in training and analytical applications. This would forestall the tendency to distort IO to fit into existing combat modeling interactions.

It may also be necessary or desirable to add IO to a federation without imposing unrealistic requirements on the definition of the FOM. IO effects may not be recognizable by many combat federates that are being targeted. The very character of real IO is that it enters the battlefield independent of those that it targets and without requiring their compliance to be effective.

IO attempts to stand between a combat object and its access to information. The purpose is to deny, delay, or distort the information accessible to the combat object. A simulation technique that could place the IO simulation conceptually between the combat object and its information source would be ideal. It is impractical or impossible for the IO software to be inserted into a federation such that it is physically between the combat object and its information flow. Therefore, we are searching for ways to model IO such that it is conceptually in that position for any combat object in the federation [3].

3. High Level Architecture Management Object Model

The High Level Architecture is the mandated architecture for bringing simulation systems together into an interoperable federation. This is accomplished through the imposition of a set of HLA Rules, an Interface Specification, and the Object Model Templates. The interface specification provides services that allow simulations to exchange data in a managed and consistent manner. These HLA services support a wide variety of operations as defined in [5]. One class of these services is known as the Management Object Model. The HLA Interface Specification document introduces these with:

> "Management object model (MOM) facilities can be used by federates and the RTI to provide insight into the operations of federates and the RTI and to control the functioning of the RTI, the federation, and individual federates. The ability to monitor and control elements of a federation is required for the proper functioning of the federation execution."

The MOM services are specifically intended to allow one federate to "control the functioning of … individual federates". These services were envisioned as being invoked by a federate charged with the responsibility of managing and monitoring the performance of the federation. However, they are also accessible to simulation federates, like an IO federation or operator terminal. The MOM provides services that allow any federate to access and influence the information available to any other federate. In this sense they are performing the same types of functions carried out under IO missions.

4. MOM Services Useful for Information Operations

Certain MOM services could be combined to support IO simulation that is effective against all combat objects in the federation. These services could be used without imposing unreasonable demands on the FOM that is developed for a combat-oriented federation. To demonstrate how such an IO federate could be created, we will focus on the RTI operations that would be performed, and will omit discussions about specific modeling software that could represent IO effects.

The HLA MOM is structured around services that could be organized into an IO package as shown in Table 1. In this table we have identified services that must be invoked to begin a specific IO function. An operational IO federate would also invoke services from the "standard" categories of the HLA interface specification and would make use of the reporting services that are triggered by

the services listed below. The detailed use of those supporting services is not described in this paper to allow us to focus on the core IO effects we would like to insert into a federation. More details on those services can be found [5].

Table 1. MOM Services Useful to an Information Operations Simulation

MOM Service	Description of Operation
RequestPublications	Request the publication data of a federate.
RequestSubscriptions	Request the subscription data of a federate.
RequestObjectsOwned	Request a list of objects owned by a federate.
PublishObjectClass	Cause a federate to begin publishing an object attribute
UnpublishObjectClass	Cause a federate to stop publishing an object attribute
PublishInteractionClass	Cause a federate to begin publishing an interaction
UnpublishInteractionClass	Cause a federate to stop publishing an interaction
SubscribeObjectClassAttributes	Cause a federate to begin subscribing to object attributes
UnsubscribeObjectClass	Cause a federate to stop subscribing to object classes
SubscribeInteractionClass	Cause a federate to begin subscribing to interaction classes
UnsubscribeInteractionClass	Cause a federate to stop subscribing to interaction classes

These twelve services can be used by an IO federate to influence the information that is received by or transmitted from any federate in the simulation. Combining these with software models or operator actions it is possible to stop, redirect, delay, or alter the content of information delivered to a federate. The MOM describes many other services can could be used for different types of Information Operations. This list is intended simply to demonstrate the feasibility of this approach.

Table 2. MOM Services for Preparing IO Federate Actions

Mom Service	IO Representation
RequestPublications	Acquire a list of all object classes, attributes, and interactions being published by the combat federate to be targeted by IO.
RequestSubscriptions	Acquire a list of all object classes, attributes, and interactions being subscribed to by the combat federate to be targeted by IO.

