ERDC/CRREL TR-15-11

US Army Corps of Engineers®
Engineer Research and Development Center

ERDC
INNOVATIVE SOLUTIONS for a safer, better world

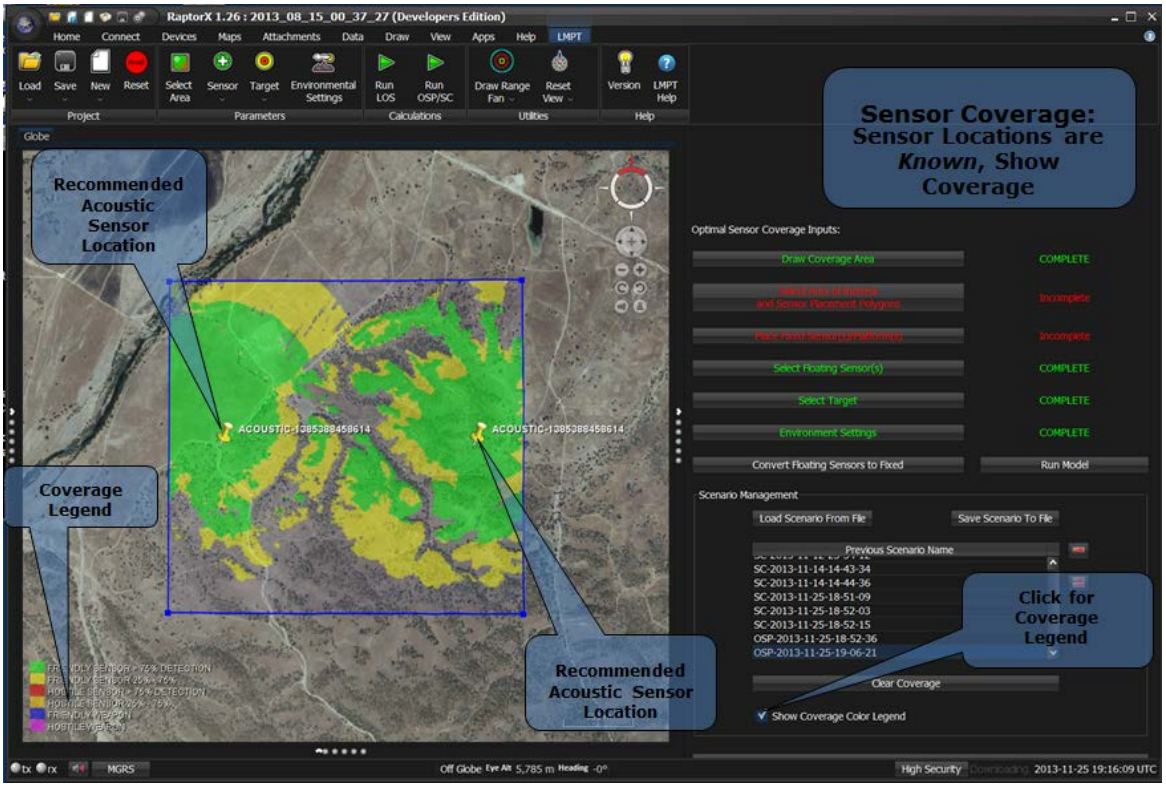
Geospatial Research and Engineering

Development of Integrated Software Capabilities for Battlefield Signal Modeling and Force Protection

Leaders Mission Planning Tool (LMPT)

Kenneth K. Yamamoto, John J. Gagnon, Donald G. Albert, and D. Keith Wilson

July 2015



Cold Regions Research and Engineering Laboratory

The U.S. Army Engineer Research and Development Center (ERDC) solves the nation's toughest engineering and environmental challenges. ERDC develops innovative solutions in civil and military engineering, geospatial sciences, water resources, and environmental sciences for the Army, the Department of Defense, civilian agencies, and our nation's public good. Find out more at www.erdc.usace.army.mil.

To search for other technical reports published by ERDC, visit the ERDC online library at http://acwc.sdp.sirsi.net/client/default.

Development of Integrated Software Capabilities for Battlefield Signal Modeling and Force Protection

Leaders Mission Planning Tool (LMPT)

Kenneth K. Yamamoto, John J. Gagnon, Donald G. Albert, and D. Keith Wilson

Cold Regions Research and Engineering Laboratory (CRREL)
U.S. Army Engineer Research and Development Center (ERDC)
72 Lyme Rd.
Hanover, NH 03755-1290

Final Report

Prepared for U.S. Army Corps of Engineers
Washington, DC 20314-1000

Under AT42 GRE, "Exploiting Sensing for Patterns—Environmental Awareness for Sensor and Emitter Employment (ESP EASEE)"

Abstract

With increasing resource constraints in the Department of Defense, it is becoming critical to develop technologies that unburden small and minimally equipped teams of Soldiers carrying out a mission. As part of the Deployable Force Protection (DFP) program by the U.S. Assistant Secretary of the Army for Acquisition, Logistics, and Technology, ASA(ALT), we developed the Leaders Mission Planning Tool (LMPT) to help Soldiers with little specialized training in sensor technologies effectively protect combat outposts by optimizing selection and placement of limited sensor resources. This product is the result of collaborating with Night Vision and Electronic Sensors Directorate, interacting with DFP sensor teams, and incorporating Soldier feedback from multiple test exercises and demos. It is a powerful and intuitive user interface for the Environmental Awareness for Sensor and Emitter Employment (EASEE) computational engine (which runs on a standard laptop) designed by the U.S. Army Engineer Research and Development Center (ERDC) for battlefield signal modeling. This report discusses the role of battlefield signal modeling for effective sensor use, rationale and approaches for using markup file exchange to interface multiple software applications, and the capabilities and features of LMPT. Details about the EASEE markup interface are documented for collaborating software developers.

Contents

Illustrations

Figures

Preface

Funding for the work was primarily provided by the Deployable Force Protection (DFP) program by the Office of the Assistant Secretary of the Army for Acquisition, Logistics, and Technology, ASA(ALT). Some technical developments were funded by the U.S. Army Engineer Research and Development Center (ERDC) Geospatial Research and Engineering (GRE) program under Project AT42, "Exploiting Sensing for Patterns—Environmental Awareness for Sensor and Emitter Employment (ESP EASEE)," including modeling of radio-frequency transmission and monostatic radar scenarios and specification of directional patterns for emitter radiation and sensor fields of view.

The work was performed by Kenneth K. Yamamoto, Dr. Donald G. Albert, and Dr. D. Keith Wilson (Signature Physics Branch, Dr. Loren Wehmeyer, Acting Chief) and John J. Gagnon (Engineering Resources Branch, Jared Oren, Chief), U.S. Army Engineer Research and Development Center (ERDC), Cold Regions Research and Engineering Laboratory (CRREL). At the time of publication, Dr. Loren Wehmeyer was Chief of the Research and Engineering Division; and Dr. D. Randy Hill was the Technical Director for Geospatial Research and Engineering. The Deputy Director of ERDC-CRREL was Dr. Lance Hansen, and the Director was Dr. Robert Davis.

The Principal Investigator of the Leaders Mission Planning Tool (LMPT) was Samuel Sweatt of Night Vision and Electronic Sensors Directorate (NVESD). The Technical Lead Investigator was Dr. Donald Albert of ERDC-CRREL. Members of the NVESD team for the development of the LMPT graphical user interface and training material were Scott Hansen, Joe Kaplan, Jacob Bowman, Eric Slominski, Turk Little, and Keith Madden.

The authors thank the Office of ASA(ALT) for funding the development of LMPT under the DFP program. The authors greatly appreciated the leadership of the DFP program, including Dr. Jerry Ballard and Alex Baylot, for their help in guiding the project as leads of the DFP Science and Technology line of effort and Mike Cahill for providing direct and clear perspective on Army needs and appropriate direction for the project.

The authors would also like to thank many whose help and advice contributed to the success of the project, including all personnel involved in setting up and running the annual DFP demonstrations at Fort Polk, LA, and Fort Benning, GA, and the Technical Support Operational Activity events at Camp Roberts, CA; other DFP technical teams for providing information about their sensors for incorporation into LMPT; and developers of Raptor and the Integrated Sensor Network for guidance on using their products.

The authors would especially like to thank the many Soldiers who tested LMPT and provided extremely valuable feedback during its development. LMPT evolved into a much better product because of their input. Special thanks go to MAJ Chris Ingenloff for providing expert perspective on the utility of LMPT for military applications.

The authors would like to acknowledge Nathan J. Reznicek and Nicholas J. Reznicek for contributing in the software development work and Michael T. Ekegren for helpful feedback during the later stages of the project. The underlying computational battlefield signal-modeling software, Environmental Awareness for Sensor and Emitter Employment (EASEE), originated from previous work led by Dr. D. Keith Wilson.

LTC John T. Tucker III was the Acting Commander of ERDC, and Dr. Jeffery P. Holland was the Director.

Acronyms and Abbreviations

ARL	U.S. Army Research Laboratory
ASA(ALT)	Assistant Secretary of the Army for Acquisition, Logistics, and Technology
CRREL	Cold Regions Research and Engineering Laboratory
DFP	Deployable Force Protection
DOD	Department of Defense
EASEE	Environmental Awareness for Sensor and Emitter Employment
EMPIRE	Electromagnetic Propagation Integrated Resource Environment
ERDC	U.S. Army Engineer Research and Development Center
ESP	Exploiting Sensing for Patterns
GFPE	Green's Function Parabolic Equation
GIS	Geospatial Information System
GPS	Global Positioning System
GRE	Geospatial Research and Engineering
HMMWV	High-Mobility Multipurpose Wheeled Vehicle
I/O	Input/Output
JAXB	Java Architecture for XML Binding
JSON	Java Script Object Notation
KMZ	Zipped Keyhole Markup Language File
LMPT	Leaders Mission Planning Tool
MODTRAN	MODerate Resolution Atmospheric TRANsmission
MUSES	Multi-Service Electro-optio Signatures

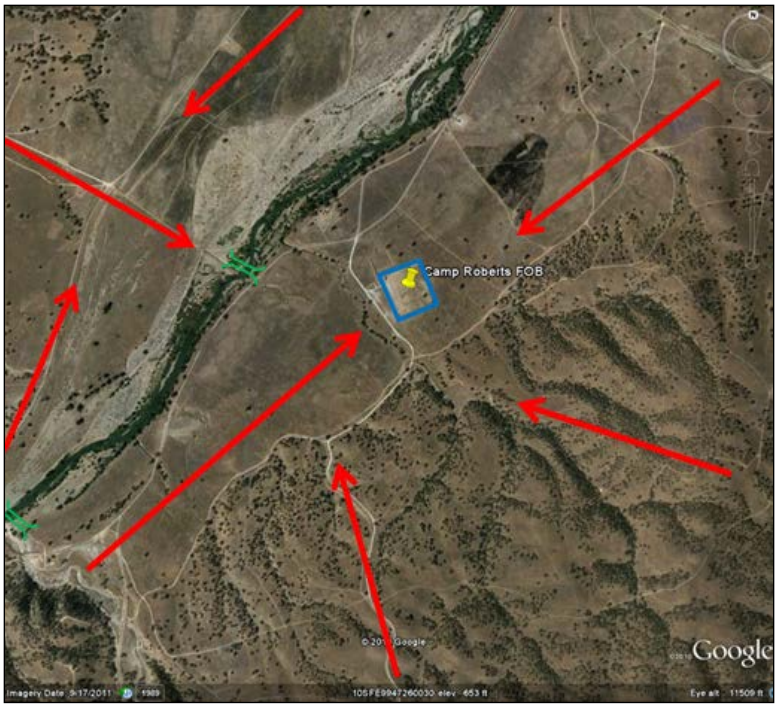
NASA	National Aeronautical and Space Administration
NLCD	National Land Cover Dataset
NVESD	Night Vision and Electronic Sensors Directorate
OASES	Ocean Acoustic and Seismic Exploration Synthesis
OGC	Open Geospatial Consortium
OSP	Optimal Sensor Placement
pdf	Probability Density Function
RF	Radio Frequency
SPEBE	Sensor Performance Evaluator for Battlefield Environments
TCP/IP	Transmission Control Protocol/Internet Protocol
TSOA	Technical Support Operational Analysis
URI	Uniform Resource Identifier
UTM	Universal Transverse Mercator
WMS	Web Map Service
XML	eXtensible Markup Language
XSD	XML Schema Definition

1 Background

1.1 Deployable force protection purpose and context

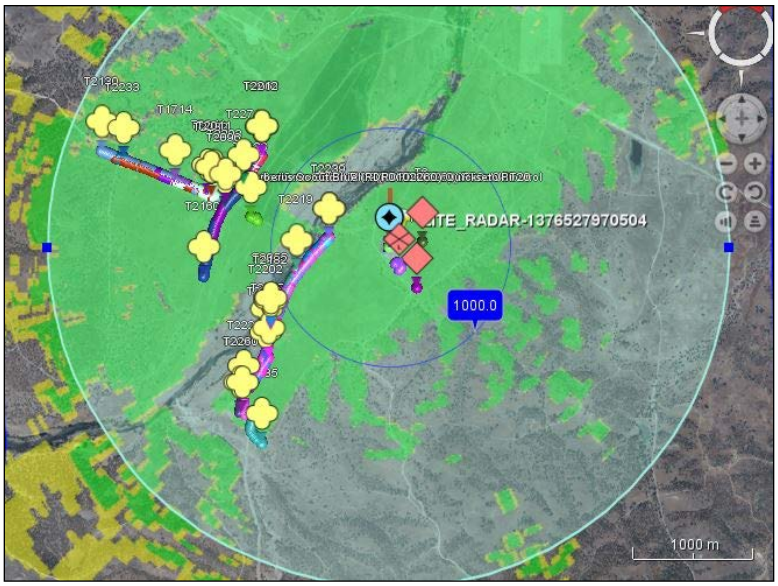
The Deployable Force Protection (DFP) program through the Office of the Assistant Secretary of the Army for Acquisition, Logistics, and Technology (ASA[ALT]) emphasizes technology development for force protection in small, forwardly deployed combat outposts in potentially remote and hostile territories (Figure 1). A significant focus is on enhancing situational awareness by improved and automated surveillance provided by sensor technologies. In doing so, a variety of technologies operate in concert with one another. As Soldiers simultaneously run and analyze outputs from multiple sensor devices to make decisions, force protection and defense planning resembles a complex systems problem. Many interrelated inputs for and outputs of various systems reveal separate pieces of the overall situation that Soldiers must interpret.

Figure 1. Defense vulnerabilities and enemy avenues of approach of a forward operating base. Surrounding mountains provide cover and concealment for enemy observation and direct- and indirect-fire attacks on the base. Vegetation and terrain inhibit monitoring enemy movement west of the river.



By integrating the execution and visualization of multiple systems in a common operating picture and Geospatial Information System (GIS) environment, a commanding Soldier is able to grasp quickly a comprehensive picture of a dynamically evolving situation, intelligently using a suite of surveillance technologies simultaneously, and to effectively direct groups of Soldiers on mission tasks (Figure 2). A major objective of the DFP program is to introduce a wide assortment of technologies to Soldiers yet make them as accessible and unified as possible. If Soldiers are to be empowered, rather than overwhelmed, by a myriad of new technologies, it is necessary to design an intuitive and cohesive workflow.

Figure 2. Integrated defense configuration of sensors and weapons.



1.2 Effective sensor use in the battlefield

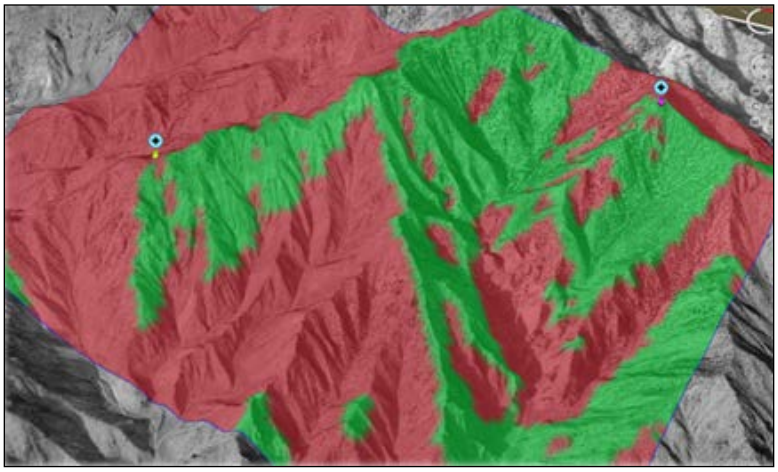
With increasing constraints on resources and labor in the Department of Defense (DOD), it is becoming critical to develop technologies that unburden small and minimally equipped teams of Soldiers carrying out missions, especially in isolated and dangerous battlefield environments. However, the ability of Soldiers to benefit from new technologies depends on how quickly and effectively they can operate these tools.

