

# SYSTEM OF SYSTEMS ANALYTICS LEVERAGING AZURE AND THE DISCRETE AGENT FRAMEWORK

---

Kris Ezra, PhD  
Research Scientist  
Purdue University

# BOTTOM LINE UP FRONT

---

- Purdue University has developed a specialized modeling and simulation capability that follows MOSA principles and, when coupled with tailored analytics, supports:
  - investigating new SoS architecture concepts,
  - analyses of alternatives,
  - and system-level trade studies
- Purdue's agent-based M&S approach focusing on enterprise integration strategies provides ways of reasoning about SoS problems of interest and exploring concepts of interest like flexibility, resilience, and robustness
  - Our approach is well-suited to rapidly characterize architecture and system design spaces in a way that is unique, but complementary to other existing toolsets
- SoS analysis and design questions lean heavily on Monte Carlo methods to interrogate uncertainty and explore design spaces
  - Monte Carlo approaches are computationally expensive and benefit from cluster infrastructure (local or in the cloud) and can be enabled by cloud services like AWS or Azure.

# WHY THE SoS MODELING AND ANALYSIS IMPERATIVE ?

## What is a System of Systems (SoS)?

*SoS- A collection of (often heterogeneous, distributed) systems that intentionally interact for some novel purpose, yet each retains some operational and managerial independence.*

See Maier\* / DeLaurentis\* for more details

## Common SoS Questions:

- How best to allocate functions in an architecture?
- Where to infuse new technology?
- What are the attributes of good configuration solutions?
- What are the metrics that best capture these attributes?
- How to manage data for robust sense making and graceful degradation?
- How best to balance complexity/cost and performance?

\* Maier, M. W., "Architecting Principles for System-of-Systems," *Systems Engineering*, Vol. 1, No. 4, 1998, pp. 267-284.

\* DeLaurentis, D. A., Calloway, R. K., "A system-of-systems perspective for public policy decisions," *Review of Policy Research*, Vol. 21, Issue 6, 2004, pp. 829-837.

# SoS MODELING AND SIMULATION

- Integrated solutions require **integrated** modeling
  - Not element-specific models
- Flexible solutions require **flexible** models
  - Models designed to exactly reflect current system don't help!
- General solutions require **general** models
  - Detailed models don't give extra insight at mission-/campaign-level