5. Conceptual IO Simulation Application

To illustrate how the above services could be packaged into an IO federate we will provide examples of actions that can be carried out with these services. For the purposes of illustration and clarity we assume that the IO federate is a GUI-based tool that allows human operators to specify the actions that should be taken, but which hides the complexity of invoking the HLA services. This will allow us to de-couple the implementation of IO in a distributed federation from the details involved in creating IO software models to invoke these actions automatically. It also allows us to describe these actions in very generic terms that do not imply real IO capabilities which may be protected information.

All of the operations described below begin by invoking the MOM services *RequestPublications*, *RequestSubscriptions*, and *RequestOwnedObjects*. The RTI will deliver the requested information to the IO federate through the "Report" service corresponding to each of the "Request" services listed above [5]. These services allow the IO federate to construct a perception of the objects and interactions that are published by each federate, subscribed to by each federate, and the objects owned by each federate. This information can then be presented to a human operator to allow him/her to plan and execute operations against specific objects and federates. The IO federate may use standard subscription mechanisms to maintain current attribute values for each of the objects discovered through the IO use of the MOM services. This functionality is included in many existing HLA products that implement plan-view displays and stealth viewers. There is no need to implement this differently for IO federates.

5.1 Deny Information

IO can be used to deny the enemy access to information. This may be done through techniques like electronic warfare or the destruction of command units.

Upon understanding which objects and interactions are being received by each federate, the IO operator may designate which of these could be prevented through the application of a specific IO asset or technique. In response to the operator's desire to stop information flowing to a specific combat federate, the IO federate would invoke *SubscribeObjectClassAttributes* or

UnsubscribeInteractionClass on behalf of the combat federate being influenced. The arguments passed to the RTI from the IO federate are the designator of the combat federate and the attributes of an object class to which that federate will now subscribe. To deny access to an attribute the IO federate would specify a list of attributes that omits the one or more that IO has the ability to deny the receiving combat federate. Unsubscribing to an attribute is accomplished by re-specifying the list of attributes desired and leaving out the ones that are to be "unsubscribed". For interactions the RTI MOM provide a specific service for unsubscribing to an entire class. The IO federate's Local RTI Component (LRC) will deliver this command to the LRC of the combat federate. The LRC will change its subscription appropriately, based on the command it has received. The fact that the command did not actually originate from the simulation models served by this LRC does not prevent the command from being accepted and acted upon.

These types of actions can be captured by any recorder or analysis software that is logging distributed simulation interactions. This allows exercise analysts to identify that IO assets are being employed and against which targets. This traceability is important in insuring that IO is induced responsibly and that the training audience is debriefed on the causes of combat events experienced. When the IO action is terminated the IO federate is responsible for reversing the actions taken against the federates subscriptions. This is accomplished through the invocation of the *SubscribeObjectClassAttributes* and/or *SubscribeInteractionClass* services and specifying the original attribute list or interaction class that the combat federate was subscribed to.

The disadvantage of this approach is that it can not be used selectively against specific instances of objects within the combat federate while leaving the subscriptions of other objects in the federate intact. This seems to be an excessive application of the technique. It subjects all of the objects modeled by the federate to the same level of IO intervention. If the federate is a manned flight simulator controlling a single aircraft, this may be very appropriate. But if the federate is a SAF which controls multiple objects distributed around the battlefield, this technique does not have the fidelity to accurately represent the effects of IO.

A related IO effect can be imposed by invoking the *UnpublishObjectClass* or *UnpublishInteractionClass* on behalf of the federate that is publishing this information. An IO technique may be so dominant on the originator of information that this approach would be more appropriate. A major disadvantage of this is that all system data collectors also lose access to the information. This can negatively impact the ability of control personnel to monitor the progress of simulation objects. Note that similar care must be taken to specify the attributes of objects that are being published and unpublished.

Table 3. MOM Services for IO Denial of Information

Mom Service	**IO Representation**
UnsubscribeInteractionClass	Blind a federate to an entire class of interactions. e.g. Electronic Warfare jams receipt of all radio transmissions.
SubscribeInteractionClass	Discontinue IO action against the federate interactions. e.g. Termination of Electronic Warfare against radio transmissions.
SubscribeObjectClassAttributes	Change the list of class attributes to which a federate is subscribed. e.g. Electronic Warfare jamming blinds federate to aircraft locations.

5.2 Delay Delivery

An alternative to simply cutting a federate's access to all objects or interactions of a specific class is to delay the delivery of information for some amount of time. To accomplish this the FOM must contain two varying representations of the same object or interaction class. These pairs contain predominantly the same information, though some additional attributes or parameters may exist to assist in implementing IO.

