Enabling GEODSS for Space Situational Awareness (SSA)

Sam Wootton
The MITRE Corporation

I. ABSTRACT

The Ground-Based Electro-Optical Deep Space Surveillance (GEODSS) System has been in operation since the mid-1980's. While GEODSS has been the Space Surveillance Network's (SSN's) workhorse in terms of deep space surveillance, it has not undergone a significant modernization since the 1990's. This means GEODSS continues to operate under a mostly obsolete, legacy data processing baseline. The System Program Office (SPO) responsible for GEODSS, SMC/SYGO, has a number of advanced Space Situational Awareness (SSA)-related efforts in progress, in the form of innovative optical capabilities, data processing algorithms, and hardware upgrades. Each of these efforts is in various stages of evaluation and acquisition. These advanced capabilities rely upon a modern computing environment in which to integrate, but GEODSS does not have one—yet. The SPO is also executing a Service Life Extension Program (SLEP) to modernize the various subsystems within GEODSS, along with a parallel effort to implement a complete, modern software re-architecture. The goal is to use a modern, service-based architecture to provide expedient integration as well as easier and more sustainable expansion. This presentation will describe these modernization efforts in more detail and discuss how adopting such modern paradigms and practices will help ensure the GEODSS system remains relevant and sustainable far beyond 2027.

II. BACKGROUND

The Ground-Based Electro-Optical Deep Space Surveillance (GEODSS) System operates in a multi-tower configuration in three locations around the globe. It has been operational since the mid-1980s, serving as a deep space surveillance workhorse with no significant technological update since the 1990s. This means GEODSS continues to operate under a mostly obsolete, legacy data processing baseline. Since GEODSS' most recent, significant modernization, hardware and software technology has profoundly evolved.

The System Program Office (SPO) responsible for GEODSS, SMC/SYGO, has a number of advanced Space Situational Awareness (SSA)-related efforts in progress, in the form of innovative optical capabilities, data processing algorithms, and hardware upgrades in various stages of evaluation and acquisition. These efforts support SMC/SYGO's rapid and aggressive vision committed to providing premier space surveillance via ground-based optical mission systems. The SPO is currently exploring partnerships and integration opportunities with many SSA capability providers across Government, Industry and Academia. These capabilities will rely upon a modern data processing environment in which to integrate, but GEODSS does not yet have one.

III. IMPACTS/DISCUSSION

Over the coming years, the SPO will continue to execute numerous mission infrastructure and capability improvements. These planned enhancements include continued deployment of a multi-phase Service Life Extension Program (SLEP), which modernizes various subsystems throughout GEODSS. SLEP's primary goals are to extend GEODSS' technological life and reduce long-term sustainment costs, with ancillary goals of lowering capability integration costs and easing computing hardware expansion. The current phase focuses on computing hardware and features scalable solutions based upon hardware virtualization hosted on x86 commodity hardware.

Scalability and hardware virtualization are foundational to evolve GEODSS from where it is today into what it has the potential to become, which SLEP features. A horizontally scalable architecture allows pooling of additional nodes to achieve increased performance or storage, while a vertically scalable architecture allows the addition of more resources, such as processors or memory, to existing hardware for increased performance. Both attributes are important to GEODSS' future capabilities as storage, processing, and modernization requirements expand. Leveraging commodity hardware with virtualization is key to this solution as well. While commodity hardware expedites procurement, virtualization simplifies integration, as well as improves system reliability, maintainability, and availability.

In close alignment with SLEP, the SPO is also modernizing the data processing software architecture. A multitude of contractors, developers, and leaders sustained GEODSS' software for more than thirty years and produced today's "spaghetti code" that causes future sustainment challenges.   Some examples include code copied and pasted instead of referenced, archaic messaging, and data storage design and techniques that are not aligned with modern practices. The result of not addressing the architecture is more cost and time expended to fix or enhance software capabilities, or less capability for a given cost. Capability integration into the legacy software is not expedient nor sustainable as experimental prototyping requires significant hardware and software hardwiring. Additionally, the legacy code base and structure are not aligned with current and future software generations, making talent acquisition more difficult.

