Abstract— Advances in both low-level and high-level fusion continue to be made at an increasingly accelerated pace. With the value of Information Fusion now being recognized as applicable to several domain problems, comes the need to develop a systematic and cohesive approach to address the Enterprise Multi-Level Fusion Lifecycle. It is no longer sufficient to advance fragmented capabilities without formal processes with which to deploy them in an integrated fashion. While the importance of Systems Engineering has been previously addressed by the Fusion Community of Interest, advances in technology and in the way that systems are deployed, e.g., Service Oriented Architecture, Distributed Systems, Information Assurance, and Real-Time/Mobile applications, and especially the complexity of Systems-of-Systems which characterize Multi-Level Fusion, warrant a fresh look at developing a lock-step approach. This paper draws from real-world projects and explores how Systems Engineering can be leveraged to deploy Enterprise Multi-Level Fusion applications and bridge the integration gap between Lower-Level and Higher-Level capabilities that continues to challenge the Fusion community.

Keywords – Systems Engineering; Information Fusion; Multi-Level Fusion, Fusion Systems Architecture.

I. INTRODUCTION

The classic Joint Directors of Laboratories (JDL) Fusion model provides a description for High and Low Level Fusion [1]; notwithstanding, it was intended to be neither an architectural guide nor a comprehensive model on how to integrate and address the lifecycle of an Enterprise Multi-Level Fusion system. Systems Engineering has been recognized as a discipline with which to analyze Fusion Systems, from requirements to systems architecture [2, 3]. More recently Blasch outlines the importance of incorporating the role of the user, thereby accentuating the necessary synergy among mission objectives, users, and capabilities [4, 5]. The value that these efforts bring to the Fusion community is that they offer integrating views and concepts beyond that of single monolithic solutions that may result from implementing fragmented fusion levels.

Furthermore, this last decade has introduced significant paradigms. To add to the complexity, Information Fusion has not only grown in its solution space, but also in its problem space.

The solution space has transformed itself from a focus on Low-Level Fusion to more comprehensive Multi-Level fusion applications, spanning the entire information fusion workflow: from collection to exploitation, including dissemination and collaboration. A relatively new development is that the problem space has grown as well. Although it is true that the potential of information fusion to other fields, beyond the original defense applications, has always been recognized, it is not until recently that applications in the medical, transportation, and cyber security fields are now being actively developed and deployed [6, 7].

Furthermore, no fusion system should be designed as a stand-alone system, but rather considerations for integrating with other systems should be thoroughly examined. Multi-Level Fusion systems have grown somewhat fragmented. Each of the various fusion levels is rather specialized, with its own algorithms, and input and output requirements, easily qualifying it to be a system based on its own merit. Notwithstanding, when considering the interaction among the Information Fusion workflow, with the added complexity of a diverse set of users and capabilities, yet sharing requirements and mission objectives, the individual systems form a Family of Systems (FoS). Therefore integration among fusion levels can be enabled by applying System-of-Systems (SoS) engineering principles, which include among others: extensibility, interoperability, and reliability (fail-over, stand-by, replication, redundant systems) [8].

Given this vertical (throughout multiple fusion levels) and horizontal (across several domains) Information Fusion integration challenges, Systems Engineering provides a mature and proven methodology with which to analyze the domain problem and synthesize the domain solution. Here lies the value in applying Systems Engineering (SE) to fusion processes: Low-Level fusion and High-Level fusion have been developing by and large separately, although recognized as different facets in the end-to-end fusion chain, yet the ability
to integrate the various advances in low and high level fusion remains a challenge.

This paper explores leveraging Systems Engineering as a formal discipline to successful implement Enterprise Multi-Level fusion systems based lessons learned from large complex projects and considering the comprehensive end-to-end workflow of the Tasking, Collection, Processing, Exploitation and Dissemination (TCPED) lifecycle. Single level fusion processes, e.g., Low-Level or High-Level applications or deployments will be referred to herein as Fusion Systems, whereas the comprehensive Enterprise Multi-Level Fusion SoS refers to an end-to-end integrated multi-node distributed fusion implementation.

II. RELEVANCE OF SYSTEMS ENGINEERING TO FUSION

System Engineering is about considering all facets of the whole and often requires trade-off analysis between the parts in order to make the best selection that fits the purpose of the mission; i.e., the objective at hand should drive critical design decisions, which are often captured in Key Performance Measurements (KPMs).

