United States Marine Corps
Command and Staff College
Marine Corps University

2076 South Street
Marine Corps Combat Development Command
Quantico, Virginia 22134-5068

MASTER OF MILITARY STUDIES

TITLE:
MODERNIZING APPLICATION FRAMEWORKS AND INTEGRATING DATA FUSION FOR LOGISTICAL RESOURCE MANAGEMENT SYSTEMS TO ENABLE SUSTAINMENT FOR FUTURE USMC OPERATING ENVIRONMENTS.

SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF MILITARY STUDIES

AUTHOR:
Major Brian Mitchel Carthon

AY 2018-19

Mentor and Oral Defense Committee Member:
JD Work

Approved:
Date: 26 APR 2019

Oral Defense Committee Member: MATTHEW FLYNN
Approved:
Date: 4/26/19

LTCOL D.W. HARROW
D.W. Harrow
190426

S.L. OCHOA
COL USMC

Executive Summary

Title: Modernizing Application Frameworks and Integrating Data Fusion for Logistical Resource Management Systems to enable Sustainment for Future USMC Operating Environments.

Author: Major Brian Mitchel Carthon, United States Marine Corps

Thesis: Current logistical resource management systems lack the flexibility and integration needed to provide the support for the 2025 operating scenarios as described in the Marine Corps Operating Concept (MOC). In order for these systems to be effective, logistical resource management systems will need to leverage emerging technologies and industry best practices for application frameworks, development, and utilize a comprehensive approach to integrating data flows to facilitate decision making.

Discussion: Future conflicts and operating environments will entail reduced access to ports, increased reliance on Joint and host nation support all while supporting smaller dispersed units across larger areas of operations utilizing contested networks. To enable logistics sustainment operations in these future environments, the Marine Corps will need systems that are light-weight, dynamic, and robust enough to provide logistical resource management. Data from across the battlefield will need to be integrated to create a logistical common operating picture for decision makers. Current logistical resource management systems (LRMS) lack the flexibility and integration needed to provide the support for the 2025 operating scenarios as described in the Marine Operating Concept (MOC). Many of these future scenarios are heavily reliant on Joint operations. Current

Marine Corps systems do not integrate efficiently with other sister services' LRMS. In order for these systems to be effective, LRMS will need to leverage emerging technologies and industry best practices for application frameworks, development, and utilize a comprehensive approach to integrating data flows to facilitate decision making.

Conclusion: In future operating environments the Marine Corps will need the capabilities to establish and maintain information dominance. As most war fighting professionals would focus on the maneuver elements, the logistics community will need the same attention as we move forward in our next conflict. Marine Corps LRMS will need to be reevaluated in order to be effective in future operating environments and provide the comprehensive logistics common operating picture (LogCOP) needed. By leveraging modern application frameworks such as virtual machines, microservices and containers, the Marine Corps will have a more portable and resilient platform to host and operate LRMS. By providing a common application development and test environments, employing software development kits, and application programing interfaces, the Marine Corps will be able to rapidly adapt to emerging needs and capability gaps. Given the proposed data management architectures and employing industry best practices, LRMS will be able to provide the capabilities needed to allow for improved LogCOP and opportunities to leverage modern data management tools such as predictive analytics and artificial intelligence tools to provide effective logistic support and facilitate decision making at the speed of relevance.

Disclaimer

THE OPINIONS AND CONCLUSIONS EXPRESSED HEREIN ARE THOSE OF THE INDIVIDUAL STUDENT AUTHOR AND DO NOT NECESSARILY REPRESENT THE VIEWS OF EITHER THE MARINE CORPS COMMAND AND STAFF COLLEGE OR ANY OTHER GOVERNMENTAL AGENCY. REFERENCES TO THIS STUDY SHOULD INCLUDE THE FOREGOING STATEMENT.

Table of Contents

Executive Summary ii
Disclaimer iv
Illustrations vi
Preface/ Acknowledgments vii
Introduction 1
Future Operating Environments 1
Current Logistical Resource Management Systems 3
Modern Application Frameworks 5
- Virtual Machines 6
- Application Programming Interfaces 8
- Microservices 10
- Containers 12

Application Development 14
- Application Development and Test Environments (ADTE) 15
- Software Development Kits 16

Data Management 17
- Authoritative Data Sources 18
- Data Schemas 19
- Multidimensional Databases 20

Solution Set 22
- LRMS Application Modernization 22
- LRMS ADTE 24
- LRMS API Standardization 24
- Data Management Enablers 25

Conclusion 26
Future Work 27

Illustrations

Figure 1: Example of Three Tier Architecture 4
Figure 2: Example of VM Architecture 6
Figure 3: Monolithic / Microservices Approach Comparison 11
Figure 4: Example of Container Architecture 13
Figure 5: Example of Multidimensional Database Architecture 21
Figure 6: Proposed LRMS Data Management Architecture 26

Preface/ Acknowledgments

During my tour at Marine Corps Systems Command from 2015-2018, I was working in the field of Application Migration and Marine Corps Enterprise hosting solutions. During my work there, I was able to use my background in Computer Science to advise senior leaders on an array of Cyber and information technology related issues. As in many organizations, adopting new technologies can be challenging due to the significant amount of time and resources expended to determine "What" technologies will bring value to an organization but fail to have a comprehensive approach to determine "How" to implement such initiatives. As the Marine Corps continues to innovate and take of advantage of emerging technologies, I was able to experience firsthand some of the challenges the Marine Corps will need to overcome to implement these emerging technologies. As a logistician by trade, I realized that the Marine Corps will need to fundamentally change the way it develops and builds its logistics systems and handles data flows to be effective in future operating environments.

I would like to thank my lovely wife Trashonda for providing devoted support during this process as well as my Command and Staff faculty mentors Mr. J.D. Work and Dr. Matthew Flynn for their mentorship and guidance through this process.

Introduction

Future conflicts and operating environments will entail reduced access to ports, increased reliance on Joint and host nation support, and providing support to smaller dispersed units across larger areas of operations utilizing contested networks. To enable logistics sustainment operations in these future environments, the Marine Corps will need systems that are light-weight, dynamic, and robust enough to provide logistical resource management and use data from across the battlefield to create a logistical common operating picture to facilitate decision making. Current logistical resource management systems lack the flexibility and integration needed to provide the support for future operating environments. In order for these systems to be effective, logistical resource management systems (LRMS) will need to leverage emerging technologies and industry best practices for application frameworks, development, and utilize a comprehensive approach to integrating data flows to facilitate decision making.

Future Operating Environments

As Marines continually refine their warfighting skills, logistics capabilities will also need to improve to keep pace the dynamic array of mission sets expected to be encountered now and in the near future. The 2018 National Military Strategy (NMS) is expecting the "Joint Force to sustainably compete to deter aggression in three different regions simultaneously while defending U.S. interest from challenges from challenges from below the level of armed conflict".[1] In order to support this mandate, logistical resources will need to be optimized and shared across the Joint Force in order to be successful.

Planners have developed visions of future operating environments and have outlined them

in seminal documents such as the Expeditionary Force 21 and the Marine Corps Operating Concept (MOC).[2] Both documents outline future warfighting environments that are highly contested and will need a highly integrated Joint Force able to share information at the speed of relevance in order to defeat the enemy. As adversaries continue to challenge the ways the U.S. projects power, the Marine Corps will continue to innovate to find and exploit opportunities. Innovative concepts such as Expeditionary Advance Basing Operations (EABO) utilizing distributed operations supporting joint elements will be the norm.[3] Marine application workloads may need to operate out of Navy ships or Army datacenters. Current logistical resource management systems will need to be designed or modified with this construct in mind. This future operating environment will require elements of the Joint Forces to have visibility of and to efficiently tie into all logistical resources available in an area of operations. The dynamic nature of conducting EABO will provide scenarios where Air Force units may be logistically supporting Marine aviation elements with repairs, fuel and weapons on one island chain while Navy units may be intermittently logistically supporting Marine and Army ground units distributed across several island chains due to adversary anti access/ anti denial capabilities. To enable support in this dynamic environment the Marines along with the Joint Force will need interoperable systems that are able to monitor readiness, supply chain performance, health service statistics, distribution networks, operational contracting support while feeding that information in real-time or near real-time to enable decision making at the speed of relevance.

