Middleware for Runtime Assessment of Information Assurance

Partha Pal
BBN Technologies
10 Moulton Street
Cambridge, MA 02138
+1 617 873 2056
ppal@bbn.com

Rick Schantz
BBN Technologies
10 Moulton Street
Cambridge, MA 02138
+1 617 873 3550
schantz@bbn.com

Joe Loyall
BBN Technologies
10 Moulton Street
Cambridge, MA 02138
+1 617 873 4679
jloyall@bbn.com

ABSTRACT
We describe a middleware service and supporting methodology we have developed to facilitate runtime assessment of information assurance (IA) properties. The methodology structures the IA requirements of mission stakeholders along four axes, organizes the measurements and observations contributing to IA into a manageable number of metric classes, and provides a framework for constructing mission specific assessment rules derived from the metrics. The middleware service consists of runtime engines that connect with enterprise management databases and other sources to obtain metric values, evaluate the rules, and present the assessment results through visual interfaces. Results from evaluation of our initial proof of concept are encouraging, and we are continuing to refine and extend the prototype capabilities.

Keywords
Middleware, Information Assurance, Mission Impact, Assessment

1. INTRODUCTION
Middleware has traditionally been used as a common software engineering approach for bridging the gap between disparate system layers, for transparency in distributed operations, and as integration glue for multiple software or hardware mechanisms across machine boundaries. Distributed object middleware such as RMI or CORBA makes distributed inter-process/inter-object communication appear as if it is local, and transparently addresses host platform heterogeneity. In component oriented systems [1], survivable systems [2], and now in Service-oriented Architecture [3], middleware integrates multiple functions, capabilities and services. Advanced middleware has enabled Quality of Service (QoS) management and resource awareness; QoS/resource managed fault tolerance and adaptation in distributed systems; and survivable systems with dynamic and adaptive security.

All middleware performs some runtime assessment. For example, marshalling/de-marshalling in distributed middleware depends on assessing a number of factors including the implementation language, OS and the type of object being acted upon. Middleware for QoS-based adaptation routinely evaluates various runtime system characteristics and resource availability. However, despite its integrating role and advantageous position in survivability architectures, as yet we see no security assessment done in the middleware, especially of the mission impact of security incidents. This is not completely surprising, and indicative of more fundamental issues. First, security evaluation is fundamentally difficult. The state of the art consists of various forms of testing (e.g., stress testing, penetration testing, and red teaming) and security focused analyses (e.g., white boarding and security audits) that are often qualitative and more appropriate for human interpretation than automation. Second, most of the evaluation is done before the system is deployed. Although modern information systems report numerous “security incidents” at run time, stakeholders during a mission do not know whether the delivered level of IA remains adequate for their mission roles.

We are unavoidably dependent on (distributed) information systems. Many of our missions run in a contested environment, where adversarial forces, be it cyber criminals trying to steal identity or money, or nation states trying to subvert critical infrastructure, are constantly practicing and perfecting their attacks. We argue that it is not enough to know that the systems we are using passed certain tests and checklists at some point in the past, or were built by satisfying certain procedural requirements. Just as the padlock icon in the browser provides a level of confidence about website authenticity and message confidentiality for HTTPS interactions, we need a way to determine and present the delivered level of IA at runtime. Knowing that the system was certified at a certain security level is helpful, but without any runtime assessment, stakeholders are left with an “all or nothing” view of IA—either the system appears to remain in an assured state, no matter what happens at run time, or it does not. This leads to unnecessary risk taking and missed opportunities. Report of an attack or attack induced failure prevents the user from using the system even if the parts and services he needs are not affected by the attack or failure—there is no way to know. Similarly, there are times when users and operators modify or reconfigure the system or its security mechanisms to perform mission critical functions or to achieve a desired level of QoS, but there is no way to judge the risk they incur for themselves, the system, or other users of the system.