Given all the surveillance technologies available, it is clear that improving defense and protection is possible. However, maximizing the extent of their benefits depends on how effectively they are used, especially when sensor resources are limited. After deploying sensors, it is also important

and prudent to maintain awareness of their detection limitations for anticipating areas where an enemy may approach undetected.

In particular, a variety of terrain and weather conditions significantly affects sensor detection performance (Figure 3); and environmental factors affect various signal modalities in different ways and extents. As users of sensor technologies but without expert knowledge of signal physics and phenomenology, Soldiers need software tools that inform sensor performance limitations in complex terrestrial and atmospheric situations.

Figure 3. Dramatic topographical effects on the detection extent of high-frequency radar.



1.3 General software for battlefield signal modeling

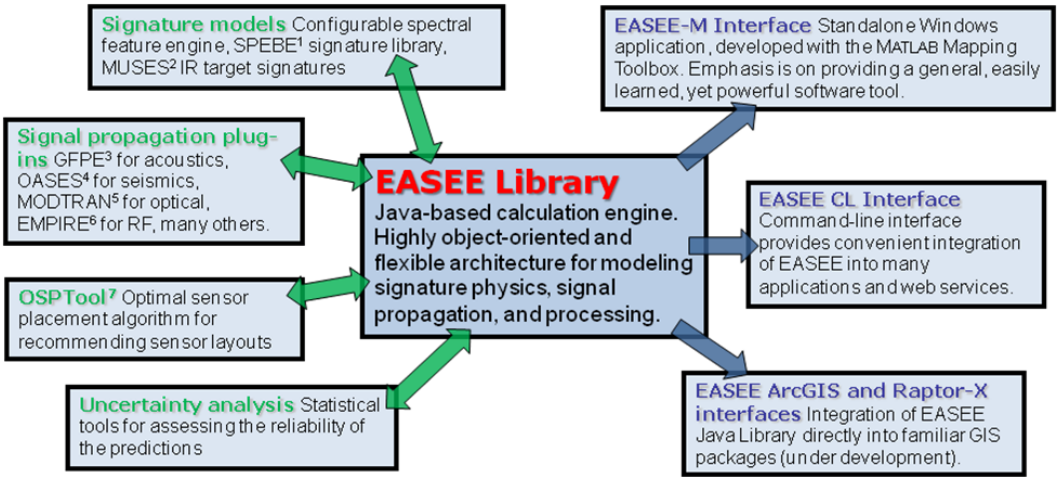
For non-expert users of sensor technologies, software for modeling detection performance based on realistic environmental conditions can provide the understanding required for effectively selecting and emplacing sensors for enemy surveillance in complex terrain and weather. The U. S. Army Engineer Research and Development Center (ERDC) began developing such software several years ago and continues to improve these tools.

The general software platform, called Environmental Awareness for Sensor and Emitter Employment (EASEE), characterizes complex terrain and weather effects on target signatures, signal propagation, and sensor systems. Its flexible, object-oriented software architecture combines multiple processes in signal transmission and sensing (i.e., generation, propagation, reception, and processing) for a wide range of signal modalities, in-

cluding optical, acoustic, seismic, magnetic, radio frequency, chemical, and biological.

The EASEE software contains an expansive library of realistic physics models and, to compute probabilities of detection and false alarm, implements statistical methodologies to account for uncertainties in signal and noise features (Figure 4). EASEE uses a sensor-placement algorithm that optimizes sensor selections and placements based on sensor supply limitations, detection coverage preferences, and sensor-placement constraints. All computational algorithms in EASEE are designed to be accurate yet also efficient for running on a standard laptop computer. More background and details about the EASEE software is available in (Yamamoto et al. 2012).

Figure 4. Components of the EASEE software.



[1]Sensor Performance Evaluator for Battlefield Environments developed by the U. S. Army Research Laboratory (ARL) and ERDC

[2]Multi-Service Electro-optic Signatures developed by ThermoAnalytics, Inc.

[3]Green's Function Parabolic Equation developed by ARL, ERDC, and the University of Mississippi

[4]Ocean Acoustic and Seismic Exploration Synthesis developed by the Massachusetts Institute of Technology

[5]MODerate resolution atmospheric TRANsmission developed by the U. S. Air Force Research Laboratory

[6]Electromagnetic Propagation Integrated Resource Environment developed by Remcom, Inc.

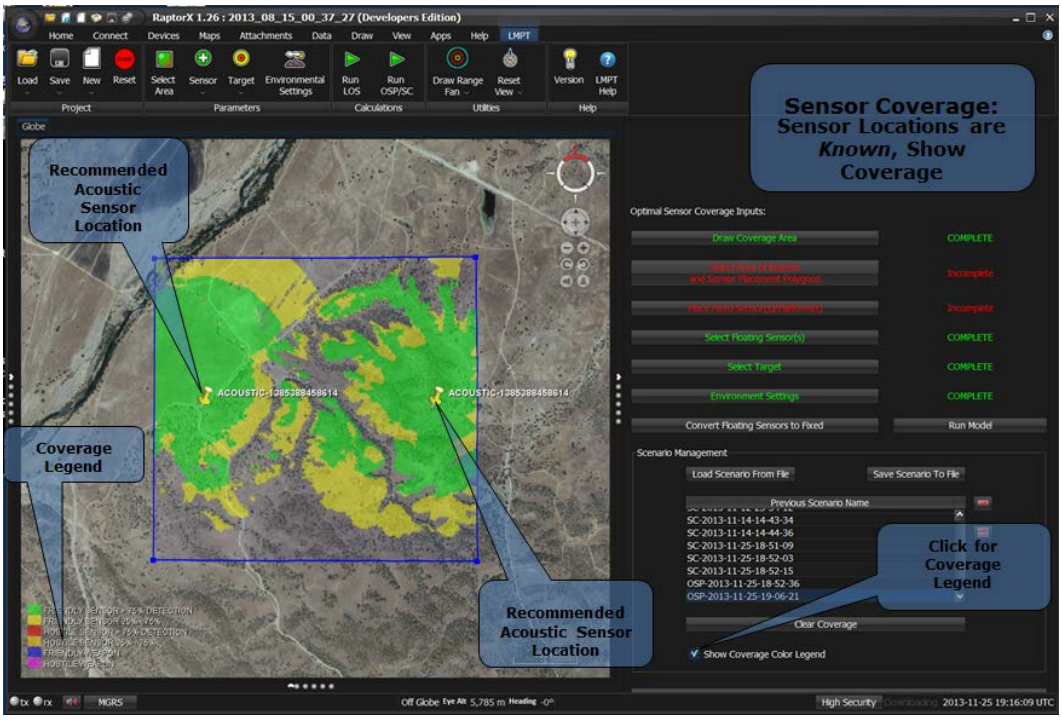
[7]Optimal Sensor Placement Tool developed by ERDC

1.4 Portable software for sensor predictions

From its inception, EASEE has been designed to be a portable computational engine that can be readily embedded into various software environments to enhance usability, visualization, and interactions among users of different software applications. In particular, we completed the Leaders

Mission Planning Tool (LMPT) project with the goal of integrating the EASEE software within a powerful, visually rich, and three-dimensional GIS browser, serving as a common operating picture for multiple other software tools and devices (Figure 5). This GIS environment, called Raptor, is a government off-the-shelf product from the U. S. Department of Energy that is based on NASA (National Aeronautical and Space Administration) World Wind.

Figure 5. The Leaders Mission Planning Tool (LMPT) is a plug-in application within the Raptor geographic information system to run the EASEE computational engine.



The portability of the EASEE software is made possible by a variety of techniques and features, including use of the Java programming language, highly object-oriented programming approaches, and an interface with an input/output (I/O) scheme using standard markup files such as eXtensible Markup Language (XML) and Java Script Object Notation (JSON).

Previous papers and reports discuss details and benefits of EASEE's Java object-oriented architecture (Yamamoto et al. 2012; Wilson et al. 2009). This report will focus mainly on the design and use of EASEE's newly developed interface for standard I/O.

2 Software Interface Using Markup File Exchange

The EASEE software is a continuously expanding computational library of considerable size and complexity, consisting of many interacting Java classes that represent and run calculations. The setup and execution of these Java data structures can be nontrivial. Even with a thorough understanding of the object-oriented software code, an extensive amount of programming may still be required to map desired scenarios into proper Java objects in EASEE.

Rather than instantiate and manipulate Java objects directly, it is typically easier for external software applications to access EASEE calculations by sending and receiving inputs and outputs based on universal markup languages (e.g., XML and JSON). By using this approach, external software applications may request EASEE calculations by simply generating and passing markup input files to EASEE. This is a less cumbersome and less error-prone alternative to porting and running actual EASEE source code within other software codes. Both external software and EASEE are run separately, and the interface between them is reduced to sending and receiving input and output files.

2.1 Motivation and benefits

There are many benefits to developing and using interfaces based on exchanging standard markup files. In general, this approach makes intuitive sense because a user application of EASEE may access and run calculations without having expertise on all of the details of the internal software code. One must only conceptually understand EASEE software's functionalities enough to produce inputs accordingly.

A major advantage is that it is easier to identify errors and bugs within an interface using markup files as such inputs and outputs may be standardized and validated by a schema (e.g., XML Schema Definition [XSD] and document type definitions). Schemas define the allowable data content and structure of inputs and outputs to ensure that they fulfill desired criteria, including sufficient completeness (i.e., the presence of required ele-

ments and specifications), correct data types (e.g., positive integers for grid resolution, listed string tokens for friendly or hostile platform affiliations and background noise categories, etc.), and predefined data nesting and order for proper parsing and interpretation by interacting software.

Furthermore, an interface based on a standard I/O protocol is substantially easier to maintain and deploy for the longer term as interacting software are continually modified. A significant issue with multiple software applications making direct calls to various internal constructors and methods is that such operations may cease to exist or fail to function as expected when software is further developed and modified. Whenever an interface is implemented in this way, changes in associated software products would often require follow-on updates to the interface code. Typically, the subsequent reworking of the interface code can be quite difficult and involved because developing and maintaining such an interface requires a rather substantial understanding of internal software code and, particularly, the details of ongoing modifications and improvements therein.

However, if an interface is based on exchanging standardized markup files, associated software products may individually focus on continuing to send and receive markup inputs and outputs that conform to an agreed upon data content and structure (i.e., schema). That is, as each software application is modified, related code within each are updated to still correctly generate and parse markup files accordingly. As long as each ensuing software improvement completes this, the interface may continue to remain stable and functional as intended.

When EASEE receives new capabilities or considerable revisions, it may make sense to alter the schema so that input and output markup files represent corresponding additions and improvements. In such cases, both interfaced software applications would need updates to exchange markup files to conform to the new schema. Typically, this requires generating or parsing some newly added or reorganized markup elements. If hardwiring source code between two software applications instead, it becomes necessary to undergo a usually more intricate and tricky task to revise the interface code to access new capabilities or to conform to extensive revisions and reorganization of source code within either software application.

Similarly, there may be instances when, for example, a significant restructuring and refactoring of the EASEE source code does not necessarily result in any additional capabilities but only internal code improvements, such as improved efficiency, information flow, or object-oriented design, etc. In such cases, the markup file inputs and outputs to EASEE may remain exactly the same although some code within EASEE for generating and parsing them may need to be updated due to alterations to the internal source code.

Consequently, if there are several interfaces to EASEE based on markup file exchange, they may all execute the newly improved EASEE software without performing any follow-on modifications in their interfaces to EASEE because the markup input and output files and schema have not changed. However, if all the software applications were interfaced by directly executing EASEE source code at various points within their own source code, it would be necessary to make potentially extensive corrections to connect and run with the newly improved version of EASEE.

As interacting software continue to develop, interfaces based on direct calls to source code or on exchanging markup files require maintenance; however, it is often easier and less risky to take the latter approach. In essence, a file-exchanging interface allows development of multiple software applications to remain separate and independent of each other as much as possible. Software interactions are controlled by a schema for markup file exchange where only a higher level, conceptual understanding of functionalities is required to implement and maintain the interface.

2.2 Overview of the XML schema and interface for EASEE

Because of its universality and widespread support, XML markup is used in the file-exchanging interface for EASEE. XML is a markup format that is textual, declarative, and hierarchical. Nested tags are used to describe data elements, which may be contained within each other in a logical structure. The name, content, and organization of elements may be defined in an XSD, which serves as a blueprint for valid XML that may be passed between software applications. Many tools exist for developing software with XML, including the Java Architecture for XML Binding (JAXB), which is used, for example, in EASEE to import XML schemas and to process XML files.

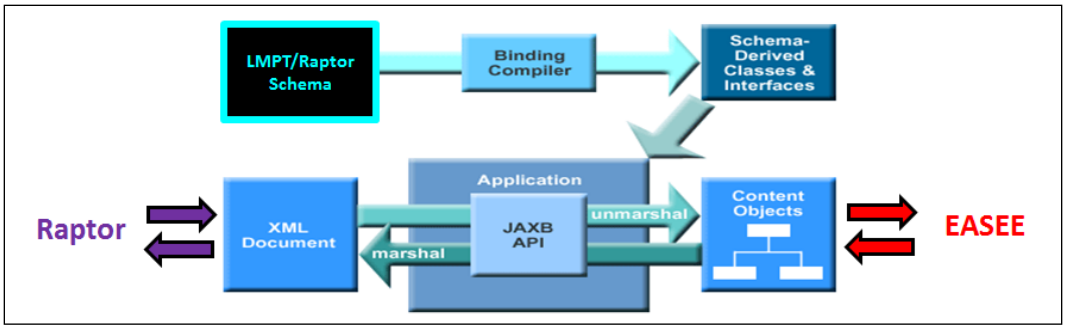
The XML schema for EASEE has been defined to describe fully the wide variety of battlefield signal transmission and sensing calculations that EASEE supports. It is comprehensive and flexible to allow various combinations of XML elements for different types of scenarios, including for sensor coverage and optimal sensor placement. While the schema enforces that XML always contain certain basic elements (e.g., corners of the rectangular calculation domain, a variable platform, category and type enumerations for each platform, etc.), it also allows optional specification of some advanced parameters, including detailed environmental conditions (e.g., wind speed and direction, atmospheric stability, land cover types, etc.) and other elements (e.g., the explicit name of the signal propagation model, the area-of-interest and sensor-placement polygons for optimal sensor placement, non-omnidirectional fields of regard for sensor sand weapons, etc.). Hence, the XML schema and interface for EASEE accommodate for a variety of user applications that may choose to control and run any type of EASEE calculation, from the most basic to the advanced, as desired.

This report focuses on the general modeling capabilities of EASEE that can be accessed by the newly developed interface for markup files, specifically as part of the objectives for the LMPT project.

Appendix A documents for software developers extensive and precise details on the structure and features of the XML schema elements themselves. The XML documentation reflects the current copy of the XML schema as of the writing of this report. While the schema may evolve over time, this version of the schema is still useful for illustrating much of its general concept and design.

Finally, it is worth noting that EASEE uses JAXB to simplify programming with XML in Java (Figure 6). JAXB is a useful tool available within Java Integrated Development Environments (e.g., Netbeans, Eclipse, IntelliJ IDEA, etc.). Using JAXB, it is possible to import an XML schema into a Java project and map its XML elements into Java objects. Then, there are convenient methods for transforming between Java objects and corresponding XML elements.

Figure 6. EASEE–Raptor XML interface data flow using Java Architecture for XML Binding.



With JAXB, Java developers may instantiate and manipulate Java object representations of XML elements within Java source code, assemble them together, and then simply invoke a method to transform them into XML. This is typically easier than directly writing XML using a text writer and making sure all the syntax is correct (e.g., that elements are properly nested and all start and end tags are present). Conversely, XML elements may be transformed into Java objects whose data contents may be easily queried using get methods in Java rather than setting up a text reader and directly parsing through XML tags. Furthermore, JAXB automatically identifies during compile time coding errors related to generating and parsing XML. This is possible because JAXB requires a valid schema to be specified for mapping XML elements to corresponding Java objects. Consequently, only valid Java objects are available for construction, manipulation, and transformation to and from XML.