Ongoing initiatives such as the development of the Naval Logistics Integration (NLI) playbook, aims to address the following in the Joint environment: Logistics responsiveness and agility, Improved and sustained combat support readiness, Reduce the logistics workload both

afloat and ashore, and Recapitalize the funding of naval logistics processes for more efficient use of resources.[4] This initiative has been on-going since 2012 and has made progress in NLI but is hampered by the lack of interoperable logistical resource management systems between the Marine Corps, Navy, and Coast Guard.

Current Logistical Resource Management Systems

Current Marine Corps and Joint LRMS are not suited to provide the logistical common operating picture needed to support the future operating environments as described by the MOC, Expeditionary Force 21, and innovative concepts such as EABO. The ability to share information across the Joint Staff at the speed of relevance is lacking. Services are responsible for logistically supporting elements at the operational and tactical level.[5] Component forces deploy with redundant capabilities and are heavier than may be required due to the inability to efficiently tie into Joint Force logistics frameworks. This has allowed for services to continue to develop logistical resource management systems that are stove-piped and isolated. Data streams generated from these systems circulate within the service and are aggregated inefficiently by inputting data from one system to another at the Joint operational level to build a logistical common operating picture (LogCOP). These LogCOPs often consist of incomplete, stale, or irrelevant data that hinders logistical planning and decision making at the "speed of relevance". The Marine Corps is the only service that is not tasked to be responsible for overall operational logistics to the Combatant Commander but is reliant on sister services to provide operational and theater logistic support.[6] This dynamic is especially problematic as few of the Marine Corps logistical resource management systems have the ability to integrate into Joint or sister service operational resource management systems.

Majority of current LRMS in the Marine Corps are built using obsolete technology and frameworks such as mainframes, traditional three tier architectures, and relational databases. Mainframes are designed to handle high input/output. Mainframes are known for their large size and amount of storage and processing power. Though reliable, mainframes are expensive to maintain, have specialized operating systems, lack the redundancy and resilience for their workloads. If a mainframe is damaged, all workloads are stopped until repaired. Application portability is an issue within this construct as physical system hardware will need to be duplicated in order to move an application to another location. Traditional three tier architectures are client /server architectures where process logic, data storage, and user interface are developed and maintained in independent modules on separate physical platforms (Figure 1). These architectures are reliable but are complex and also fairly monolithic. Changing data streams or adding new functionality requires significant refactoring. Scaling resources to meet demands is an issue as physical hardware will need be added to expand capacity. Constant management of resources is required to maintain peak performance. These archaic methods have contributed to stove-piped, monolithic systems that do not allow for the dynamic, highly portable applications and data fusion needed to provide the level of logistical support.

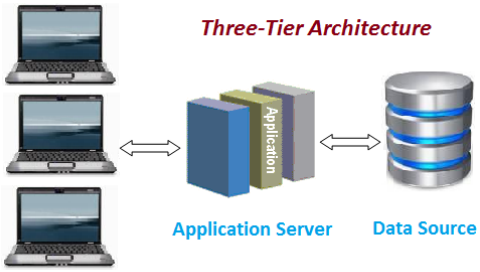

Figure 1: Example of Three Tier Architecture
Source: STC, "Difference between Two Tier and Three Tier Architectures", Software Training Class, January 13, 2013.

During an assessment conducted in 2017 of LRMS operating out of the Logistics Command (LOGCOM) datacenter in Albany, Georgia, out of the twenty-four applications and systems assessed thirteen were found to be operating using traditional three tier architectures or mainframes[7]. The assessment team also documented the lack of a comprehensive enterprise architecture for all applications and data interfaces within the data center. The applications are managed by a plethora of different contractors responsible for their specific system. The lack of modern application architectures degraded LOGCOM's application portability, survivability, and security. This is problematic as LOGCOM handles the preponderance of the operational logistics for the Marine Corps. This assessment did not encompass all LRMS utilized in the Marine Corps but gives a sample to infer possible trends across the enterprise.

Data aggregated from the DOD IT Portfolio Repository – Department of the Navy (DITPR DON) in March 2019 shows that the Marine Corps currently operate sixteen applications on mainframes. Fourteen of the sixteen mainframes currently operate LRMS.[8] Some of these key systems include Marine Corps Total Force System (MCTFS), Contract Divisions Documentation Control System (CDDCS), Automated Procurement System (APS), Transportation Management System (TMS) and the Marine Corps' newest LRMS, Global Combat Service Support Marine Corps (GCSS-MC).

Modern Application Frameworks

By analyzing the future environment and gaining some understanding of the current disposition of LRMS, the problem set can be framed on how the Marine Corps begins to modernize its LRMS in order to meet the future operating environment demands. To meet these

future demands, the Marine Corps and the Joint Force will need LRMS systems that are lightweight, portable, robust, secure, and integrated in order to provide the LogCOP needed to support maneuver elements and the Combatant Commander. In order to accomplish this, the Marine Corps will need to employ modern application frameworks such as Virtual Machines, Application Programing Interfaces, Microservices, and Containers.

Virtual Machines

At the core of most modern application frameworks is the Virtual Machine (VM). VMs are software programs or operating systems that have the functionality of a physical computer.[9] VMs are capable of executing programs like a physical computer. It is possible for a single physical computer to host multiple VMs. Storage and compute resources from a physical host computer are allocated internally to the VM for use. These resources are controlled by a hypervisor, which allows for communication between the physical host computer and the VMs. An example of VM architecture can be found in Figure 2. Limitations of storage and compute resources are dependent on the amount of resources available within the physical computer.

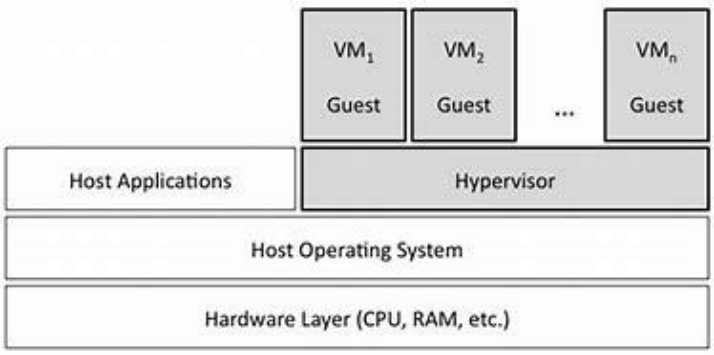

Figure 2: Example of VM Architecture
Source: Research Gate, "Typical Virtual Machine Architecture", Researchgate.net, August 12, 2016

The advantages of this framework are multiple independent VMs can run on a single

computer. VMs can be programmed to mimic hardware which allows for a reduction in the amount of physical hardware needed to support applications. Utilizing VMs gives the ability to abstract interfaces and visible resources onto the interface and resource of another system. These systems could be a physical system or other VMs.[10] This abstraction capability allows for systems that may have incompatible sub-components to be able to integrate and share information. This framework allows for a more efficient use of physical computer resources and allows for more workloads to be executed in parallel. Traditional three tier architectures are executed more efficiently utilizing a VM framework.

Because VMs are software based and not dependent on a specific hypervisor, VMs are highly portable and can be hosted on a plethora of platforms. As long as the host computer is operating a compatible hypervisor, multiple VMs performing different tasks can be managed from a single console. This application portability allows for systems to be hosted in standalone servers, datacenters, and cloud environments with minimal refactoring of the application.

