OLD DOMINION UNIVERSITY

Hard Real-Time Computing Performance in a Cloud Environment

Alvin Murphy

BSEE (VT, '91), MSSE (GMU, '07)

Director:

Dr. James D. Moreland, Jr., Ph.D.

Members: Andres Sousa-Poza, Samuel Kovacic, Saikou Diallo

Defense of Dissertation Research

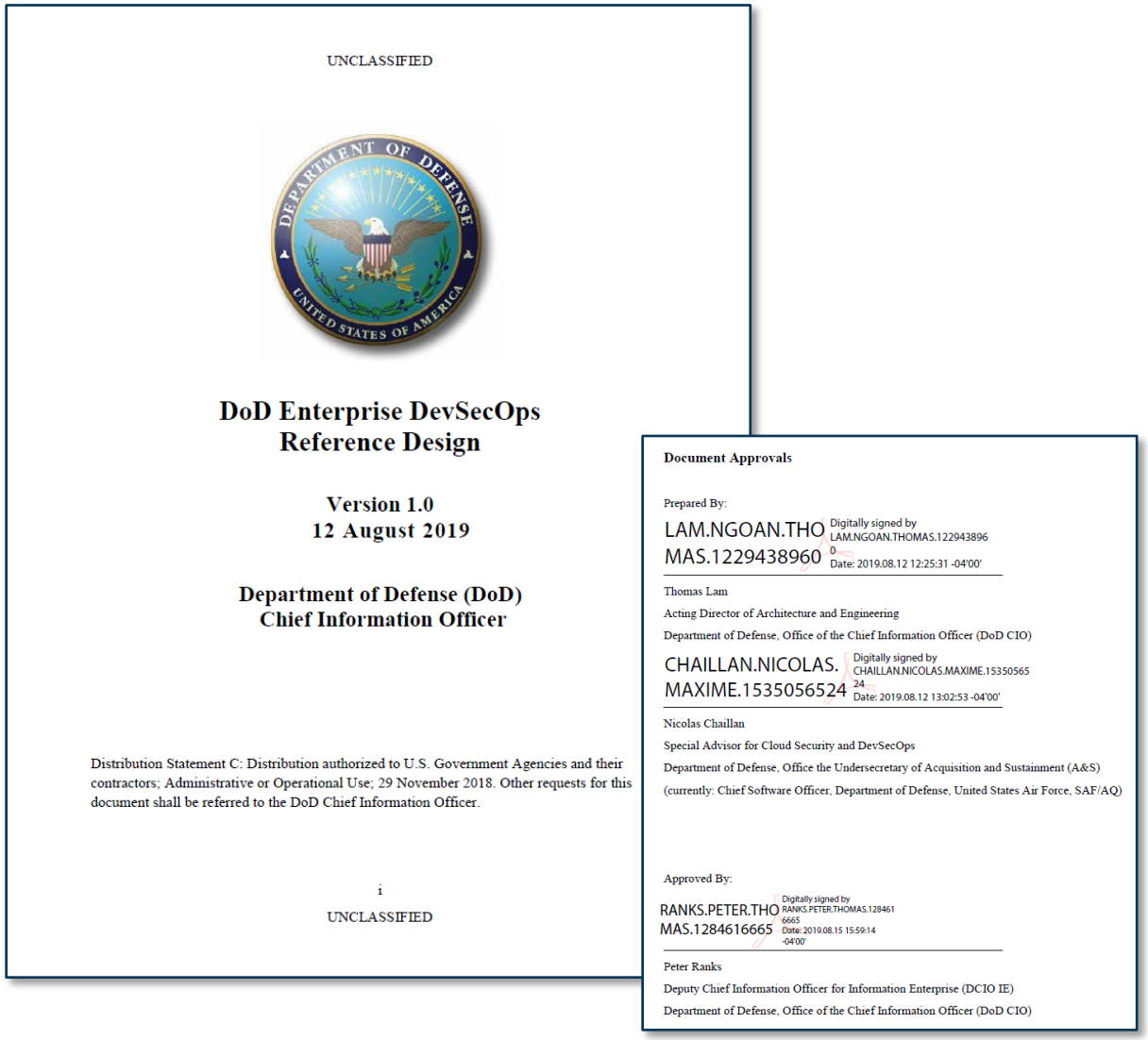
10 November 2022

Outline

- Introduction
  - Research Motivation
  - Research Process Overview
  - Problem Statement
  - Technology Overview (e.g. microservices, containers)
  - EMSE Focus Area
- Literature Review
- Hypothesis
- Mission Engineering
- Prototype
- Analysis
- Conclusions

Research Motivation

- DoD is mandating moving to a culture of digital engineering and DevSecOps
  - Analysis of the suitability of technologies is non-existent within the surface community
  - Guidance is needed regarding how to engineer, develop, and assess hard-real-time performance
  - Desire to deliver capabilities to warfighter “at the speed of relevance.”
- The US Surface Navy is developing an Integrated Combat System (ICS) that will require modern technologies to ensure success and ability to pace future threats
  - Future Surface Combat Force (FSCF) Initial Capabilities Document (ICD) signed by JROC in Sept 2018
  - ICS TLR signed by OPNAV N9 in May 2021
- Surface Navy Integrated Warfare System program office efforts ongoing to explore DevSecOps technologies and practices
  - Other Transaction Agreement (OTA) contract awarded to Solute in April 2019
  - Limited research to date in the suitability of microservice and container technologies



UNCLASSIFIED

DoD Enterprise DevSecOps Reference Design

Version 1.0
12 August 2019

Department of Defense (DoD)
Chief Information Officer

Distribution Statement C: Distribution authorized to U.S. Government Agencies and their contractors; Administrative or Operational Use; 29 November 2018. Other requests for this document shall be referred to the DoD Chief Information Officer.

i

UNCLASSIFIED

Document Approvals

Prepared By:

LAM.NGOAN.THOMAS.1229438960 — Digitally signed by LAM.NGOAN.THOMAS.1229438960 Date: 2019.08.12 12:25:31 -04'00'

Thomas Lam
Acting Director of Architecture and Engineering
Department of Defense, Office of the Chief Information Officer (DoD CIO)

CHAILLAN.NICOLAS.MAXIME.1535056524 — Digitally signed by CHAILLAN.NICOLAS.MAXIME.15350565 24 Date: 2019.08.12 13:02:53 -04'00'

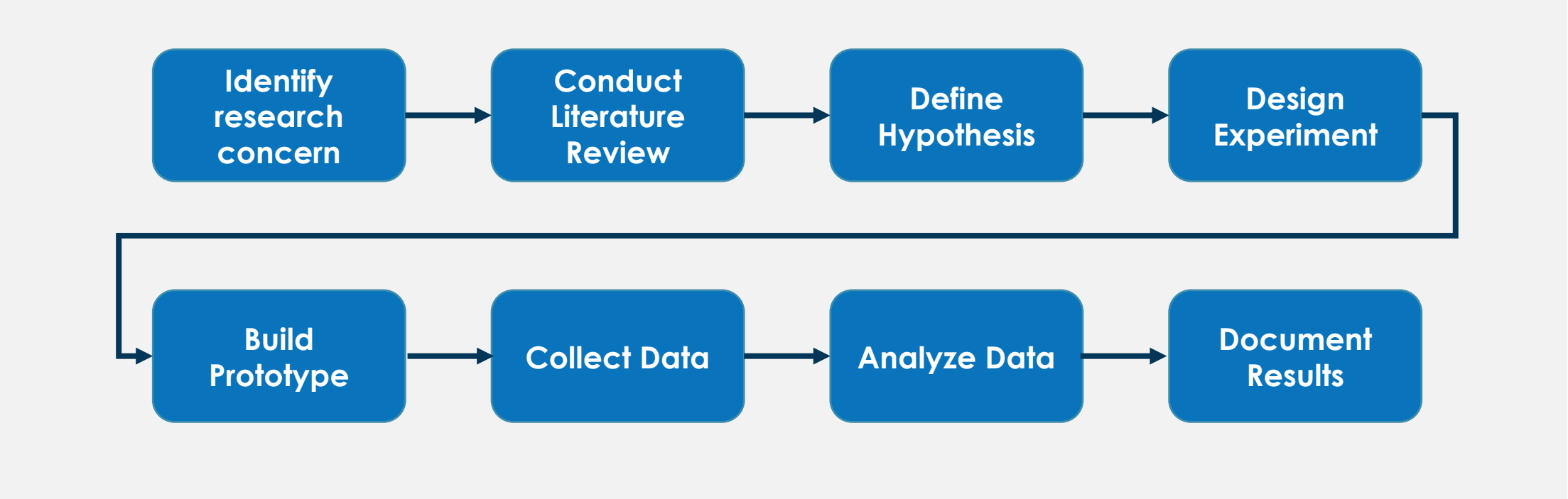
Nicolas Chaillan
Special Advisor for Cloud Security and DevSecOps
Department of Defense, Office the Undersecretary of Acquisition and Sustainment (A&S)
(currently: Chief Software Officer, Department of Defense, United States Air Force, SAF/AQ)

Approved By:

RANKS.PETER.THOMAS.1284616665 — Digitally signed by RANKS.PETER.THOMAS.1284616665 Date: 2019.08.15 15:59:14 -04'00'

Peter Ranks
Deputy Chief Information Officer for Information Enterprise (DCIO IE)
Department of Defense, Office of the Chief Information Officer (DoD CIO)

Research Process



Problem Statement

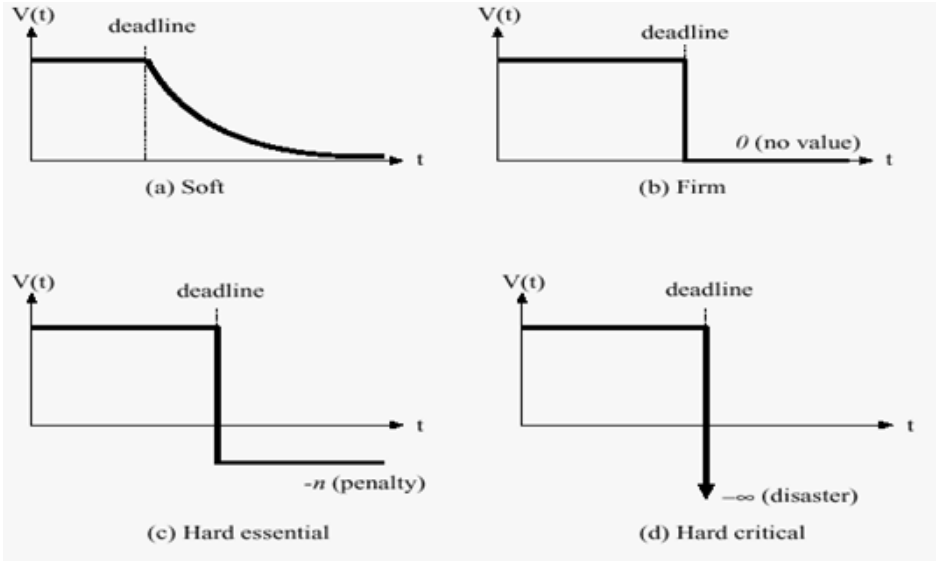
While commercial technologies and approaches provide an opportunity for rapid fielding of capabilities to pace threats, the suitability of commercial technologies to meet hard-real-time requirements within a surface combat system is unclear.

Gap: Industry microservice evidence exists (e.g. Amazon, Netflix, SpaceX, self-driving cars); however, limited public research information regarding hard-real-time in safety critical systems and how to assess.

Technology Focus: Microservices, containers, mission engineering

cBoK-1: Analytical quantitative hard-real-time microservices evidence
cBoK-2: Method for analysis (mission architecture, design patterns, statistical analysis)

What is hard-real-time?