We introduce a middleware service and supporting methodology to track delivered levels of IA against stakeholders’ requirements.
throughout a mission. This work combines two ongoing QoS middleware [4] and survivability [5] research threads, and ultimately aims to support automatic QoS-IA tradeoff management when all IA and QoS requirements cannot be met simultaneously. The methodology structures the IA requirements along several axes, organizes the measurements and observations contributing to IA into a manageable set of metric classes, and provides a common framework to combine and aggregate the metric values to construct mission-specific assessment rules. The middleware service has assessment engines that evaluate the rules, connects with enterprise management databases and other data sources for measurements and observations, and offers a visual interface to present assessment results. We have applied the initial prototype in a military context to evaluate the concept of mission-oriented and continuous assessment-focused middleware. Initial results are encouraging, and we are continuing to refine and enhance the methodology as well as the prototype.

The rest of this paper is organized as follows. In section II we describe the middleware service and the underlying methodology. Section III describes an early evaluation application of our middleware. Section IV concludes the paper with a brief summary and some planned next steps.

2. SOLUTION APPROACH
2.1 Methodology
Structuring required/delivered IA: Missions in network-centric (military and business) enterprises consist of multiple stakeholders cooperating to achieve a common objective, following a workflow, whose execution involves multiple different sessions within a supporting information system. Stakeholders have different mission roles, i.e., hold different perspectives of concern, such as an end-user stake, a system administration stake, or an ownership stake. Each stakeholder requires different levels of IA at different points within the mission, depending on which mode the mission is in. The mission mode is defined by various conditions including elapsed time (e.g., the 1st hour of the mission is WAIT mode, and when an hour is spent in WAIT mode, the mission moves to LOGOUT mode) and mission events (e.g., moving from BROWSING to CHECKOUT when the user sends a ready to checkout confirmation). As an example of varying IA requirements in different mission modes, strong encryption is not required during BROWSING, but is essential during CHECKOUT. A military mission may have a WAIT phase when connectivity to backend systems is not necessary, but in the REPORTING phase that sends sensitive information, a secure line of communication is absolutely essential. We define assessment to mean contrasting the level of IA delivered by the system at any given point against the required level, where the levels are descriptive terms with relative ordering (e.g., high, medium-high, medium, etc.). Treating IA assessment this way eliminates the need for absolute quantification, and aligns it with QoS assessment.

Of course, neither IA nor QoS is monolithic. Usually, they are expressed in terms of attributes. Some attributes, such as confidentiality and integrity are associated with IA, while attributes like timeliness and fidelity are associated with QoS. There are some attributes such as availability that are used to describe both QoS and IA. Our middleware service for mission-oriented IA assessment, henceforth called CMA (for Continuous Mission-oriented Assessment) currently supports Confidentiality, Integrity, Availability, Access Control/Authentication, Timeliness and Fidelity attributes. In a mission, a stakeholder may not have requirements for all attributes or all of the time. But attributes by themselves do not mean much unless they are accompanied by a spatial context. For example, a user needs to specify confidentiality of what—usually a service or a network link, not just the need for confidentiality. Stakeholder role and responsibility may change during a mission. Therefore, we must accommodate differences in the requirement over time. The problem space of mission-oriented assessment is multidimensional. Therefore, in our methodology the required and delivered levels that are compared for assessment purposes are described with respect to the following coordinates: <stakeholder, IA/QoS attribute, spatial context, and mission mode>.