VMs do provide improved security by providing the capability for VM components to be logically separated on the same host computer. Though VMs are logically separated, there are potential ways to gain access and exploit VMs. As stated before, VMs are software programs that have the functionality of a physical computers and need to be protected with the same due diligence of a physical computer. Many attack vectors are focused on the host systems hypervisors as this is the main interface between the host computers and guest VMs.[11] If a host computer is compromised and access to the hypervisor is achieved then an attacker will have access to the VMs on the host computer. To mitigate this, the hypervisor should be secured and

privileged access to the hypervisor should be closely managed and monitored. The ease of creating VMs is another security concern. VMs frameworks allow for VMs to be created and destroyed rapidly. Pre-established templates and automated scripts allow for VMs to be created with little effort. This capability unfortunately allows for resources to be allocated and forgotten about. These forgotten VMs continue to take up finite resources on the host computer, which can compromise a system.[12] If an attacker was able to get access to the host computer and compromise the hypervisor, exploiting a VMs resources could be a way to degrade or destroy a system. To mitigate security vulnerabilities standard pre-approved templates or "images" should be utilized to reduce malicious code into a system. Though VMs do provide a capability to implement a defense in depth, there are some security risks but they can be mitigated through a combination of robust physical security and ruthless enforcement of security policies.

Application Programming Interfaces

Application programming interfaces (API) is another core of modern application frameworks. APIs are sets software protocols that allow for applications to exchange or extract information with each other.[13] APIs allow for applications to communicate with other systems without having to know how they are implemented. When utilizing APIs, instructions are sent to a system via internet or Local Area Network (LAN) to a server. The server receives the instructions, reads them, executes the required actions, and transfers the results back to the requesting application. APIs can simplify systems and application development allowing for new or updated functionalities to be implemented into application frameworks quickly. This is critical in dynamic operating environments where gaps are identified and need quick solutions to be innovated and rapidly implemented. APIs allow for light-weight applications to be built as

the majority of the processing is conducted by the requested system. Only the data of the request and results are transmitted across the network.

There are three ways APIs are utilized: public, partner, and private.[14] Public APIs are released for the public use by an organization. This allows open source developers to integrate or innovate with a specific system or application. Partner APIs are shared with specific organizational partners that can add value or increase productivity. An example of this would be payment companies allowing a third party retail company's approval to develop payment solutions with their APIs. Private APIs are for internal organizational use. No data is shared outside the organization. These APIs are primarily for internal organizational system integration and reporting.

Google Maps is an example of a popular API. Google gives the public access to the Google Maps APIs. In turn, open source developers are able to build applications utilizing Google Map functionality and data. Utilizing this framework allows developers to saving time and money in developing their own custom solution. An example of an employment of Google Maps API is how restaurants utilize Google maps APIs for a restaurant's location and directions. Restaurants embed Google Map functionalities into their websites for use in locating and providing driving directions to their place of business. The restaurant did not have to create their own location or map functionality but instead leveraged Google Maps existing capabilities utilizing their public APIs.

APIs provide a layer of security by limiting the exposure to primary systems by sharing only

the data that is needed. APIs allow for organizations to gain access to a limited amount of system resources. In most organizations APIs are standardized, reinforced by disciplined security practices and governance. APIs are managed just like traditional software through the software development cycle of designing, testing, building, managing, and versioning. There are known attack vectors for APIs to be exploited by malicious actors. Structured Query Language (SQL) injection attacks are common on public and partnered APIs due to deliberate access to API documentation for third party developers.[15] These threats can be mitigated through the use of firewalls to validate incoming requests. Impersonation attacks are also a common method to breach APIs. Impersonation attacks are conducted by utilizing stolen API key codes, which identify legitimate systems to the API.[16] Those with API keys can use them to inject malicious scripts as an impersonated legitimate system. This can be mitigated by implementing additional authentication protocols in addition to API keys. A third attack vector for APIs are Man in the Middle (MITM) attacks. MITM attacks consists of an attackers positioning themselves in the middle of the API and requesting system. The attacker can intercept data or impersonate API or a potential user. Using secure data exchange protocols such as Transport Layer Security (TLS), Secure Socket Layers (SSL) or utilizing tunneling protocols can facilitate mitigating MITM attacks.[17]

Microservices

Microservices are a method of constructing an application using a group of loosely coupled small, lightweight, independent services. These services are smaller applications that execute a distinct task and can communicate with other services through APIs.[18] An example of the architecture can be found in Figure 3. These services can be developed, deployed, maintained, and scaled independently. Decomposing monolithic application functionality and providing

individual functionalities as services for a user or other applications provides modularity, scalability, and resilience to an application. Isolating these services at a refined level provides the ability to build resilient applications that can run services across different platforms. If there is a disruption in service of one microservice workload in an application, only the functionality of that service is affected.[19] Disrupted services requests can be shifted immediately to microservices operating in another location that is available. The rest of the application functionality will still be operational. Microservice architecture allows for individual modules to be written in different programming languages and be able to integrate into other components as long as the APIs are consistent through the framework. Because microservices are small and portable, they can be distributed to operate across different platforms. This provides an added defense in depth for security considerations. If attackers are able to gain access or compromise one function of the application, they will not get access to other functions of the application.

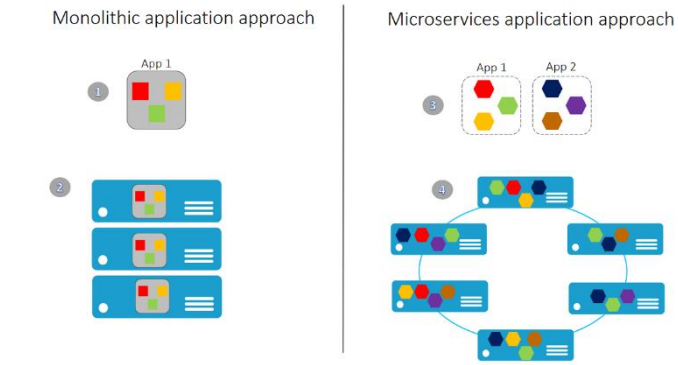

Figure 3: Monolithic / Microservices Approach Comparison

Source: Microsoft Azure, "Comparison between Application Development Approaches", Microsoft Azure.com, April 19, 2018

Though there is a lot of benefits to microservices, applying these frameworks does add to the level of complexity of an application and there are security implications. Employing microservices does provide a possible attack vector to a system with each instance created.[20] This potentially creates a wider attack surface for an application. To mitigate this, utilizing

trusted and validated code repositories for code reuse, ruthless enforcement of security policies must be enforced, and automated tools need to be utilized to maintain a hardy security posture. Because microservices utilize APIs to integrate into larger applications, employing robust API security practices will be paramount to facilitating accessibility of services. Due to the ease of scalability, utilizing automated security tools will be needed to securely scale microservices to meet demands.[21]

Containers

Containers are a standard way to divide applications into distributed objects consisting of microservices. This method can be used to componentize whole systems, abstracting them from physical platforms allowing the capability to move applications from environment to environment agnostic of platform[22]. Segmenting applications provides the ability to host them on different physical and virtual machines, on-premise, or in a cloud environment. Containers are similar to VMs in the relation to both can scale resources on demand providing elasticity and fault tolerance. Containers are dependent only on a specific operating system meaning a container can migrate to wherever that operating system is running.[23] Operating system resources are shared and orchestrated through the use of container management tools. A graphical representation of the architecture can be found in Figure 4.