“Dream Aircraft” as designed by individual production teams

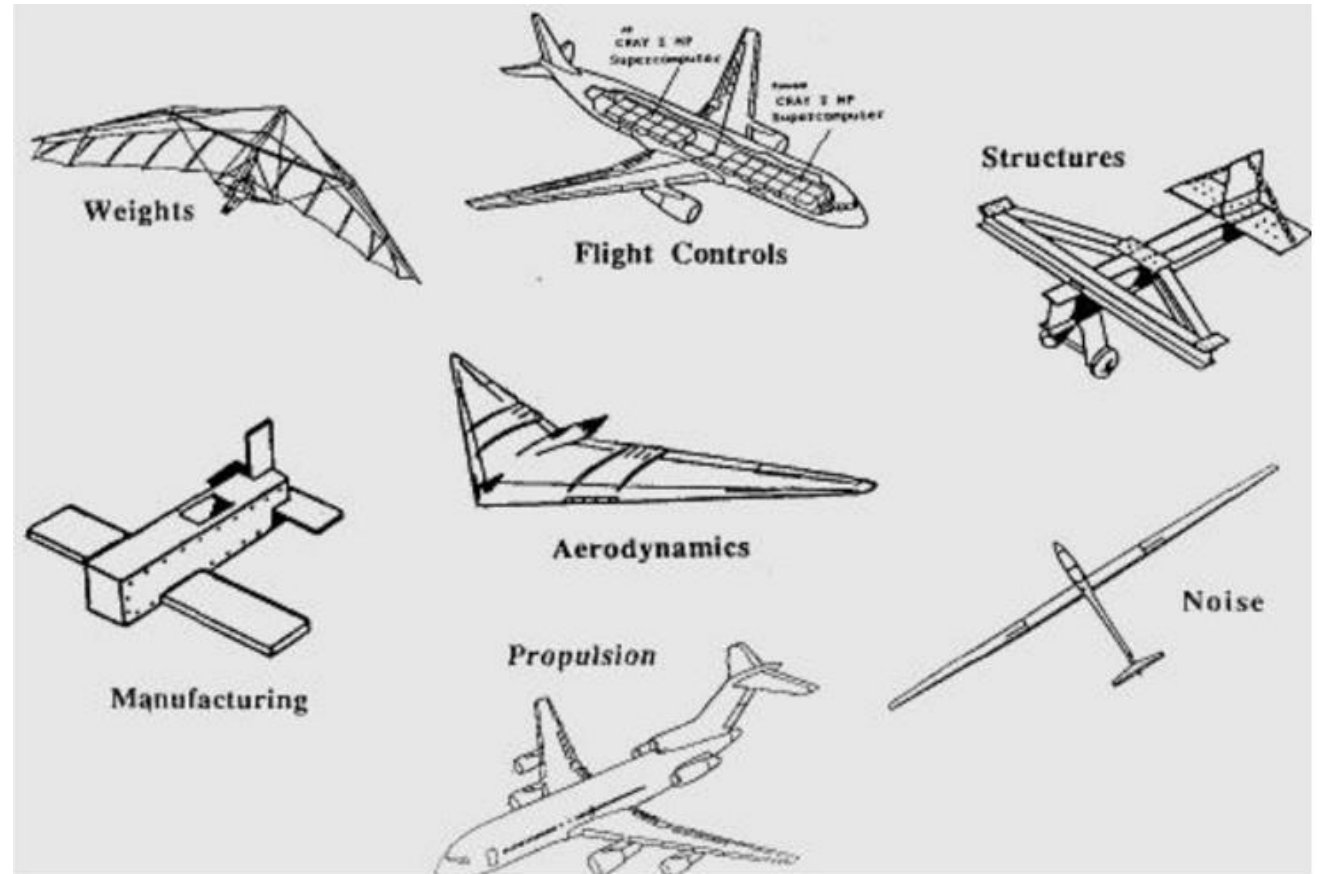


Image credit: <https://dirkgorrisen.com>

To explore design space, a generalized, flexible, mission-level analysis toolset is required.

**DAF 2.0**

---

# VISION STATEMENT

---

- Provide a unified starting point for System-of-Systems analysis with a focuses on:
  - Rapid deployment
  - Design space exploration
  - Transparent logging
  - Composable agents and analysis models for multi-fidelity modeling
- Create a constantly growing library of functions, agents, and analyses based on “literature in action” to tackle adjacent questions in similar design spaces
- Reduce the barrier to entry for agent-based modeling in a discrete-event simulation environment for students, staff, and customers exposed to Purdue research

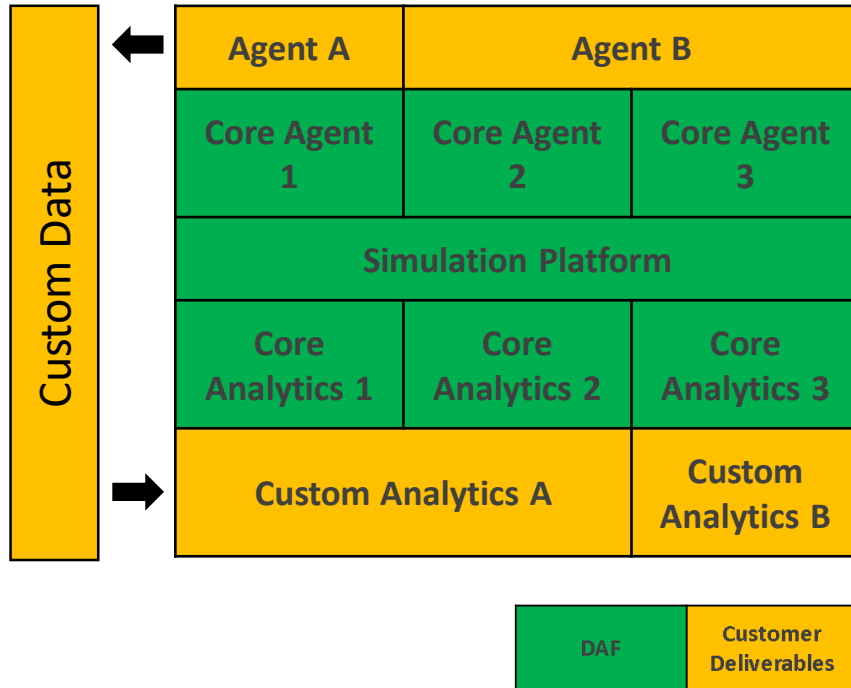
# DAF: WHAT IS IT?