Organizing measurements and observations: Many factors influence the level of IA delivered in the system. They are mostly system specific, and there is no universally applicable or well accepted set of metrics. In our methodology, we have organized the measurements and observations that contribute to the assessment of IA into a small number of metric classes. This is described in [6]. Here, we provide a brief summary. We assume that modern mission critical systems will be defense-enabled to include a number of security mechanisms and tools (defense mechanisms) organized into a survivability architecture providing defense in depth, containment and dynamic management (of the defense mechanisms). The DEF STAT class represents the status of the defense mechanisms—OFF/ON state or configuration settings indicating stricter or more permissive modes of operation. The RES STAT class represents the status of system resources. Measurements involving incident reports issued by defense mechanisms are in the DEF REP class. The DEF EFF class represents measurements indicative of the effectiveness of the defense mechanisms. Factors like newly released vulnerability or advisory reports are represented by the EXT class. The classes AIQ and POM represent measurements and observations indicative of the architectural qualities of the system (e.g., the number of defensive layers an adversary needs to overcome to control a critical service) and process and organizational maturity (e.g., patch status, password renewal policy).

Framework to combine metrics in assessment rules: While it is intuitive that measurements from the metric classes influence the level of IA delivered by the system, much like QoS assessment, there is no well accepted all-encompassing formula for combining the metrics to determine the delivered level. We have developed mechanisms to determine the delivered level of QoS in our QoS-managed middleware work earlier [4]. We follow a similar approach for IA assessment. As in the case of QoS, a number of system analysis, development, and integration steps are needed before deployment. For this reason, in addition to the middleware service, we introduce a new software engineering role, namely, the assurance engineer that is analogous to the QoS engineer role.

However, in QoS engineering, the relations between metrics (e.g., jitter in packet delay, frame rate, CPU load) and attributes (e.g., fidelity of audio or video streams) are derived from concretely observable phenomena (e.g., packet jitter over a threshold makes the audio or video quality unusable for the user) and natural laws (e.g., inability of human eye to distinguish certain frame rates, or a highly loaded CPU means sluggish response for all processes running on that host). Variations in delivered IA (e.g., loss of confidentiality) may not be directly visible to the stakeholder, and there is no natural law to assist us. To address this gap, we have
developed a framework that enables us to relate metrics and attributes in a context sensitive way.

In this framework, delivered level of assurance $L_{A,S,H}$ for an IA attribute A and spatial context S the stakeholder H is interested in (we will use $A$ in $S$ for $H$ as a shorthand henceforth) at a given time equals $V(B_{A,S,H})$ where $B_{A,S,H}$ is a base level assessment and $V$ is a variance function. Only the DEF STAT and RES STAT metrics, which are easily available in most modern systems, are mandatory in this framework. All other metrics are optional to accommodate the fact that these measurements may not be available in some systems. The base level $B_{A,S,H}$ is assessed by establishing a mapping from the collection of DEF STAT values relevant for $S$ to the desired number of levels the stakeholder used to express his requirements. This process is described next.

Given the spatial context and the IA attribute of interest for each stakeholder-specified requirement, it is possible to determine, with a fair degree of accuracy, the system resources, defense mechanisms and business functions that are involved in servicing the requirement, using a straightforward security focused system analysis. Let us assume that there are $N$ defense mechanisms $D_1$, $D_2$, ..., $D_N$ (distributed throughout the system, not necessarily local to $H$) that contribute to the security attribute A of the spatial context S that the stakeholder H has an interest in.

A defense mechanism $D_i$ can be in $m_i$ states ordered in terms of the strength of protection. The states can be as simple as OFF and ON, with the ON state providing stronger protection, but a defense mechanism will typically have more than 2 states. For instance, a firewall can be set to use rule sets with increasing strictness; an encryption mechanism can support 128, 256, 512 and 1024 bit keys. For each $D_i$, we assign a DEF STAT metric $d_i$ whose value can be integers between 1 and $m_i$, where 1 represents the setting providing the weakest protection and $m_i$ the strongest.