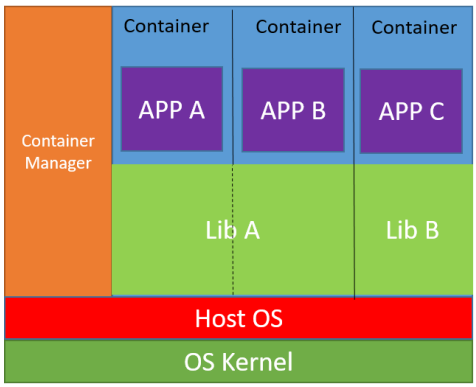

Figure 4: Example of Container Architecture

The employment of containers can be built into applications during development or legacy applications can be reengineered to utilize containers. Containers are ideal for use within Hybrid Information Technology (Hybrid IT) environments which will consist of a combination of on-premise datacenters and cloud-based solutions. APIs tie the application into the infrastructure provided. Automated tools are available to migrate containerized applications from environment to environment, as well as manage and monitor container capacity/ consumption resources across multiple environments.[24] Containers provide flexible portability across multiple hosting platforms and provide a capability to distribute application functions across multiple environments in the Marine Corps Hybrid IT infrastructure. Containers can efficiently monitor application performance from centralized management consoles.

Though containers provide flexibility and robustness to application frameworks, there are security considerations. Because container environments share operating system resources, a compromised container would have access to operating system functions.[25] To mitigate, hardening the host environment and access management of personnel and applications are pivotal. Similar to VMs, containers can be deployed based on a standard template or "image" of an approved set of functionalities.[26] These images can contain vulnerabilities which attackers

can use as possible vectors for exploitation. It is important to scan these images for vulnerabilities and to continually patch as needed. Containers make it easy to quickly build and deploy but security needs to be at the forefront of any modernized application framework.[27]

In a review of published case studies, the commercial logistical community is currently employing containers and microservices within their application and infrastructure modernization efforts. Companies such as Amazon are using containers and microservices to scale resources to fill requests, provide in transit visibility and inventory management services on demand to its millions of customers around the world as well as gain efficiencies and simplify in its own IT architecture.[28] Maersk, an international leader in shipping services, integrated containers and microservices to modernize its software resulting in savings in resource costs, faster application employment, and simplification of IT resource management and improved in transit visibility of assets[29]. The United Parcel Service (UPS) currently employs containers and microservices as the backbone in building applications to optimize package operations and delivery. Utilizing these frameworks increased UPS's application availability by giving them the ability to run, modify, and migrate applications across multiple hosting environments.[30]

Application Development

Adopting modernized application frameworks is one pillar in a comprehensive approach to application modernization. Current industry best practices employ speedy development operations by integrating development teams and operational teams together in a common environment and utilizing common development tool sets. These frameworks give the commercial sector the ability to innovate, develop, and deliver solutions at the rate needed to

keep pace with the dynamic array of requirements needing solutions to bring business value to an organization. Current Marine Corps development operations still use a stove-piped approach to development with little reliance on automation or common development tools sets. In order to develop, test, and deploy LRMS capabilities at the speed of relevance the Marine Corps will need to address its current antiquated development and test (Dev/Test) environments. To facilitate this capability the Marine Corps will need to employ standard automated Dev/Test environments and employ software development kits.

Application Development and Test Environments (ADTE)

Application development and test environments (ADTE) are platforms for building and testing applications. ADTEs facilitate rapid development of applications to meet new requirements or be modified to gain efficiencies as new technologies emerge. The Marine Corps currently hosts applications in different hosting environments consisting of enterprise datacenters, regional datacenters, and commercial cloud environments.[31] Each of these environments have their own unique Dev/Test environments and migration paths to production environments. Many application owners utilize their own isolated development sandboxes to innovate or test new concepts of functionalities. Once these new functionalities are matured, the application is manually transferred to the designated hosting Dev/Test environment for independent verification and validation before being deployed in a production environment for operation[32]. Utilizing this antiquated framework, the application owner builds the application three times before it is deployed. The average time for migrating applications into the Marine Corps enterprise datacenter is 260 days.[33] This timeline can be greatly extended if issues are experienced in the migration process. Current trends in delays consists of application owners

acquisition programmatic issues involving funding and personnel resources for development as well as frustrations in complying with Marine Corps Enterprise Network (MCEN) operating parameters and security protocols. This inefficient use of resources and time coupled with the lack of automated tools to shift applications through the development operations cycle degrades the Marine Corps ability to develop, test, and deploy at the speed of relevance.

Software Development Kits

As the Marine Corps begins to expand into multiple hosting environments, and standardize LRMS development platforms, constructing software development kits (SDKs) will be needed to rapidly create applications to meet operational demands. Software development kits are a set of tools and documentation used for writing applications for a specific platform or environment[34]. SDKs facilitate the integration of functionalities and data for applications. SDKs provide documentation on data access points within APIs to enable functionalities from other applications. SDKs are important because they provide the building blocks and roadmaps to create applications. This drastically cuts down on development time as discovery learning is reduced for application developers.

The current Marine Corps model allows for a plethora of software development tools to be used to develop applications. This is problematic as a developer will have to custom build applications depending on hosting environment parameters, types of operating systems, databases, APIs available as well as security requirements needed to operate. As each application developer comes up their own boutique solution, the complexity for integration across platforms increases. Often systems must be refactored in order to add functionality or

integrate with other systems. To minimize the spread of antiquated software development tools and simplify the integration of applications, LRMS stakeholders will need to establish a LRMS SDK.

In addition to providing a standard LRMS development and test environment, the creation of SDKs will facilitate the development of applications by accelerating the development and deployment of applications. Creating a standard SDK will allow LRMS stakeholders the ability to ensure that software development lifecycle best practices are utilized. Employing SDKs also ensures software quality is maintained and the ability to integrate functionalities are standardized. A standard SDK would improve the security of LRMS by potentially reducing the amount of bad code within an application that could be used for exploitation by malicious actors. SDKs can also be used to enforce current Marine Corps and DOD cyber security protocols. The SDKs created could be built to comply with newly revised Risk Management Frameworks (RMF) and expedite the Authority to operate (ATO) process[35].

Data Management

As the Marine Corps and the Joint Force look to integrate systems to support sustainment operations, the ability to share across the Joint Force will be a critical capability. The volume, variety and velocity of data will rise exponentially over the next ten years[36]. In future operating environments, the Marine Corps will look to exploit data from sources and sensors across the battle space. The ability to process and leverage this plethora of data into actionable information will be needed to facilitate sustainment in future operating environments. To handle this problem set the Marine Corps will need data management frameworks to enable the ability to

turn this raw data to information to actionable information. In order to provide the LogCOP needed to support the sustainment, the Marine Corps will need to look to modernize its data management framework to leverage data fusion and LRMS interoperability. To enable this capabilities, LRMS stakeholders will need to reevaluate authoritative data sources around the enterprise and establish data schema standards. In order to enable an improved data analysis capability, the Marine Corps will need to integrate multidimensional databases into its IT architecture.

Authoritative Data Sources

The current Marine Corps Data Management Strategy is a network-centric data strategy promoting "the use of shared resources, independence of data and data/ information exchange from applications /systems."[37] Despite this guidance, the Marine Corps currently employs a stove-piped approach to data management. The Marine Corps current framework is centered on authoritative data sources (ADS) provided by selected systems around the enterprise. Per the current Marine Corps Data strategy, an ADS is a recognized or official production source to publish reliable and accurate data for subsequent use by an organization[38]. An authoritative data source can be a mixture of numerous, separate data sources[39]. Many of these systems utilize traditional relational database management systems (RDBMS). Though reliable, RDBMSs are not effective in managing structured and unstructured data sets. Many of these systems have different ways and tools to exchange and access data with these ADSs. This can be problematic when looking to integrate data from multiple ADSs. Many applications often replicate redundant information outside of ADS which can cause issues with data quality, reliability and security. ADSs are often exposed to security exploits as they are often queried directly to the database

vice utilizing APIs.