---

- Discrete Agent Framework
  - For agent-based modeling (ABM) and simulation design
  - Generic, intended to support any type of ABM at medium fidelity or lower
- Built on lessons learned from the last ~10 years of simulator development across application areas in ballistic missile defense, air and missile defense, littoral combat, space exploration, and machine learning
- Heavily leverages OOP and some MATLAB “quirks”
- Designed to be used at Purdue across multiple projects including PhD and thesis research with ready extensions to industry customers and the greater academic community

# IMPLEMENTATION

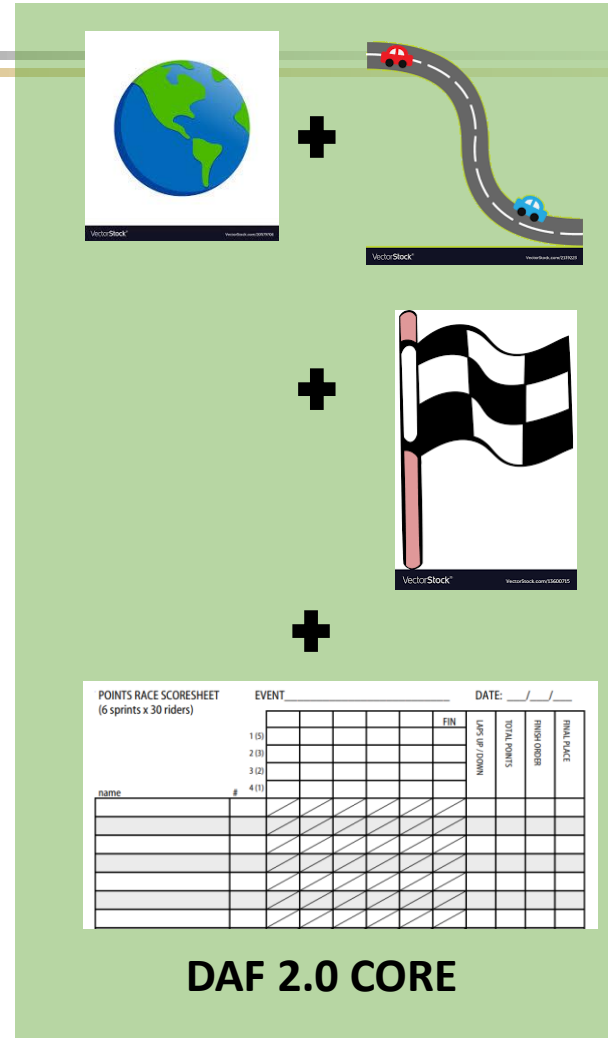
## Simulation Architecture



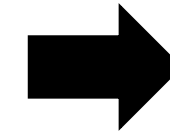
## DAF 2.0 CORE

- DAF 2.0 enables 1) rapid, custom simulation development and analysis, and 2) radical extensibility and flexibility to attack new problems
- Product of >10 years of research and is a Purdue SoSL cornerstone tool