The configuration of the defense mechanisms relevant for A in S for H can therefore be represented as a vector of DEF STAT metrics $[d_1, d_2, ..., d_N]$. The possible values of this vector forms a partially ordered lattice with $[m_1, m_2, ..., m_N]$ at the top and the unit vector of size N at the bottom. Some combinations of defense mechanism states are invalid. For example, if one DEF STAT value indicates that one side of S is set up to encrypt and another DEF STAT indicates that the other side is set up to handle clear text, this configuration is not workable. Similarly, some combinations may not offer different levels of assurance. Therefore the lattice will have some invalid nodes that we exclude, and some equivalent nodes that we treat as a group. After eliminating the invalids and grouping the equivalents, the lattice is partitioned into stakeholder specified number of levels such that the nodes in the upper part of the lattice correspond to higher levels of assurance, a process we call banding. If the specified number of levels exceeds the number of possible bands, the assurance engineer reconciles the conflict with the stakeholders.

This process is illustrated in Figure 1 with 4 defense mechanisms, where $d_1$ and $d_2$ are binary valued (they can be either 1 or 2), and $d_3$ and $d_4$ have values in the range $[1, 3]$. Invalid configurations are shown by the red boxes in Figure 1, indicating configurations where $(d_1 = 1$ and $d_2 = 2)$ or $(d_1 = 2$ and $d_2 = 3)$ are incompatible. Equivalent configurations are shown in Figure 1 by the dashed red shape, indicating that the configurations where $(d_1 = 1, d_2 = 1, d_3 = 1, d_4 = 3)$ and $(d_1 = 1, d_2 = 1, d_3 = 1, d_4 = 2)$ are effectively the same. The figure also shows the banding for 3 required levels. Note that the invalids and equivalents can appear anywhere in the lattice, and to be meaningful, nodes that are not far from each other are clustered together (e.g., putting $(1,1,1,1)$ and $(2,3,2,3)$ in the same band would imply that the weakest and strongest configurations offering the same level of assurance.)

The algorithm to assess the base level $B_{A,S,H}$ (which will also be the assessed level in the absence of the variance function) then becomes membership checking: which band contains the current configuration indicated by the observed DEF STAT values? But explicit membership checking on a large graph can be time consuming (the total number of nodes in the graph is the product of $m_i$ and $s$, and can be extremely large). To address that issue, we make use of the atLeastAsSecureAs relation, which captures the partial ordering implicit in the DEF STAT space, between two nodes defined as follows:

\[
\text{atLeastAsSecureAs}([],[]).
\]

\[
\text{atLeastAsSecureAs}([F1|R1],[F2|R2]) :- F1 >= F2,
\]

\[
\text{atLeastAsSecureAs}(R1,R2).
\]

In the above code fragment, a node $N$ is defined as a list of values (e.g., $N = [a, b, c, d]$ and the $[F | R]$ notation implies F is the first element in the list and R the remainder of the list (e.g., $F = a$ and $R = [b, c, d]$) for the list $[a, b, c, d]$). It is straightforward to define the various colored bands in terms of the invalid, equivalent, and atLeastAsSecureAs relations.

The variance function $V$ may not be applicable to all assessments. More specifically, the mandatory RES STAT measurements modify the base level assessment of availability only. The DEF REP, DEF EFF and EXT measurements, when available, are used to modify the base level assessment of confidentiality, integrity, and access control/authentication strength. The POM and AIQ measurements, when available, are used to reduce the impact of EXT, DEF REP, and DEF EFF measurements on the assessed value. Although somewhat intuitive (e.g., many alerts about the confidentiality of a link should cause a downward revision of all assessments that have the link within its spatial context, and confidentiality as the IA attribute), the variance function is highly complex. Difficulties include identifying which assessments are to be modified when a DEF REP or EXT measurement changes, the extent of damping if a POM or AIQ value is available, and the extent and direction of revision. We are actively working on expanding these topics. In the interim, the assurance engineers are tasked to determine the variance functions on a case by case basis.

### 2.2 Middleware Service

It should be clear that assessment involves intelligent and focused manipulation of multiple types of information originating in
various parts of the system. The middleware service facilitates the flows, aggregation and processing of the information required for such assessments, and presenting the results. The runtime components of our middleware service consist of **assessment engines** that can be customized with assessment rules for computing V and B_{A,S,H} as described earlier. The rules, defined in terms of values of system condition objects are implemented by a combination of Java and Prolog code. System condition objects exist within the process boundary of the assessment engines, but represent the measurements that are needed for the assessment.