Mission Objective: Useable information in time to support human and automated decisions (e.g. AI)

Questions

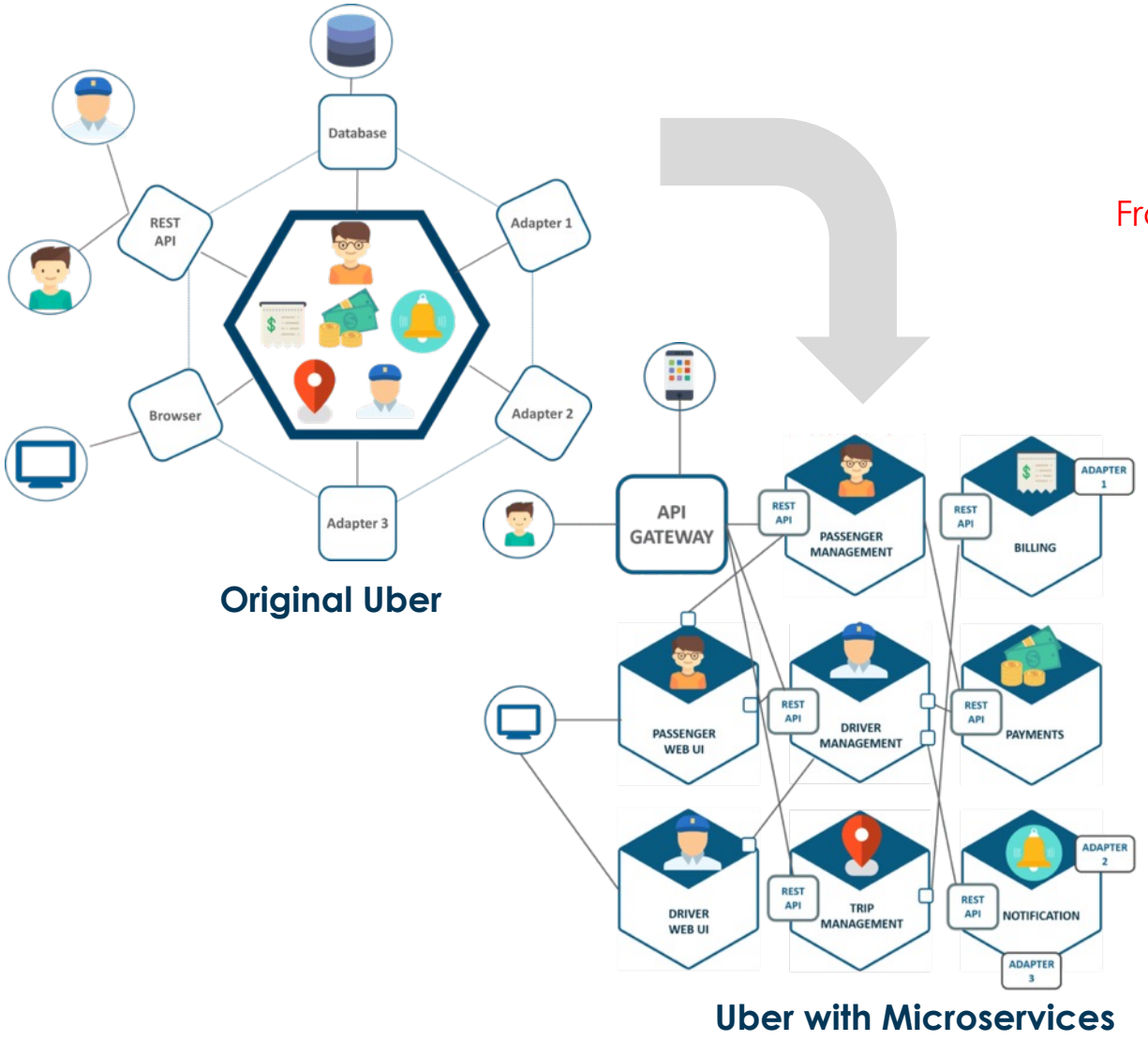
Q1: Is it possible to synthesize cloud computing and open-source technologies to realize a microservices architecture for a hard-real-time deterministic Combat Management System (CMS)?

Q2: Does end-to-end (E2E) mission thread analysis increase SoS interoperability and reduce integration issues?

Q3: What benefits are gained from a microservice based DevSecOps approach?

Q4: Is the resultant architecture hard-real-time deterministic?

Modern Technology Enabler – Microservices

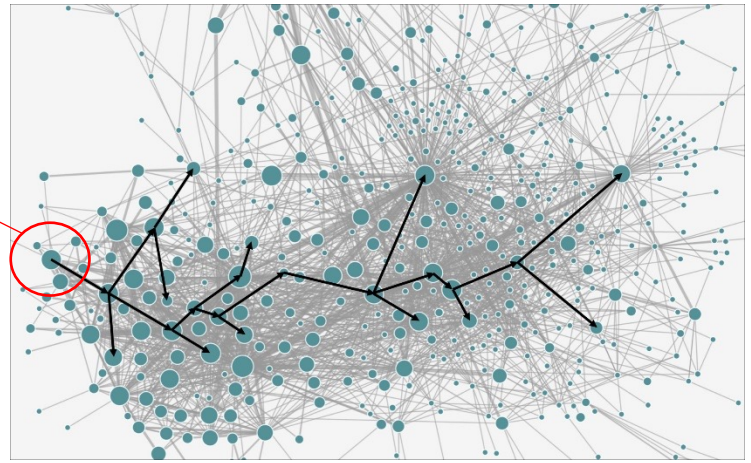


Original Uber

Uber with Microservices

Numerous BackEnds

FrontEnd



Visual representation of a subset of Uber's microservices and a hypothetical transaction (Shkuro, Mastering Distributed Tracing, 2019)

Modern Technology Enabler – VMs and Docker Containers

*700 MB * 3 = 2.1 GB*
+ CPU and Memory Resources

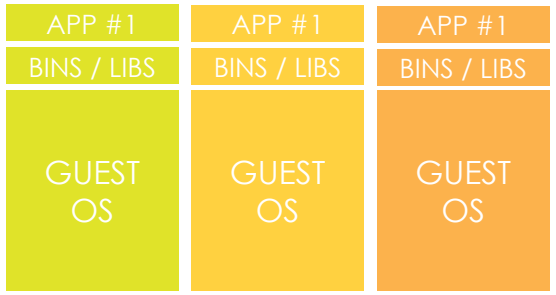
Type 1: Hyper-V (Win), HyperKit (OS X)
Type 2: VirtualBox / VMWare

Linux, Windows, OS X

Laptop, Desktop, Cloud Instance (e.g. EC2)

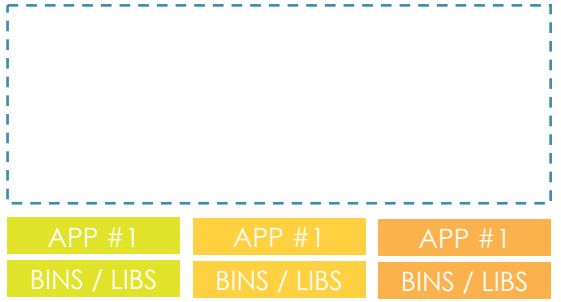
Reference: Virtual Machines vs. Docker Containers – Dive Into Docker on YouTube.com

Virtual Machines

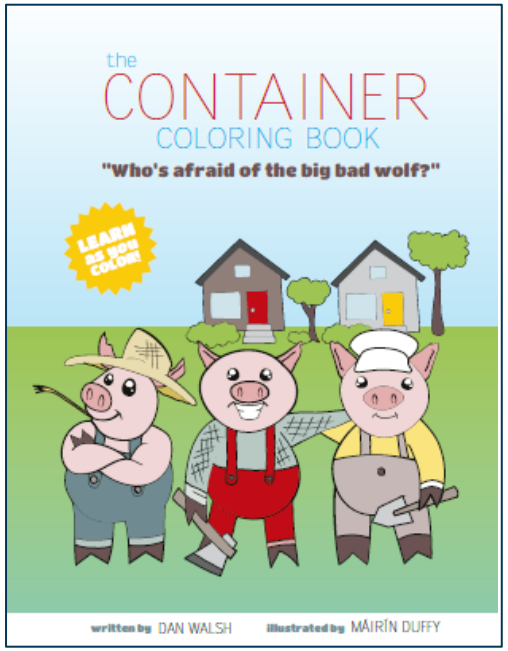


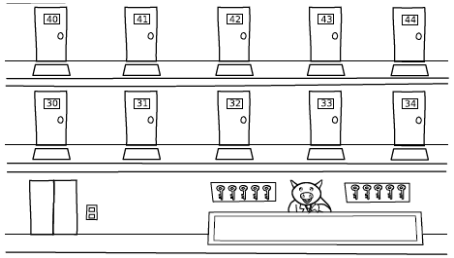
- Isolate Systems
- Starts in Minutes
- Wastes Resources

Docker Containers



- Isolate Applications
- Starts in Milliseconds
- Saves Resources





House (VM) vs. Apartment (Container) (e.g., Studio, Penthouse)

Modern Technology Enabler – Example Software Principles

- SOLID (Martin, 2017)
  - Single-responsibility principle: Every class (e.g., application) should have only one responsibility (SRP)
  - Open-closed principle: Software entities should be open for extension, but closed for modifications (OCP)

- 12-Factor App (Wiggins, 2017 and 12factor.net)
  - One codebase tracked in revision control, many deploys (1)
  - Store configuration in the environment (3)
  - Treat backend services as attached resources (4)
  - Export services via port bindings (7)
  - Treat logs as event streams (11)

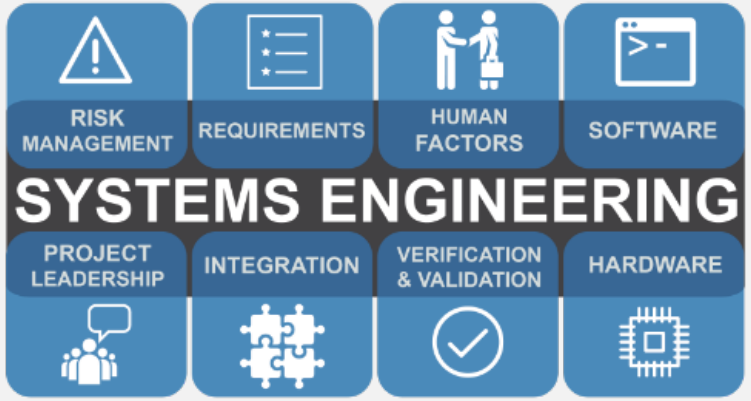
- Cloud Native (Gannon et al., 2017)
  - Built on the assumption that cloud infrastructure is fluid and failure is constant
  - Designed so that upgrade and test occurs seamlessly without disruption to production
  - Security is not an afterthought – security is part of the underlying application architecture

EMSE Research Focus

Mission Engineering
(Adapted from USD(R&E) Mission Engineering Guide (2020))

- Establish mission context; e.g., mission characterization
- Identify desired SoS mission outcomes; e.g., mission metrics
- Define mission threads; e.g., "kill-chains"
- Define SoS mission allocations
- Assess SoS health and identify mission gaps; e.g., mission analysis
- Assess new technologies that may add mission advantage; e.g., **DevSecOps technologies**

Systems Engineering
(Adapted from INCOSE and IEEE Guidance (2015))



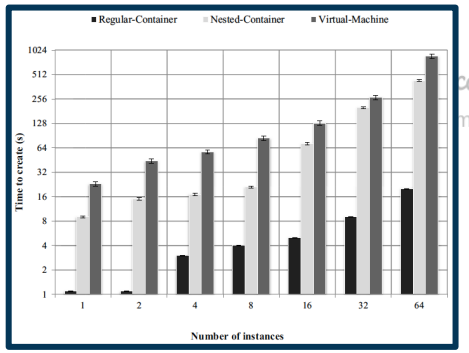
- Establish system context within SoS
- Define requirements
- Human factors analysis
- Allocate requirements to software and hardware solutions; e.g., subsystem architecture definition
- System analysis (e.g. **models, prototypes**)
- System integration
- Verification and validation (V&V)
- Risk management
- Project leadership

EMSE Focus Area

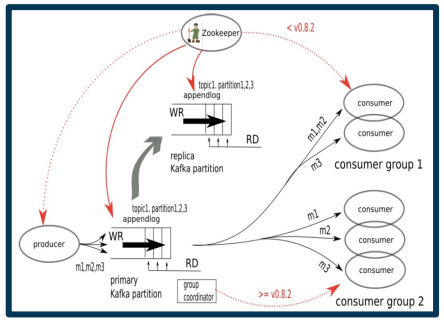
Computer Science
(Adapted from various university catalogs and syllabi)

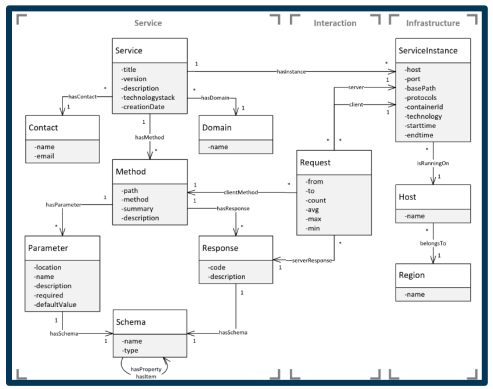
- Software requirements analysis
- Detailed software architecture and software design
- Application of design patterns
- Technology selection; e.g., operating system, coding languages, database
- Software integration
- Software testing and evaluation

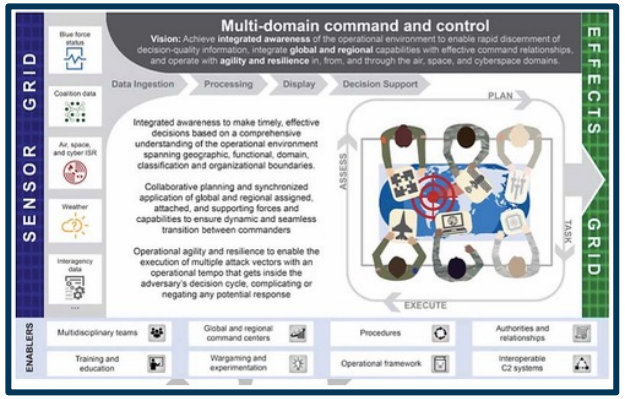
Review of Prior Research



- Literature Review
  - Microservice Performance (Q1)
    - VMs vs. Containers (Felter et al., 2014)
    - ★ Container Deployment; e.g. Pods (Amaral et al., 2015)
    - ★ Container Launch Time (Wei et al., 2018)
  - Architecture (Q4)
    - Kafka vs. AMPQ Msg Broker (John & Liu, 2017)
    - Kafka vs. RabbitMQ pub/sub (Dobbelaere & Esmaili, 2017)
    - Kafka Reliability (Wu, Shang, & Wolter, 2019)
  - E2E Analysis (Q2)
    - ★ Architecture Extraction (Mayer & Weinreich, 2018)
    - DockerSim (Nikdel, Gao, & Neville, 2017)
    - ContainerCloudSim (Piraghaj et al., 2017)
    - Microservice Testing Tools (Sotomayor et al., 2019)
  - C2 Implementations (Q3)
    - ATHENA Kubernetes Auto-Scaling (Kho Lin, et al., 2018)
    - MDC2 Overview (Bruza, M., & Reith, M. 2018) ★
    - AOC Pathfinder (Thesis) (Bruza, M., 2018)
    - CERN SCADA w/ Kafka Messaging (Ledeul et al., 2019) ★
    - Technology Survey (Bogner et al., 2019) ★







★ Discussed in candidacy exam

Gaps in Previous Research

Question	Limitations
Q1: Microservices Performance	Lacks node architecture that is representative of CMS components. Variance in container start up times is tolerated without assessment of design alternatives.
Q2: E2E Analysis	Environment used to generate architecture products and conduct analysis is not representative of CMS architecture and challenges of CMS interface management (e.g. APIs).
Q3: C2 Implementation	MDC2 example is similar to CMS challenges; however, the results are qualitative not quantitative. Focus is on change in DevOps culture.
Q4: Architecture	Provides detailed assessment of Kafka vs. RabbitMQ; however, a mix may be required within a CMS to meet unique CMS domain requirements (e.g. external interface to weapons/sensors vs internal processing). Analysis is required to facilitate experiment design.