Systems Engineering is not a waterfall process whereby one stage follows another one. This may seem obvious but it is important to understand why. In today’s complex systems, design is often iterative – requirements are usually not fully known at the outset, and dependencies may be difficult to identify at the beginning of the design. For example, most systems now employ Commercial-of-the-Shelf (COTS) products as a way to achieve standardization and realize cost savings. However, today’s complex multi-tier architectures comprising database layers, middleware layers, application layers, and presentation layers, as well as distributed nodes with multiple access control paradigms, present unique interoperability issues that require an iterative process to refine and resolve the unknowns.

Additionally, it may not always be possible to design the best solution in an Enterprise Multi-Level Fusion SoS from the outset, as it often is the case that integration with legacy systems (existing fusion platforms; e.g., Sensors) will limit many design options. These legacy systems may be part of existing Family-of-Systems, which in turn result in additional requirements being levied that force specific integration approaches and impact the overall enterprise architecture. Accordingly, SE is well suited to Information Fusion, as fusion nodes are often part of a larger Enterprise (a Fusion Systems-of-Systems) comprising many capabilities, serving many organizational objectives as well as different organizations and different Communities-of-Interest (COI). Additionally, the interdisciplinary nature of System Engineering enables this approach to dovetail with the Information Fusion’s diverse challenges.

Another aspect that is critical to the implementation of Information Fusion systems as a sponsored project, is the value that Systems Engineering brings regarding budget, schedule, O&M (Operations and Maintenance). SE concerns itself with not only the technical aspects of an implementation, but also with the business aspects of realizing a system – and when and how to deprecate it (upgrade, replace, or retire). The rise of Systems Engineering in the defense field became inevitable with the exponentially growing complexity of deployed systems. SE provides an effective way, including both technical and business consideration, for managing change requests during the entire product lifecycle. From conceptual design, development, testing, and O&M, the cost of escaped defects rises exponentially. These out-of-phase defects are a consequence of not following a Systems Engineering approach, whereby verification and validation is performed throughout the SE lifecycle. For example, in Information Fusion systems, an escaped defect may manifest itself by propagating defective incomplete data from sensors or collectors high-level fusion processing nodes. Since this issue was not caught during the requirements phase, the cost to repair the gap becomes almost prohibitive once the system has been deployed.

Germaine to Systems Engineering is a wealth of knowledge regarding continuous improvement and optimization, which can be leveraged to integrate Process Refinement (Level 4), an integral part of Information Fusion, which is much too often overlooked or underemphasized. Process Refinement is arguably responsible for improving the overall fusion process. However Process Refinement should be built-in – from the outset of the fusion system design, with clear performance metrics established, and using continuous improvement to derive new metrics. SE looks at establishing clear definition of interfaces between component boundaries, such as functions and data exchanged among the different fusion levels thereby ensuring that data degradation does not occur; no amount of downstream processing can make up for the deficiencies of upstream data collection [9].

Another key aspect related to fusion is that of traditional deployment models, such as big bang vs. incremental. Big Bang represents systems that are released into the market as a single roll-out, whereas an incremental approach represents improvements such as software upgrades to accommodate new requirements. This is also representative of how fusion systems come about, improvements in some aspects are integrated into the system, this is especially true for those with system lifecycles and service periods which are longer, or system that have become ingrained, either because their criticality in operations is such that replacement would result in unacceptable disruption, or because the replacement cost is prohibitive. This imputes a degree of uncertainty when designing tomorrow’s fusion systems with today’s technology, and it becomes paramount to mitigate the risk of premature obsolescence with extensibility, reliability, and deprecation strategies (i.e., when and how are systems going to be
upgraded or replaced). It also brings about a very important trade-off: that of technical debt, which is the point in time when a system can no longer be upgraded without introducing significant loss of robustness (i.e., such as results from continuous patching). When this point is reached, new technologies need to be introduced, or the replacement of an entire system needs to be considered. For example when divergent databases cannot be effectively replaced until the next generation, database gateways and adapters are created to coexist with new technologies such as SOA, thereby extending the life of the legacy system. At some point, the legacy system can no longer be kept up-to-date without introducing significant degradation to the system. Multi-media and Spatial data types are an example of new data types requiring next generation, database gateways and adapters are created to coexist with new technologies such as SOA, thereby extending the life of the legacy system. At some point, the legacy system can no longer be kept up-to-date without introducing significant degradation to the system. Multi-media and Spatial data types are an example of new data types requiring significant loss of robustness (i.e., such as results from new defects being introduced with upgrades, i.e., breaking functionality, which previously worked. While some “new” defect introduction is inherent to upgrades, it is imperative that a Fusion SoS perform thorough end-to-end regression testing, including from low-level fusion to high-level fusion, to ensure that the workflow has not been compromised.