We observed that modern Enterprise System Managers (ESMs) (as well as incident and log management systems) already collect most of the information that stakeholder assessments need in their databases. Therefore, the CMA assessment engine provides (see [Figure 2](#)) the plumbing necessary to draw the required value from such databases and mirrors them among the other assessment engines using a light-weight web-service.

Connecting a system condition object with a measurement that is maintained by an ESM involves specifying the appropriate database query, which the runtime mechanism then uses to fetch the value. If the system condition value is already available in another assessment engine, only a peering relation needs to be specified and the runtime mechanism will perform the mirroring. In cases where the value must be obtained directly from the system, e.g., it is not possible to publish the measured value to the ESM, the system condition interfaces with the measurement instrumentation code. The system condition interface does not expose a listening port for obvious security reasons, but in addition to being able to “pull” from the assessment engine, also supports a push interaction model through registered callbacks.

A specialized system condition object in each assessment engine is used as a **mission progress tracker**. This system condition subscribes to mission level events. The mission progress tracker can be set to receive arbitrary events and customized with rules to parse the received data. If the rules signify a change in mission mode, an assessment cycle is automatically triggered.

The assessment engines have a GUI to **present assessment results**. An assessment engine is responsible for multiple assessments, involving substantial (so far we have tested with ~50) number of system condition objects. The GUI can be used to continuously observe the system condition values as well.

Once the assessment engines are set up with the assessment rules and appropriate connections to obtain metric values, the assessments are performed in an ongoing manner, as the mission is executing. The trigger and frequency of the assessment is configurable—for a user-centric application, the assessment can be configured to run automatically at a regular interval so that the current assurance state, i.e., whether the system is delivering what the stakeholder requires, is always visible to him at any point during the mission.

### 2.3 Roles Played by the Assurance Engineer

The assurance engineers are responsible for capturing the stakeholder requirements and instrumenting the information system to use our middleware service.

**Capturing IA requirements**: Our methodology requires the stakeholders to shoulder some responsibility in the assessment process by specifying the level of IA they require in order to successfully execute their mission roles. Assurance engineers capture the IA requirements through interviews. Since stakeholders are not QoS or security experts, only a high level specification is expected. The key information the assurance engineers seek to capture from the stakeholders are the qualitative ordering of security requirements, defined in terms of the IA attributes, for the functions and services they need at various mission modes. Stakeholder H’s requirement about an IA attribute A (e.g., Confidentiality) with respect to a spatial context S (e.g., a link between a client and a service) R_{A,S,H} is captured as a sequence of <T_i, l_i> pairs, where T_i denotes a mission mode and l_i denotes an ordered level value (i.e., given two level values l_i and l_j, either l_i=l_j or l_i<l_j or l_i>l_j).

**Interfacing with the runtime**: This process is driven by the metrics combining framework described in Section 2.1. First, the assurance engineers identify the metrics that can be measured from the given system, enumerate the DEF STAT lattice for each R_{A,S,H}, deconflict any mismatch in required and supportable number of levels, and then derive the assessment rules to compute V (B_{A,S,H}). Next, the assurance engineers configure the assessment engines with the appropriate rules. A subpart of this activity involves setting up the system condition objects so that they can obtain their respective measurements at runtime. Obviously, measurement obtained through the ESMs is preferable because the ESMs are anticipated to already have a reliable, low overhead and relatively secure way of transporting observations to databases. If the desired metrics are not available, the assurance engineers instrument the system to collect them and submit them to the ESM, or interface with the system conditions directly if that is not possible. Finally, they load the rules for mission mode transition into the assessment engines and connect the progress trackers with the appropriate event streams.