Quantitative Hypothesis

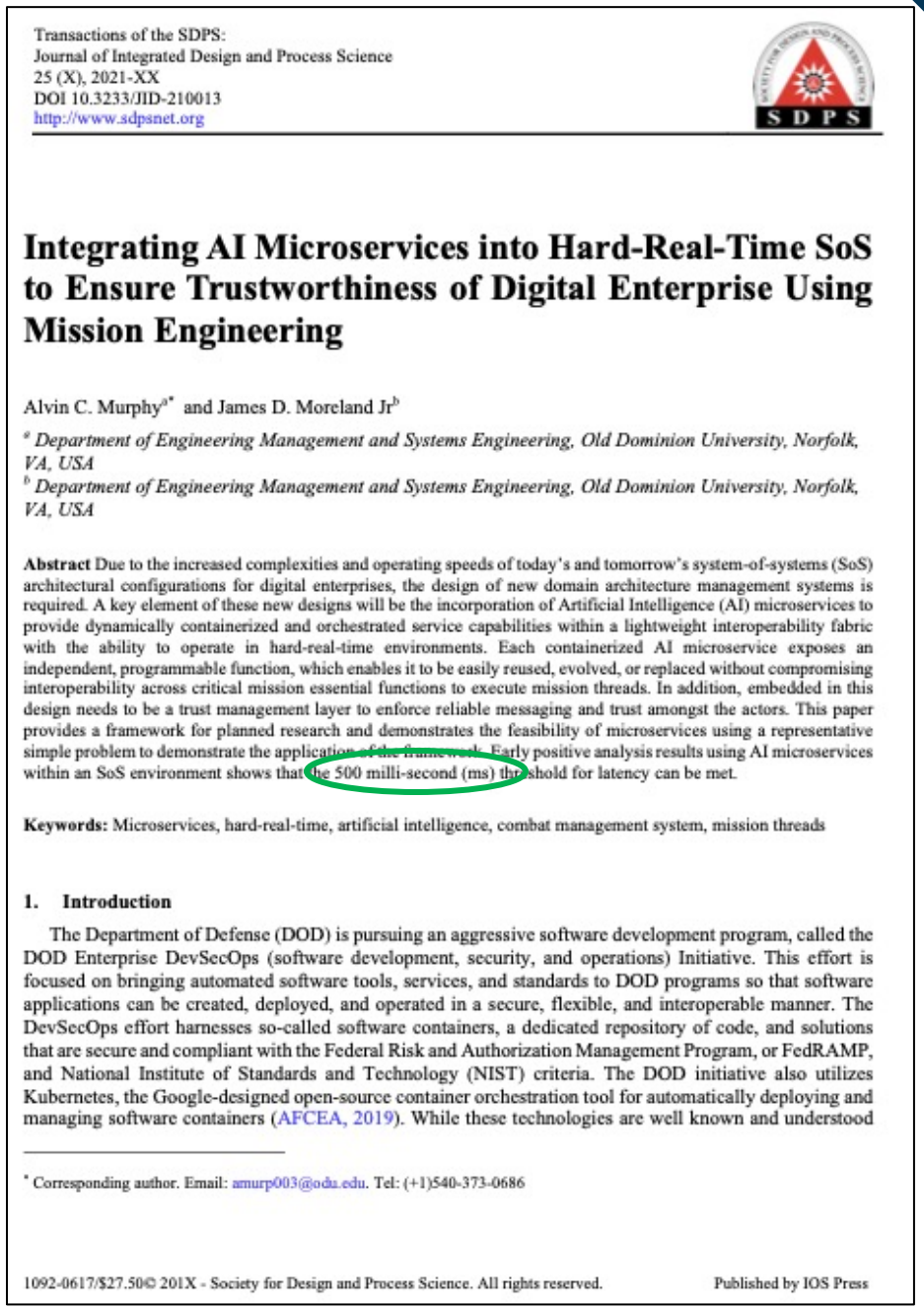
Modern DevSecOps architectures can be designed to meet hard-real-time latency (μ) requirements using modern computing environments and computing infrastructure:

H0: μ ≤ tbd ms with jitter within latency bounds

Ha: μ > tbd ms with jitter exceeding latency bounds

$\propto = 0.05$

500 ms analysis target defined in "peer-reviewed" JIDPS article (published Fall 2021)



Transactions of the SDPS:
Journal of Integrated Design and Process Science
25 (X), 2021-XX
DOI 10.3233/JID-210013
http://www.sdpsnet.org

Integrating AI Microservices into Hard-Real-Time SoS to Ensure Trustworthiness of Digital Enterprise Using Mission Engineering

Alvin C. Murphy[a*] and James D. Moreland Jr[b]

[a] *Department of Engineering Management and Systems Engineering, Old Dominion University, Norfolk, VA, USA*
[b] *Department of Engineering Management and Systems Engineering, Old Dominion University, Norfolk, VA, USA*

Abstract Due to the increased complexities and operating speeds of today's and tomorrow's system-of-systems (SoS) architectural configurations for digital enterprises, the design of new domain architecture management systems is required. A key element of these new designs will be the incorporation of Artificial Intelligence (AI) microservices to provide dynamically containerized and orchestrated service capabilities within a lightweight interoperability fabric with the ability to operate in hard-real-time environments. Each containerized AI microservice exposes an independent, programmable function, which enables it to be easily reused, evolved, or replaced without compromising interoperability across critical mission essential functions to execute mission threads. In addition, embedded in this design needs to be a trust management layer to enforce reliable messaging and trust amongst the actors. This paper provides a framework for planned research and demonstrates the feasibility of microservices using a representative simple problem to demonstrate the application of the framework. Early positive analysis results using AI microservices within an SoS environment shows that the 500 milli-second (ms) threshold for latency can be met.

Keywords: Microservices, hard-real-time, artificial intelligence, combat management system, mission threads

1. Introduction

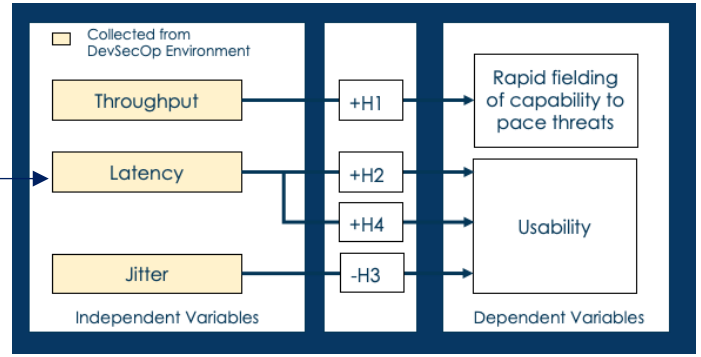
The Department of Defense (DOD) is pursuing an aggressive software development program, called the DOD Enterprise DevSecOps (software development, security, and operations) Initiative. This effort is focused on bringing automated software tools, services, and standards to DOD programs so that software applications can be created, deployed, and operated in a secure, flexible, and interoperable manner. The DevSecOps effort harnesses so-called software containers, a dedicated repository of code, and solutions that are secure and compliant with the Federal Risk and Authorization Management Program, or FedRAMP, and National Institute of Standards and Technology (NIST) criteria. The DOD initiative also utilizes Kubernetes, the Google-designed open-source container orchestration tool for automatically deploying and managing software containers (AFCEA, 2019). While these technologies are well known and understood

* Corresponding author. Email: amurp003@odu.edu. Tel: (+1)540-373-0686

 Published by IOS Press

Additional Predictions and Hypothesis Statements

Research Quantitative Analysis Focus



Research Framework

coggle
made for free at coggle.it

H1: The scaleability of microservices as new data sources are added to an architecture enables maintaining high **throughput** and predicted to have a positive impact on **rapid fielding of capability**.

(I) Throughput

(D) Rapid Fielding

H2: Microservices orchestration through DevSecOps technologies enables maintaining low **latency** service call responses and predicted to have a positive impact on **usability**.

(I) Latency

(D) Usability

(I) Jitter

(D) Usability

H3: Web based user interfaces will increase display **jitter** and predicted to have a negative impact on **usability** and the deterministic performance needed for positive control of ownship weapons.

(I) Latency

(D) Usability

H4: A system architecture model can predict end-to-end **latency** within a mission kill chain to quantify SoS **usability**.

System

ME System-of-Systems

Problem Statement:
"While commercial technologies and approaches provide an opportunity for rapid fielding of capabilities to pace threats, the suitability of commercial technologies to meet hard-real-time requirements within a surface combat system is unclear."

1. Problem Statement

2. Predictions

3. Hypotheses

Dissertation Research

Q1: Is it possible to synthesize cloud computing and open source technologies to realize a microservices architecture for a hard-real-time deterministic Combat Management System (CMS)?

Q2: Does end-to-end (E2E) mission thread analysis increase SoS interoperability and reduce integration issues?

Q3: What benefits are gained from a microservice based DevSecOps approach?

Q4: Is the resultant architecture hard real-time deterministic?

P1: A microservices based architecture is predicted to have a positive impact on time to implement capability upgrades and development costs.

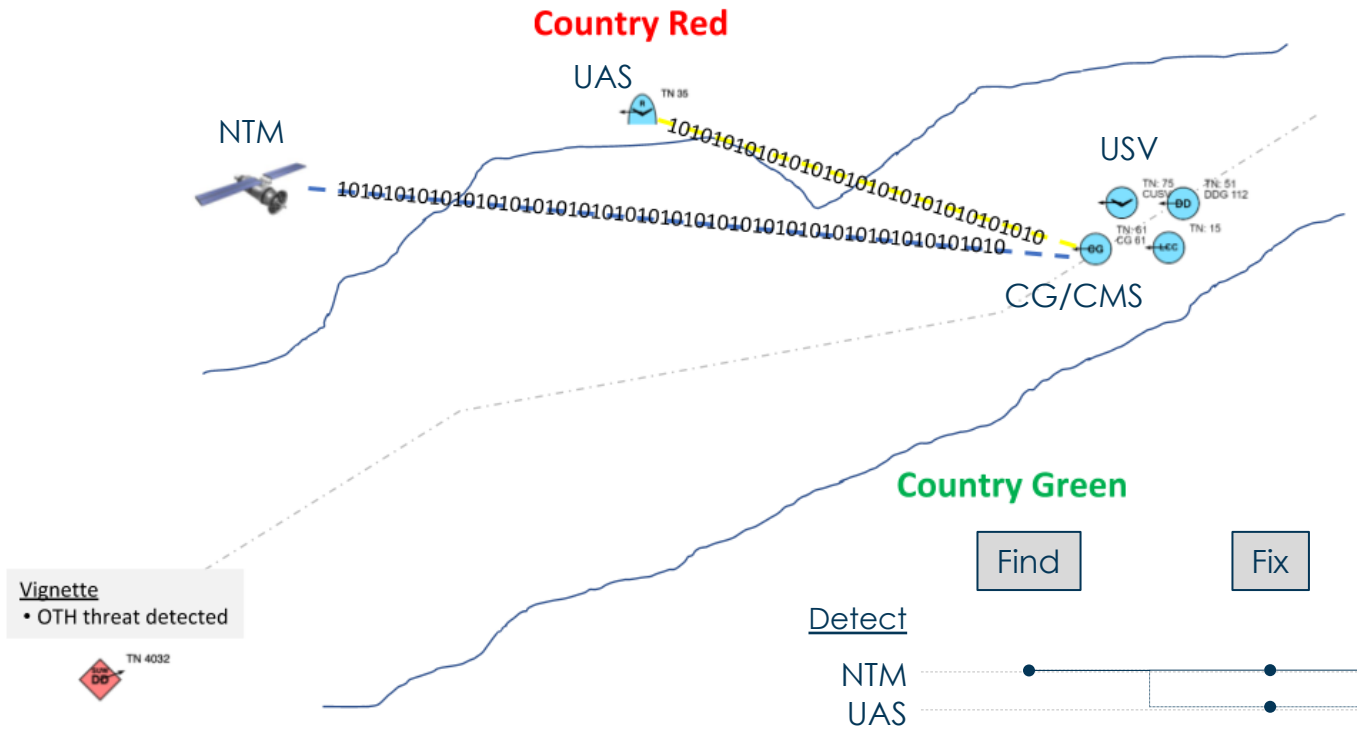
P2: DevSecOps container orchestration technologies are predicted to have a positive impact on CS availability and latency due to an ability to tune deployment configurations and load balance.