III. SYSTEM ENGINEERING KEY CONCEPTS

Two key concepts of Systems Engineering are its process lifecycle and the “V” model, pronounced “Vee”. Different but related to these two concepts is the product or system lifecycle, which is essentially the usage characteristic a product or system undergoes from introduction into the market (product or system release) to retirement from the market; the criticality to fusion is that understanding how two cycles are interdependent will yield a more robust design. Since the developing an Enterprise Multi-Level Fusion SoS by definition comprises several fusion systems or nodes (single level or multi-level), each having its own product lifecycle, it becomes critical to understand what updates or upgrades overlap nodes, and how some systems will be deprecated, and new ones introduced. This is not only true from a technical perspective, but the inherent revenue and utility associated with a product lifecycle will also drive critical decisions, e.g., introduction of new technologies.

A. The Product Lifecycle

Figure 1 depicts the product lifecycle identifies the utility (or market penetration) of a Fusion system with respect to its users. Metrics on the vertical scale can include revenue generated by commercial fusion systems, number of users, or number of installations for non-commercial systems, e.g., government procured. Usually the analysis phase is not included in a typical product lifecycle, but it should be considered, as feasibility studies and prototypes for example, are all viable pre-introduction strategies. A typical Fusion node lifecycle comprises the following phases: (a) development, (b) introduction, (c) growth, (d) maturity, (e) saturation, and (f) decline, which ultimately leads to retirement, either through replacement, or all together capability phase-out. Of particular interest is the growth through saturation phases. In the growth phase, Systems Engineering is critical in handling the Change Requests (CRs), some CRs are due to escaped defects, those who were not captured in the regular SE cycles (up to testing), other CRs may come from Engineering Change Requests (ECRs), those which are new (but compatible) functionality that the end-user has requested to incorporate into the existing system; and finally, regular maintenance, such as upgrades. To the extent that extensibility (the ability to add functionality) has been designed into the Fusion system from the outset, will drive its robustness. Lack of robustness will result in increasing number of new defects being introduced with upgrades, i.e., breaking functionality, which previously worked. While some “new” defect introduction is inherent to upgrades, it is imperative that a Fusion SoS perform thorough end-to-end regression testing, including from low-level fusion to high-level fusion, to ensure that the workflow has not been compromised.

B. The Systems Engineering Lifecycle

Different organizations e.g., NASA, DoD, commercial endeavors have their own variant of lifecycles, the Systems Engineering lifecycle comprises the following generic phases: (a) conceptual stage, (b) development stage, (c) production stage, (d) operations and maintenance with overlapping utilization and support stages, and (e) retirement stage. During the Exploratory and Concept stages, requirements are identified, or rather elicited. Ironically, everyone intuitively understands how important requirements definition is, and in fusion most Subject Matter Experts (SMEs) succinctly and accurately list their top requirements. Yet an often neglected effort is to identify the gamut of stakeholders that interact with entire multi-level Fusion SoS. For example, a Low-Level Fusion expert often clearly understands its own Input Process Output (IPO) requirements, but may overlook requirements from the downstream or upstream fusion workflow. SE provides enablers, such as Requirement Traceability Matrices (RTM), and Producer-Consumer lists, which track user needs from the mission objectives down to the individual single level fusion systems. Figure 2 highlights how the SE lifecycle model can be generalize to adapt to a variety of lifecycle models used by government and commercial organizations [10]. The red box highlights the main development and implementation period.
To develop fusion systems it is important to understand how current technologies dovetail with mission and user requirements, and leverage the exploratory and concept stage to achieve a high degree of technical maturity. Such an effort is usually done via prototyping and can be assessed with tools such as Technical Readiness Levels, which ensure that when the system is procured the implementation is tenable; e.g., with High-Level fusion systems, there is a risk of designing beyond the current technical capabilities, e.g., classifiers or pattern detection algorithms may have only been tested with simulated data, then during deployment and integration with more mature Low-Level fusion systems, disconnects often occur resulting in significant reengineering efforts.