2.3 Extensions of interface to support JSON and other markup inputs

Although XML is the markup format used for the schema for both validation and translation into EASEE Java code, a different markup format may be preferred in various software environments. For example, the JSON standard has become a very popular and useful markup language among some developers and use cases, particularly in browser and mobile applications.

As the backbone of the EASEE markup interface is XML-based, it is necessary to translate inputs and outputs to and from XML when another markup format is used for the interface. If several client applications may potentially prefer a certain non-XML markup format, it makes sense to support the translation to and from XML within the EASEE interface code

for reuse by multiple user applications rather than its being duplicated multiple times in various individual client software programs. This would expand the use of EASEE by a variety of client applications and expedite development of integrated software capabilities.

In particular, to accommodate web-based applications of EASEE, we have improved the markup interface for EASEE to allow also the exchange of JSON files. In most mobile and browser environments, JSON is the preferred format for delivering data to and from services. Thus, support for JSON within the markup interface for EASEE would be a very useful feature.

To translate JSON into XML for use by the existing XML-based interface, it is necessary to have clean and robust transformations between the markup languages. Because of subtle but significant differences in both the data model and serialization formats, this transformation can be challenging. Several libraries exist that perform desired markup translations between JSON and XML. Most of those that are open source implement some basic syntax conversions with various limitations. It is possible to hand code the transformations for specific applications, but this can be time consuming to implement.

The EASEE interface uses the library by Douglas Crockford, the creator of JSON. However, there are some limitations when transforming from JSON into XML, resulting in inexact ordering of XML elements and instabilities with XML attributes. This may lead to issues in validating and interpreting the XML translated from JSON if the schema enforces a particular ordering and uses attributes. To accommodate for this, the schema definition of the XML interface for EASEE has been loosened so that child elements are used instead of attributes and so that ordering of various XML elements does not matter as long as the required ones are still present. Even with the slightly relaxed restrictions, the schema will still identify and alert for incomplete and invalid XML inputs, both directly provided or translated from JSON, that describe unviable calculations for EASEE.

3 Basics of EASEE Calculations

This section serves as a basic introduction of the types of calculations in EASEE, their input parameters, and computing probability of detection. Subsequent sections of the report provide further details and Appendix A supplies documentation on the XML schema for EASEE.

3.1 Calculation modes

The two basic calculation modes that EASEE supports include probability of detection and optimal sensor placement. Probability-of-detection calculations can perform two types of predictions:

1. Footprint of the sensor (i.e., locations of an emitter where a fixed-position sensor would detect)
2. Footprint of the emitter (i.e., locations of a sensor where a fixed-position emitter would be detected)

For sensor footprints, probabilities of detection are computed with the signal emitter's position varying across the spatial calculation grid and the sensor position fixed. For emitter footprints, probabilities of detection are computed with the sensor's position varying and the signal emitter's position fixed.

The optimal sensor-placement calculation mode optimizes selection and placement of sensors by analyzing detection coverage of candidate sensors at various positions. Detection coverage of candidate sensors is obtained by performing sensor footprint calculations for each candidate sensor.

3.2 Input parameters

Any calculation in EASEE requires a few basic input parameters:

1. A rectangular geographic domain for the calculation
2. An emitting or sensing platform with varying positions over the calculation domain

3. At least one emitting/sensing platform with a fixed position for an emitter/sensor footprint calculation or at least one candidate sensing platform for an optimal sensor-placement calculation

There are other fundamental input parameters that are relatively important and always used in EASEE but that are not required to be customized by the user. If unspecified, these parameters are set to default values by EASEE. Such parameters include the following:

1. A background noise model that characterizes noise from distributed sources, such as roadway traffic and wind
2. Environmental conditions, including terrain elevation, land cover, atmospheric stability, wind direction and speed, and others
3. Specification of the signal propagation models to be used for different signal modalities

Finally, there are optional, advanced input parameters:

1. Resolution of the calculation by specifying either (1) the desired number of points sampled vertically and horizontally in the calculation domain or (2) the desired uniform spatial resolution in meters. In the first approach, the number of calculation points is specified to sample the edge of the calculation domain with the greater physical distance, where the resultant spatial increment between sampled points along the longer edge is applied when sampling the shorter edge. Consequently, the shorter edge is sampled by the same number of points as along the longer edge (for a square domain) or fewer (for a non-square domain). If unspecified, 150 points are sampled along the longer edge of the rectangular calculation domain, and the resultant spatial sampling increment is applied when sampling the shorter edge.
2. Polygons defining areas of interest and sensor-placement constraints for optimal sensor-placement calculations. Area-of-interest polygons may include two attributes, coverage priority and probability-of-detection threshold. Sensor-placement polygons may include one attribute, placement cost. If unspecified, the entire calculation domain has a probability-of-detection goal of 0.95 with coverage priority of 1 and sensor-placement cost of 0. See Section 4.7.3 for details.
3. Number of candidate sensor locations to be generated and analyzed by the optimal sensor-placement algorithm. If unspecified, EASEE generates a

default number of candidate sensor locations for optimal sensor placement. See Section 4.7.4 for details.

4. Additional characteristics of emitting/sensing platforms, including custom fields of regard and orientation. If unspecified, default radiation patterns and sensor directional gains are used for various emitters and sensors. The orientation is set to zero pitch and zero roll with yaw set to northward. See Section 4.4 for details.

3.3 Calculating probability of detection

To account for random uncertainties in signals subject to irregular signal generation, propagation, and sensing phenomena, EASEE uses statistical models to represent signals (Yamamoto et al. 2010). Namely, probability density functions (pdfs) describe signals from targets of interest, $s(x)$, and noise, $n(x)$, for scalar values, x, of a signal feature (e.g., acoustic sound level, seismic intensity, radio-frequency power, concentration of chemical or biological agent). Then, probabilities of detection, P_d, and false alarm, P_{fa}, are computed by the following integrals, which are required to determine a sensor's receiver operating characteristic:

$$P_d = \int_{\beta}^{\infty} (s+n)(x)dx$$

$$P_{fa} = \int_{\beta}^{\infty} n(x)dx,$$

where

$s(x)$ = pdf of signals from targets of interest,
$n(x)$ = pdf of signal from noise sources,
β = sensor detection threshold.

A variety of processing algorithms exist in EASEE for computing the sensor detection threshold, β, including

1. Neyman-Pearson (constant false alarm rate) criterion (Burdic 1984),
2. absolute thresholding,
3. relative thresholding,

4. error minimization, and
5. Bayes risk minimization.

The default detection processing algorithm in EASEE uses the Neyman-Pearson criterion with the constant false alarm rate set to 10^{-3}, where the detection threshold, β, is set such that $P_{fa} = \int_{\beta}^{\infty} n(x)dx = 10^{-3}$.

4 Modeling Capabilities for Battlefield Force Protection

Discussions within this section focus on new modeling capabilities developed within EASEE as part of the LMPT project. Other references describe the previously existing features in EASEE and additional, concurrent developments (Yamamoto et al. 2012, 2013; Wilson et al. 2013; and Vecherin et al. 2011).

4.1 Radio-frequency transmission and monostatic radar

The internal, object-oriented Java architecture of EASEE is flexibly designed to support efficient integration and simulation of diverse signal modalities via extensible libraries of target signatures, signal propagation models, and sensor systems. Thus, if available, external software may be integrated within EASEE to model particular aspects of the signal-transmission and sensing process. An apt example of this is the integration of the U.S. Navy Electromagnetic Propagation Integrated Resource Environment (EMPIRE) software suite in EASEE for modeling radio-frequency (RF) transmission and monostatic radars. This way, EASEE is able to access the nearly twenty different realistic RF propagation models contained in EMPIRE.

Both one-way transmission and two-way monostatic radar calculations by EMPIRE have been interfaced in EASEE. The one-way calculations simulate the emission and reception of signals between distinct transmitting and receiving antennas. Two-way calculations simulate monostatic radar scenarios where transmitting and receiving antennas are co-located on a single radar system (i.e., signals are transmitted by the radar, reflected off a target, and received by the same radar). Transmission loss over an entire radar path is predicted by combining transmission loss from both forward (from radar to target) and inverse (from target to radar) directions and applying a target radar cross section. By default, EMPIRE supports a single-valued radar cross section, independent of incident, scattered angles, and the field component.

The EASEE–EMPIRE integration, like the EASEE–Raptor interface, uses an XML-based interface as described by Yamamoto et al. (2013). The XML inputs and outputs serve as standardized data models between EASEE's Java and EMPIRE's C++ software in accordance with a detailed XML schema included with the EMPIRE software distribution. The XML data exchange sequence is as follows:

1. EASEE requests an RF propagation calculation from EMPIRE by sending an XML file to EMPIRE with input parameters for the computation.
2. EMPIRE parses input XML from EASEE and performs the specified RF propagation calculation.
3. EMPIRE sends a new XML file to EASEE with RF calculation results (i.e., transmission loss on a cylindrical grid).
4. EASEE parses the output XML from EMPIRE.

The XML file from EASEE to EMPIRE may contain a variety of input parameters for the RF computation (step 1). These parameters generally include transmitter position and frequency, uniform cylindrical sector of target and receiver positions, transmitting and receiving antenna types, radar cross section of the target, terrain elevation data, and a specification of a specific propagation model within EMPIRE. Figure 7 shows an example.

Figure 7. Example XML input to EMPIRE for a monostatic radar calculation. For illustrative purposes, a large series of spaced-delimited values for terrain elevation is substituted by "..." within the ElevationData element.

```
<!DOCTYPE EMPIRE SYSTEM "EMPIRE.dtd">
<EMPIRE>
    <PropCalc>
        <PS_MonoRadarSectorJob>
            <PS_Model name="TIREM"/>
            <Frequency>9797.958971132713</Frequency>
            <StartPoint lat="35.752914" lon="-120.79148"
alt="20.0" altref="AGL"/>
            <StartAntenna>
                <IsotropicAntenna>
                    <Orientation Bearing="0.0" Pitch="0.0"
Roll="0.0"/>
                </IsotropicAntenna>
            </StartAntenna>
            <Terrain>
                <PS_TerrainGrid LatCount="47.0" LonCount="100.0">
                    <SWCorner lat="35.74046" lon="-
120.849858333333" altref="AGL"/>
                    <NECorner lat="35.768275" lon="-120.772736"
altref="AGL"/>
                    <ElevationData count="4700.0">
389.40298667691576 402.29787820350106 389.30874528569495 ...
232.0 228.19323269564094 228.0 </ElevationData>
                </PS_TerrainGrid>
            </Terrain>
            <PS_SectorBounds MinHeight="2.5" MaxHeight="2.5"
MinRange="0.0" MaxRange="5530.45024334612" StartBearing="0"
EndBearing="360" ref="AGL"/>
            <PS_SectorSamples HeightCount="1.0" RangeCount="81.0"
BearingCount="1580.0"/>
            <RadarCrossSection>
                <RCSSimple rcs="90.0">90.0</RCSSimple>
            </RadarCrossSection>
        </PS_MonoRadarSectorJob>
        <Calculate Output="EASEELib\tmp\empireOutput.xml"
DataType="float"/>
    </PropCalc>
</EMPIRE>
```

Given that a geographic calculation domain in EASEE is represented as a unstructured Universal Transverse Mercator (UTM) grid corresponding to a regular latitude and longitude geographic grid, it is important to define an appropriate uniform cylindrical sector for receiver (for one-way transmission) or target (for monostatic radar) positions in the input XML for EMPIRE. Specifically, the resolution of the uniform cylindrical calculation grid in EMPIRE must be defined so that all calculation points are resolved in the unstructured UTM grid calculation domain in EASEE, particularly along the edges, during subsequent interpolation. Yamamoto et al. (2015) provides details on the method for defining appropriate uniform cylindri-

cal grids in EMPIRE for a given non-uniform UTM-grid calculation domain in EASEE.

When parsing output XML from EMPIRE (step 4), it is important to note that the transmission loss on the uniform cylindrical grid from EMPIRE is oriented with the northward bearing pointing towards true north (i.e., where the geographic longitudes converge). Thus, during interpolation onto the original non-uniform UTM grid calculation domain in EASEE, a correction for the UTM meridian convergence angle must be applied. See Yamamoto et al. (2015) for details.

4.2 Access to web-based geospatial data services

For enhanced usability and convenience, various modeling and simulation applications are beginning to leverage geospatial data from networked and web-based servers in real time. This is becoming increasingly possible as open standards for geospatial content and services are being developed. In particular, the Open Geospatial Consortium (OGC) was established in 1994 to foster collaboration and consensus in practices and data models among various commercial, government, and other organizations. Current and future developments of EASEE seek to include interfaces for various OGC compliant geospatial data models and services.

Currently, an interface in EASEE can access the NASA World Wind web-database terrain-elevation data caches on a local client's machine. The database is powered by Web Map Service (WMS) servers, which is an OGC compliant protocol for supplying geo-referenced maps over the Internet. The NASA World Wind environment is one example of an open source WMS implementation that enables sharing and delivering map data. Other WMS software environments exist that are both open source (e.g., GeoServer and MapServer) and proprietary (e.g., ArcGIS Server).

Specifically, the NASA World Wind terrain cache reader in EASEE extracts cached terrain elevation data from the local machine's hard drive. This data cache on the client machine would be populated during real-time downloads by external World Wind applications. In particular, the Raptor GIS environment is an example of a World Wind application that automatically downloads imagery and terrain elevation data from the NASA World Wind

server onto a local cache for those areas that a user zooms into on the virtual globe (Figure 8).

Figure 8. The NASA World Wind terrain elevation and imagery data downloaded by Raptor.

4.3 Range-limited line of sight for sensors and weapons

Quick, low-fidelity predictions of fire or detection by a variety weapons and sensors may be based on a simple, range-limited line-of-sight model. Such a model is not physics based and, hence, requires minimal computational time to run. It is intended only as a rapid and approximate alternative to realistic physics-based models.

Range-limited line-of-sight models compute line of sight based on the heights of both the start and end points while also limiting the length of line-of-sight paths to an ascribed value. In general, range limits may be a function of the sensor, target, and weather conditions. The range limit may also be a parameter that is customizable by a user, as is allowed by the XML interface for EASEE.

4.3.1 Realistic detection range limits based on sensor, target, and weather

When using range-limited line-of-sight models to represent realistic detection extents by various visual and thermal imagers, it is important to use appropriate range limits based on various combinations of sensor capabilities, target signal emissions, and weather conditions. A full physics-based

approach would compute these multiple interactions in real time. However, for range-limited line-of-sight models, various range limits may be pre-computed and stored in a look-up table.

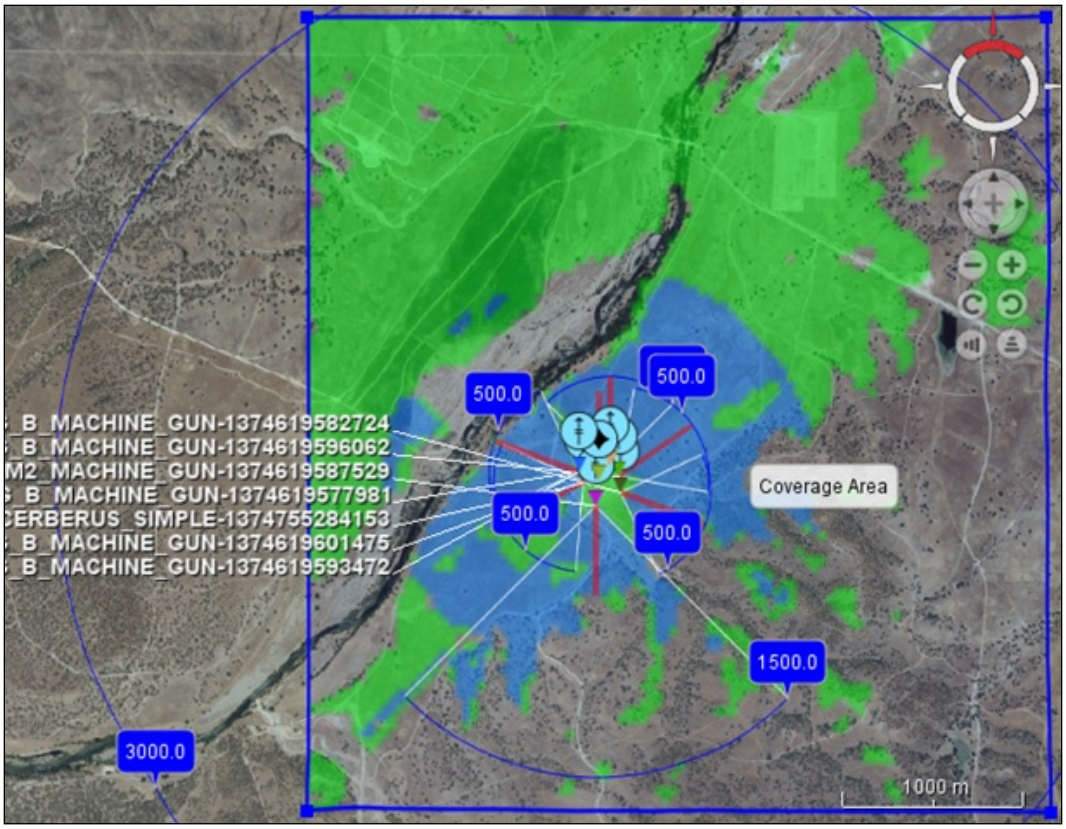
As part of LMPT, we included these look-up tables in EASEE for use by simple range-limited line-of-sight models. Range-limit values for various combinations of sensors, targets, and weather conditions were pre-computed by NVTherm, an expert electro-optical and infrared modeling software. Alternatively, the user may select a custom sensor and set any arbitrary detection range limit desired.