P3: Web based user interface technologies are predicted to have a negative impact on deterministic real-time performance needed for positive control of ownship weapons.

P4: A systems architecture model is predicted to have a positive impact on microservices architecture performance prediction and prediction of associated mission impacts.

(I)= Independent, (D) = Dependent variable

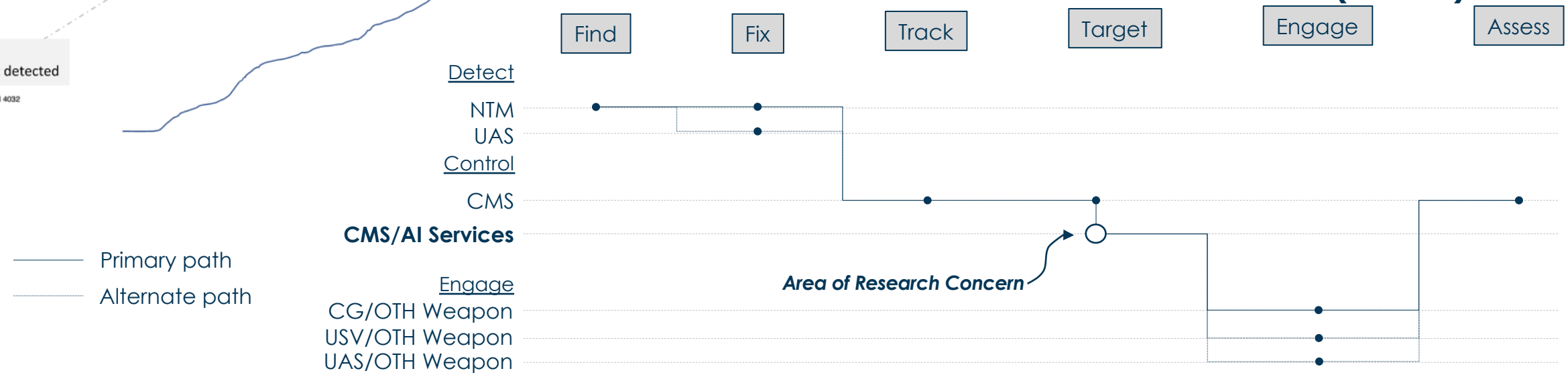
Mission Scenario (OV-1) and Kill Chain



Strait Transit OTH-SUW Kill Mission Scenario

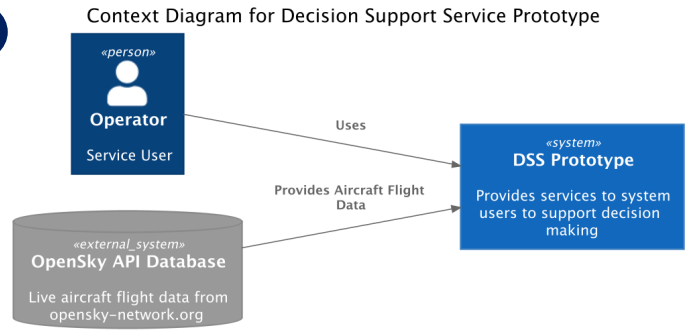
1. SAG enters Strait between Country Red and Country Green
2. NTM provides reports of a Country Red surface threat near the end of the Strait. A non-organic UAV is tasked to provide additional targeting information (e.g. Triton). Joint assets may also be tasked to provide support.
3. A targeting solution is developed by the CMS using weapon coordination and AI.
4. OTH weapons are employed.

Strait Transit OTH-SUW Kill Chain (F2T2EA)