C. The Systems Engineering “V” Model

Figure 3 depicts the Systems Engineering “V” model, which captures the SE lifecycle stages and incorporates what is arguably the most notable feature of SE: its integrated verification and validation at various levels in the system decomposition hierarchy (i.e., system, sub-system, component, subcomponent). Along the SE “V” path are rigorous process based on industry best practices such as Six Sigma, Capability Maturity Model Integrated (CMMI), and Decision Gates, that ensure that the subsequent stage is only executed when previous stage deliverables have been successfully met. Typical gates within government contracting include Preliminary Design Review (PDR), Critical Design Review (CDR), Test Readiness Review (TRR), and Factory Acceptance Test (FAT). These gates are by no means exhaustive, but highlight key events from a business and technical perspective: they feature the decomposition level from high-level of abstraction to detailed design, which is when the decomposition has reached a level where development can begin. Too often Fusion systems tend to get ahead of themselves, deploying algorithms or creating data models before the details of how the various nodes interact have been fleshed out – this invariably results in fragmented and brittle Enterprise Multi-Level Fusion SoS.

IV. SE PROCESSES FOR INFORMATION FUSION

Systems Engineering by virtue of its multi-disciplinary nature is an excellent fit to dovetail with the development of Enterprise Multi-Level Fusion SoS, as by its own nature requires expertise in several technical fields such as database model, algorithm development, sensor (collection platform) expertise. The following sections highlight key SE processes and describe some challenges and opportunities encountered when implementing Multi-Level Fusion systems.

A. Requirements Management

A stakeholder is defined as anyone who participates in the system. For fusion this has special implications, as it should
include anyone along the entire end-to-end fusion workflow: from those who define the mission requirements down to the users of collection platforms (i.e., sensors), and then back up to the analysts in fusion centers where situation awareness and predictive algorithms are applied to develop actionable intelligence. Additionally, multiple organizations may be involved, and overlaps in mission requirements need to be reconciled; for example, adopting standards for data exchanges and processing among fusion nodes. Another challenge is that sensor platforms (e.g., radars, EO/IR, SIGINT) may already be deployed, so there is little that can be done to flow requirements down to the Low-Level fusion collection points. A way to work with these constraints is to standardize the immediate post-processing, which can usually be done with software upgrades without forcing hardware changes to the sensor. Having a Process Refinement (i.e., feedback) loop on types of data that provide the most impact to achieve a particular objective may lead to the procurement of new types of collection platforms. Hence, having representation from all stakeholders present as well as understanding the procurement cycle is critical in gradually improving the Enterprise Fusion SoS. Establishing requirements for the often forgotten metadata, that facilitates processing and discovery is also a key consideration that requires input from all stakeholders. Multi-Level fusion is unique from Low-Level fusion in that it uses a very diverse number of sources; by diverse it is meant that the provenance of the low-level data itself comes from multiple organizations that impose their own need-to-know or access control to their data. This may be frustrating for the fusion analyst, but it is here where a chief systems engineer can be valuable in managing the data requirements, and flowing them down to the collection platforms. It is often possible to generate alternative product derivatives (e.g., different resolutions, or redacted data) that can still serve as input to downstream processing and yet preserve the integrity of need-to-know. Accordingly, without such enterprise requirement decomposition and traceability effort, the result is often a fragmented enterprise fusion process where many nodes may lack the adequate data. Once a comprehensive list of stakeholders has been generated, an enabling tool is to develop a producer-consumer matrix. This matrix includes not only data, but also services (e.g., processing functionality, retrieval, dissemination). The advantages of this SE process is that each fusion center (node) can align their Input-Processing-Output (IPO) workflow with their business needs, and ensure that there are no gaps. The following is a list that is often used in the SE process for identifying stakeholders:

- Individuals or organization responsible for the mission
- Individuals in the entire fusion workflow (from low-level to high-level fusion)
- Third-Party Individuals or Organizations that Interact with the System
- Individuals involved in indirect support activities, e.g., Systems Administrator, Database Administrators, System Architects, Information Assurance