### 2.4 Discussion

**QoS-IA tradeoff**: The CMA runtime operates in conjunction with existing middleware services for enterprise system management, QoS management and survivability architectures, and leverages them for collection and transportation of measurements. Survivability architectures facilitate monitoring security incidents and mounting defensive responses. Similarly, QoS management facilitates monitoring the delivered level of QoS and manipulating system resources and application behavior to deliver acceptable QoS. The CMA middleware monitors the delivered levels of IA, and alerts the stakeholders, including the system administrators responsible for managing the system’s defenses, when their requirements are not being met. The ideal case of satisfactory levels of both IA and QoS is not always achievable even with
autonomic and human administered QoS and survivability management. Stakeholders’ requirements, even the QoS and IA requirements of a single stakeholder, often compete with each other. Moreover, services and defenses use the same system resources making security and service delivery a zero-sum game. This motivates the need for QoS-IA tradeoffs, a capability that we are working on and plan to incorporate in our middleware in the future. The combined treatment of QoS and IA also enables us to assess the mission impact of known and unknown attacks uniformly because in order to achieve their objectives, all attacks, except those that focus only on exfiltration, must visibly affect the QoS of the mission-essential functions provided by the system.

Related runtime IA assessment work: To the best of our knowledge, runtime IA assessment approaches have been limited to vulnerability scanners (e.g., NESSUS [7]), or situation awareness and alert correlation services (e.g., M-Corr [8]). Neither of these approaches really assesses the mission/business impact. In that sense our approach is unique.

3. APPLICATION

As an initial step toward validating and evaluating our approach, we applied our methodology to a scenario based on a military mission, and built a demonstration prototype that uses CMA middleware services in an environment simulating the mission.

System and mission context: The information system (depicted in Figure 3, and based on experience with operational system experiments) consists of 3 enclaves connected by different networks with different security and performance attributes. The mission involves a detached unit (the Client Enclave in Figure 3) in enemy territory that regularly publishes ground intelligence to an air operation center (AOC) and calls in air support when they spot critical targets. The interaction with the AOC, a designated subnet in the ground enclave, is handled via a battlespace information management system (IMS) that supports exchange of rich data in a more efficient and automated manner than voice over field telephones. The IMS client is at the detached unit. Other IMS components are in the ground enclave and an airborne enclave reachable by SATCOM and radio links respectively.

Assurance engineering steps: We played the assurance engineer role with an eye to validating the underlying principles of our approach. In this exercise, we focused mostly on the end-user stake, i.e., using the detached unit as the stakeholder H, whose main interest is to communicate with the AOC using the IMS. Analyses of the mission, use cases and the supporting system confirmed that the mission can be organized in elapsed time and mission event based modes (see Figure 4) and $R_{3,5,6}$ can be captured in terms of ordered levels (see Figure 5). In this example, the stakeholder’s requirements turned out to be rather simple: only availability requirement showed 3 levels, all other requirements were expressed using two levels. For brevity, we skip the rationalization behind the end user’s requirements in this paper.

Analysis of the exemplar system also identified the cybersecurity defense mechanisms relevant to the detached unit’s mission role. They include the redundancy offered by the airborne and the ground enclaves in hosting the IMSs, and network level encrypting devices at the enclaves that provide the option of secure communication over encrypted channels. In addition, the IMSs offer three levels of service: in the first, the user needs to authenticate; in the second, he does not but requests must be accompanied by a HMAC or a token based on an a-priori shared secret; and in the third, he does not need to authenticate at all, and requests do not include any crypto token. The number of ways these defense mechanisms can be configured is more than sufficient to support the desired number of levels, and multiple bandings are possible. For brevity, we show the banding we encoded in the CMA runtimes in tabular form in Figure 6.