To address the growing volume and variety of data collected, organizations employ modern data management frameworks utilize data warehouses in their IT architectures to support reporting, queries and decision making. A data warehouse is a collection of technologies used integrate data from multiple different sources under a common platform to make data available for new operational or decision support applications[40]. Data warehouses allow decision makers to gather, and analyze data from multiple sources, providing a comprehensive data set to aid analysis and decision making. Data in a data warehouse data is loaded and accessed in mass as a static "snapshot"[41]. When a change in data occurs, a new snapshot is created to access for processing. This ability to collectively integrate large pools of data from different sources provides the ideal platform for predictive analytic and artificial intelligence tools to be utilized to facilitate decision making.

Data Schemas

As the data volume, variety, and velocity increases, searching and retrieving relevant data will become increasingly difficult. One of the fundamental elements of data management are data schemas. A data schema is a framework for organizing and categorizing data in a database. Data schemas are critical as they define how data is related and associated with each other[42]. Data schemas include formats for semantics and content for data points such as nomenclature, date, locations and any specific elements needed by a specific mission area[43]. Many Marine Corps LRMS utilize data schemas related to their specific system or application. This can be problematic in integrating data from multiple data sources as it would be difficult to correlate

data from different systems. In some integration efforts, ad hoc data interfaces or mediators are created to convert one data schema to another before being ingested by another system.

Multidimensional Databases

Once authoritative data sources are identified, data warehouses are created and data schemas are validated, modern databases will be needed to managed and store data in a way where data can be aggregated and process queries quickly to support decision making at the speed of relevance. A multidimensional database management systems (MDDMS) is a database that is optimized for data warehouses and online analytical processing applications. RDBMS are optimized to manipulate individual records. RDBMS records are accessed by using a structured query language which is unsuited for data analysis applications since some grouping queries are difficult to express.[44] Multidimensional databases allow users to analyze large groups of records from multiple ADS. MDDMS store data in a cube format which allows for the data to be seen and understood from many dimensions or perspectives.[45] These cubes are collections of data structures used to organize data in a logical way. Each dimension is organized in accordance with data domains at different levels. An example of this architecture can be found in Figure 5. This high degree of structure associated with this framework allows for a simple and more natural language query system to express complex queries. Users can pivot the data to see information from a different view point, go to a lower dimension to find more detailed

information, or view a higher dimension to see an overview of information.

The commercial industry is currently employing a combination of RDMS and MDDMS in data management architectures in order to support data aggregation and analysis. Southwest airlines is currently employing MDDMS in order to identify requirement shortfalls associated with customer travel plans, fuel prices and security requirements.[46] MDDMS facilitated the creation of alternative cash flow models and 15-month predictive forecasts of potential requirements for the company. Ford Motor Company used MDDMS in analyzing energy consumption patterns within auto assembly plants with the goal of rescheduling peak usage to coincide with the time of the day where the cost of power was the cheapest. Ford was able to consolidate all energy meter readings for the company into a data warehouse and used a MDDMS to generate time series analysis to determine optimal production times to reduce energy costs.

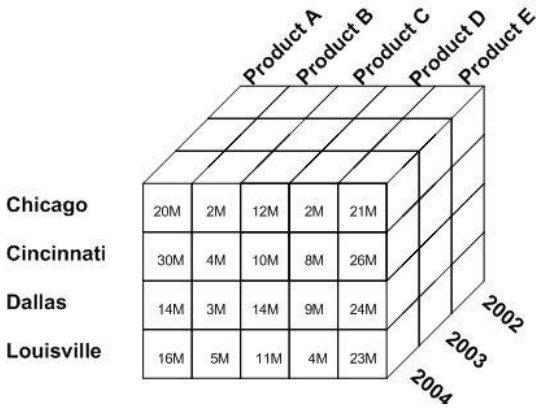

Figure 5: Example of Multidimensional Database Architecture
Source: Ramachandran, Regesh, "Multidimensional Databases", BayT October 8, 2014

Solution Set

The Marine Corps and the DOD are modernizing application hosting platforms and IT architectures to provide a more flexible and resilient capability. This effort consists of employing a combination of traditional datacenters, private and commercial cloud environments. In future engagements adversaries will target Marine Corps and Joint LRMS to disrupt the ability to force generate and sustain combat forces. Applications will need to be distributed across these multiple platforms in order to provide the resilience needed to adequately mitigate the expected disruption in Joint and coalition networks in a near peer engagement. In order to mitigate this threat, the Marine Corps will need to employ modern application frameworks. The Logistics Functional Area Mangers (FAMs) and key stakeholders will need to conduct a comprehensive review of all LRMS applications for modernization. This coalition will need to advocate for a common ADTE with APIs and SDKs to facilitate application development, testing and deployment for LRMS. In order to facilitate the data fusion a rationalization of LRMS including ADS need to be conducted as well an aggressive data schema standardization initiative. Incorporating all of these recommendations will allow for the Marine Corps to be in a position to provide sustainment in future operating environments.

LRMS Application Modernization

An aggressive application rationalization effort by Functional Area Managers (FAMS) and key stakeholders will need to be conducted to determine the viability, number of interfaces, data composition and data flow analysis of all LRMS. After a LRMS enterprise architecture is constructed, systems with similar capabilities need to be consolidated and redundant systems

should be cancelled. All systems and applications at a minimum need to utilize some sort of virtualized application architecture either VMs, containers or microservices were appropriate. These architectures will provide the hosting portability and resilience needed in future operating environments. As the likelihood for cyber-attack to Joint IT networks will be high, these architectures allow applications to distributed over multiple hosting environments providing resilience to service disruption. These architectures also provide a light-weight platform for LRMS to be pushed to the tactical edge when needed. Any system that cannot be refactored into a modern architecture need to be retired with those system functionalities absorbed into an existing system or new system developed.

Though GCSS-MC is classified as a mainframe by the DITPR-DON, it is a large system of systems that utilizes VMs at the core of its architecture running on a hyper-converged platform. In the Marine Corps development of GCSS-MC, it created a modern system that was far superior to older systems but employed a traditional framework that significantly limits deployability to the tactical edge. Breaking up the current GCSS-MC monolithic approach into a microservices and containers-based architecture would add flexibility, robustness and resilience to the system. Utilizing this approach would eliminate the need for expensive custom vendor specific hyper-converged platforms and would allow for the system to operate on Marine Corps Enterprise Datacenter commodity hardware be hosted on government cloud infrastructure or federated across both cloud and tactical on-premise solutions. Utilizing a container and microservice approach would provide a framework for a light weigh deployable version of GCSS-MC that can be operated at the tactical edge.

LRMS ADTE

The ability to quickly develop, test and deploy new applications or modify existing applications with new functionalities will be needed to keep pace with gaps in capabilities and integrating new emerging technologies. In order for the Marine Corps to securely expedite development operations for LRMS, stakeholders will need to advocate for a standard ADTE in the software development lifecycle for all LRMS. Employing these environments in a cloud-based format provides an efficient way for an environment to be quickly constituted and erased once on longer needed. This implementation can facilitate multiple types environments based on the hosting environments determined. Providing a common LRMS ADTE with all standard APIs for integration and SDKs needed to build applications will significantly reduce the time needed to develop and deploy a new capability. A common environment also allows applications to be build more securely as they will be able to take advantage of automation in the form of automated independent vulnerability discovery and validation to the greatest extent possible. In addition to utilizing pre-accredited platforms and templates vetted through the RMF process, this would improve the fidelity of the proof of secure development and resilience to exploitability.

LRMS API Standardization

Establishing standard APIs will facilitate the integration of systems and allow for lighter applications to be developed by utilizing functionalities that already exist in other applications. The Marine Corps should employ a combination of private and partnered API's. Utilizing Private APIs to allow for integration of internal Marine Corps systems and processes. Partnered API's would be released to and utilized by the Joint Force, operational partners and selected

commercial partners to facilitate interoperability, integration and data exchange. Utilizing this framework would allow the Marine Corps the ability to monitor and manage the access of applications have to other systems on the MCEN or other operational networks.