Range-limited line-of-sight models with look-up tables of pre-computed detection extents are intended for relatively rough but quick predictions on visual- and thermal-imager performance. More realistic, physics-based models that use higher fidelity terrain and weather data are being incorporated into EASEE.

4.3.2 Direct-fire weapon models

Because the ability of a Soldier to fire on a target largely depends on line of sight to the target and the ballistic range of the weapon, it is suitable to model the effective weapons range with range-limited line-of-sight models (Figure 9). As with imagers described earlier, line of sight takes into account the heights of the start and end points of the line-of-sight path. For weapons scenarios, these correspond to the heights of the weapon and target, respectively, where line of sight improves for taller targets and when the weapon position is elevated (e.g., from lying on the ground to standing up to posting in a tower). A look-up table defines range limits for various weapons to reflect their effective, lethal firing ranges. Alternatively, a user may also select a custom weapon platform and specify any arbitrary firing range desired.

Figure 9. Sensor coverage (*green*) and areas of fire (*blue*) by friendly sensors and weapons.



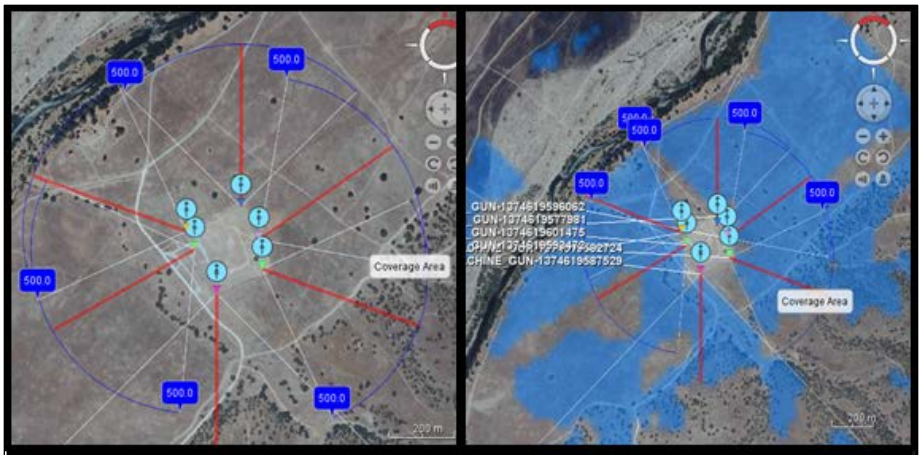
4.4 Customizable fields of regard for sensors and weapons

A field of regard is the extent of the physical environment that can be perceived at any given moment. For a sensor, it is the solid angle through which a detector is sensitive to signals. For weapons, it is the solid angle through which a weapon can be fired. In general, various target and sensing platforms have directional radiation and sensor gain patterns. Realistic use of weapons is limited by angular fields of fire. Although it may be appropriate in some instances to make predictions using omnidirectional models for target emissions, sensor gains, and weapons fire, it may also be desirable to model directional radiation and sensing patterns.

In particular, the XML interface for EASEE allows specification of non-omnidirectional fields of regard for sensing and weapon platforms (Figure 10). It is also possible to indicate orientation for platforms with non-omnidirectional fields of regard. Non-omnidirectional fields of regard are applicable to a subset of sensing platforms. Namely, acoustic microphone and seismic geophone sensors are always modeled as omnidirectional in

EASEE. It is, however, possible to customize fields of regard for all direct-fire weapon platforms in EASEE.

Figure 10. Direct-fire weapons with customized fields of regard and orientations.



Position	Orientation	FOV	Left Limit	Right Limit
NE Tower	0	110	305	55
E Tower	55	110	0	110
SE Tower	110	110	55	165
SW Tower	180	110	125	235
Gate	240	110	185	295
NW Tower	290	110	235	345

A custom field of regard is indicated by both azimuthal (horizontal) and elevation (vertical) angular extents, which are restricted to values from 0° to 360° and 0° to 180°, respectively. The azimuthal and elevation field of regard angles represent the total viewing and firing extents where the center of the angular ranges are aligned with and pointed outwards from the front of the platform.

4.5 Customizable platform orientation

Platform orientation (a.k.a., attitude) is specified by three degrees of freedom:

1. Yaw—This parameter is restricted to values between 0° and 360°, where zero degrees aligns the front of the platform towards north. The parameter rotates the platform around the axis orthogonal to the Earth's ellipsoid and passing through the center of the platform, where degree values increase clockwise from north (i.e., towards the eastward direction).
2. Pitch/tilt—This parameter is restricted to values between −90° and +90°, where zero degrees aligns the axis passing through the front and back of the platform with the tangent to the Earth's ellipsoid. The parameter rotates the platform around the axis passing through the sides of the platform, where the front of the platform points upwards and downwards for positive and negative degree values, respectively.
3. Roll—This parameter is restricted to values between −180° and +180°, where zero degrees aligns the axis passing through the sides of the platform with the tangent to the Earth's ellipsoid. The parameter rotates the platform around the axis passing through the front and back of the platform, where rotations counterclockwise are positive degree values when looking from the front of the platform into the back. This parameter is always set to zero degrees in EASEE for LMPT.

An arbitrary three-dimensional orientation of a rigid body is uniquely described by these three degrees of freedom by explicitly defining the specific sequence of rotations as yaw-pitch-roll in EASEE. This convention describes orientation defined by Tait-Bryan angles by using intrinsic rotations where a series of three elemental rotations occur about the orthogonal axes of the platform's local coordinate system, which modifies its orientation after each elemental rotation. Note that, if an explicit sequence of rotations is not enforced, the three degrees of freedom are not unique (i.e., one set of values for the three degrees of freedom do not uniquely define a given orientation).

The customizable fields of regard described earlier are a simple case of directional source radiation and sensor gain patterns that represent single beams with prescribed limits on azimuthal and elevation angles. In gen-

eral, emission and gain patterns can be arbitrarily complex, and expansions to EASEE are being completed to support them.

The object-oriented structure of the EASEE software enables such generalizations to be made efficiently. For example, the single beams are a specific subclass implementation of general radiation patterns. Other, more complicated radiation pattern implementations include a radiation raster that ascribes a series of radiation or gain factors at discrete points from the emitter/sensor, as desired. Combinations of smoothly varying patterns (e.g., monopoles, dipoles, etc.) may use models in EASEE that compute spherical harmonics.

Conceptually, scattering patterns may be represented as a combination of directional gain and radiation patterns (Yamamoto et al. 2013). Developers are currently working to incorporate directional scattering patterns into EASEE. Such models would, for example, describe target radar cross sections as a function of incident and scattering angles.

4.6 Force-on-force scenarios

4.6.1 Background on sensor- and weapon-platform characteristics

EASEE includes an expansive library of various target, sensing, and weapon platforms of military interest. Each platform is identified by category and type. For example, the wheeled ground vehicle category includes a variety of types, such as a sport utility vehicle, generic truck, and high-mobility multipurpose wheeled vehicle (HMMWV). The direct-fire category includes various weapon types, such as the M16 rifle, AK47 rifle, M249 machine gun, M203 grenade launcher, etc.

A platform may have either a fixed position at a single coordinate or a variable position that varies across the gridded calculation domain. For any calculation, there can be only one variable platform although any number of fixed platforms is allowed. In essence, the overall calculation on an N × M gridded domain iterates over a sequence of N × M individual calculations where the variable platform is moved to each of the N × M points in the gridded domain. The calculation output raster displays the probability-of-detection value at each of the N × M points for the corresponding calculation.

4.6.2 Modeling friendly and hostile scenarios

To model various defensive and offensive scenarios, friendly or hostile affiliation may be indicated for sensing and weapon platforms. For any calculation, it is possible to simultaneously have sensor and weapon platforms of both affiliations. Traditionally, two basic situations are of interest for such calculations: (1) friendly monitoring hostile and (2) hostile monitoring friendly.

In the first situation (i.e., friendly monitoring hostile), probability is computed that at least one of all friendly sensors would detect any hostile target platform. In this case, friendly target platforms, if present, emit nuisance noise signals. If friendly weapons are specified, their ability to fire at the variable platform is also computed. In the second situation (i.e., hostile monitoring friendly), the situation is reversed; probability is computed that at least one of all hostile sensors would detect any friendly target platform. In this case, hostile target platforms, if present, emit nuisance noise signals. If hostile weapons are specified, their ability to fire at the variable platform is also computed. In weapons calculations, the target is always the variable platform, regardless of its affiliation.

4.6.3 Visualization and interpretation

To simultaneously visualize sensing and firing abilities of both friendly and hostile assets (i.e., all monitoring all), EASEE includes a new calculation situation. In this case, it is challenging to formulate a suitable way to visualize the result, especially as the number of various friendly and hostile assets increases.

Below is a color scheme for presenting results in a way that fully captures all the desired information while being intuitive and easily interpreted for various situations:

1. All monitoring all with only friendly assets. Or friendly monitoring hostile.

    a. Locations where fixed friendly weapons can fire at the variable platform are colored blue.
    b. Locations where at least one friendly sensor can detect any hostile target platform are indicated by the following colors corresponding

to different values for probability of detection (Pd): transparent for $Pd < 0.25$, yellow for $0.25 \leq Pd \leq 0.75$, and green for $Pd > 0.75$.

c. Friendly fire ranges indicated in blue override the friendly sensor detection color wherever both coincide.

2. All monitoring all with only hostile assets. Or hostile monitoring friendly.

   a. Locations where fixed hostile weapons can fire at the variable platform are colored magenta.
   b. Locations where at least one hostile sensor can detect any friendly target platform are indicated by the following colors corresponding to different values for probability of detection: transparent for $Pd < 0.25$, orange for $0.25 \leq Pd \leq 0.75$, and red for $Pd > 0.75$.
   c. Hostile fire ranges indicated in magenta override the hostile sensor detection color wherever both coincide.

3. All monitoring all with both friendly and hostile assets.

   a. Locations where fixed friendly weapons can fire at the variable platform are colored blue.
   b. Locations where fixed hostile weapons can fire at the variable platform are colored magenta.
   c. Locations where at least one friendly sensor can detect any hostile target platform are indicated by the following colors corresponding to different values for probability of detection: transparent for $Pd \leq 0.75$ and green for $Pd > 0.75$.
   d. Locations where at least one hostile sensor can detect any friendly target platform are indicated by the following colors corresponding to different values for probability of detection: transparent for $Pd \leq 0.75$ and red for $Pd > 0.75$.
   e. Friendly fire ranges indicated in blue override the friendly sensor detection color wherever both coincide.
   f. Hostile fire ranges indicated in magenta override the hostile sensor detection color wherever both coincide.

Note that situation 3 is exceedingly more complex to visually represent as the following cases for overlapping colors exist:

1. Areas of simultaneous detection by friendly and hostile sensors

   a. Green and red
   b. Green and orange
   c. Yellow and red
   d. Yellow and orange

2. Areas of coinciding fire by friendly and hostile weapons
   a. Blue and magenta

3. Areas of fire by friendly weapons overlapping with areas of detection by hostile sensors
   a. Blue and red
   b. Blue and yellow

4. Areas of fire by hostile weapons overlapping with areas of detection by friendly sensors
   a. Magenta and green
   b. Magenta and yellow

5. Areas of fire by friendly weapons overlapping with areas of simultaneous detection by friendly and hostile sensors
   a. Blue with any of the four possible colors from case 1

6. Areas of fire by hostile weapons overlapping with areas of simultaneous detection by friendly and hostile sensors
   a. Magenta with any of the four possible colors from case 1

7. Areas of coinciding fire by friendly and hostile weapons overlapping with areas of simultaneous detection by friendly and hostile sensors
   a. Blue and magenta with any of the four possible colors from case 1

Various overlapping colors are represented by mixing colors by the subtractive color model, which is analogous to mixing different color paints (Figures 11 and 12). This contrasts with an additive color model based on mixing different color light. We preferred implementing color mixtures by the subtractive color model because it is more intuitive to infer which original colors were combined to achieve various blended colors. In essence, this intuition originates from relatively common experiences in mixing paint and dyes.

Figure 11. Simultaneous visualization of friendly and hostile weapons fire. Bottom left corner shows the color legend for friendly and hostile sensor detection and weapon fire.