Delay of delivery of information begins with the same "unsubscription" services described above. Once disconnected from its original information source, the combat federate would be ordered to subscribe to the related object or interaction class that was built into FOM for that purpose. This would be accomplished through the *SubscribeObjectClassAttributes* and *SubscribeInteractionClass* services and specifying the alternate object attributes or interaction that was placed in the FOM to accomplish this redirection. The IO federate would subscribe to the original object or interaction class that the combat federate was removed from. The IO federate is then in control of the data flow needed by the combat federate. It would republish the data it receives from the original object or interaction into the alternate object or interaction that the combat federate was directed to listen to. The IO federate would impose a delay time on the retransmission of the data to the combat federate. This would cause the object attribute or interaction to be received at "time constrained" federates at the appropriate

Table 4. MOM Services for IO Delay of Information

Mom Service	IO Representation
UnsubscribeInteractionClass	Disconnect a combat federate from its original source of interactions.
SubscribeInteractionClass	Reconnect the combat federate to the IO generated interactions.
SubscribeObjectClassAttributes	Disconnect the combat federate from its original source of object attributes. Then connect the combat federate to the IO generate object attributes.

point in the future. However, enforcing the delayed receipt of information at a federate that is not time constrained would require the IO federate to hold onto the captured information for the desired delay time.

One of the benefits of this approach is that it allows the IO federate to distinguish between individual object instances being published and to apply IO techniques only to those that should be effected. A disadvantage is that it places the IO federate in the data flow path of all of the objects on the combat federate being affected. This would often result in additional network traffic for the objects or interactions that are being retransmitted. The disadvantage of impacting all objects on the targeted federate remains.

5.3 Deceive Content

Corrupting Content

Deceiving federates about the content of information would be accomplished through the delay technique described above with additional logic in the IO federate. Once the object or interaction information had been redirected such that it was being brokered by the IO federate, the IO federate will modify the information that it is controlling. These modifications may be carried out from an operator window or by an IO software model. The use of a human operator would obviously add noticeable delay to the delivery of the modified information.

The disadvantage of this approach remains that it imposes the same degree of deception on all objects within the combat federate that is being affected.

Injecting Misleading Content

Deception operations can also be carried out by simply publishing additional object instances or interactions that are already described in the FOM. An IO federate may be responsible for representing radar reflectors, chaff clouds, and physical decoys on the battlefield. All of these may draw combat objects into engagements with them rather than with real assets. An IO federate may also publish communications or radar emission interactions deceiving the enemy into acting on incorrect or misleading data. These operations do not require the use of the MOM, but can be accomplished through the "standard" HLA services.

6. Conclusion

This paper presents some of the first concepts for implementing information operations in a distributed, interactive simulation. These ideas are in their formative stages and are presented here to demonstrate avenues that can be pursued immediately by an organization that is interested in adding IO effects to an HLA federation.

The disadvantages listed in each section are indications that additional thought and experimentation are needed. At this point in time it is not clear that a military service or government agency is interested in bringing IO into the distributed simulation domain. Our hope is that papers of this type will generate discussion on the topic and attract organizations that would benefit by exploring these areas.

7. References

[1] Allen, Patrick D. and Demchak, Chris C. "The Need for, and Design of, an IO-ISR Federation of Simulations". *2000 Spring Simulation Interoperability Workshop*. March 2000.

[2] Haeni, Reto E. "Information Warfare: An Introduction". White Paper, George Washington University. January 1997.

[3] Smith, Roger D. "Information Operations in Training Simulation". *2000 Spring Simulation Interoperability Workshop*. March 2000.

[4] Waag, Gary L. and Loental, Dave. "Information Operations M&S: An Overview of Recent Activities and a Role for Standards". *2000 Spring Simulation Interoperability Workshop*. March 2000.

[5] U.S. Department of Defense. "High Level Architecture Interface Specification". http://hla.dmso.mil/ . April 1998.

Author Biography

ROGER D. SMITH is the Technical Director for BTG Inc. working on next-generation simulation applications

and technologies. He is also the creator and instructor for a series of military simulation courses that have educated hundreds of simulation professionals. He is the Area Editor for Distributed Simulation for *ACM Transactions on Modeling and Computer Simulation* and has just completed his term as Chair of the ACM Special Interest Group on Simulation.