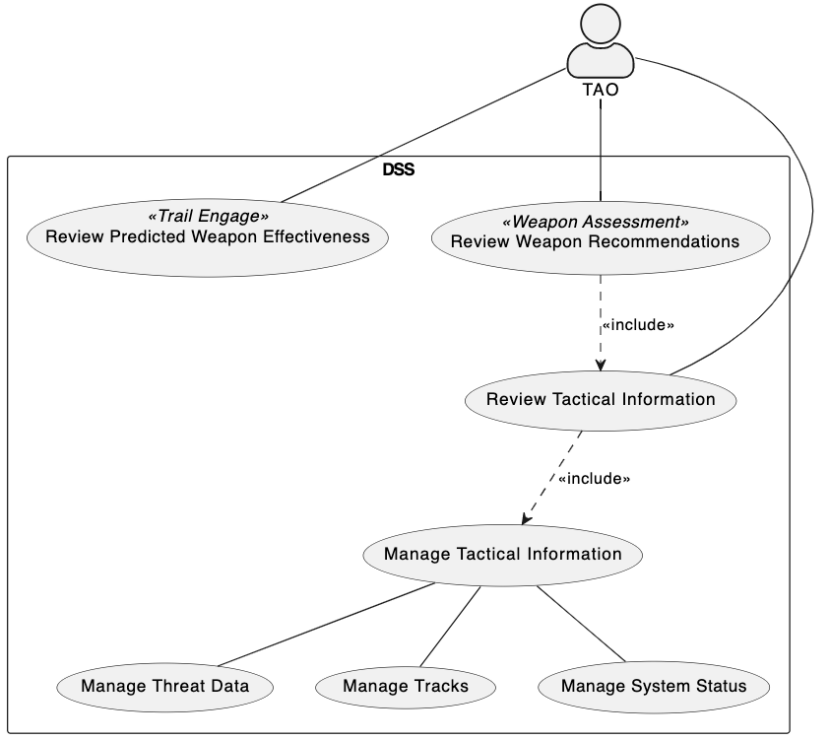


DSS Context/Use Cases

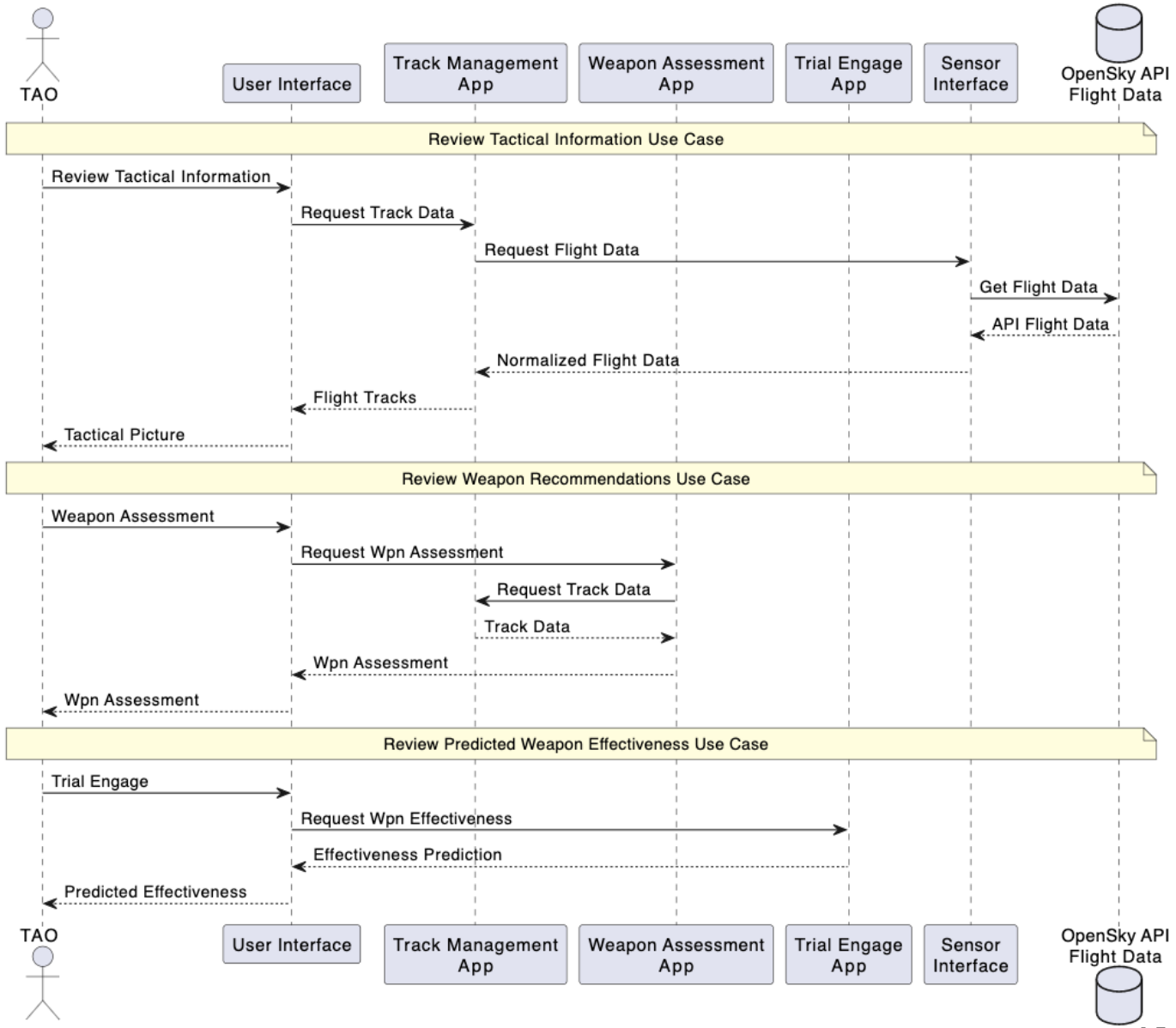
1



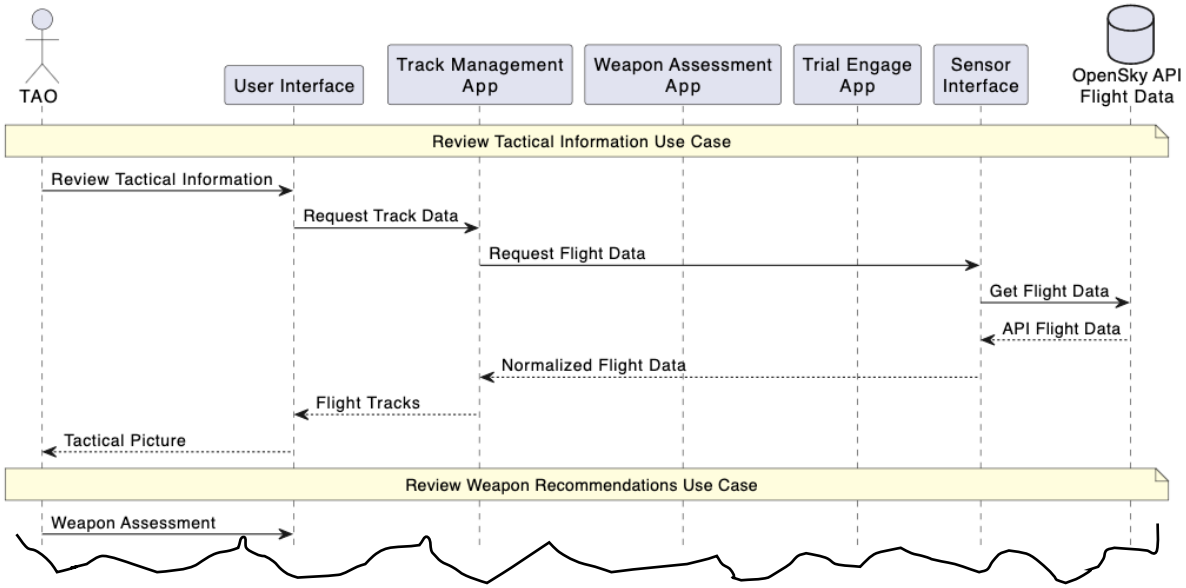
2



3



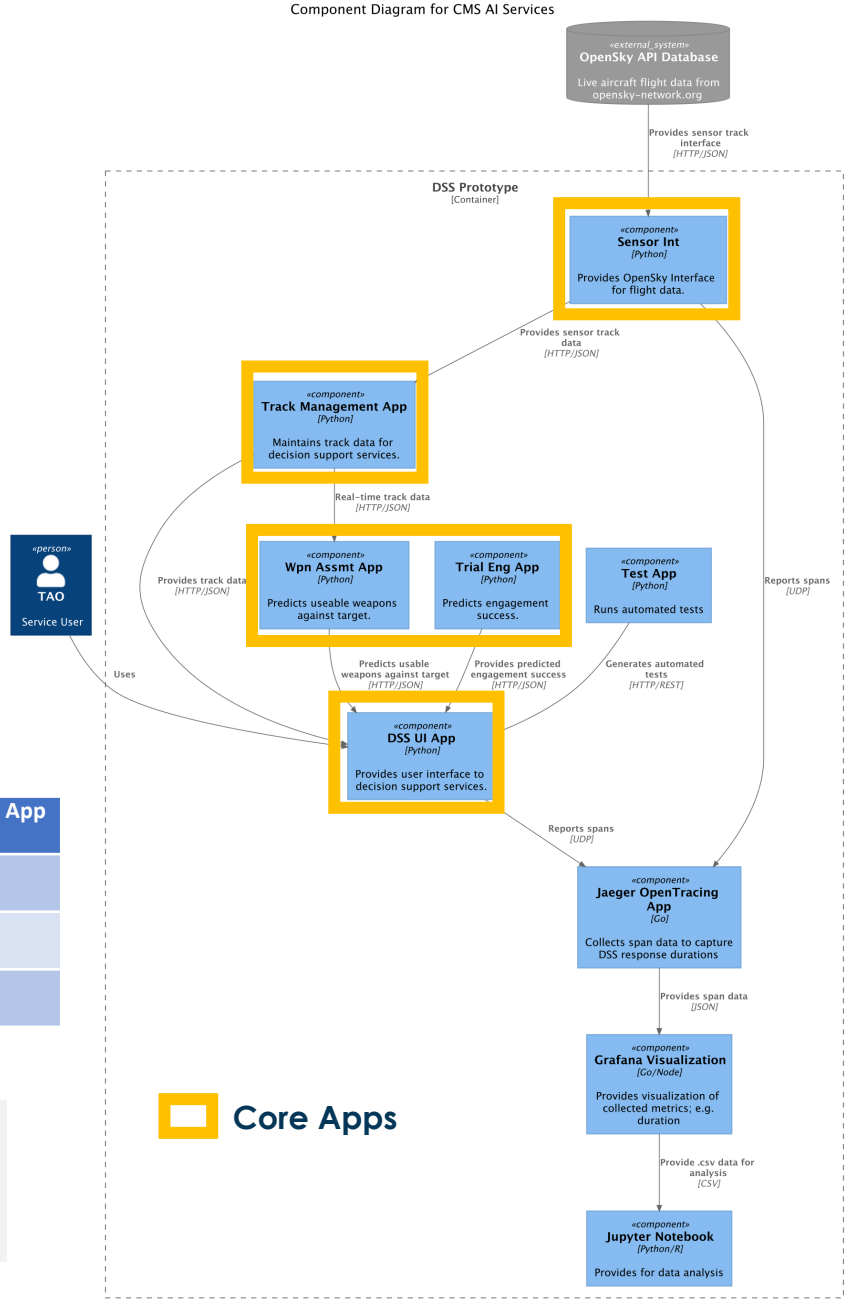
DSS Prototype Component Diagram



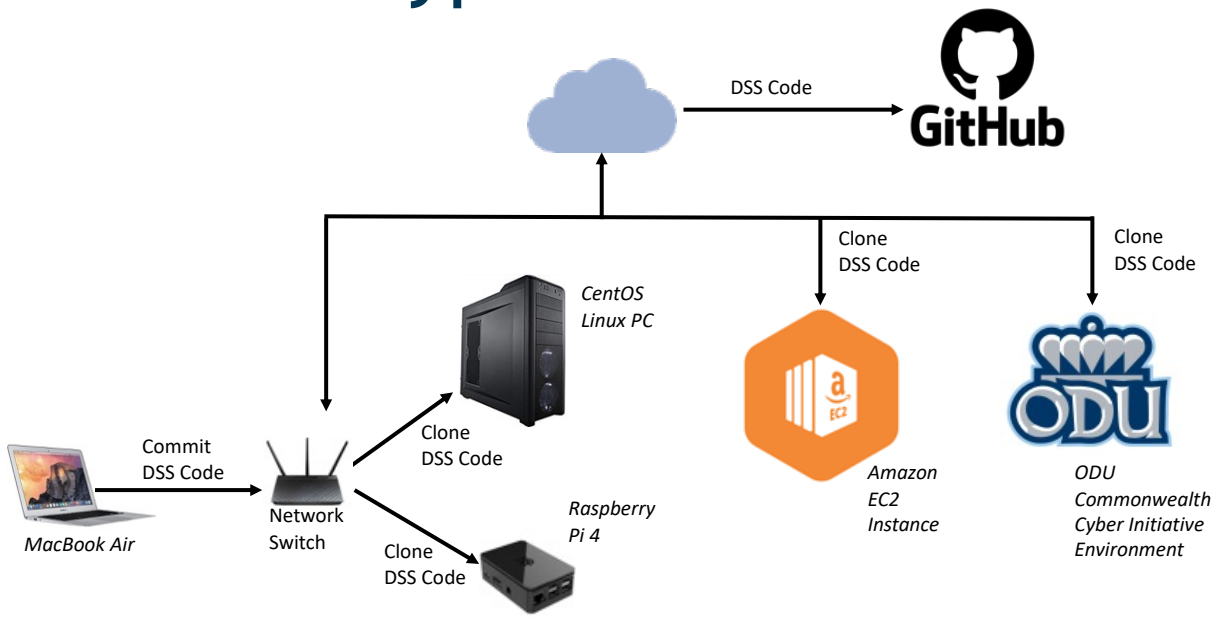
Mission Use Case	User Interface App	Track Management App	Weapon Assessment App	Trial Engagement App	Sensor Interface App
Review Tactical Information	X	X			X
Review Weapon Recommendations	X	X	X		
Review Predicted Weapon Effectiveness	X			X	

Container Based System-of-Systems

*H1: The scalability of microservices as new data sources are added to an architecture enables maintaining high **throughput** and predicted to have a positive impact on **rapid fielding of capability of capability**.*
*H2: Microservices orchestration through DevSecOps technologies enables maintaining low **latency** services call responses and predicted to have a positive impact on **usability**.*



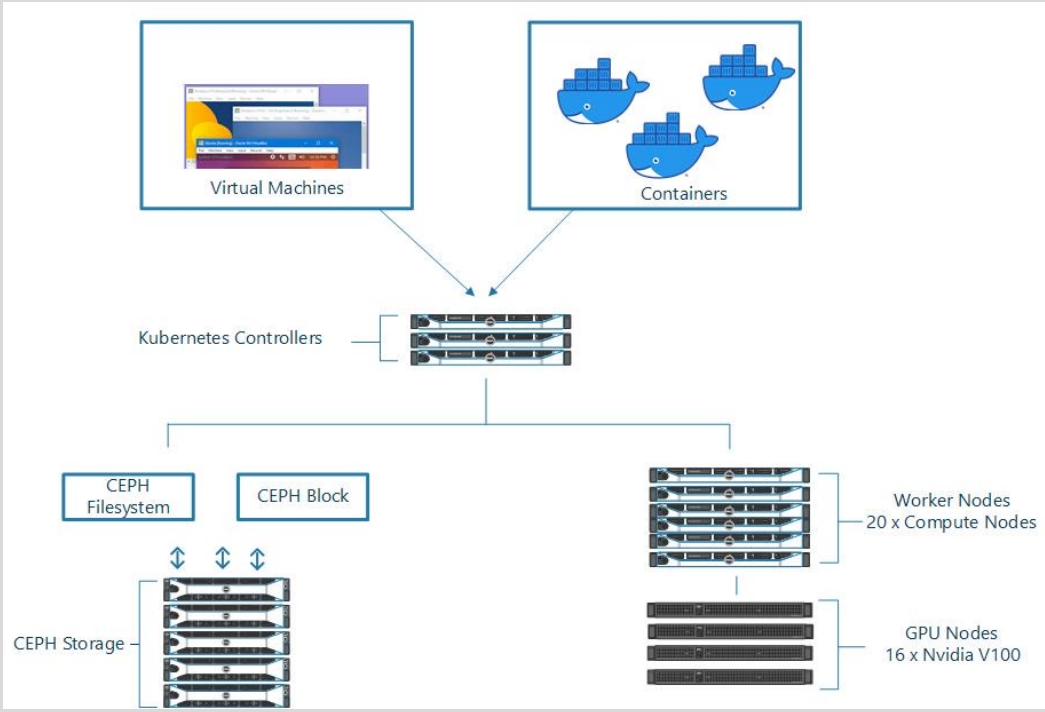
DSS Prototype Environment



Development	Integration Testing	Production

Env ID	Platform	Chipset	Processor	Memory	OS
0	MacBook Air (2017)	Intel Core i5	Dual-Core Intel Core 5 @ 1.8 GHz	8 GB 1600 MHz DDR3	MacOS 12.4 (Monterey)
1	Linux PC (2012)	Intel Core i7	Intel(R) Core(TM) i7-3770K CPU @ 3.50GHz	16 GB 1600 MHz DDR3	CentOS Linux 8 (Core)
2	Raspberry Pi 4 (2020)	Broadcom BCM 2711	Quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5 GHz	4 GB LDDR4-3200 SDRAM	Debian GNU/Linux 11 (bullseye)
3	Amazon Elastic Compute Cloud (EC2): t2.micro	Intel Xeon	Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz	1 GB	Debian GNU/Linux 10 (buster)
4	ODU Commonwealth Cyber Initiative (CCI)	Intel Xeon	Intel(R) Xeon(R) CPU E5-2683 v4 @ 2.10GHz	128 GB	Red Hat Enterprise Linux 8.5 (Ootpa)

ODU CCI Research Environment



- Automated deployment of changes from ODU GitLab Repo via ArgoCD GitOps
- Kubernetes (RKE2) container orchestration
- **Great example of "DevOps" in practice**
- Briefed project to ODU CCI Steering Group on 27 June 2022

*H1: The scalability of microservices … predicted to have a positive impact on **rapid fielding of capability of capability**.*

Collected Data from Prototype

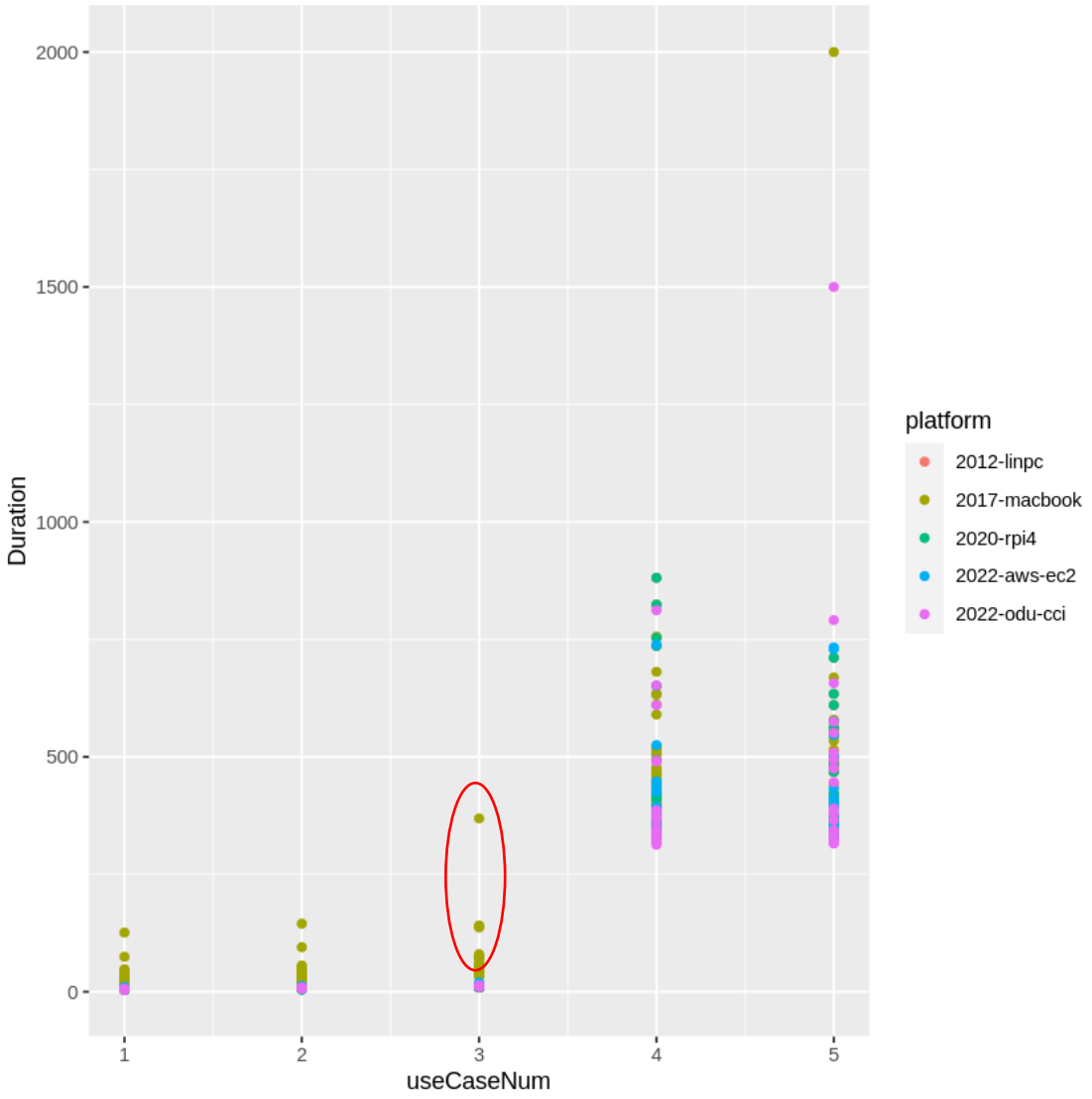
Data Sources

```
macData <- read.csv('DSS_SpanData-mac-2022-05-02 18_38_26_s10-5-1.csv', header = TRUE)
linpcData <- read.csv('DSS_SpanData-linuxpc-2022-06-06 17_38_29_s10-5-1.csv', header = TRUE)
rpi4Data <- read.csv('DSS_SpanData-rpi4-2022-06-06 17_52_59_s10-5-1.csv', header = TRUE)
awsEC2Data <- read.csv('DSS_SpanData-aws_ec2-2022-06-07 17_44_08_s10-5-1.csv', header = TRUE)
cci_Data <- read.csv('DSS_SpanData-odu_cci-2022-06-28 17_47_20_s10-5-1.csv', header = TRUE)
```