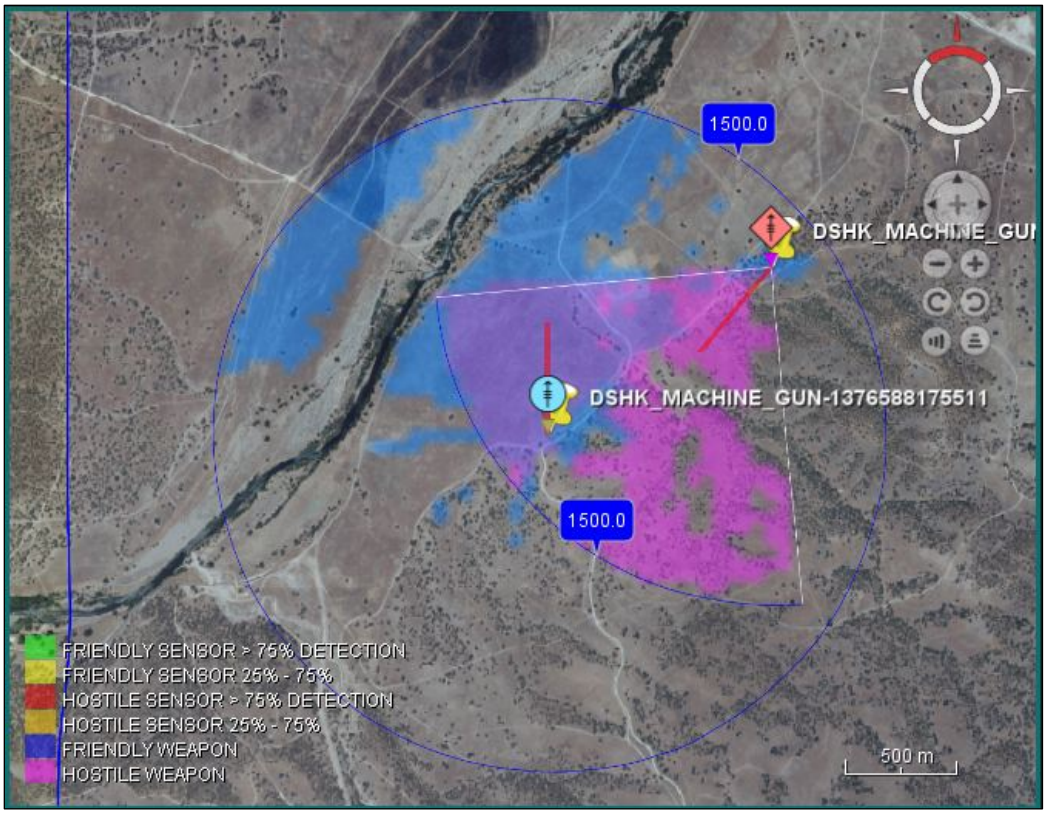
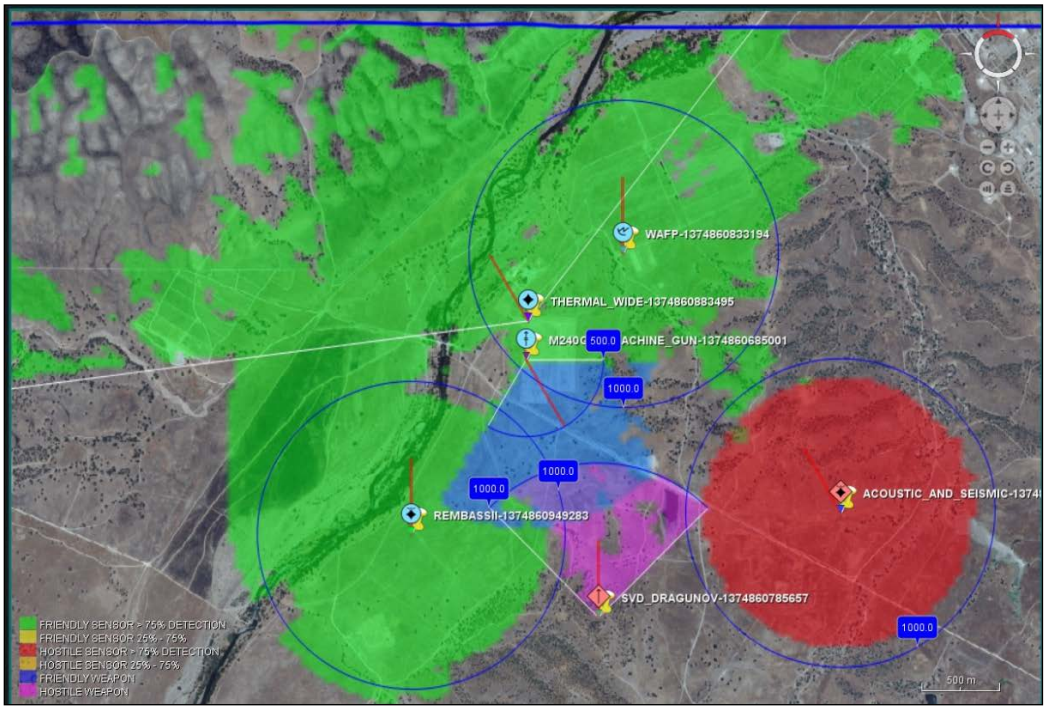
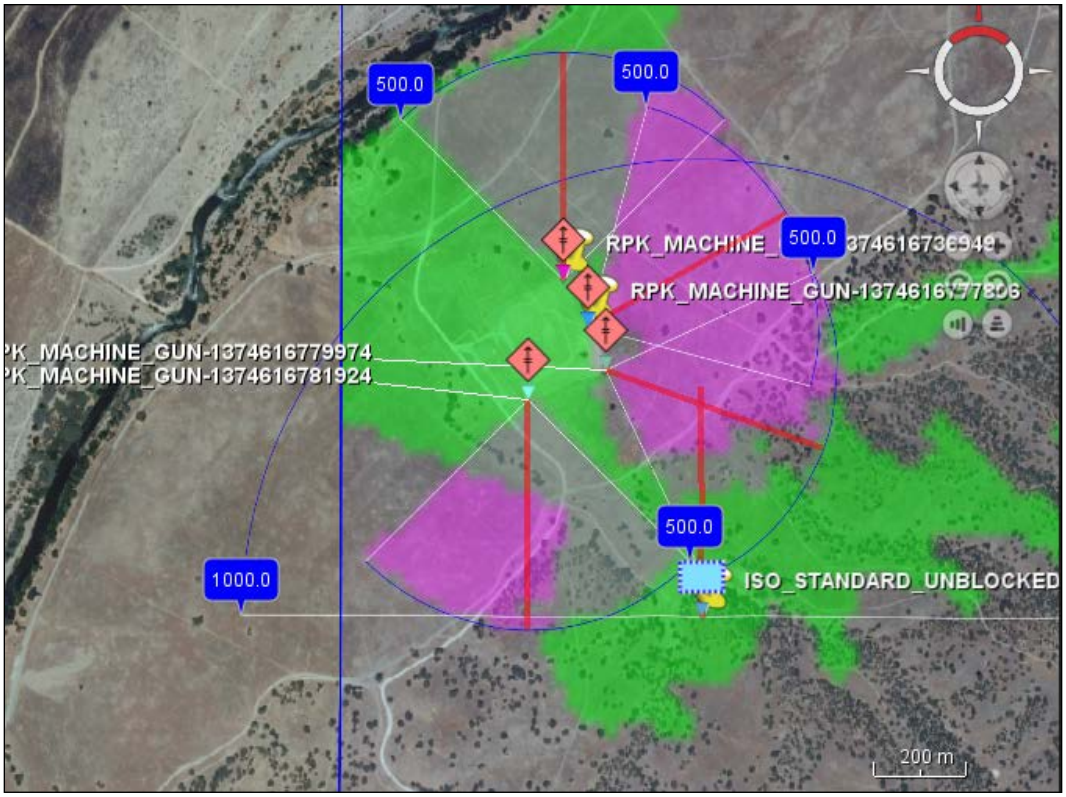


Figure 12. Sensor coverage and weapons fire of both hostile and friendly affiliations.



For some combinations of overlapping colors, there may be additional complications in visualizing results when the blended color closely resembles the background terrain imagery of the GIS map. For example, areas of fire by hostile weapons overlapping with areas of detection by friendly sensors blend magenta and green colors (i.e., case 4a in the list immediately above), resulting in a gray color that closely matches the imagery color of desert terrain within Raptor (Figure 13).

Figure 13. The blended color of magenta (for areas of fire by hostile weapons) and green (for areas of detection by friendly sensors) is a gray color that closely matches the color of the background terrain imagery.



4.7 Optimization of sensor selection and placement

4.7.1 Background

LMPT uses optimization of sensor selection and placement, a practical and powerful application of advanced sensor performance models in EASEE. The OSP algorithm incorporated in EASEE finds a near-optimal, approximate solution to a binary linear programming problem that has been defined based on probabilistic analyses of sensor performance (Vecherin et al. 2008, 2011). The algorithm optimizes selection and placement of vari-

ous types of sensors to satisfy specified detection coverage preferences while minimizing the number and cost of sensors and placement in unfavorable locations (e.g., due to inaccessibility or risk of vandalism).

For OSP scenarios, it is important to distinguish between two types of gridded points within the calculation domain. The first are sensor coverage points, at which probabilities of optimized sensors detecting a specified target are computed. The second are candidate sensor locations defining a finite sample of points within the calculation domain for potential placement of optimized sensors.

Because the OSP algorithm chooses optimal sensor locations among only the specified sample of finite candidate locations, the OSP solution is sensitive to the spatial sampling of the candidate locations. In general, sensor optimizations improve with greater sampling of candidate locations within the calculation domain but at the cost of increased computational time.

4.7.2 Input parameters

Performing sensor optimizations requires three basic inputs:

1. The probability-of-detection coverage goal for all coverage calculation points in the calculation domain
2. A list of available sensors of any type
3. Finite candidate locations for sensor placement within the calculation domain.

It is also possible to include several optional specifications:

1. Maximum numbers of available sensors of various types
2. Relative costs of sensors of various types
3. Relative costs for placement in various candidate locations
4. Different probability-of-detection coverage goals over the calculation domain
5. Priority areas for detection coverage

4.7.3 Polygonal inputs for sensor optimization

Recent improvements to the EASEE Java library and XML interface now allow optional specifications 3–5, listed in previous subsection, to be attributed via polygons over geographic areas.

A polygon is defined by a list of at least three vertices in latitude and longitude coordinates. Edges are generated by sequentially connecting vertices in the list. The generated edges must not intersect so that the resulting polygon is closed and simple. At least three vertices are required, and all points may not be collinear so that the polygon is not degenerate (i.e., a line). There are two polygon types for OSP:

1. An area-of-interest polygon attributed with a probability-of-detection goal (between 0 and 1) and coverage priority (as a positive integer)
2. A placement polygon attributed with a placement cost (as a nonnegative integer)

Area-of-interest polygons attribute coverage calculation points that lie within them with specified probability-of-detection goals and coverage priorities. Placement polygons attribute candidate sensor locations that lie within them with specified placement costs. Point-in-polygon tests are performed by a ray-casting algorithm within the Java Topology Suite.

Finally, whenever placement polygons are specified in EASEE, it is also possible to specify a placement cost (as a nonnegative integer) for the area that lies outside the polygons within the calculation domain.

For simplicity, polygon attributes are always set to the following specific values in LMPT that cannot be changed by the user:

1. Area-of-interest polygons have a probability-of-detection goal of 0.95 and a coverage priority of 1. Areas that lie outside area-of-interest polygons within the calculation domain have a probability-of-detection goal of 0. Note that coverage priority in these areas outside the polygons is unimportant when the probability-of-detection goal is 0 (i.e., these areas are completely ignored in coverage calculations when optimizing sensors).

2. Placement polygons have a placement cost of 0. Areas that lie outside placement polygons within the calculation domain have a placement cost of infinity whenever placement polygons are specified.

When no area-of-interest polygons are specified, the entire calculation domain has a probability-of-detection goal of 0.95 and a coverage priority of 1 although the priority value does not matter when there is only one priority level for the OSP calculation. When no placement polygons are specified, the entire calculation domain has a placement cost of 0.

4.7.4 Generation of candidate sensor-placement locations

As described earlier, the OSP algorithm chooses optimal sensor locations among only a sample of finite candidate locations. Sensor optimizations generally improve with greater sampling of candidate locations within the calculation domain but at the cost of increased computational time.

An XML input to EASEE for an OSP calculation may indicate a desired sampling rate of candidate sensor locations by a single positive integer. This specified sampling rate is used to generate candidate sensor locations for OSP in different ways, depending on whether placement polygons are specified or not.

When no placement polygons are specified, the physical distance of the perpendicular edges of the rectangular calculation domain on the Earth's surface is first computed. Coordinates along the longer axis of the rectangular domain are uniformly sampled by the specified sampling rate. Coordinates along the shorter axis are sampled by a number computed to match the distance spacing between sampled points along the longer edge, where the sampling rate along the shorter edge is always less than or equal to the rate along the longer edge and is equal only when the calculation domain is a perfect square. The candidate sensor locations are then generated on a grid containing all combinations of sampled coordinates along the longer and shorter axes of the rectangular calculation domain, where the coordinates along both axes are in uniform latitude and longitude degree angles.

When placement polygons are specified, a bounding rectangle is first computed around each individual polygon. For each bounding rectangle, can-

didate sensor locations are generated within the rectangle as described in the previous paragraph. Point-in-polygon tests are performed to identify generated candidate points that lie inside each polygon. Depending on the shape of a given polygon, it is possible that no candidate points exist in the polygon after executing this procedure. If this is the case, the procedure is repeated for such a polygon, where the sampling rate along the longer axis of the bounding rectangle is incrementally increased by one until there is a minimum of three sampled points that lie inside the polygon. Sensor optimizations are performed on all generated candidate sensor locations that lie within placement polygons via this method.

LMPT uses this process to generate candidate sensor locations whenever placement polygons are specified because placement polygons in LMPT have a zero placement cost and because areas outside the polygons are defaulted to an infinite placement cost. A different process for generating candidate sensor locations would be appropriate when the area outside the polygons has a non-infinite placement cost. Namely, in this case, candidate sensor locations should be generated over the entire calculation domain with specified placement costs depending on which polygons they lie within, if any.

4.7.5 Optimization around existing fixed sensors

In certain instances, a network of deployed sensors may already exist; and one desires to optimize a set of sensors to fulfill coverage preferences in areas that still have insufficient surveillance by the existing sensors. In such scenarios, it is possible to specify multiple fixed sensors in the environment and then a set of available sensors for subsequent placement by the OSP algorithm. In LMPT, the sensors to be optimized are called “floating sensors.”

4.7.6 Optimizing orientation for directional sensors

When optimizing placement of sensors with directional gains or non-omnidirectional fields of regard, it becomes necessary to determine not only the best geographic coordinates but also orientations of emplaced sensors. The existing OSP algorithm in EASEE can be extended to optimize sensor orientation and placement by introducing various combinations of candidate orientations and locations. Subsequently, the algorithm would select the best combination of orientations and locations to meet

coverage preferences while minimizing either the number or cost of sensors.

Currently, the OSP capability in EASEE does not include optimization of orientation for directional sensors. All sensors for placement by the OSP algorithm are modeled as omnidirectional; orientation does not apply. A future version of EASEE may include improvements to support optimization of orientation for directional sensors. However, it is still presently possible to optimize sensors around fixed sensors with directional gains or non-omnidirectional fields of regard.

5 Suggested Future Capabilities Based on Soldier Feedback

During various evaluation events over the duration of the LMPT project, including demos and Technical Support Operational Analysis (TSOA) exercises, we trained teams of Soldiers on the LMPT software and interviewed them for feedback. In general, the comments were positive, highlighting the relevance and utility of the sensor predictions, particularly during initial planning of combat outpost setup and subsequent reorganizations, and the usability of the powerful, intuitive interface within Raptor. Nonetheless, there were several suggestions for future improvements to the software tool. This section documents some of the more significant recommendations that were shared on multiple occasions.

When possible, we added to the LMPT software during the course of the project some of the commonly desired features that were relatively simple to include. Most of these additional capabilities that Soldiers requested are available in the current version of the tool and have been described in previous sections in this report, including the following:

1. Automated access to geospatial data from networked and web-based servers in real time. Specifically, we added an interface with LMPT software to access terrain-elevation data caches on a local hard drive from the NASA World Wind web database. See Section 4.2 for details.
2. Advanced polygonal inputs for areas of interest and sensor-placement constraints in optimal sensor-placement scenarios. See Section 4.7.3 for details.
3. Customizable fields of regard for sensor and weapons. See Section 4.4 for details.
4. Customizable platform orientations. See Section 4.5 for details.
5. Modeling direct-fire coverage using line of sight to the target and effective lethal range of various weapon systems. See Section 4.3.2 for details.
6. Force-on-force scenarios for simultaneous visualization of sensing and firing abilities of both friendly and hostile assets. See Section 4.6 for details.
7. A quick user guide for setting up, running, and interpreting software simulations. See Appendix C for a copy of the quick user guide.

However, we could not address within the existing project scope other noteworthy recommendations and ideas from Soldiers, mostly due to the greater level of research and development efforts required. These suggested capabilities would be worthwhile to pursue in the future:

1. Realistic foliage effects on propagation of various signal modalities. Although the software can presently ingest and utilize datasets for bare Earth terrain elevation data and land cover in signal propagation calculations, it does not model the decay of signals penetrating through vegetation and other surface structures, both natural and manmade, with adequate realism for various signal modalities. A promising approach is to extract information about surface vegetation and structures by analyzing high-resolution LiDAR point-cloud datasets for use in signal-propagation calculations.
2. Direct user customization of environmental features in simulations. In some instances, the conventional input data for terrain elevation and land cover do not accurately represent the actual surveillance environment upon arriving at and observing the scene. This may be because such datasets are out of date or sampled at insufficient resolutions. Generally, certain environmental details significantly affect signal-modeling predictions, which highly depend on the accuracy of the environmental input data. When no up-to-date and high-quality environmental data exists, an interactive option for directly modifying and re-constructing the simulated environment would be desirable. For example, tree lines, berms, buildings, various structures within an established combat outpost, and others could be manually inserted in the model environment via point and polygonal inputs with attribute menus where exact locations could be marked using GPS (Global Positioning System) devices.
3. Rapid display and toggling on and off detection contributions by individual sensors when multiple sensors are present (e.g., by clicking or hovering the cursor over particular sensors on the map). While it is useful to predict the probability of detecting a target by at least one of multiple sensors, many expressed that it would be also helpful to quickly consider the independent detection extents by the separate sensors within this prediction. In particular, the analysis and decision flow would be enhanced if this was possible to do as instantly as possible without setting up and running another calculation. At present, detection offered by individual sensors would need to be recomputed in a separate calculation, despite the fact that it has already been computed during the calculation of the combined coverage of multi-

ple sensors. A reasonable approach may be to store in a matrix the probabilities of detection by individual sensors. This matrix could be dynamically accessed to compute fused probabilities for various sensor combinations for visualization in real time. This recommendation also extends to visualizing areas of fire by individual weapons when multiple weapons are present.

4. The option to selectively visualize simultaneous detection coverage by various combinations of multiple sensors. Currently, the probability of detection by at least one sensor is computed when multiple sensors are present. In some applications, it is useful to assess locations where at least two, three, or more sensors can detect simultaneously (e.g., for radio localization by RF receiver arrays).
5. The additional optimization of platform orientation and placement. See Section 4.7.6 for details.
6. Implementation of the software on a tablet device (e.g., Android or iPad). This would allow the software to be available "on the go" to re-evaluate sensor coverage when slightly modifying sensor emplacement from the original plans, to assess detectability by hostile forces while actively engaged in a covert operation, to visualize one's position within sensor prediction outputs via real time GPS tracking on tablet devices, etc. For example, the display of where one could be detected (e.g., visibility, acoustically, by radar, etc.) by an enemy could be updated once per minute on the tablet device during travel.