However, using the Input-Process-Output (IPO) model alone may not produce a comprehensive list, and it may leave out important documentation that has been produced during the exploration and concept phases. The following documents should be part of the requirements analysis process as they are a useful source to reconcile and align requirements with Enterprise Multi-Level Fusion SoS objectives:

- Concept of Operations (ConOps)
- Concept of Deployment
- Concept of Support
- Concept of Deprecation
- Measurements of Effectiveness (including MOE data)
- Requirement Traceability Matrix

Once the high-level Enterprise Fusion SoS requirements have been compiled, then the decomposition starts. This is where requirements are “translated” from user needs and high-level mission language to technical functional requirements and constraints that are implemented by the Fusion architects, those who will design the different multi-level fusion components. Although technically requirements should only specified “what” not “how”, there should not be any ambiguity in the requirement language. Requirements are broken down to the point where implementation and development can start, and should be reconciled with technical standards and interfaces among all Multi-Level fusion components. Many Enterprise Multi-Level Fusion SoS, which are fragmented are often a result of poor requirement management. The following list highlights select key considerations when specifying requirements:

- Requirements should be necessary (not optional)
- Clear, complete and consistent
- No subjective language (e.g., “user friendly”)
- Achievable and bounded (i.e., not open-ended)
- Traceable along the decomposition hierarchy
- Verifiable (testable)

A key in establishing a Fusion Multi-Level architecture baseline is to formally document all interfaces. Interface definition fall into two broad categories: (a) Internal to the System (b) External to the System. Traditionally, the boundary of external systems is that point beyond the purview of the design and development effort, e.g., legacy systems or a system being developed by another effort. In the case of a legacy system, there should be an existing Interface Control Document (ICD), which guides the design requirements. Internal interfaces are within the boundary of the system, e.g., component to component, or interfaces between the application layer and the database layer.

**B. Integration**

Technically, integration is the process of assembling a system that is consistent with its architectural design, by
combining components to form the complete product specified by the requirements [10]. Multi-Level fusion presents a unique challenge in this aspect, as it is a complex SoS comprising hardware, software (including Multi-Level fusion algorithms), data, processes, and policies. The implication is that integration should be done not only end-to-end but also throughout the many layers described above. Additionally, integrating the fusion workflow is a challenge as this process is performed with a bottoms-up approach. This means that true end-to-end test may not be possible until after the Enterprise Multi-Level Fusion SoS has been completely integrated. Unfortunately, if approached in this manner, it won’t be possible to detect any defects until it is virtually too late. Several SE processes ensure that this risk is mitigated.

Clear definition of requirements, formal documentation of all ICDs, simulation, and test data, will go a long way in ensuring that fusion nodes are compatible. The strategy is to actually move these integration activities to the development phase, so that unit tests for the fusion nodes can be executed with real data (usually available from legacy systems), or simulated data from the data producer as early as possible. Additionally, given that fusion nodes within an Enterprise belong to various organizations, which implies multiple jurisdictions, cross-organizational integration considerations should be captured in the requirements phase and formal requests, such as Government Furnished Equipment (GFE), and Government Furnished Information (GFI). Although the final product may indeed not be ready until final SoS integration, there are usually prototypes available for the purpose of allowing preliminary testing. The approach should be captured in an Integration Strategy document, and signed off (approved) by all pertinent parties.

C. Verification and Validation

Verification and Validation (V&V) is performed throughout the entire SE lifecycle. The difference between verification and validation is subtle, yet important. Validation is ensuring that the end system (or SoS) meets the customer expectations (i.e., has the right system been built), whereas verification ensures that the system meets specifications (i.e., has the system been built right). Ironically, some fragmented Enterprise SoS systems are due because the validation process was overlooked – the verification process confirmed that the individual fusion components are working as designed, and even approved – yet the overall Enterprise Multi-Level Fusion SoS does not meet the user needs, and accordingly the mission cannot be met. Following a Systems Engineering process ensures that verification and validation is executed throughout the architectural hierarchy and aligned with all stakeholders. Tools such as a Requirements Verification Traceability Matrix (RVTM) ensures that stakeholders are actively involved, that the right parties are accountable, and that the capabilities being developed are aligned with the mission. Testing is the major activity where verification takes place, and it comprises the following types of verification methods: (a) inspection: where an examination of a component is made against the corresponding documentation, (b) analysis: to show theoretical compliance via e.g., simulations, such as when verifying fusion algorithms, (c) demonstration: running a functional performance test or capability, e.g., information exchange or conversion functions such as used in fusion Extract, Transform and Load (ETL) processes, (d) Test: executed under controlled conditions (real or simulated) and with the ability to quantify and qualify the results for further analysis if needed, and (e) certification: usually performed by a third party to comply with rules and regulation. An example of this would be to certify a fusion algorithm on an aerial platform (e.g. UAV) to comply with 178-B safety standards.