Data Management Enablers

Industry best practices are pushing to eliminate inconsistent data schemas within their data management architectures. Given the complexity of the DOD IT enterprise, standard data schemas across the DOD may not be achievable. Despite this, there is opportunity to focus data schema standardization across LRMS platforms. This will provide a common schema framework for all LRMS ADS and applications to build upon. Utilizing standard data schemas would assist in contextualizing data for greater analytical value. Though this solution would facilitate a homogenous data environment within the LRMS ecosystem, there will still be a need to interact outside of the ecosystem. Utilizing data mediators at these LRMS ecosystem data seams, would allow for data integration with the rest of the Marine Corps and DOD IT enterprise. The incorporation of data warehouses and multidimensional databases into the existing Marine Corps and Joint IT infrastructure would enable the data aggregation and analysis needed to facilitate decision making at the speed of relevance. The proposed LRMS enterprise data management architecture in Figure 6 would be a baseline to enable the data fusion needed to integrate LRMS across the Marine Corps as well as integrate data across the Joint force.

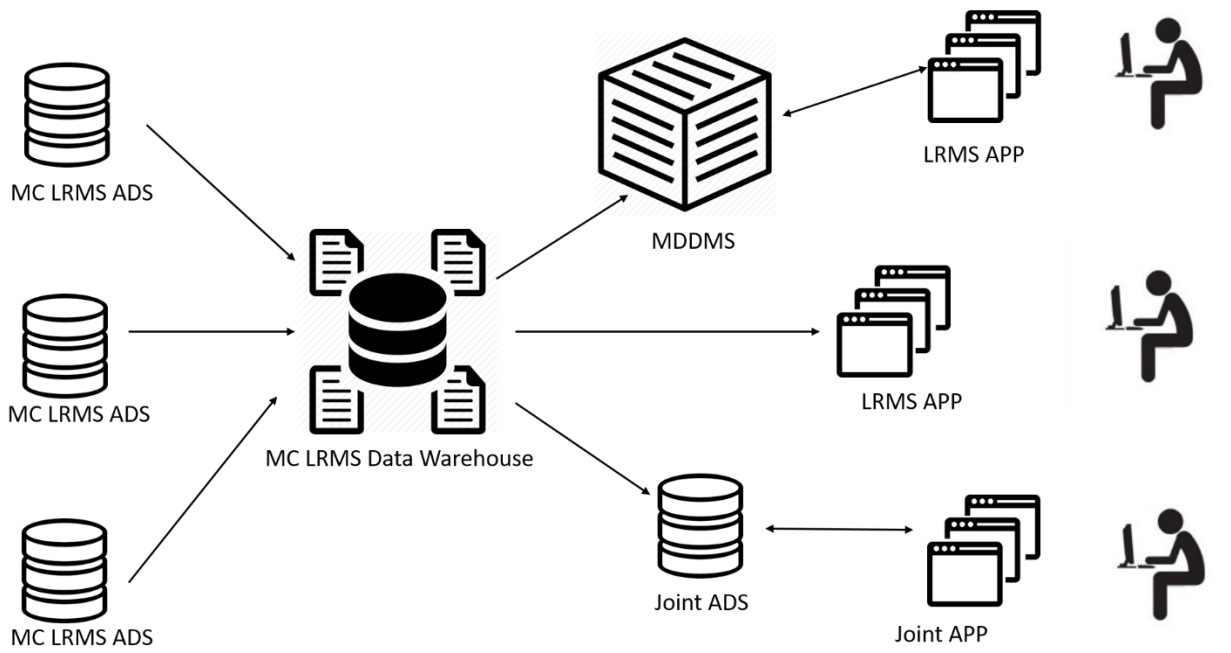

Figure 6: Proposed LRMS Data Management Architecture

Conclusion

In future operating environments the Marine Corps will need the capabilities to establish and maintain information dominance. As most warfighting professionals focus on modernizing maneuver element capabilities, the logistics community will need the same attention to deliver relevant capabilities to enable sustainment in next conflict. The Marine Corps LRMS will need to be reevaluated in order to be effective in future operating environments and provide the comprehensive logistics common operating picture (LogCOP) needed. By leveraging modern application frameworks such as VMs, microservices and containers, the Marine Corps will have a more portable and resilient platform to host and operate LRMS. Utilizing these modernized application frameworks will put the Marine Corps in a position to host applications anywhere including Joint or coalition hosting environments. This flexibility will be necessary to avoid

disruptions in services through kinetic or cyber fires as we will have the ability to automatically reconstitute entire systems when needed to continue to sustain the warfighter.

By providing a common LRMS ADTE employing standardized SDKs, the Marine Corps will be able to rapidly adapt to emerging needs and capability gaps. Standardized APIs will facilitate the integration of internal Marine Corps LRMS as well as with Joint and coalition partners when needed. Standardized APIs will allow the Marine Corps to manage the access that applications have to systems on the MCEN or other operational networks. Applying standardized data schemas and implementing the proposed data management framework (Figure 6), will set a baseline to aggregate the data needed instantiate an improved LogCOP. Enabling these data management frameworks will allow for opportunities to leverage tools such as predictive analytics and artificial intelligence tools to provide a robust and effective logistics support to facilitate decision making at the speed of relevance.

Future Work

In order to facilitate the modernization efforts outlined in this thesis, key enablers driven from logistics headquarters elements and the partnered stakeholders will need to be pursued in order to set conditions for success. Some of these key enablers are: Business process reform to drive data management architectures, pilot opportunities, accelerating the acquisitions process to provide capabilities at the speed of relevance, and exploiting partnerships with research institutions.

As operating concepts such as the MOC and EABO continue to mature, new business processes will need to be developed or modified to support them. These business process

reforms will need to be captured and drive the data management architectures of the future. By instantiating these process reforms, the logistics community will have better fidelity on who must execute a process, what tasks need to be completed and when they need to be completed. This business process management will provide the backbone for warfighter sustainment activities and the data management needed to support those activities.

As the Marine Corps continues to pursue its logistics modernization initiatives, capitalizing on pilot opportunities will continue drive innovative approaches. As a pilot opportunity, a container/ microservice based GCSS-MC with new partnered APIs to enable Joint interoperability should be developed. Core functionalities can be demonstrated in exercises such as Steel Knight, Bold Alligator, and Trident Juncture. Once initial testing is completed and performance parameters are met, a follow on proof of concepts could be initiated with the Marine Expeditionary Unit (MEU). The MEU would be an ideal platform to demonstrate the new supportability, deployability and interoperability of a container/ microservices based, API enabled GCSS-MC as the MEU is supported by sister services within the Combatant Commander's area of responsibility while deployed.

Though the proposed ADTE, SDK and APIs recommended in this thesis will assist in rapidly developing new capabilities at the speed of relevance, the acquisitions process is currently too slow to generate the material solutions needed when requirements surface. The current acquisitions apparatus is designed to emphasize competition and fairness among vendors but not flexibility and speed of delivery to the warfighter. As these new operating concepts mature and new material solutions are needed, the acquisitions community and lawmakers will need to look

for ways to streamline the convoluted laws, statues and policies currently in place that are hindering the ability to get the capabilities needed to the warfighter.

As the Marine Corps continues to innovate in the logistics arena to improve sustainment, the Marine Corps will need to partner with research institutions in order to leverage the latest innovations that can be used in the logistics functional area. Partnering with institutions offer access to a unique breath of knowledge and experiences along with highly skilled research expertise. Universities are incubators for knowledge, discoveries and innovation that the Marine Corps logistical community can leverage to bring business and warfighting value to the enterprise. In addition to reaching out the external research institutions, the logistical community should work to reinforce relationships with internal DOD research institutions such as the Naval Postgraduate School and The Krulak Center at Marine Corps University as platforms integrate emerging technologies with current and future operating concepts.