6 Deployment of EASEE Software for External Applications

There are a few necessary steps for deploying and running EASEE. Foremost, the EASEE distribution includes three components: (1) a Java executable of EASEE (EASEELib.jar), (2) an ext folder, and (3) a tmp folder. All three of these components should be placed in the same directory.

From this directory, it is possible to run the markup interface for EASEE in one of two ways:

1. Command line. Write markup input to a file (e.g., tmp\example.txt). Then run the following system call: `java -cp ext\*;ext\gdal\*;* mil.army.usace.easee.JsonXmlProcessor tmp\example.txt.`
2. TCP/IP (Transmission Control Protocol/Internet Protocol) socket. Stream markup through socket port 4444 after running the following system call: `java -cp ext\*;ext\gdal\*;* mil.army.usace.easee.SocketPort.`

A user may directly perform the above steps for a manually written markup input in XML or JSON, or a client application may perform the above steps, as is done by the LMPT plug-in in Raptor. In either case, EASEE will execute the calculation and save the results in a KMZ file at the user-specified location indicated within the markup input. For optimal sensor-placement scenarios, EASEE will also produce an output XML with selections and locations of optimized sensors. This output is saved at a specified location or streamed through the TCP/IP socket port.

7 Conclusions

An appropriately designed network of surveillance sensors can provide much-needed protection of forces at small, forwardly deployed bases in hostile areas. Completing this task effectively, however, is not straightforward and requires expert understanding of the terrain and weather impacts on detection by various sensor systems. It is critical to understand limitations on sensor performance, given the realistic conditions of the surveillance environment, and to account for these effects when planning sensor configurations.

Because they often have limited specialized expertise in sensor technologies, Soldiers can benefit from software tools that characterize environmental effects on sensor performance during defense planning. To address this need, the ASA(ALT) DFP-funded project LMPT has incorporated the ERDC-developed EASEE software for battlefield signal modeling within the powerful Raptor GIS environment. This integrated software capability enhances usability and enables simultaneous visualization of relevant information from other products within Raptor.

The integration of EASEE and Raptor is enabled by an interface based on markup file exchange and standardized using an XML schema. Several advantages to this approach include key design considerations, such as compatibility, extensibility, maintainability, and reusability.

In conjunction with this, the LMPT project added to the core EASEE computational engine additional modeling capabilities that address issues relevant to force protection and surveillance planning for combat outposts and that Soldiers suggested in test events. Added features include realistic RF transmission and radar models, limits on detection ranges of optical and infrared cameras, estimation of areas of fire for weapons, customizable fields of regard for sensors and weapons, visualization of force-on-force scenarios with friendly and hostile assets, and optional polygonal inputs for specifying areas of interest and sensor placement for sensor optimization. Other capability recommendations from Soldier feedback have been documented and will be considered in future research and development.

In the future, the markup interface for EASEE developed as part of LMPT will be continually expanded and reused within a variety of client applications and environments. One of the main purposes of this report is to facilitate such technology transition efforts by serving as a helpful and thorough resource for collaborating software developers by describing the markup interface, documenting details of its XML schema elements (in Appendix A), and reviewing the modeling capabilities of EASEE.

References

Burdic, W. S. 1984. *Underwater Acoustic System Analysis*. Englewood Cliffs, NJ: Prentice-Hall.

Vecherin, S. N., D. K. Wilson, and C. L. Pettit. 2008. *Optimal Sensor Placement with Terrain-Based Constraints and Signal Propagation Effects*. ERDC/CRREL TR-08-24. Hanover, NH: U.S. Army Engineer Research and Development Center.

Vecherin, S. N., D. K. Wilson, and C. L. Pettit. 2011. Optimal sensor placement with signal propagation effects and inhomogeneous coverage preferences. *International Journal of Sensor Networks* 9 (2): 107–120.

Wilson, D. K., R. Bates, and K. K. Yamamoto. 2009. *Object-Oriented Software Model for Battlefield Signal Transmission and Sensing*. ERDC/CRREL TR-09-17. Hanover, NH: U.S. Army Engineer Research and Development Center.

Wilson, D. K., C. T. Borden, E. S. Bettencourt, and K. K. Yamamoto. 2013. Characterization of infrared imaging performance within a general statistical framework for environmental impacts on battlefield signals and sensing. In *Proceedings of SPIE, Modeling and Simulation for Defense Systems and Applications VIII, 11 June*, ed. E. J. Kelmelis, 8752:87520J. doi:10.1117/12.2018246.

Yamamoto, K. K., D. K. Wilson, and C. L. Pettit. 2010. *Probability and Statistics in Sensor Performance Modeling*. ERDC/CRREL TR-10-12. Hanover, NH: U.S. Army Engineer Research and Development Center.

Yamamoto, K. K., S. N. Vecherin, D. K. Wilson, C. T. Borden, E. Bettencourt, and C. L. Pettit. 2012. General software for multimodal battlefield signal modeling and optimal sensor placement: Environmental awareness for sensor and emitter employment (EASEE). In *Proceedings of the IEEE International Carnahan Conference on Security Technology (ICCST), 15–18 October, Boston, MA*, 66–77. doi: 10.1109/CCST.2012.6393539.

Yamamoto, K. K., N. J. Reznicek, and D. K. Wilson. 2013. Integration of radio-frequency transmission and radar in general software for multimodal battlefield signal modeling. In *Proceedings of SPIE, Modeling and Simulation for Defense Systems and Applications VIII, 11 June*, ed. E. J. Kelmelis, 8752:875203. doi:10.1117/12.2018182.

Yamamoto, K. K., Nathan J. Reznicek, and D. K. Wilson. 2015. *Integration of Radio-Frequency Transmission and Radar in Multimodal Signal-Modeling Software: Environmental Awareness for Sensor and Emitter Employment (EASEE)*. ERDC/CRREL TR-15-12. Hanover, NH: U.S. Army Engineer Research and Development Center.

Appendix A: Description of XML Schema Elements for EASEE Software Interface

This appendix describes the data content and structure of elements within XML input and output files for the EASEE markup interface. We intend it as a reference for software developers interfacing client applications with EASEE via its markup interface. In particular, it provides a thorough explanation of the XML schema for the EASEE markup interface. This schema standardizes the XML markup for input files requesting EASEE calculations and for output files with calculation results from EASEE.

The following sections provide details on the elements in the XML schema for the EASEE markup interface at the time of the LMPT project. Although the schema will surely evolve, descriptions of the original schema may remain relevant, assuming that the schema's general design remains the same.

Input/output markup files

The root element in the XML schema for EASEE is called `EASEE`. All input and output XML files for the EASEE markup interface must contain this `EASEE` root element. Inside the `EASEE` root element, it is possible to have two child elements, called `Simulation` and `Results`. An input XML file requesting an EASEE calculation must contain the `EASEE` root element with the `Simulation` element. An output XML file with EASEE calculation results would contain the `EASEE` root element with the `Results` element. The `EASEE` root element for the output XML file could also have a copy of the `Simulation` element from the input XML file for the calculation if Boolean `echoInput = true` within `XmlOutputType` within `Output` within `Simulation` within the `EASEE` root element of the input XML file.

Child elements of the Simulation element within the root EASEE element for setting up calculation request

BackgroundNoise element

The background noise model that EASEE uses can have a significant effect on calculation results for sensor performance.

Models for background noise must be specified for each modality (i.e., `acoustic`, `infrared`, `radiofrequency`, and `seismic`). Select background noise models from enum lists defined for each modality (i.e., `BackgroundNoiseAcousticType`, `BackgroundNoiseInfraredType`, `BackgroundNoiseRadioFrequencyType`, and `BackgroundNoiseSeismicType`).

Environment element

The atmosphere, land cover, and terrain can greatly affect predictions of sensor performance.

Although the `Environment` element is required within the `Simulation` element, all of the child elements in `Environment` are optional. If they are not specified, these would be mapped to EASEE defaults. These child elements of `Environment` are described here in separate subsections ("Atmospheric element," "Landcover element," and "Terrain element").

Atmospheric element

The Object `afwa` field within this element is intended for using numerical weather predictions from the Air Force Weather Agency. At the time of this writing, this is a placeholder to be used in the future.

The `WindStabilityType` field allows for specification of wind characteristics in the environment. Within this field, there are other subfields:

1. `CloudCover` for specifying fractional coverage for low-, mid-, and high-altitude clouds
2. `StabilityCategoryType` for specifying atmospheric stability based on Pasquill-Gifford stability types
3. `WindCategoryType` for specifying wind strength
4. Double `windDirection` for specifying wind direction by using Cartesian convention (i.e., in radians, 0 = to east, positive counterclockwise)

Landcover element

EASEE can include land cover effects in signal predictions. Land cover data can be specified in two widely used formats: GeoCover and the National Land Cover Dataset (NLCD) 2001.

There are BasicLandcoverType fields within Landcover for both GeoCover and NLCD2001 land cover data. Within BasicLandcoverType are other subfields:

1. String filePath specifies the uniform resource identifier (URI) to the land cover data.
2. String type specifies a single land cover type to be applied either over the entire calculation domain or where data in a land-cover raster file is missing. The string must be one of the enumerations defined within the GeoCoverLCTypes or NationalLandcoverDataset2001Types Java classes in EASEE for GeoCover and NLCD2001 land cover data, respectively.

Terrain element

The Terrain element contains only one field, string source, for specifying URI to the terrain data file (e.g., GeoTIFF). If this element is left blank, EASEE will attempt to parse terrain elevation data from the NASA World Wind data cache on the local machine's hard drive.

GeographicDomain element

A rectangular geographic domain for the calculation must be specified.

The GeographicDomain element contains a GridType child element, which contains the following fields:

1. An optional field, AltitudeModeType altitudeMode, has an enum list of how altitude coordinates for the geographic domain are referenced (e.g., ABOVE_GROUND_LEVEL, ABOVE_MEAN_SEA_LEVEL, or ABSOLUTE). This is not necessary when defining a two-dimensional coordinate for the platform, and if unspecified, EASEE will use the default altitude mode and value for the variable platform category and type, defined by the VariablePlatform element.
2. DoubleListType southwest specifies the most southwest latitude, longitude, and altitude coordinate of the geographic domain.
3. DoubleListType northeast specifies the most northeast latitude, longitude, and altitude coordinate of the geographic domain.
4. String srsName specifies a name that is based on OpenGIS—at the time of this writing, this is a placeholder for future use.

The order of coordinates in `List<Double> value` within `DoubleListType` `southwest` and `northeast` is (x,y) = (longitude, latitude) for when `BigInteger dimensions` = 2. It is (x,y,z) = (longitude, latitude, altitude) for when `BigInteger dimensions` = 3.

For increased flexibility, the geographic domain can be specified in three-dimensional space. At minimum, it is fine to specify the geographic domain in two dimensions only.

Along with the `GridType` child element, the `GeographicDomain` element also contains an optional subfield, `ResolutionType`, which contains the following fields: (1) `GridDivisionType`, which allows specifying a fixed number of calculation points over the geographic domain, and (2) double `meters` for specifying a uniform spatial calculation resolution in meters over the geographic domain.

The `GridDivisionType` field contains further subfields:

1. `BigInteger columns` for specifying the number of calculation points sampled horizontally on the rectangular geographic domain in uniform longitudinal coordinates
2. `BigInteger rows` for specifying the number of calculation points sampled vertically on the rectangular geographic domain in uniform latitudinal coordinates

The higher the number of calculation points sampled horizontally and vertically, the greater the calculation resolution with increased computation times.

Because the `ResolutionType` element is optional, if it is not specified, EASEE will use default calculation resolution settings.

OptimalSensorPlacement element

The only element within `Simulation` that is optional is the `OptimalSensorPlacement` element. If this element is not provided, then OSP is not performed and a sensor/emitter footprint is calculated instead.

The OptimalSensorPlacement element contains one required child element for specifying the inventory of candidate platforms for OSP (specified in PlatformInventoryType). The PlatformInventoryType element contains a list of PlatformInInventoryType elements with the following subfields:

1. Two required string tokens, category and type, for platform category and type identify the candidate platform. The platform category string must be the name of one of the Java classes in the platform package in EASEE. The platform type string must be one of the PlatformTypes enumerations defined within the Java class in EASEE for the specified platform category.
2. An optional field, double cost, specifies the (relative) cost of the candidate platform. If unspecified, EASEE defaults to cost = 1.
3. An optional field, BigInteger supply, specifies available supply of the candidate platform. If unspecified, EASEE defaults to supply = 1.
4. Two optional string tokens, groupName and name, name the candidate platforms.

The OptimalSensorPlacement element also contains two optional child elements for defining area-of-interest and sensor-placement polygons (i.e., CoverageGoalsType and PlacementConstraintsType, respectively).

CoverageGoalsType and PlacementConstraintsType each have inner child elements with the same name, Polygon. Despite the same name, these Polygon elements within CoverageGoalsType and PlacementConstraintsType are different (i.e., CoverageGoalsType.Polygon and PlacementConstraintsType.Polygon).

Common fields in each Polygon element are (1) an optional field, AltitudeModeType altitudeMode, with an enum list of how altitude coordinates for the geographic domain are referenced (e.g., ABOVE_GROUND_LEVEL, ABOVE_MEAN_SEA_LEVEL, or ABSOLUTE); (2) DoubleListType coordinates for specifying the latitude, longitude, and altitude coordinates of the polygon vertices; and (3) string srsName for specifying a name that is based on OpenGIS—at the time of this writing, this is a placeholder for future use.

The order of coordinates in List<Double> value within DoubleListType coordinates is (x,y) = (longitude, latitude) for when BigInteger dimensions = 2. It is (x,y,z) = (longitude, latitude, altitude) for when BigInteger dimensions = 3.

As with the rectangular calculation domain defined by the `GeographicDomain` element, these polygons must be specified in two dimensions at minimum but can also be specified in three-dimensional space, if desired. However, at the time of this writing, only polygons specified in two dimensions are supported; and, in turn, the field `AltitudeModeType altitudeMode` and the altitude coordinate in the field, `DoubleListType coordinates`, are unused.

The following are differences between `Polygon` elements within `CoverageGoalsType` and `PlacementConstraintsType` (i.e., `CoverageGoalsType.Polygon` and `PlacementConstraintsType.Polygon`, respectively):

1. `Polygon` within `CoverageGoalsType` (i.e., `CoverageGoalsType.Polygon`) contains a required field, `BigInteger priority`, for specifying a positive integer for the priority of the area-of-interest polygon, where 1 = highest priority. The optional field, double `probabilityOfDetection`, is for specifying the desired probability-of-detection threshold, between 0 and 1, to be achieved by sensors optimized by OSP, given a particular constant false alarm rate whose default value is set to 10^{-3} in EASEE. If unspecified, EASEE defaults to `probabilityOfDetection` = 0.95.
2. `Polygon` within `PlacementConstraintsType` (i.e., `PlacementConstraintsType.Polygon`) contains an optional field, double `cost`, for specifying a cost for placing a sensor within the sensor-placement polygon. If unspecified, EASEE defaults to `cost` = 0.

Finally, the following are the two remaining optional fields in the `OptimalSensorPlacement` element:

1. Double `defaultCost` specifies the cost associated with placing a sensor within the calculation domain (defined by the `GeographicDomain` element) but outside sensor-placement polygons (defined by the `PlacementConstraintsType` element). If unspecified, `defaultCost` is either set to (1) infinity when sensor-placement polygons are specified within the `PlacementConstraintsType` element, so that optimized sensors are never placed outside sensor-placement polygons, or (2) zero when no sensor-placement constraint polygons are specified, so that optimized sensors may be placed anywhere within the calculation domain specified by the `GeographicDomain` element.

2. `BigInteger candidateGridSize` controls a number of candidate sensor locations to be generated and analyzed by the OSP algorithm. Increasing this number results in more candidate sensor locations for the algorithm and, in turn, more optimal sensor configurations although with longer computation times.

Output element

The `Output` element is for specifying the URI to the KMZ (zipped Keyhole Markup Language file) and XML calculation outputs generated by EASEE (i.e., the string `kmz` and `XmlOutputType xml` fields, respectively). The `kmz` and `xml` URI fields are both optional. If unspecified, KMZ and XML files with calculation results will not be saved.