During validation, objective evidence is presented to ensure that the Enterprise Multi-Level Fusion SoS meets its intended use in an operational environment (as it will be used by the end-users to achieve their mission objectives). Rather than ensuring technical requirements have been met, validation looks at other user and SoS documentation such as ConOps, law and regulations, industry standards, and organizational policies to ensure that the language has been correctly transcribed into requirements so that they can be verified. This process should be iterative and done throughout the SE lifecycle. A common practice in deploying Enterprise Multi-Level Fusion SoS is to, where possible, provide prototypes or beta-releases and allow end-users to try them and capture their feedback - stakeholder involvement should be proactively managed throughout the SE lifecycle.

D. Operations Maintenance & Deprecation

While in SE Operations & Maintenance, and Retirement (Deprecation) are considered distinct processes, for Multi-Level Fusion systems they should be tailored and be done in lock-step. Successful development and integration alone does not guarantee that the Fusion SoS will enjoy real-time effectiveness. A Multi-Level Fusion SoS is imputed with uncertainty stemming for its use of forensic and predictive analytics. This creates a highly dynamic system where feedback is critical to refine the process. While the Fusion SoS should have been designed with fusion Process Refinement level in mind, it is not until the Fusion SoS is operating in the “field” under real-time conditions, and executing “live” missions that the system can be actually tuned. The challenges this presents to a Multi-Level fusion system is that it is more likely that several fusion nodes along the fusion workflow need to be adjusted. Here is where Measurements of Performance (MOPs) with enough granularity to identify components, subsystems, or systems that require adjusting becomes paramount to help evaluate the overall SoS effectiveness in achieving the mission. Techniques such as usage analysis, can identify which systems and which data contributes the most to the development of a successful course of action. This data can be used to either determine the next
upgrade, or whether a fusion node that is ineffective should be replaced or retired altogether.

V. DoDAF AND ENTERPRISE FUSION

The System Engineering concepts presented herein are high-level descriptions, as such they are not prescriptive, i.e., they do not provide enough guidance to implement a Systems Engineering process for Enterprise Multi-Level Fusion SoS. In this vein the Department of Defense Architecture Framework (DoDAF) offers a rich set of enablers, which can serve as a template to formulate a Systems Engineering plan for Multi-Level Fusion SoS. DoDAF offers multiple views, which act as enablers for visualizing and comprehending the large scope which is typical of very large and complex Multi-Level Fusion Systems. These views can capture the special fusion behavior, community of interest vernacular or taxonomy, data and metadata, structures, and interfaces in a cohesive and traceable fashion, which guarantees architectural consistency. Figure 4 depicts the views of DoDAF 2.0; DoDAF 2.0 Views extends and adds to DoDAF 1.5 views. Note that the DoDAF views can be integrated with the System's Engineering “V” and other design principles, such as Axiomatic Design.

While DoDAF evolved from the C4ISR AF (Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance Architecture Framework), and accordingly optimized for large complex SoS in the defense domain, it is nevertheless well suited for Enterprise Multi-Level Fusion SoS, irrespective of domain. With special considerations for including standards, data and information, services, capabilities and operations, DoDAF threads these views together, enabling a cohesive and integrated SoS architecture, one that is (a) vertically aligned: consistent across multiple layers of granularity, and (b) horizontally aligned: cohesively integrated among all of its architectural components [11]. Figure 5 depicts the Operational Views (OVs), Technical Views (TVs), and Service Views (SVs), and how subordinate reference architectures and taxonomies can be cohesively mapped to an overarching Enterprise Architecture (EA) reference model.