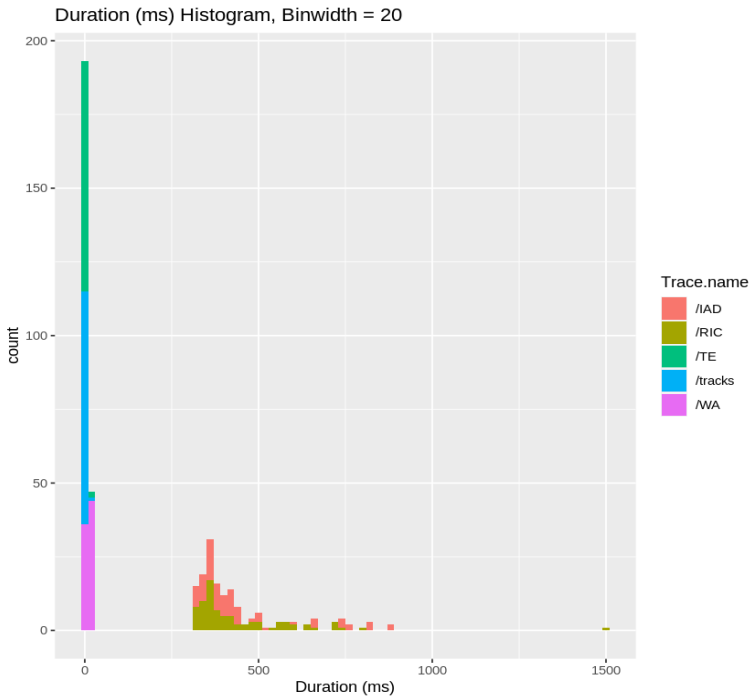
Data Attributes

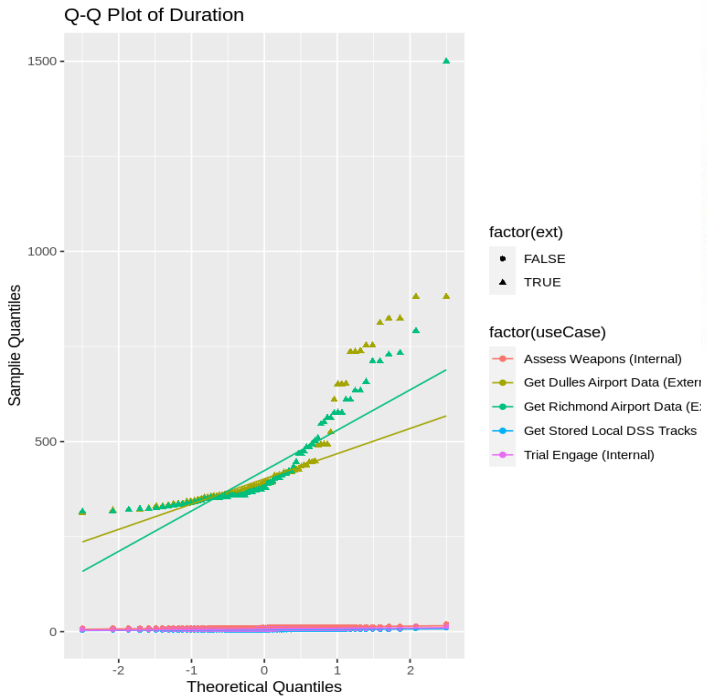
```
Rows: 500
Columns: 9
$ Trace.ID   <chr> "9ee3577fb1b427bc4fc17fecc5154d7d", "f05ddc4dc13aff5c309801…
$ Trace.name <chr> "/TE", "/tracks", "/IAD", "/RIC", "/WA", "/TE", "/tracks", …
$ Start.time <chr> "2022-05-02 10:25:01.366", "2022-05-02 10:25:00.309", "2022…
$ Duration   <dbl> 36.0, 43.3, 464.0, 494.0, 139.0, 30.3, 30.0, 478.0, 546.0, …
$ platform   <chr> "2017-macbook", "2017-macbook", "2017-macbook", "2017-macbo…
$ env        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,…
$ useCase    <chr> "Trial Engage (Internal)", "Get Stored Local DSS Tracks (In…
$ useCaseNum <dbl> 2, 1, 4, 5, 3, 2, 1, 4, 5, 3, 2, 1, 4, 5, 3, 2, 1, 4, 5, 3,…
$ ext        <lgl> FALSE, FALSE, TRUE, TRUE, FALSE, FALSE, FALSE, TRUE, TRUE, …
```

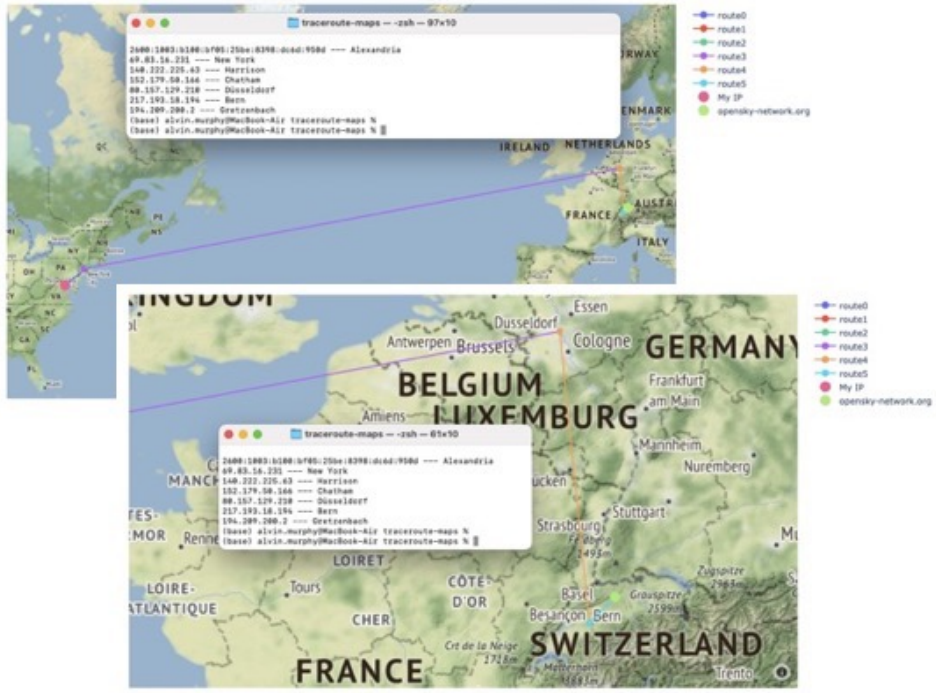
Removed macData in subsequent analysis; e.g. 400 data points (rows)



Latency Histogram and Q-Q Plot (i.e. Duration Summary)

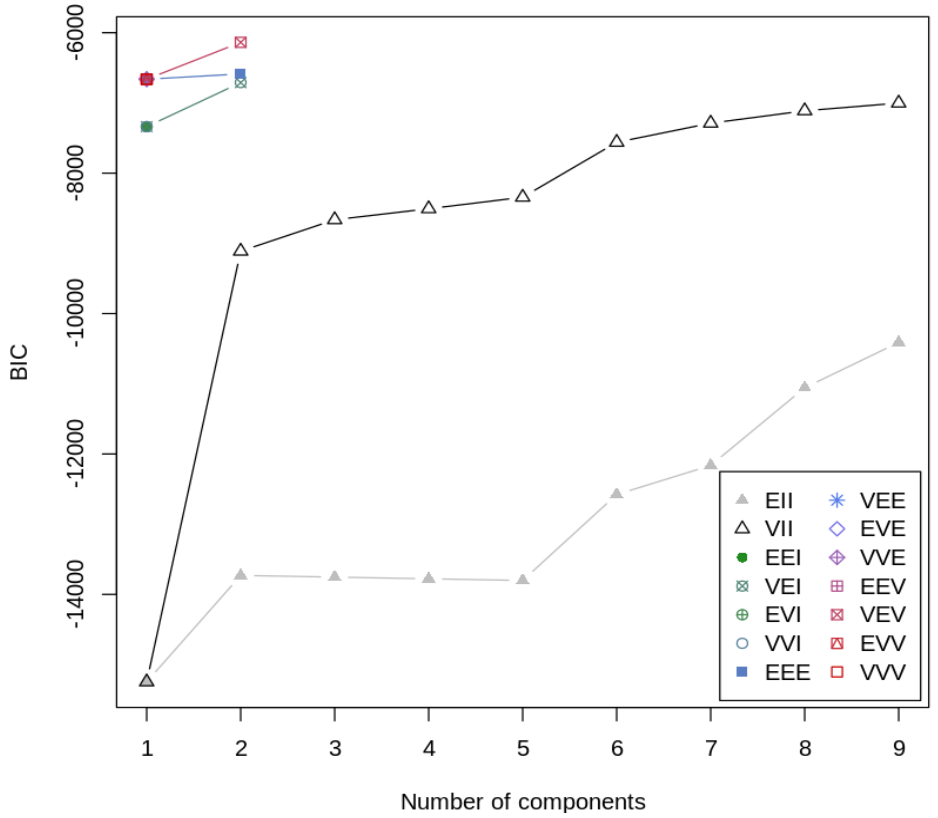




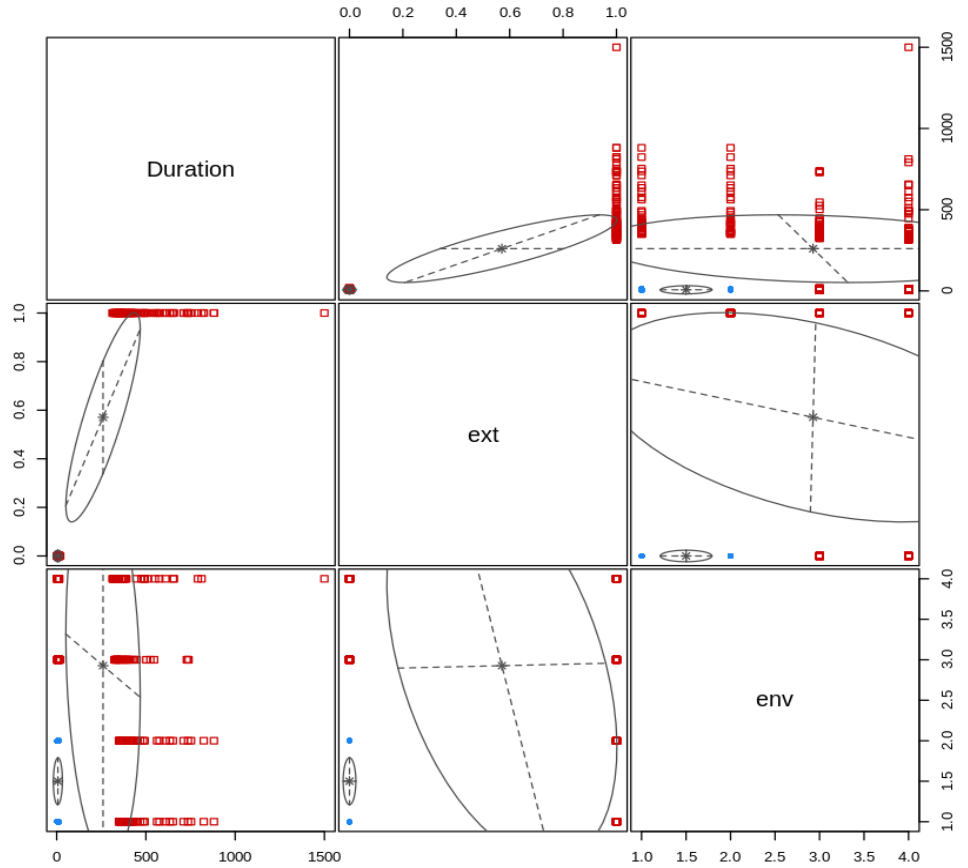


*H3: Web-based interfaces (e.g. RESTful HTTP) will increase **jitter** and predicted to have a negative impact on **usability** and the deterministic performance needed for positive control of ownship weapons.*