Typically, at least one of the two files should be specified to obtain calculation results. For example, only the KMZ output file may be desired when performing a sensor/emitter footprint calculation. If only the coordinates of the optimized sensors are desired, then the XML output file is sufficient. However, if the corresponding detection coverage by the optimized sensors is also desired, a URI for the KMZ output file must also be specified.

The `XmlOutputType` element contains two subfields:

1. A required field, string `path`, specifies the URI where the XML output should be saved.
2. An optional field, Boolean `echoInput`, specifies whether the XML output should contain a copy of the `Simulation` element from the input XML file with the EASEE calculation request. If `echoInput = true`, the output XML file with EASEE calculation results would include a copy of `Simulation`. If unspecified, EASEE defaults to `echoInput = true`.

PlatformList element

At the minimum, all EASEE calculations, for both sensor/emitter footprints and OSP, require specification of one, and only one, variable platform. This platform is moved to all points in the calculation domain, which is defined by the `GeographicDomain` element. This results in multiple probability-of-detection calculations at each spatial location, visualized by a colored pixel in the generated KMZ output.

The PlatformList element contains two child elements:

1. An optional field, FixedPlatformList fixedPlatforms, specifies one or more fixed platforms for the EASEE calculation. This field is not necessary when performing an OSP scenario where the optimized sensors are placed by EASEE; however, they may still be specified for OSP calculations if sensor optimization around already deployed sensors is desired.
2. A required field, VariablePlatform variablePlatform, specifies the single variable platform for the EASEE calculation.

The FixedPlatformList and VariablePlatform child elements are described here in separate subsections ("FixedPlatformList element" and "VariablePlatform element").

FixedPlatformList element

The FixedPlatformList element is for specifying one or more fixed platforms for an EASEE calculation and contains a list of Platform elements with the following subfields:

1. Two required string tokens, category and type, for platform category and type identify the candidate platform. The platform category string must be the name of one of the Java classes in the platform package in EASEE. The platform type string must be one of the PlatformTypes enumerations defined within the Java class in EASEE for the specified platform category.
2. An optional field, Boolean active, specifies the status of the platform. If active = true, the platform is included in sensor/emitter footprint for OSP calculations. In OSP calculations, sensor optimizations are performed around active fixed sensors, taking the coverage they offer into account. If active = false, the platform is not included in sensor/emitter footprint calculations. In OSP calculations, an inactive fixed sensor is treated as a candidate fixed-position platform for OSP if CalculationModeType is set to OPTIMAL_SENSOR_PLACEMENT within CalculationType within Scenario within Simulation. In this case, if OSP decides that the inactive fixed sensor is optimal, the active field for the platform would be converted to active = true and returned within the list of optimized sensors in the output XML file with EASEE calculation results. If unspecified, EASEE defaults to active = true.

3. An optional field, `AffiliationType affiliation`, specifies the platform's affiliation (i.e., `UNKNOWN`, `FRIEND`, `NEUTRAL`, and `HOSTILE`). If `SituationType` is set to `FRIENDLY_MONITORING_HOSTILE` within `Scenario` within `Simulation`, then all friendly sensing platforms would treat signals from all hostile emitting platforms as signals of interest and signals from all friendly emitting platforms as interfering noise. Also, whenever friendly direct-fire weapon platforms are specified, their ability to fire at hostile platforms is computed. If `SituationType` is set to `HOSTILE_MONITORING_FRIENDLY` within `Scenario` within `Simulation`, then all hostile sensing platforms would treat signals from all friendly emitting platforms as signals of interest and signals from all hostile emitting platforms as interfering noise. Also, whenever hostile direct-fire weapon platforms are specified, their ability to fire at friendly platforms is computed. Signals from unknown emitting platforms are always treated as signals of interest. Signals from neutral emitting platforms are always treated as interfering noise. If unspecified, EASEE defaults to `affiliation = UNKNOWN`.
4. An optional field, `FieldOfRegardType fieldOfRegard`, allows specifying azimuth and elevation angles and a finite range for a platform's field of regard (for sensors) and area of fire (for direct-fire weapons). This field contains the following optional subfields: (1) double `azimuthWidth`, restricted to values from 0 to 360, for the azimuthal (horizontal) angular extent in degrees (if unspecified, EASEE defaults to `azimuthWidth` = 360); (2) double `elevationWidth`, restricted from 0 to 180, for the elevation (vertical) angular extent in degrees (if unspecified, EASEE defaults to `elevationWidth` = 180); and (3) double `range` for specifying a positive detection range in meters (if unspecified, EASEE defaults to `range` = 10^5).
5. Two optional strings, `groupName` and `name`, name the platform.
6. An optional field, `StateHistoryType stateHistory`, specifies platform state information (e.g., attitude, location, and velocity) as a function of time. At the time of this writing, this is a placeholder for future use in setting up EASEE calculations over multiple time steps. For static (single time step) calculations, it is sufficient to use the `StateType` state field described next. This field, `StateHistoryType stateHistory`, is optional; however, if unspecified, the field, `StateType state`, must be specified.
7. An optional field, `StateType` state, specifies platform state information for a static (single time step) EASEE calculation.

The `StateType` element contains three child elements:

1. An optional field, `AttitudeType attitude`, specifies pitch, roll, and yaw of the platform. This field contains the following optional subfields:

    a. Double `pitch`, restricted from −90 to +90, specifies the pitch of the platform in degrees, where zero degrees aligns the axis passing through the front and back of the platform with the tangent to the Earth's ellipsoid. This parameter rotates the platform around the axis passing through the sides of the platform, where the front of the platform points upwards and downwards for positive and negative degree values, respectively. If unspecified, EASEE defaults to `pitch` = 0.
    b. Double `roll`, restricted from −180 to +180, specifies the roll of the platform in degrees, where zero degrees aligns the axis passing through the sides of the platform with the tangent to the Earth's ellipsoid. This parameter rotates the platform around the axis passing through the front and back of the platform, where rotations counterclockwise are positive degree values when looking from the front of the platform into the back. At the time of this writing, this parameter is not customizable and is always set to zero degrees in EASEE.
    c. Double `yaw`, restricted from 0 to 360, specifies the yaw of the platform in degrees, where zero degrees aligns the front of the platform towards north. This parameter rotates the platform around the axis orthogonal to the Earth's ellipsoid and passing through the center of the platform, where degree values increase clockwise from north (i.e., towards eastward direction).

If the entire field, `AttitudeType attitude`, is unspecified, EASEE defaults to `pitch` = `roll` = `yaw` = 0.

2. A required field, `LocationType location`, specifies the location of the platform. This field contains the following subfields: (1) an optional field, `AltitudeModeType altitudeMode`, with an enum list of how altitude coordinates for the geographic domain are referenced (e.g., `ABOVE_GROUND_LEVEL`, `ABOVE_MEAN_SEA_LEVEL`, or `ABSOLUTE`), which is not necessary when defining a two-dimensional coordinate for the platform; and, if unspecified, the default altitude mode and value in EASEE for the particular platform category and type is used; (2) `DoubleListType point` for specifying the latitude, longitude, and altitude coordinate of the plat-

form; and (3) string srsName for specifying a name that is based on OpenGIS—at the time of this writing, srsName is a placeholder for future use.

The order of coordinates in List<Double> value within DoubleListType coordinates is (x,y) = (longitude, latitude) for when BigInteger dimensions = 2. It is (x,y,z) = (longitude, latitude, altitude) for when BigInteger dimensions = 3.

As with coordinates in GridType, CoverageGoalsType.Polygon, and PlacementConstraintsType.Polygon elements, the platform location coordinate must be specified in two dimensions at minimum but can also be specified in three-dimensional space, if desired.

3. VelocityType velocity specifies the velocity of the platform based on double values for bearing, ground speed, and vertical velocity. At the time of this writing, this element is a placeholder for future use.

VariablePlatform element

The VariablePlatform element is a modified version of the Platform element. The VariablePlatform contains many of the same child elements as the Platform element except for the following:

1. The optional field, Boolean active, for specifying the status of the platform as this should always be active = true for the variable platform in a valid sensor/emitter footprint or OSP calculation in EASEE.
2. The optional field, StateHistoryType stateHistory, for specifying platform-state information (e.g., attitude, location, and velocity) as a function of time or the optional field, StateType state, for specifying platform-state information for a static (single time step) EASEE calculation. These are omitted in the VariablePlatform element because they require defining a LocationType element for specifying a fixed location for the platform when the variable platform, by definition, does not have a fixed coordinate.

The VariablePlatform element does contain an optional field, AttitudeType attitude, for specifying pitch, roll, or yaw of the platform.

Guidance on using the Platform element for optimal sensor-placement and

sensor/emitter footprint calculations

For an OSP scenario, a variable emitting/target platform is all that is strictly required, in which case `PlatformInventoryType` within `OptimalSensorPlacement` would also need to specify the available candidate sensing platforms for placement. The KMZ that is generated in this case is the probability-of-detection footprint of all the optimized sensors for detecting the specified (variable) emitting/target platform.

However, it is also possible to provide a list of fixed-position platforms for OSP (i.e., `List<Platform>` within `FixedPlatformList`). As described in the section "FixedPlatformList element," fixed-position platforms with Boolean `active` = false would be treated as candidate fixed-position platforms for OSP. It is also possible to specify fixed-position platforms with `active` = true for an OSP scenario. In this case, these fixed-position platforms with `active` = true would be considered as already deployed and active; and the OSP algorithm would optimize sensing platforms around these already deployed fixed-position platforms, accounting for the detection coverage provided by these already deployed fixed-position platforms.

For a sensor/emitter footprint scenario, a variable platform and at least one fixed-position platform are required for a valid EASEE calculation.

Scenario element

The required `Scenario` element is for specifying basic characteristics of the EASEE calculation.

The `Scenario` element contains two required child elements:

1. `CalculationType calculation` specifies the desired type of calculation. The `CalculationType` element is described in more detail in the next subsection ("CalculationType element").
2. `SituationType situation` specifies the calculation situation by choosing between three enums—`FRIENDLY_MONITORING_HOSTILE`, `HOSTILE_MONITORING_FRIENDLY`, and `ALL_MONITORING_ALL`.

The `CalculationType` and `SituationType` child elements are described here in separate subsections ("CalculationType element" and "SituationType element").

CalculationType element

The CalculationType element contains a required child element CalculationModeType, which defines an enumerated list of many calculation modes (i.e., PROBABILITY_OF_DETECTION, PROBABILITY_OF_FALSE_ALARM, OPTIMAL_SENSOR_PLACEMENT, OPTIMAL_SENSOR_ACTIVATION, SIGNAL_TO_NOISE_RATIO, SIGNAL_STRENGTH, and NOISE_STRENGTH). Some of these calculation modes are not yet supported. At the time of this writing, the two modes for CalculationModeType, PROBABILITY_OF_DETECTION and OPTIMAL_SENSOR_PLACEMENT, are supported by the EASEE markup interface.

The optional Booleans within CalculationType for different modalities (i.e., acoustic, chemBio, infrared, radiofrequency, seismic, and visible) allow turning on and off certain signal modalities for the EASEE calculation. If platforms in the calculation emit or sense a signal modality that is turned off (e.g., acoustic = false, radiofrequency = false, etc.), then signals for that modality would not be emitted or sensed. If the Boolean field for turning on and off is unspecified for a signal modality, EASEE defaults to Boolean = true for that signal modality.

SituationType element

The SituationType element defines an enumerated list describing the detection mode of the calculation (i.e., FRIENDLY_MONITORING_HOSTILE, HOSTILE_MONITORING_FRIENDY, and ALL_MONITORING_ALL).

If SituationType is set to FRIENDLY_MONITORING_HOSTILE, all friendly sensing platforms treats signals from all hostile emitting platforms as signals of interest and signals from all friendly emitting platforms as interfering noise. Whenever friendly direct-fire weapon platforms are specified, their ability to fire at hostile platforms is computed. If SituationType is set to HOSTILE_MONITORING_FRIENDLY, all hostile sensing platforms treats signals from all friendly emitting platforms as signals of interest and signals from all hostile emitting platforms as interfering noise. Whenever hostile direct-fire weapon platforms are specified, their ability to fire at friendly platforms is computed.

The ALL_MONITORING_ALL enumeration for SituationType is intended for use with a purely emitting variable platform (defined by VariablePlatform element) that does not sense signals itself. Two separate calculations with detection mode set to FRIENDY_MONITORING_HOSTILE and HOSTILE_MONITORING_FRIENDY are performed, where the EASEE markup interface resets the affiliation of the variable platform (defined by AffiliationType element in VariablePlatform) to HOSTILE and FRIEND, respectively. Whenever direct-fire weapon platforms are also specified, the ability of friendly weapons firing on hostile targets and vice versa are separately modeled as well. All of these results are merged together using various colored pixels to represent friendly and hostile detection and fire.

Signals from unknown emitting platforms are always treated as signals of interest. Signals from neutral emitting platforms are always treated as interfering noise. If unspecified, EASEE defaults to affiliation = UNKNOWN.

SignalPropagation element

In EASEE, propagated signals are computed by a signal propagation model for each signal modality. EASEE contains libraries of multiple physics-based propagation models for each modality, including acoustic, chemical and biological, infrared, RF, seismic, and visible.

A specific propagation model in EASEE (e.g., ImpedancePlaneModel, CNParabolicEqn, SCIPUFFModel, LineOfSightPropagator, GeomSpreadPropagator, etc.) is used for each signal modality that is included in the desired EASEE simulation (i.e., all signal modalities with Boolean = true within CalculationType).

Propagation models to use for specific signal modalities are specified by an optional string token for each signal modality field within SignalPropagation (i.e., acoustic, chemBio, infrared, radiofrequency, seismic, and visible). It is possible to simply specify the following three model fidelity strings for each of the signal modality fields: “low,” “medium,” and “high.” These strings are automatically mapped to predefined propagation models of corresponding fidelity. Alternatively, it is also possible to specify explicitly a particular propagation model to use in EASEE. For this, it is necessary to provide a string with a valid Java class path name for the propagation model (e.g.,

mil.army.usace.easee.acoustic.ImpedancePlaneModel;
mil.army.usace.easee.radiofreq.EmpirePropModelFullUnstruct;
mil.army.usace.easee.propagate.GeomFootprintPropagator; etc.).

Timing element

The Timing element is required but, at the time of this writing, is a placeholder for future use in setting up EASEE calculations over multiple time steps. Until this feature is supported, it is fine to specify the following required fields within the Timing element with any values: (1) Duration interval for specifying the time duration of the calculation (e.g., "P0Y0M0DT0H0M0.000S" for zero time duration), (2) BigInteger timeSteps for specifying the number of discrete time steps for calculation within the indicated time duration (e.g., "1" for a single time step), and (3) XMLGregorianCalendar startDateTime for specifying the starting date and time of the calculation (e.g., "2013-03-05T18:46:07.919Z").

Child elements of Results element within root EASEE element with calculation results

The optional Results element in the root EASEE element contains two optional fields: (1) OSPResults optimalSensorPlacement containing selection and placement of optimized sensors from the OSP algorithm and (2) string kmzUri that indicates the URI where the KMZ output file was saved.