Another salient characteristic of DoDAF is that it does not only include the System Engineering processes, but it also covers six major processes among them: (a) Acquisition Systems, (b) Capabilities Integration and Development, (c) Capabilities Portfolio Management, (d) Planning and Budgeting, (e) Operations Planning, as well as (f) Systems Engineering. Of special interest for Enterprise Fusion Multi-Level SoS, is that DoDAF address current topics that are prevalent in today’s fusion deployments, such as Net-Centric Concepts, Service Oriented Architecture (SOA), and backwards compatibility. There is literature on how to tailor DoDAF for specific technologies such as SOA [12], which further facilitates the leveraging DoDAF. The objective would be for the fusion community to adopt principles of Systems Engineering with a framework such as DoDAF and tailor it for Enterprise Multi-Level Fusion SoS.

VI. SPECIAL SOS CONSIDERATIONS FOR FUSION

The following systems engineering criteria are integral in most approaches, however due to their particular relevance to Information Fusion they should be given special consideration.

A. Data Over-Production

There is an almost “intuitive”, albeit fallacious “feeling” that more data is better, and the more kinds of data there are, the more algorithms can be run, and more information can be extracted to develop a course of action. Nothing could be
further from the truth; the more data, additional care needs to be taken to clean the data, e.g., remove outliers and noise. Furthermore, with Process Refinement tools such as Minimum Description Length or other Information Theory based models, it can be proven that significant portions of the data may not contribute to either the forensic or predictive capability of the system – the implication, that entire sensor or collection nodes may not only be superfluous, but are in fact degrading the quality of the algorithms and should be replaced or retired.

B. Data Over-Processing

While it stands to reason that some data will feed several nodes, and that the same data may be processed multiple times with different algorithms. If care is not exercised, it is easy to lose the pedigree, and to fall into a spiral where data is being constantly degraded.

C. Data Over-Migration (Distributed Fusion)

Another aspect of distributed data fusion, executed over fusion nodes that may be under the jurisdiction of several organization, is that every node may want all the data, resulting in duplicate data what soon will become divergent. Again, the data pedigree is lost, or worse: the entire system can suffer from cross-contamination, where errors inadvertently introduced in one node are propagated throughout the entire end-to-end Multi-Fusion workflow.

D. Data Over-Retention (Stale Data)

A key concept in Information Fusion for actionable intelligence is that of timely data, from both the perspective of making it available on time, yet even more important, that information as a finite amount of time for its use – after which point it becomes stale. Yet even after missions are completed there is a tendency to retain all data on-line – this degrades overall system performance and presents a maintenance challenge. For predictive analytics usually a weighted system is used to give emphasis to more recent data. Accordingly, the best practice should be to archive stale data to tertiary media.

VII. Conclusion

An Information Fusion System requires more than bridging the gap between Low and High Level fusion, it is about focusing on the Mission - and enabling users to successfully achieve their mission. Architecting such an Enterprise Fusion system requires an understanding of the inherent variability in mission objectives brought about by the numerous and complex components – and not managing it a posteriori, but designing it in from the outset.

Ultimately, Information Fusion is about being able to generate insights based on data primitives that will lead to a course-of-action to preempt a causal force, or manipulate it to the mission’s advantage. Akin to how telephones are about communication, and cars are about transportation, Low and High-Level fusion maybe temporary abstractions, a practical way of managing the current domain problem within the context of currently available implementations and solutions. For example, there is a trend to incorporate high-level fusion capability in airborne sensors for example, which will allow the platform to execute an end-to-end fusion workflow in real-time. This requires new paradigms where data collection, exploitation and dissemination are executed concurrently in a semi-autonomous and semi-disconnected environment.

In this vein, Systems Engineering serves as an enabler to not only bridge the Multi-Level fusion gap, but also to ensure that Information Fusion advances in lock-step with the demands of the fusion users, by providing an integrated engineered solution to an ever morphing domain problem. Accordingly, to achieve a successful Enterprise Multi-Level Fusion SoS requires more than having Fusion SMEs; it is necessary for Fusion Engineers to adopt Systems Engineering principles to ensure that mission and user needs, as well as technical advances in Fusion are cohesively implemented, integrated, maintained, and managed in lock-step.

REFERENCES