To address the current software architecture challenges, the SPO requires the new architecture to be loosely-coupled, modular, and "integrable". A loosely-coupled architecture has software elements with little to no knowledge of or dependency on other elements such that modifying one element has little to no effect on other software elements. Modular software is a style that organizes functionality into separate, interchangeable modules that implements one piece of functionality. "Integrable" is a term coined by the Software Engineering Institute (SEI) at Carnegie Mellon University (CMU) meaning a component's interfaces are well-defined and well-documented [1]. Well-defined interfaces will allow capability providers to understand how to develop and implement an interface to GEODSS and effectively "plug-in" to it. Aligning with these attributes while closely following sound coding practices and standards will prevent "spaghetti code" while also simplifying sustainment and expansion.

To best achieve the SPO's technical goals and sustainment vision, the GEODSS sustainment contractor recommended a microservice architecture. A microservice architecture shares some general, strategic commonality with a Service Oriented Architectures (SOA) approach, but microservices could be viewed as a further SOA refinement or a next step in the software evolution. Early SOA patterns did not stipulate how to architect a service, but developers delivered monolithic services featuring a single, logical executable tied to a single, monolithic database and executed within a single application or web container. These early services were a step in the right direction and provided a movement away from legacy stovepipes or "cylinders of excellence". SOA patterns were not bad or unsustainable, but they were not autonomous, modular, or resilient like microservices.

Microservices are a suite of small services built around a capability with decentralized data storage providing run-time and sustainment advantages. Each service independently deploys in its own container with its own interface, functionality, and data management. System resiliency increases because each service runs in its own container, which can be individually and continuously monitored as well as automatically restarted upon failure. Legacy systems and their monolithic services required an application or system restart upon failure, thus negatively impacting system downtime, while the microservice approach provides faster and automatic restart resulting in little to no system downtime. Additionally, independent containers also provide scalability as the system can appropriately load-balance services. Heavily used or resource-heavy microservices can be reallocated within the virtualized infrastructure to ensure system loads are optimally balanced.

Fig. 1 visually depicts the architectural differences between microservices and the aforementioned monolith. Within a monolithic architecture, all services run within the same container or single application and share a single data store, as shown on the left. Microservices feature single, individually deployed services having an independent data store, running within its own container, as presented on the right. Occasionally, a data store (i.e., database) can be shared by a limited number of services, but that is not the general pattern [2].

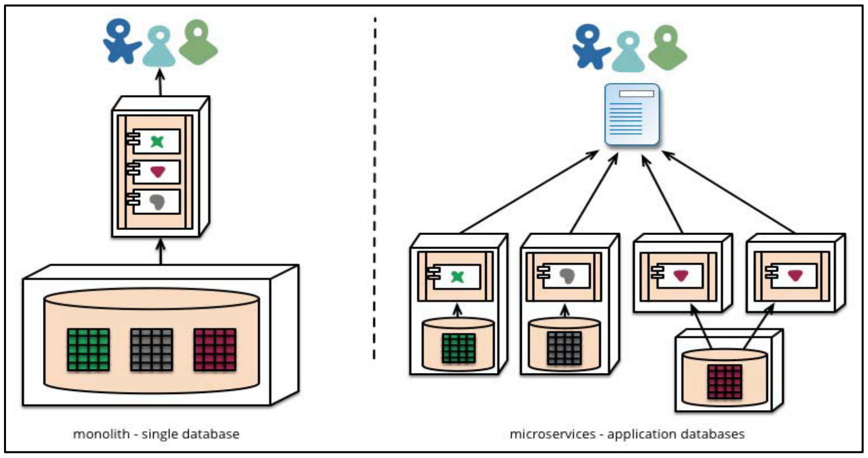
monolith - single database
microservices - application databases

Fig. 1. Monolith vs. Microservices

Microservices' modular characteristics of decentralized storage and independent functionality make them well-fit for efficient integration and long term software sustainment. Data storage belongs to an individual service, or can be shared across very few, offering less complex data models and designs. Furthermore, developers are not tied to a single language, database vendor, or specialized product. The architecture can support a multitude of languages and data storage products, which can be advantageous as GEODSS explores capability integration with external providers. Independent functionality contained within a microservice provides less complex code resulting in more expedient sustainment than monolithic services. Enhancements and/or maintenance may result in modifying numerous services or data repositories, and when this occurs, regression and unit testing become simpler due to the microservices' compartmentalized nature. Tests can be less complicated and more expedient, permitting enhanced or even automated integration testing. Large, well-known cutting-edge technological companies like Amazon, eBay, and Netflix successfully applied a microservice architecture that implements these advantages.