---

[1] U.S. Department of Defense, Summary of the 2018 National Defense Strategy of the United States of America: Sharpening the American Military's Competitive Edge. 5-7.
[2] Amos, James. *Expeditionary Force 21*. Washington D.C.: DON, 2014.; Neller, Robert B. *Marine Corps Operating Concept: How an Expeditionary Force Operates in the 21st Century*. Washington D.C.: DON, 2016.
[3] Corbertt, Arthur. Expeditionary Advance Base Operations (EABO) New Considerations for Force Development. Marine Corps Warfighting Lab, 2017. 15-23.
[4] U.S. Department of Defense. Joint Staff. Naval Logistics Integration Playbook. 3rd ed. 2012. A1-2.
[5] U.S. Department of Defense. Joint Staff. Joint Publication 4-0 Joint Logistics. 2013. III 6-II 10.
[6] U.S. Department of Defense. Joint Staff. Joint Publication 4-0 Joint Logistics. 2013. III 6-II 10.
[7] Burrill, Steve, Marvin Wallace, and Brian Carthon. *LOGCOM Application Initial Assessment Out Brief*. Report. Managed Services Organization, Marine Corps Systems Command. MSO, 2017. 1-2.
[8] Managed Services Organization, Marine Corps Systems Command. *DITPR-DON Infrastructure Data Report*. March 20, 2019.
[9] Smith, James E., and Ravi Nair. "High-Level Language Virtual Machine Architecture." Virtual Machines, 2005, 221-79. doi:10.1016/b978-155860910-5/50006-9.
[10] Li, Xiao-Feng. "Basics of Virtual Machines." *Advanced Design and Implementation of Virtual Machines*, 2016, 7-18. doi:10.1201/9781315386706-2.
[11] Reuben, Jenni Susan. "A Survey on Virtual Machine Security." Seminar on Network Security, 11 Oct. 2007, pp. 3–5., pdfs.semanticscholar.org/c17c/a55a1b269dc162f8e14ce3980c0e8f9b15e8.pdf.
[12] Reuben, Jenni Susan. "A Survey on Virtual Machine Security." Seminar on Network Security, 11 Oct. 2007, pp. 3–5., pdfs.semanticscholar.org/c17c/a55a1b269dc162f8e14ce3980c0e8f9b15e8.pdf.
[13] Siriwardena, Prabath. *Advanced API Security - Securing APIs with Oauth 2.0, Openid Connect, Jws. Apress,*

2014. 1-3
[14] RedHat. "What are API's?" Red Hat - We Make Open Source Technologies for the Enterprise. May 2, 2018. https://www.redhat.com/en/topics/api/what-are-application-programming-interface.
[15] Shahriar, Hossain, and Mohammad Zulkernine. "MUSIC: Mutation-Based SQL Injection Vulnerability Checking." The Eighth International Conference on Quality Software, IEEE, 2008, pp. 77–80., doi:10.1109/QSIC.2008.33.
[16] Hu, Phili, Ronghai Yang, Yue Li, and Wing Lau. "Application Impersonation: Problems of OAuth and API Design." *Proceedings of the Second ACM Conference on Online Networks*, 2014, 271-78.
[17] Siriwardena, Prabath. *Advanced API Security - Securing APIs with Oauth 2.0, Openid Connect, JWS*. Apress, 2014. 50-52
[18] Maniot, Dmitry, and Manfred Sneps-Sneppe. "On Micro-Service Architecture." International Journal of Open Information Technologies 2, no. 9 (2014): 24. http://injoit.org/index.php/j1/article/view/139/104.
[19] Maniot, Dmitry, and Manfred Sneps-Sneppe. "On Micro-Service Architecture." *International Journal of Open Information Technologies* 2, no. 9 (2014): 24-25. http://injoit.org/index.php/j1/article/view/139/104.
[20] Yarygina, Tetiana. Exploring Microservice Security. Master's thesis, University of Bergen Norway, 2018. Bergen, 2018. 59-71. DOI 10.1109/SOSE.2018.00011
[21] Yarygina, Tetiana. Exploring Microservice Security. Master's thesis, University of Bergen Norway, 2018. Bergen, 2018. 59-71. DOI 10.1109/SOSE.2018.00011
[22] Pahl, Claus. "*Containerization and the PaaS Cloud*." IEEE Cloud Computing 2, no. 3 (2015): 1. doi:10.1109/mcc.2015.51.
[23] Pahl, Claus. "*Containerization and the PaaS Cloud*." IEEE Cloud Computing 2, no. 3 2015: 1. doi:10.1109/mcc.2015.51.
[24] Pahl, Claus. "*Containerization and the PaaS Cloud*." IEEE Cloud Computing 2, no. 3 2015: 2-3. doi:10.1109/mcc.2015.51.
[25] T.Combe, A. Martin, and R. Di Pietro. "*To Docker or not to Docker: A Security Perspective*". IEEE Cloud Computing 3. 2016. 54-62
[26] T.Combe, A. Martin, and R. Di Pietro. "*To Docker or not to Docker: A Security Perspective*". IEEE Cloud Computing 3. 2016. 54-62
[27] T.Combe, A. Martin, and R. Di Pietro. "*To Docker or not to Docker: A Security Perspective*". IEEE Cloud Computing 3. 2016. 54-62
[28] Amazon. "Amazon Fulfillment Technologies Aurora Case Study – Amazon Web Services (AWS)." Amazon. December 17, 2018. https://aws.amazon.com/solutions/case-studies/amazon-fulfillment-aurora/.
[29] Microsoft. "Maersk Uses Cloud to Spur Development of Containerized Solutions Built on Kubernetes." Microsoft Case Studies. September 19, 2018. https://customers.microsoft.com/en-us/story/maersk-travel-transportation-azure.
[30] Red Hat. Red Hat Customer Case Study. 2018. UPS Streamlines Package Tracking and Delivery with Devops and Red Hat https://www.redhat.com/cms/managed-files/rh-ups-customer-case-study-f10927kc-201902-en.pdf
[31] U.S. Marine Corps. Dept of Navy. *Marine Corps Information Enterprise Strategy*. By Kevin Nally. 2010. 10-29.
[32] U.S. Marine Corps. Marine Corps Systems Command. *MCEITS Data Center Concept of Operations*. McLean, VA: SAIC, 2016. 6-7.
[33] Managed Service Organization, Marine Corps Systems Command. *MCEITS Migration Timeline*. April 2015.
[34] Gartner. "SDK (software Development Kit)." Gartner IT Glossary. January 16, 2013. https://www.gartner.com/it-glossary/sdk-software-development-kit.
[35] U.S. Department of Commerce. National Institute of Standardization and Technology. *Risk Management Framework for Information Systems and Organizations*. By NIST. 2018. 2-3. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-37r2.pdf
[36] Columbus, Louis. "10 Charts That Will Change Your Perspective of Big Data's Growth." Forbes. May 23, 2018. https://www.forbes.com/sites/louiscolumbus/2018/05/23/10-charts-that-will-change-your-perspective-of-big-datas-growth/#645b0e482926.
[37] U.S. Marine Corps. HQMC. *Marine Corps Data Management Strategy*. By Commandant of the Marine Corps. Washington D.C.: DON, 2009. 1-5.
[38] U.S. Marine Corps. HQMC. *Marine Corps Data Management Strategy*. By Commandant of the Marine Corps. Washington D.C.: DON, 2009. Enclosure 2-1
[39] U.S. Marine Corps. HQMC. *Marine Corps Data Management Strategy*. By Commandant of the Marine Corps. Washington D.C.: DON, 2009. Enclosure 2-1
[40] Jarke, Matthias. *Fundamentals of Data Warehouses*. 1st ed. Berlin: Springer, 2010. 1-2
[41] Jarke, Matthias. *Fundamentals of Data Warehouses*. 1st ed. Berlin: Springer, 2010. 3-5