Bayesian Information Criterion (BIC) Analysis of “Clusters”

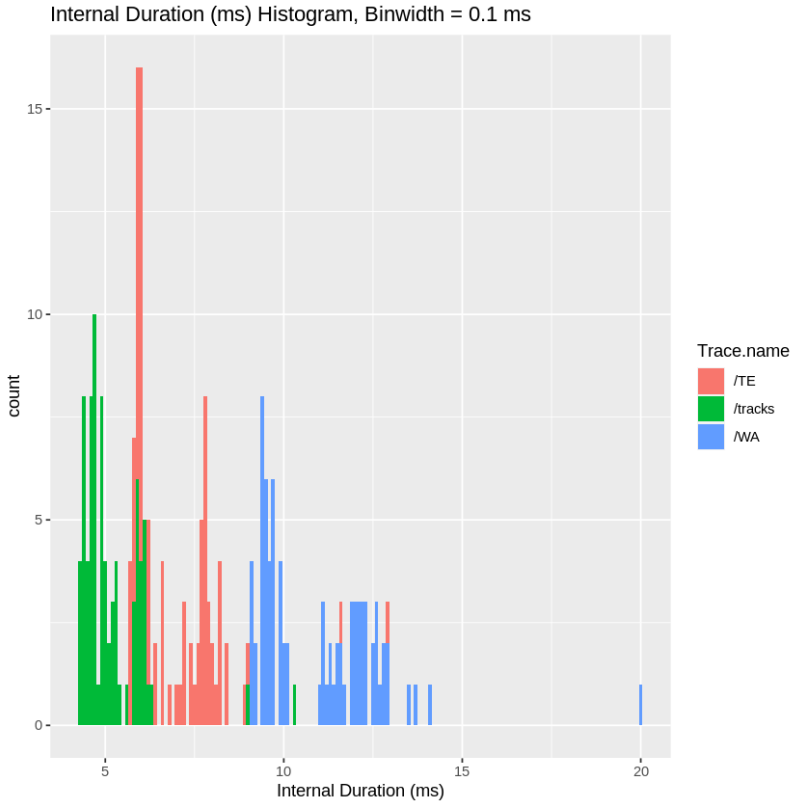


Best BIC values:			
	VEV,2	EEE,2	EEE,1
BIC	-6136.963	-6586.3351	-6662.2804
BIC diff	0.000	-449.3724	-525.3177



- VEV: varying volume, equal shape, varying orientation (ellipsoidal covariance)
- EEE: equal volume, equal shape, equal orientation (ellipsoidal covariance)

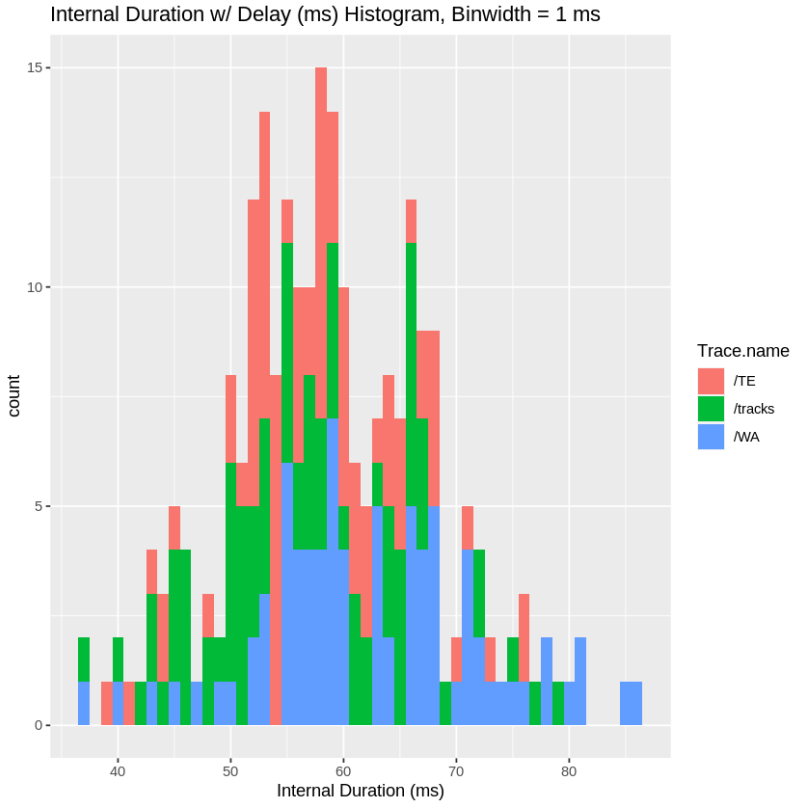
Normality Testing and Statistical Processing Adjustment (Internal Data)




```
Shapiro-Wilk normality test

data:  iSpan$Duration
W = 0.9075, p-value = 5.081e-11
```

Not Normally Distributed, Unable to Use Hypothesis Test

Added a normally distributed processing delay to each of the use cases with a mean of 50 ms and a standard deviation of 10 ms.



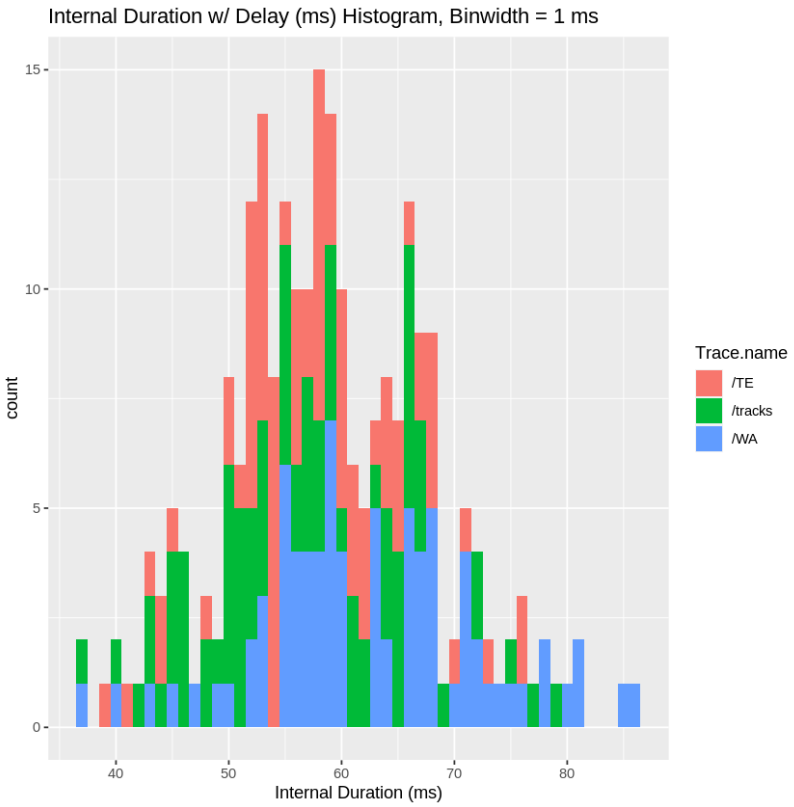

```
Shapiro-Wilk normality test

data:  pd_iSpan$Duration
W = 0.9921, p-value = 0.2265
```

Normally Distributed

The null-hypothesis of the Shapiro-Wilk test is that the population is normally distributed. Thus, if the p-value is less than the chosen alpha level, then the null hypothesis is rejected and there is evidence that the data tested are not normally distributed.

$\propto = 0.05$

Hypothesis Test (Internal Data)



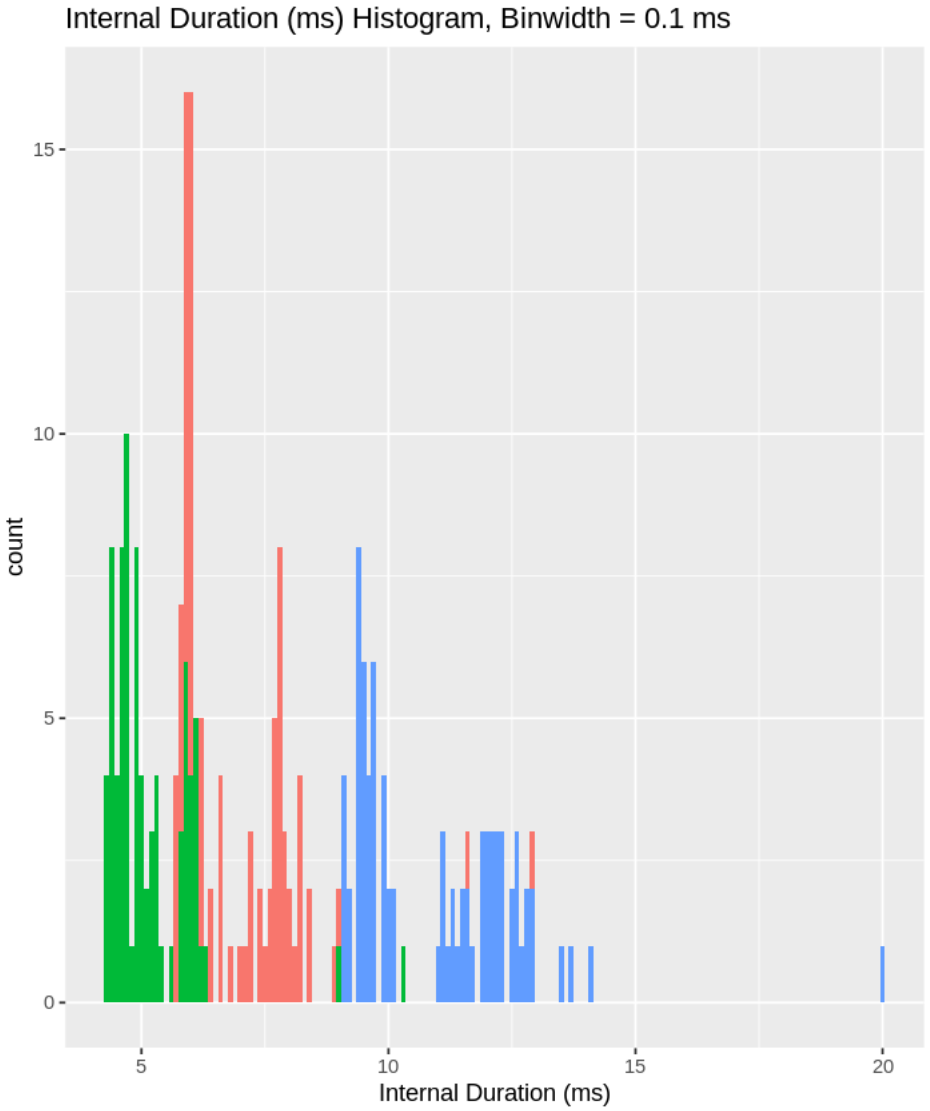
The null-hypothesis of the Shapiro-Wilk test is that the population is normally distributed. Thus, if the p-value is less than the chosen alpha level, then the null hypothesis is rejected and there is evidence that the data tested are not normally distributed.