A focus on more testing, integration, delivery, and deployment to the point where all steps become continuous is a key tenet of a "DevOps" paradigm, which microservices support. DevOps is short for "Development Operations" and involves collaboration between operations and development engineers throughout the entire development lifecycle, from design through development to production. Some important DevOps features include an increased focus on end-product delivery, improved product quality through continuous insight into delivery, and reduced time-to-field by using continuous integration feedback. Within DevOps, the customer not only works directly with the developer throughout the entire product's development, much like an Agile or Lean paradigm, but concentrates on business results by focusing on speed and quality of software through continuous insight [3]. Fig. 2 depicts the software development evolution from current, longstanding practices into the DevOps paradigm where continuous integration and deployment further evolve into continuous operations [4].

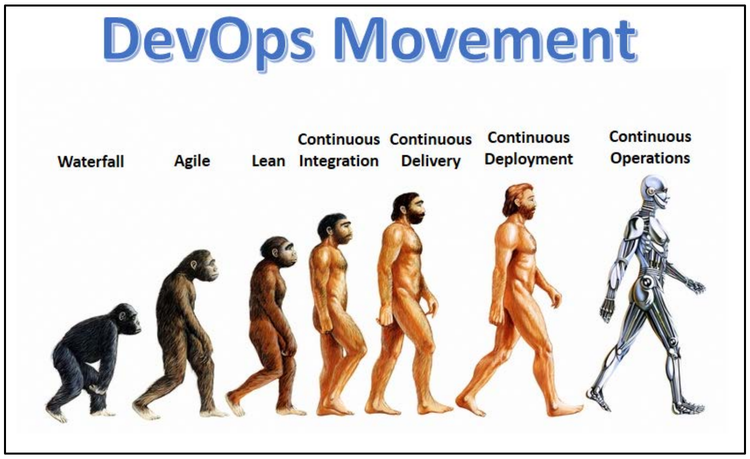

Fig. 2. DevOps Movement

The GEODSS SPO is taking steps towards a full DevOps paradigm by pursuing microservices. The GEODSS' sustainment contractor prototyped a set of microservices exploiting the new hardware and software architecture and will soon prepare those microservices for operation. SLEP's infrastructure will also permit an incremental delivery instead of a large, "big bang" delivery most programs execute. An incremental microservice fielding provides advantages such as flexible execution and less technical risk due to a number of legacy integration challenges. The SPO will continue to incrementally deliver a new set of data processing microservices until all functionality is instantiated and the legacy code can be fully retired.

IV. CONCLUSION

Upon implementing the new hardware and software environments, GEODSS and its SPO will find themselves in a position to expediently incorporate partner capabilities in a sustainable manner. By taking advantage of technology paradigms that ease sustainment and integration, new capabilities will integrate without storage or processing concerns and can "plug-in" due to well-known and defined software interfaces. This robust and extensible infrastructure provides the foundation and blueprint for continual expansion and modernization. Realizing the modern architecture's benefits and newly integrated capabilities, GEODSS will publish not only data, but also SSA knowledge, thus making its relevance last far beyond original intentions.

V. REFERENCES

[1] Software Engineering Institute. "A Framework for Software Product Line Practice". Retrieved June 2016, from http://www.sei.cmu.edu/productlines/frame_report/softwareSI.htm.

[2] Fowler, Martin. "Microservices". Retrieved June 2016, from http://martinfowler.com/articles/microservices.html.

[3] Marschall, Matthias. (July 12, 2012). "How are Lean, Agile, and Devops related to each other?" Retrieved July 2016, from http://www.agileweboperations.com/lean-agile-devops-related

[4] Correlsense, Inc. (August 27, 2013). "'Continuous Operations' a great idea – but details of execution still matter". Retrieved July 2016, from http://www.correlsense.com/continuous-operations-a-great-idea-but-details-of-execution-still-matter/.