[42] Chan, Lois M., and Marcia L. Zeng. "Metadata Interoperability and Standardization – A Study of Methodology Part I Achieving Interoperability at the Schema Level." *D-Lib Magazine*, June 2006. http://dlib.org/dlib/june06/chan/06chan.html

[43] Chan, Lois M., and Marcia L. Zeng. "Metadata Interoperability and Standardization – A Study of Methodology Part I Achieving Interoperability at the Schema Level." *D-Lib Magazine*, June 2006. http://dlib.org/dlib/june06/chan/06chan.html

[44] Cabibbo L., Torlone R. (1998) *Querying multidimensional databases*. In: Cluet S., Hull R. (eds) Database Programming Languages. DBPL 1997. Lecture Notes in Computer Science, vol 1369. Springer, Berlin, Heidelberg 2-3

[45] Vassiliiadis, Panos. "Modeling Multidimensional Databases, Cube and Cube Operations." Proceedings. July 3, 1998.https://s3.amazonaws.com/academia.edu.documents/41405394/DML1_vassiliadis98modeling.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1553622964&Signature=RFpa8mA/gjSwwo873LlZE0apGw8=&response-content-disposition=inline; filename=Modeling_multidimensional_databases_cube.pdf.

[46] Oracle. *Understanding an OLAP Solution from Oracle. Case Studies*. Oracle, 2008. 14-15. http://www.oracle.com/us/solutions/business-intelligence/064300.pdf

Bibliography

Amazon. "Amazon Fulfillment Technologies Aurora Case Study – Amazon Web Services (AWS)." Amazon. December 17, 2018. https://aws.amazon.com/solutions/case-studies/amazon-fulfillment-aurora/.

Amos, James. *Expeditionary Force 21*. Washington D.C.: DON, 2014.

Burrill, Steve, Marvin Wallace, and Brian Carthon. *LOGCOM Application Initial Assessment Out Brief*. Report. Managed Services Organization, Marine Corps Systems Command. 2017. 1-2.

Cabibbo L., Torlone R. (1998) *Querying multidimensional databases*. In: Cluet S., Hull R. (eds) Database Programming Languages. DBPL 1997. Lecture Notes in Computer Science, vol 1369. Springer, Berlin, Heidelberg

Columbus, Louis. "10 Charts That Will Change Your Perspective of Big Data's Growth." Forbes. May 23, 2018. https://www.forbes.com/sites/louiscolumbus/2018/05/23/10-charts-that-will-change-your-perspective-of-big-datas-growth/#645b0e482926.

Corbett, Arthur. *Expeditionary Advance Base Operations (EABO) New Considerations for Force Development*. Marine Corps Warfighting Lab, 2017. 15-23

Department of Defense. *Summary of the 2018 National Defense Strategy of the United States of America: Sharpening the American Military's Competitive Edge*. 2018.

Dragoni, Nicola, Saverio Gillorenzo, and Alberto Lafuente. "*Microservices: Yesterday, Today and Tomorrow".* Springer, 2017, 1-17.

Gartner. "SDK (software Development Kit)." Gartner IT Glossary. January 16, 2013. https://www.gartner.com/it-glossary/sdk-software-development-kit.

Hu, Phili, Ronghai Yang, Yue Li, and Wing Lau. "Application Impersonation: Problems of OAuth and API Design." *Proceedings of the Second ACM Conference on Online Networks*, 2014, 271-78. doi:10.1145/2660463.

Jarke, Matthias. *Fundamentals of Data Warehouses*. 1st ed. Berlin: Springer, 2010.

Li, Xiao-Feng. "Basics of Virtual Machines." *Advanced Design and Implementation of Virtual Machines*, 2016, 7-18. doi:10.1201/9781315386706-2.

Managed Service Organization, Marine Corps Systems Command. *MCEITS Migration Timeline*. April 2015.

Managed Service Organization, Marine Corps Systems Command. *DITPR-DON Infrastructure Data Report*. March 20, 2019.

Maniot, Dmitry, and Manfred Sneps-Sneppe. "On Micro-Service Architecture." *International Journal of Open Information Technologies* 2, no. 9 (2014): 1-4. http://injoit.org/index.php/j1/article/view/139/104.

Microsoft. "Maersk Uses Cloud to Spur Development of Containerized Solutions Built on Kubernetes." Microsoft Case Studies. September 19, 2018. https://customers.microsoft.com/en-us/story/maersk-travel-transportation-azure.

Neller, Robert B. *Marine Corps Operating Concept: How an Expeditionary Force Operates in the 21st Century*. Washington D.C.: DON, 2016.

Oracle. *Understanding an OLAP Solution from Oracle*. Case Studies. Oracle, 2008. 14-15. http://www.oracle.com/us/solutions/business-intelligence/064300.pdf

Pahl, Claus. "Containerization and the PaaS Cloud." *IEEE Cloud Computing* 2, no. 3 (2015): 24-31. doi:10.1109/mcc.2015.51.

Red Hat. *Red Hat Customer Case Study*. 2018. UPS Streamlines Package Tracking and Delivery with Devops and Red Hat
https://www.redhat.com/cms/managed-files/rh-ups-customer-case-study-f10927kc-201902-en.pdf

RedHat. "What are API's?" Red Hat - We Make Open Source Technologies for the Enterprise. May 2, 2018. https://www.redhat.com/en/topics/api/what-are-application-programming-interface.

Reuben, Jenni Susan. "A Survey on Virtual Machine Security." Seminar on Network Security, 11 Oct. 2007, pp. 3–5., pdfs.semanticscholar.org/c17c/a55a1b269dc162f8e14ce3980c0e8f9b15e8.pdf.

Shahriar, Hossain, and Mohommad Zulkernine. "MUSIC: Mutation-based SQL Injection Vulnerability Checking." *The Eighth International Conference on Quality Software, IEEE,* 2008, 77-80. doi:10.1109/QSIC.2008.33.

Siriwardena, Prabath. *Advanced API Security - Securing APIss with Oauth 2.0, Openid Connect, JWS,*. Apress, 2014.

Smith, James E., and Ravi Nair. "*High-Level Language Virtual Machine Architecture*." *Virtual Machines*, 2005, 221-79. doi:10.1016/b978-155860910-5/50006-9.

T.Combe, A. Martin, and R. Di Pietro. *"To Docker or not to Docker: A Security Perspective*". IEEE Cloud Computing 3. 2016. 54-62

U.S. Department of Commerce. National Institute of Standardization and Technology. *Risk Management Framework for Information Systems and Organizations*. By NIST. 2018. 2-3. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-37r2.pdf

U.S. Marine Corps. HQMC. *Marine Corps Data Management Strategy*. By Commandant of the Marine Corps. Washington D.C.: DON, 2009. 1.

U.S. Marine Corps. HQMC. *Marine Corps Information Enterprise Strategy*. By Kevin Nally. 2010. 10-29.

U.S. Marine Corps. Marine Corps Systems Command. *MCEITS Data Center Concept of Operations*. McLean, VA: SAIC, 2016. 6-7.

Vassiliiadis, Panos. "*Modeling Multidimensional Databases, Cube and Cube Operations*." Proceedings. July 3, 1998.

https://s3.amazonaws.com/academia.edu.documents/41405394/DML1_vassiliadis98modeling.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1553622964&Signature=RFpa8mA/gjSwwo873LlZE0apGw8=&response-content-disposition=inline; filename=Modeling_multidimensional_databases_cube.pdf.

Yarygina, Tetiana. *Exploring Microserivce Security*. Master's thesis, University of Bergen Norway, 2018. Bergen, 2018. 59-71. DOI 10.1109/SOSE.2018.00011