The OSPResults element contains a required inner PlatformList element (i.e., OSPResults.PlatformList). This inner PlatformList element contains a single list of Platform elements, described in the section "FixedPlatformList element." The list of Platform elements within this OSPResults.PlatformList element would contain all the platforms that the OSP algorithm either placed from the inventory of candidate platforms for OSP (specified in PlatformInventoryType) or activated from candidate fixed-position platforms (i.e., platforms in FixedPlatformList whose field, Boolean active, was changed from false to true—see the section "FixedPlatformList element"). Typically, two sets of information in the Platform element for each optimized sensor in OSPResults.PlatformList would be of interest:

1. The string tokens category and type for platform category and type that identifies the optimized sensor

2. The `StateType` `state` element with the placement location for the optimized sensor

Example markup inputs for EASEE calculation

JSON markup input example

```
{"easee": {"Simulation": {
     "Timing": {
          "startDateTime": "2013-03-05T18:46:07.919Z",
          "timeSteps": 1,
          "interval": "P0Y0M0DT0H0M0.000S"
     },
     "Environment": {
          "Landcover": {"nlcd2001": {
               "filePath": "dist\\NLCD_FtPolk.tif",
               "type": "BARREN_LAND"
          }},
          "Terrain": {"source": ""},
          "Atmospheric": {"windStability": {
               "wind": {"windSpeed": 0.4},
               "windDirection": 1.5707963267948966,
               "cloudCover": {
                    "high": 3,
                    "mid": 5,
                    "low": 7
               },
               "stability": "UNSTABLE"
          }}
     },
     "Scenario": {
          "calculation": {
               "visible": true,
               "infrared": false,
               "acoustic": true,
               "seismic": true,
               "chemBio": true,
               "type": "PROBABILITY_OF_DETECTION",
               "radioFrequency": true
          },
```

```
            "situation": "FRIENDLY_MONITORING_HOSTILE"
        },
        "PlatformList": {
            "FixedPlatforms": {"Platform": {
                "category": "UnattendedGroundSensor",
                "fieldOfRegard": {
                    "elevationWidth": 180,
                    "azimuthWidth": 360
                },
                "state": {
                    "attitude": {
                        "yaw": 180,
                        "roll": 0,
                        "pitch": 0
                    },
                    "location": {
                        "point": {
                            "value": "-120.789561 35.753567
10",
                            "dimensions": 3
                        },
                        "altitudeMode": "aboveGroundLevel"
                    }
                },
                "active": true,
                "affiliation": "FRIEND",
                "type": "RFDF"
            }},
            "VariablePlatform": {
                "category": "RadioFreqTransmitter",
                "affiliation": "HOSTILE",
                "type": "RFDF"
            }
        },
        "BackgroundNoise": {
            "infrared": "NWP_TERRAIN",
            "acoustic": "ATLAS_VERY_LOUD",
            "seismic": "CRREL_OPEN_RURAL",
            "radioFrequency": "HIGH_VHF_UHF_X"
```

```
      },
      "Output": {"kmz": "../example.kmz"},
      "SignalPropagation": {
           "visible": "low",
           "infrared": "high",
           "acoustic": "low",
           "seismic": "medium",
           "chemBio": "low",
           "radioFrequency": "high"
      },
      "GeographicDomain": {
           "grid": {
                "southwest": {
                     "value": "-120.849858333333 35.74046 1.9",
                     "dimensions": 3
                },
                "northeast": {
                     "value": "-120.772736 35.768275 10",
                     "dimensions": 3
                },
                "altitudeMode": "aboveGroundLevel"
           },
           "resolution": {"gridDivision": {
                "columns": 200,
                "rows": 200
           }}
      }
}}}
```

XML markup input example translated from JSON above

```
<easee>
    <Simulation>
        <BackgroundNoise>
            <acoustic>ATLAS_VERY_LOUD</acoustic>
            <infrared>NWP_TERRAIN</infrared>
            <radioFrequency>HIGH_VHF_UHF_X</radioFrequency>
            <seismic>CRREL_OPEN_RURAL</seismic>
        </BackgroundNoise>
        <Environment>
```

```
            <Atmospheric>
                <windStability>
                    <cloudCover>
                        <low>7.0</low>
                        <mid>5.0</mid>
                        <high>3.0</high>
                    </cloudCover>
                    <stability>UNSTABLE</stability>
                    <wind>
                        <windSpeed>0.4</windSpeed>
                    </wind>

<windDirection>1.5707963267948966</windDirection>
                </windStability>
            </Atmospheric>
            <Landcover>
                <nlcd2001>
                    <filePath>dist\NLCD_FtPolk.tif</filePath>
                    <type>BARREN_LAND</type>
                </nlcd2001>
            </Landcover>
            <Terrain>
                <source></source>
            </Terrain>
        </Environment>
        <GeographicDomain>
            <grid>
                <altitudeMode>aboveGroundLevel</altitudeMode>
                <southwest>
                    <value>-120.849858333333 35.74046 1.9</value>
                    <dimensions>3</dimensions>
                </southwest>
                <northeast>
                    <value>-120.772736 35.768275 10.0</value>
                    <dimensions>3</dimensions>
                </northeast>
            </grid>
            <resolution>
                <gridDivision>
```

```
                    <columns>200</columns>
                    <rows>200</rows>
                </gridDivision>
            </resolution>
        </GeographicDomain>
        <Output>
            <kmz>../example.kmz</kmz>
        </Output>
        <PlatformList>
            <VariablePlatform>
                <category>RadioFreqTransmitter</category>
                <type>RFDF</type>
                <affiliation>HOSTILE</affiliation>
            </VariablePlatform>
            <FixedPlatforms>
                <Platform>
                    <category>UnattendedGroundSensor</category>
                    <type>RFDF</type>
                    <active>true</active>
                    <affiliation>FRIEND</affiliation>
                    <fieldOfRegard>
                        <azimuthWidth>360.0</azimuthWidth>
                        <elevationWidth>180.0</elevationWidth>
                    </fieldOfRegard>
                    <state>
                        <attitude>
                            <pitch>0.0</pitch>
                            <roll>0.0</roll>
                            <yaw>180.0</yaw>
                        </attitude>
                        <location>

<altitudeMode>aboveGroundLevel</altitudeMode>
                            <point>
<value>-120.789561 35.753567 10.0</value>
<dimensions>3</dimensions>
                            </point>
                        </location>
                    </state>
```

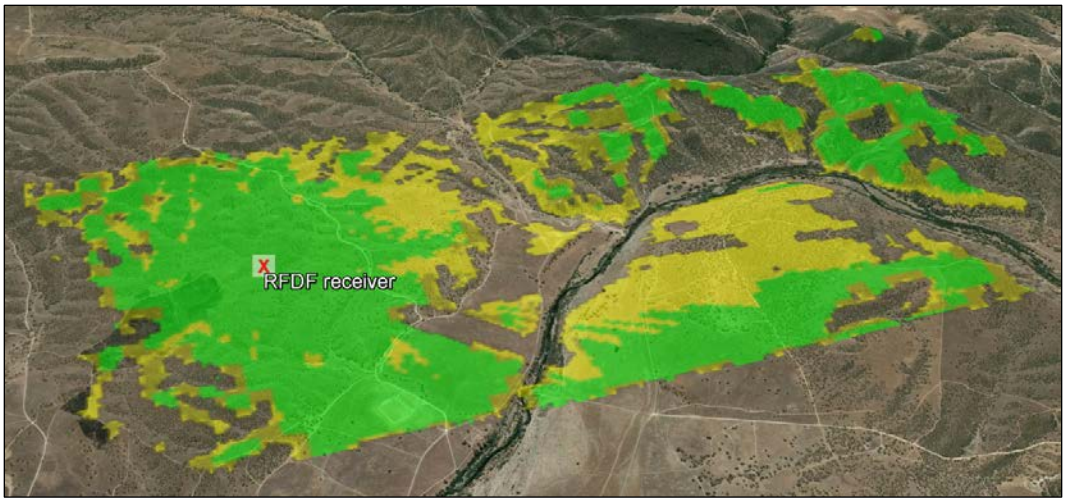
```
                </Platform>
            </FixedPlatforms>
        </PlatformList>
        <Scenario>
            <calculation>
                <type>PROBABILITY_OF_DETECTION</type>
                <acoustic>true</acoustic>
                <chemBio>true</chemBio>
                <infrared>false</infrared>
                <radioFrequency>true</radioFrequency>
                <seismic>true</seismic>
                <visible>true</visible>
            </calculation>
            <situation>FRIENDLY_MONITORING_HOSTILE</situation>
        </Scenario>
        <SignalPropagation>
            <acoustic>low</acoustic>
            <chemBio>low</chemBio>
            <infrared>high</infrared>
            <radioFrequency>high</radioFrequency>
            <seismic>medium</seismic>
            <visible>low</visible>
        </SignalPropagation>
        <Timing>
            <interval>P0Y0M0DT0H0M0.000S</interval>
            <timeSteps>1</timeSteps>
            <startDateTime>2013-03-
05T18:46:07.919Z</startDateTime>
        </Timing>
    </Simulation>
    <Results>
        <kmzUri>../example.kmz</kmzUri>
    </Results>
</easee>
```

KMZ output of the EASEE calculation for JSON and XML inputs above

Figure A1 provides an example RF transmission calculation created from the preceding JSON and XML inputs. The area of the calculation domain

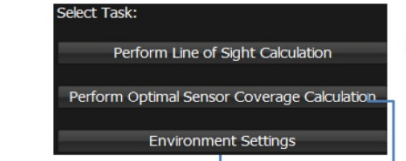
is approximately 7 × 3 km, and the location of the fixed receiver is indicated by a red "X" and is labeled "RFDF receiver." Green, yellow, and transparent pixels represent probabilities of >0.75, 0.25–0.75, and <0.25, respectively, that the receiver would receive signals from the transmitter at that location.

Figure A1. Example radio-frequency transmission calculation defined by JSON and XML markup inputs.



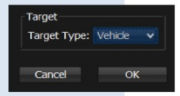
Appendix B: Quick Reference Guide for LMPT

LMPT Quick Reference Guide



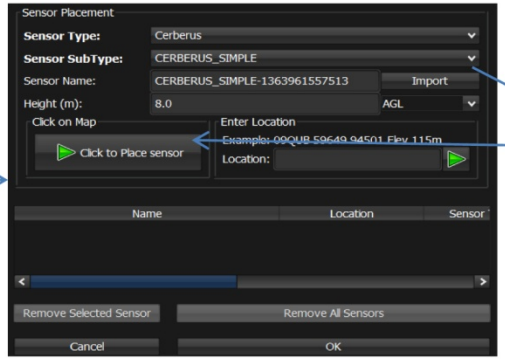
LINE OF SIGHT CALCULATION

Complete all 4 Steps
1 Set Environmental Conditions
2. Draw Coverage Area
3 Place Viewpoint
4 Select Target
then press
Please Refresh or *Run Model*
Answer will be displayed quickly

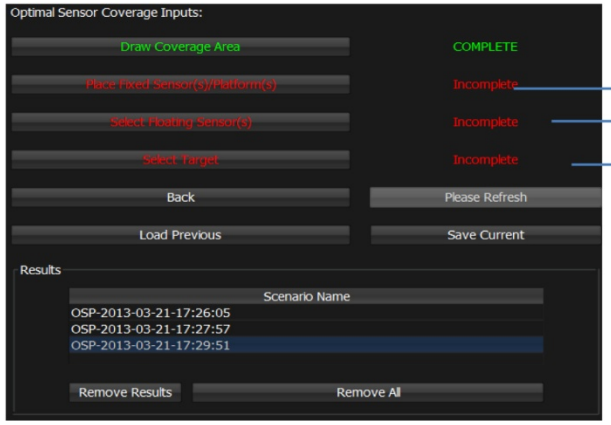


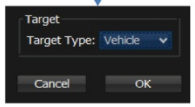
OSP / OSC

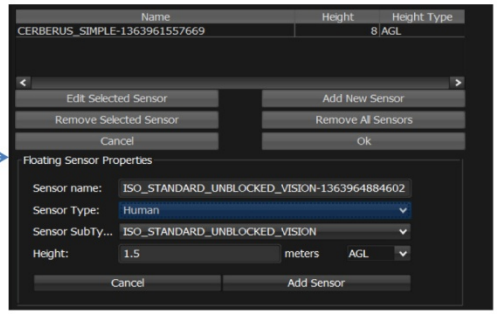
Complete all 4 Steps
1 Set Environmental Conditions
2 Draw Coverage Area
3 Place Fixed and Floating Sensors
4 Select Target
then press *Please Refresh* or *Run Model*
Answer will be displayed quickly



Select Sensor types Then You place them on map







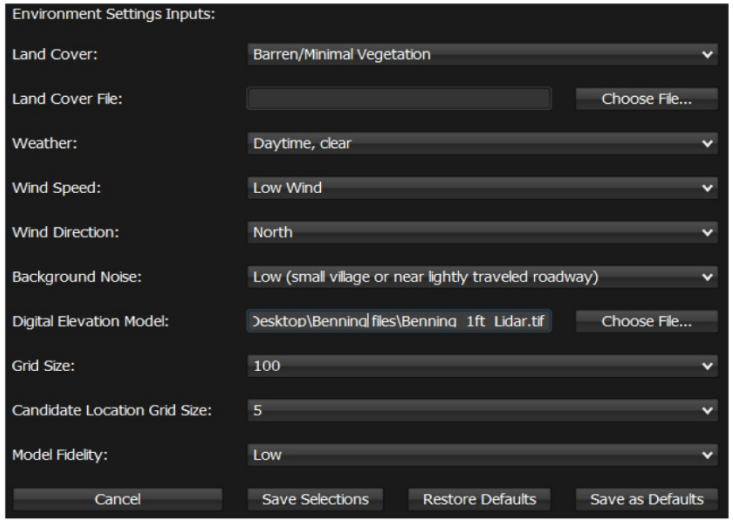
Select Sensor types to place

Press Add Sensor to add to the list of sensors

Note: Optimal Sensor Placement (OSP) takes time to run

LMPT Quick Reference Guide

POC: Sam Sweatt (samuel.t.sweatt2.civ@mail.mil)
POC: Don Albert (Donald.G.Albert@usace.army.mil)



LAND COVER: Affects Acoustic & Seismic calculation *all fidelity models

WEATHER: Affects Acoustic calculation & some imagers *medium & high fidelity models

WIND SPEED & DIRECTION: Affects Acoustic calculation *medium & high fidelity models

BACKGROUND NOISE: Affects Acoustic, Seismic & RADAR calculation *all fidelity models

GRID SIZE: Affects all calculation times and accuracy. Default is 100 (rows & columns). Larger number = longer calculation time and more accurate

CANDIDATE LOCATION GRID: Affects number of locations of placed OSP sensors. Default 5 (rows & columns). Larger number = longer calculation time and more accurate

MODEL FIDELITY: Higher Fidelity = Longer runtime & more accurate. Lower fidelity faster runtime less accuracy. For OSP, recommend LOW fidelity model

Connecting to ISA:

1. Select the "Home" tab from the Menu Bar.
2. Select the "Connection" button from the Ribbon Bar. This creates a panel on the left side of the display.
3. Select the "Add" pull down menu to activate it.
4. Select the "ISA" option from the "Add" pull-down menu. A window appears in the middle of the display.
5. Select the "main" connection from the "Saved connection" pull-down menu, and press the "Connect" button. The connection will be made and a Green "ISA Connected" Icon will be displayed
6. Select the "Explorer" button from the Ribbon Bar to see the list of Devices. The display will zoom to the first device that reports a location

Saving result to LMPT Share: (use 2 windows on desktop)

1. Open Desktop folder LMPT_RESULTS and copy latest KML file
2. Paste file in LMPT_SHARE desktop folder
3. Rename file to indicate contents

ERDC/CRREL TR-15-11

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
July 2015	Technical Report/Final	

4. TITLE AND SUBTITLE

Development of Integrated Software Capabilities for Battlefield Signal Modeling and Force Protection: Leaders Mission Planning Tool (LMPT)

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Kenneth K. Yamamoto, John J. Gagnon, Donald G. Albert, and D. Keith Wilson

5d. PROJECT NUMBER
AT42 GRE

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Cold Regions Research and Engineering Laboratory (CRREL)
U.S. Army Engineer Research and Development Center (ERDC)
72 Lyme Road
Hanover, NH 03755-1290

8. PERFORMING ORGANIZATION REPORT NUMBER

ERDC/CRREL TR-15-11

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U.S. Army Corps of Engineers
Washington, DC 20314-1000

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

Geospatial Research and Engineering

14. ABSTRACT

With increasing resource constraints in the Department of Defense, it is becoming critical to develop technologies that unburden small and minimally equipped teams of Soldiers carrying out a mission. As part of the Deployable Force Protection (DFP) program by the U.S. Assistant Secretary of the Army for Acquisition, Logistics, and Technology, ASA(ALT), we developed the Leaders Mission Planning Tool (LMPT) to help Soldiers with little specialized training in sensor technologies effectively protect combat outposts by optimizing selection and placement of limited sensor resources. This product is the result of collaborating with Night Vision and Electronic Sensors Directorate, interacting with DFP sensor teams, and incorporating Soldier feedback from multiple test exercises and demos. It is a powerful and intuitive user interface for the Environmental Awareness for Sensor and Emitter Employment (EASEE) computational engine (which runs on a standard laptop) designed by the U.S. Army Engineer Research and Development Center (ERDC) for battlefield signal modeling. This report discusses the role of battlefield signal modeling for effective sensor use, rationale and approaches for using markup file exchange to interface multiple software applications, and the capabilities and features of LMPT. Details about the EASEE markup interface are documented for collaborating software developers.

15. SUBJECT TERMS

Atmospheric effects
EASEE
Force protection
LMPT
Mission planning
Signal propagation
Signature physics
Sensors

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*
Unclassified	Unclassified	Unclassified	SAR	80	