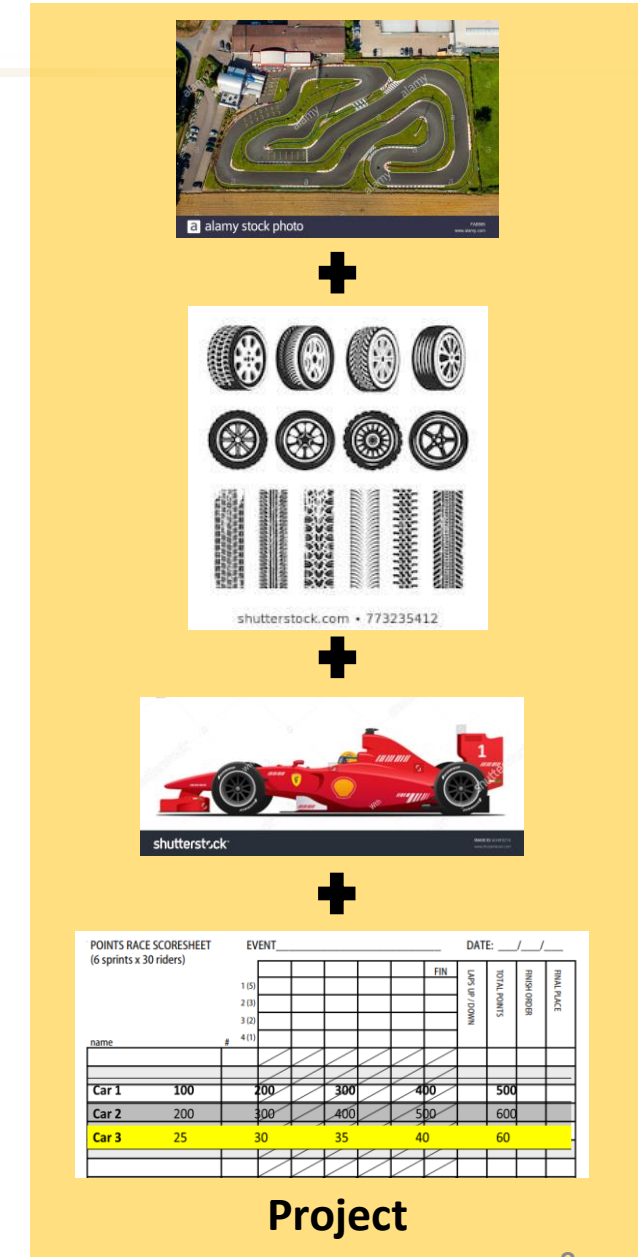
## A Racing Example



Core Software



Enables  
Simulator and  
Processing

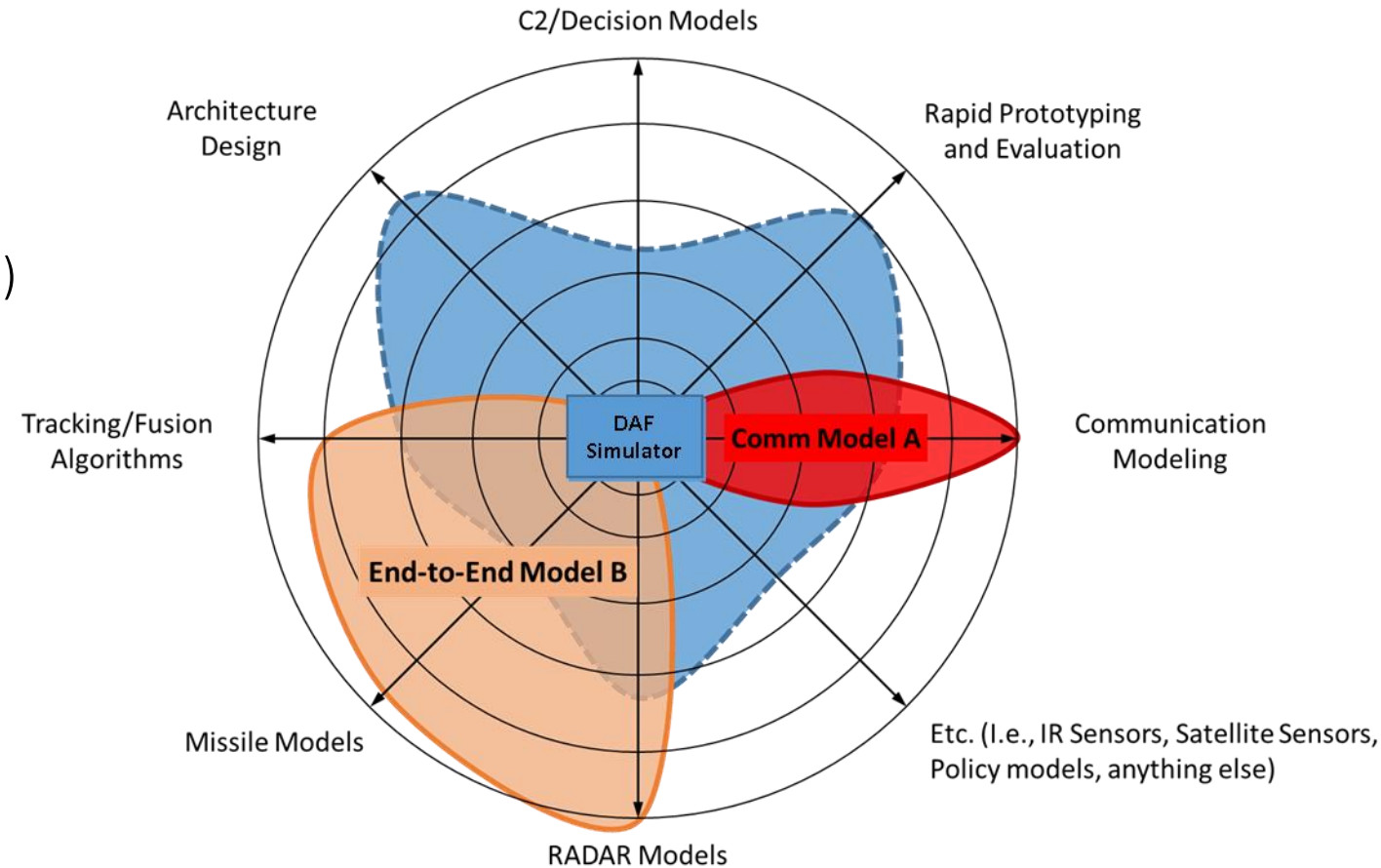




# DAF SIMULATOR STRENGTHS

- Other models are rigid (Model B) or not geared to model the multi-faceted architecture space (Comm Model A)
- DAF-based simulators address each modeling axis at desired (flexible) fidelity (Dashed Lines) to meet minimum requirements