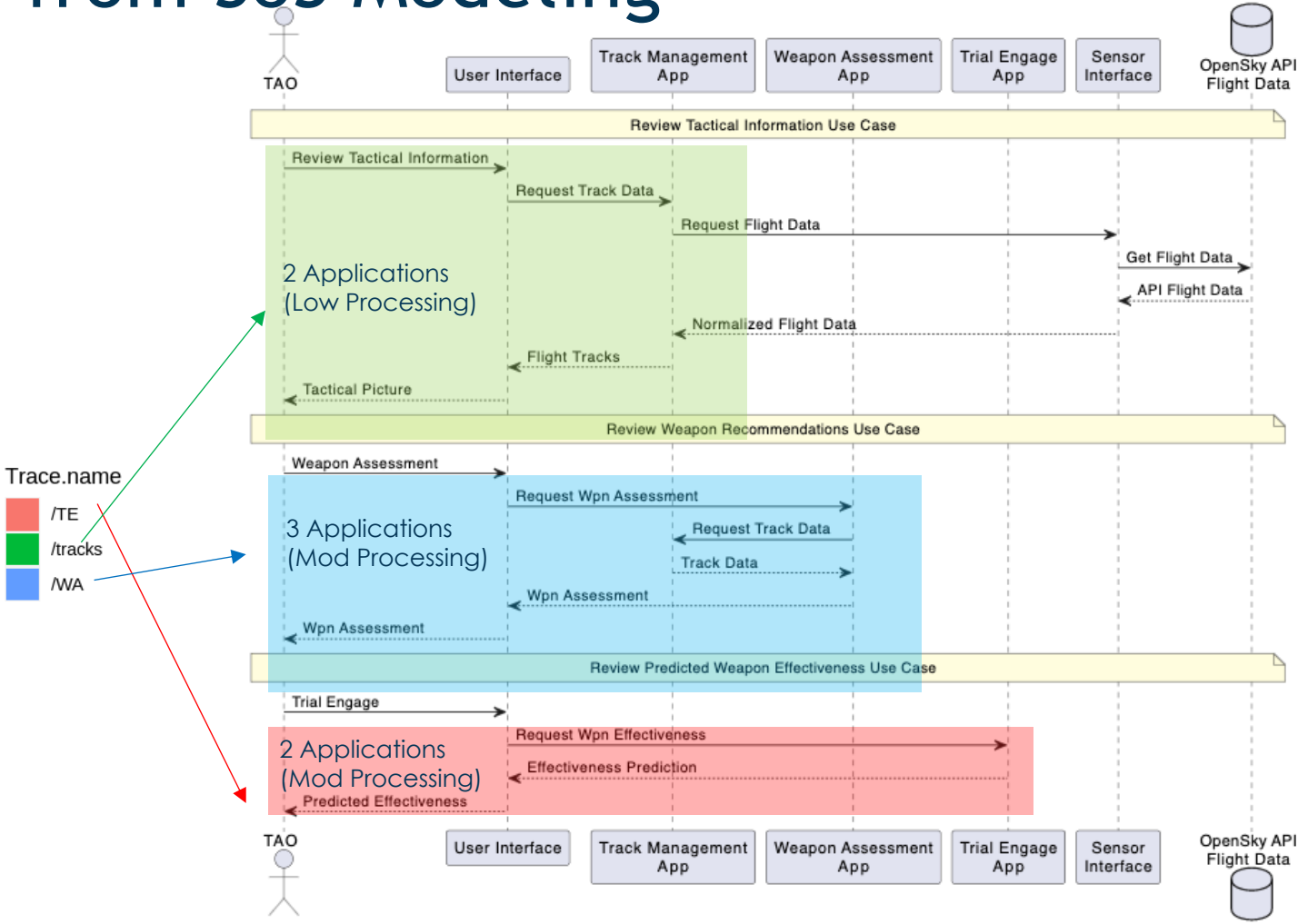
$\propto = 0.05$

```
    One Sample t-test

data:  x
t = -745.31, df = 239, p-value = 1
alternative hypothesis: true mean is greater than 500
95 percent confidence interval:
 57.9299     Inf
sample estimates:
mean of x
 58.90716
```

- "t-test" p-value converges to 1 because latency is well below 500 ms
- Significant margin available for processing delay – controllable by software architecture design

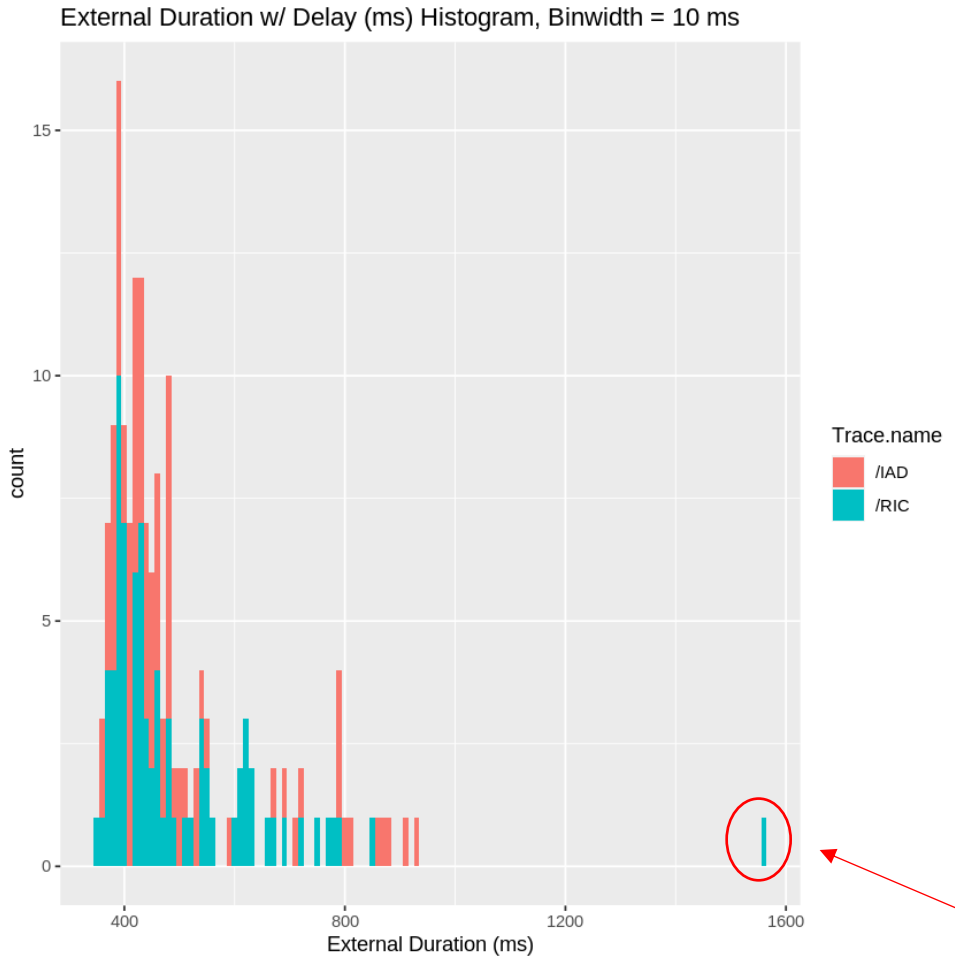
Predicted Performance from SoS Modeling





*H4: A system architecture model can predict end-to-end **latency** within a mission kill chain to quantify SoS **usability**.*

Binomial Test of HTTP Requests (External Data)



External Duration with Processing Delay

Combined Data Binomial Test Results = 89% success probability

```
   Mode   FALSE    TRUE
logical      46     354


    Exact binomial test

data:  354 and 400
number of successes = 354, number of trials = 400, p-value = 1
alternative hypothesis: true probability of success is less than 0.5
95 percent confidence interval:
 0.0000000 0.9102965
sample estimates:
probability of success
                0.885
```

External Data Binomial Test Results = 71% success probability

```
  ext          hthreshold
 Mode:logical   Mode :logical
 TRUE:160       FALSE:46
                TRUE :114


    Exact binomial test

data:  114 and 160
number of successes = 114, number of trials = 160, p-value = 1
alternative hypothesis: true probability of success is less than 0.5
95 percent confidence interval:
 0.0000000 0.7711356
sample estimates:
probability of success
                0.7125
```

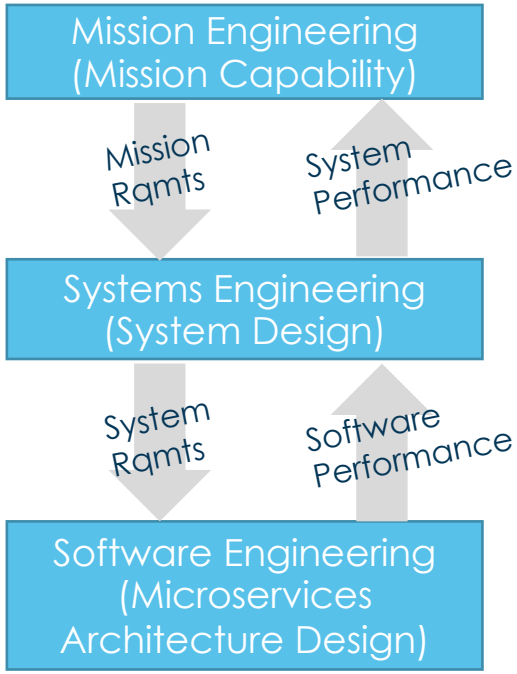
*H3: Web-based interfaces (e.g. RESTful HTTP) will increase **jitter** and predicted to have a negative impact on **usability** and the deterministic performance needed for positive control of ownship weapons.*

Qualitative Hypothesis Assessment

Hypothesis	Description	Qualitative Evidence	Assessment
H1	*The scalability of microservices as new data sources are added to an architecture enables maintaining high **throughput** and predicted to have a positive impact on **rapid fielding of capability of capability**.*	• *Ability to mediate data through containers with single responsibility with common backend interface; e.g., OpenSky sensor interface* • *Ability to rapidly field in different environments through configuration as code; e.g., Docker containers to Kubernetes (RKE2)* • *Ability for ”Ops” to get on the same page with “Dev” to rapidly field*	True
H2	*Microservices orchestration through DevSecOps technologies enables maintaining low **latency** services call responses and predicted to have a positive impact on **usability**.*	• *Demonstrated ability to manage latency through orchestration of the architecture and ensure usability*	True
H3	*Web-based interfaces (e.g. RESTful HTTP) will increase **jitter** and predicted to have a negative impact on **usability** and the deterministic performance needed for positive control of ownship weapons.*	• *Demonstrated non-deterministic performance when using external web-based interface for access to flight data* • *Mitigated problem with ”sensor interface abstraction” layer*	True
H4	*A system architecture model can predict end-to-end **latency** within a mission kill chain to quantify SoS **usability**.*	• *Demonstrated ability to use architecture models to predict system-of-system impacts once ”budgets” are applied*	True

Contribution to EMSE and Mission Engineering “Body of Knowledge”

- Insight into current microservice and container performance within context of hard-real-time combat system constraints
- Insight into system design factors affecting hard-real-time performance
- Documented traceability approach from microservices to mission capabilities
- Insight into hard-real-time implementation patterns
- Insight into microservice design documentation approach (e.g. SysML/UML)
- Insight into how to assess hard-real-time performance in mission critical systems (e.g. statistical analysis)
  - Current microservices designs have been industry based
  - Have not required ”hard real-time” assessment due to nature of business (e.g. Netflix, Amazon)
- Use of use cases to define the ”prototype” mission objectives; e.g. weapon assessment, trial engage
- Use of reusable components (e.g. containers) to define system threads within a system of systems
  - Select services that are needed to meet the mission without extra baggage
- Demonstrates that a distributed container-based system enables ability to reconfigure to meet emergent requirements beyond what was possible with a monolithic system



Conclusions

- Findings
  - Research results provide analytical quantitative evidence that containers and microservices can meet hard-real-time safety critical requirements (cBoK-1)
  - Research provides a methodology for analysis (mission architecture, design patterns, statistical analysis) (cBoK-2)
- Recommended Further Research
  - Service Mesh with mTLS (Mutual Transport Layer Security) (e.g., zero-trust authentication)
  - Distributed clusters (e.g., sensor management, track management, command and control)
  - Add representative gaussian delays to each microservice – determine and add based upon historical data
  - The source code, documentation, and analysis scripts for the prototype can be found on GitHub at https://github.com/amurp003/dss-prototype
- Next Steps
  - Publish a follow-up paper to JIDPS paper to document findings
  - JIDPS, IEEE, INCOSE, Other?

Questions?

amurp003@odu.edu

BACKUP

Domain “Profiling” – Initial Findings

- “Profiling” desired to demonstrate extensibility of research to domains outside of combat systems engineering
- Further documented evidence is needed to further EMSE body of knowledge; e.g. Space-X, transportation industry

Company	Capital One	Netflix	Pinterest	Spotify
Persona Concern https://www.cncf.io	Challenges of **resilience** and velocity. **Millions of transactions per day**. Some apps deal with critical functions like fraud detection and credit decisioning; e.g. AI.“Now, a team can come to us and we can have them up and running with a basic decisioning app in a fortnight, which before would have taken a whole quarter, if not longer.” Deployments increased by several orders of magnitude.	Challenges of **Latency**, Productivity, and Velocity. Netflix developed its own technology stack for interservice communication using HTTP/1.1. For several years, that stack supported the company’s stellar growth. But by 2015, there were pain points: Clients for interacting with remote services were often wrapped with handwritten code.	Challenges of Efficiency and Velocity. After eight years in existence, Pinterest had grown into 1,000 microservices and multiple layers of infrastructure and diverse set-up tools and platforms. The first phase involved moving services to Docker containers. Once these services went into production in early 2017, the team began looking at orchestration to help create efficiencies and manage them in a decentralized way. After an evaluation of various solutions, Pinterest went with Kubernetes.	Challenges of Efficiency, Scaling and Velocity. An early adopter of microservices and Docker, Spotify had containerized microservices running across its fleet of VMs with a homegrown container orchestration system called Helios. By late 2017, it became clear that “having a small team working on the features was just not as efficient as adopting something that was supported by a much bigger community,” he says
Applicability to CS Domain	**Directly applicable:** Resilience challenge is directly related to management of combat situations. Other concerns are also of interest but not within research scope.	**Directly applicable:** Latency concern is relatable. Productivity and velocity relate to other domain concerns but not within the research scope.	**Not applicable to this research:** However, concerns are related to broader combat system domain concerns.	**Not applicable to this research:** However, concerns are related to broader combat system domain concerns.

cBoK-1: Analytical quantitative hard-real-time microservice containers evidence
cBoK-2: Method for analysis (mission architecture, design patterns, multi-variate analysis)