Figure 3: A close air support mission context

Figure 4: Mission mode transitions

Figure 5: Stakeholder requirements

Figure 6: Assurance levels for the detached unit

Runtime setup: In the simulation, each enclave is given a CMA assessment engine. The ESMs in the ground and the air enclaves feed some of the system conditions of their collocated assessment engines, while the rest of the system conditions are peered. Some system conditions in the assessment engine at the client enclave bind to local instrumented code, while the others are peered.

Key benefits: To illustrate the benefits of the CMA middleware service we simulated various mission execution threads with different arrival time of the air asset, different loads on the IMSs,
CMA middleware offered the much needed “dial tone” effect that the end-users desperately wanted before starting to use the system—direct visibility of available services and their security and performance characteristics upon connection. The assessment engines provided the detached-unit with the assessments resulting from complex processing of multiple disparate pieces of the measurements upon start up and updated as the mission progressed. The CMA middleware guided the user’s course of action in a way that was not previously possible. They avoided aborting a new mission by postponing the call when they saw high IMS load. When the firewall at the air asset was corrupted, even though it was reachable by radio, they switched to SATCOM to talk to the ground enclave.

Civilian use cases: Although we used a military scenario, civilian domains are plagued by the same semantic gap between mission/business requirements and available low-level system information. System administrators rarely have a good understanding of the mission or business operations being performed over the IT assets they manage. Business users know the security implications of the services they use, but do not have the training/expertise to express the right security configurations or to understand the security alerts and incident reports. The CMA middleware enables an approach to assess whether the delivered level of security is adequate for business users, and provides the system administrators a better understanding of the incidents/asset compromises that impact business processes. As an example of comparable civilian use case, consider a user who wants to use SSL (https) over WPA when connecting to his bank. In a hotspot or a hotel room, CMA middleware will alert him if his connection to the bank is not secure or available data is insufficient to determine that. A repairman walking into an installation site provides another example. He requires stronger authentication of the deployed devices before connecting to them. He also requires timely interaction with a higher level of integrity and confidentiality when transferring customer data to a back office than when he connects back to look up parts. The CMA middleware will dynamically adjust to the repairman’s situation and indicate if the delivered level of assurance is not sufficient.

4. CONCLUSION
Continuous Mission-oriented Assessment middleware bridges the semantic gap between low-level incidents and measurements that are routinely collected in modern information systems, and the IA requirements for specific mission and business operations that the information system supports. We have developed a mechanism to internally represent and logically connect mission operations with IA requirements of stakeholders. We have implemented a runtime assessment engine and the connective infrastructure needed to integrate it with Enterprise System Management databases and to support Web Services-based peering of assessment engines. Putting them together, we have demonstrated continuous mission-oriented assessment in a simulated environment. Our goals for this evaluation were limited to feasibility and proof of concept within a simplified operational context. The work to date merely skims the surface, and motivates deeper investigations in each of the relevant areas. However, even this limited experience has influenced our R&D path forward. Extracting actionable awareness from highly interconnected systems composed of ill-specified individual parts is a complex undertaking. There are dangers on both ends, from overly simplified, to overly detailed. Our initial attempts here are to find a middle ground to build upon both user needs (top down) and system specific details (bottom up). As outlined in this paper, we are iteratively refining a number of points of view simultaneously. Going forward, we expect the banding process to involve vectors of significant sizes. We are exploring automated tool support to assist with identifying invalid and equivalent nodes based on pre-defined and reusable rules about defense mechanisms, and distance-based clustering of the lattice into specified number of bands. We are augmenting our requirements capture and representation to include preferences, i.e., indication of which attributes to sacrifice when the system is unable to deliver at the required level for all QoS and IA attributes. Complementing that, we are developing mechanisms to reason about the dependency among the controllable and observable aspects which need to converge for effective assessment and QoS-IA tradeoff.

5. ACKNOWLEDGMENTS
This work draws upon a number of DARPA and AFRL funded projects including most recently under AFRL contact no. FA8750-08-C-0196. The authors wish to thank Mr. Patrick Hurley of AFRL for his support and review of the paper.

6. REFERENCES