  - Note: The best model for SoS analysis is not necessarily the one that fills the entire circle
- Our simulators leverage significant effort in engineering for architecture design, system design, and model prototyping

- ***Increased fidelity moving away from center***
- ***Faster analysis moving toward center***

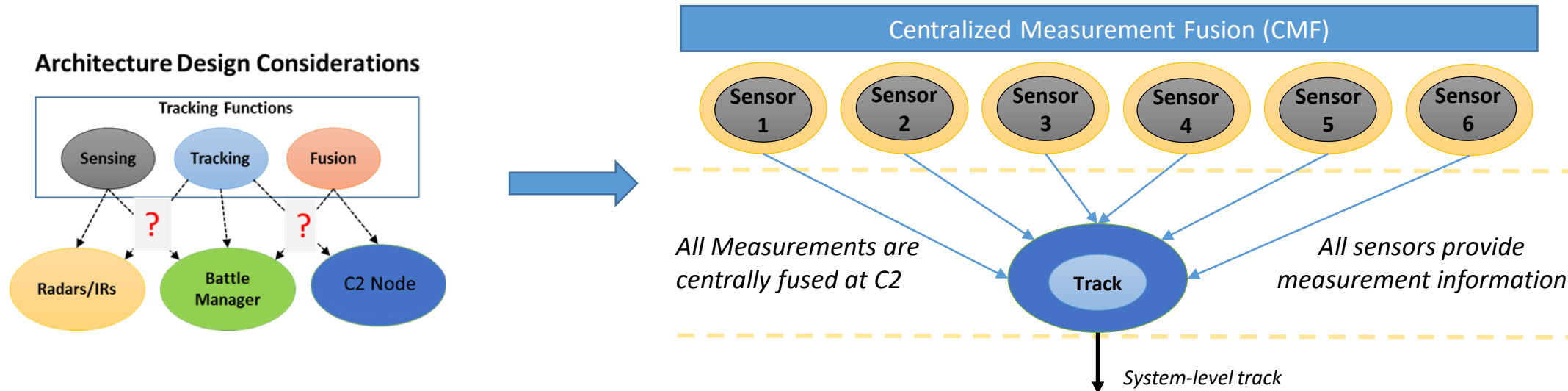


# TARGET TRACKING EXAMPLE

---

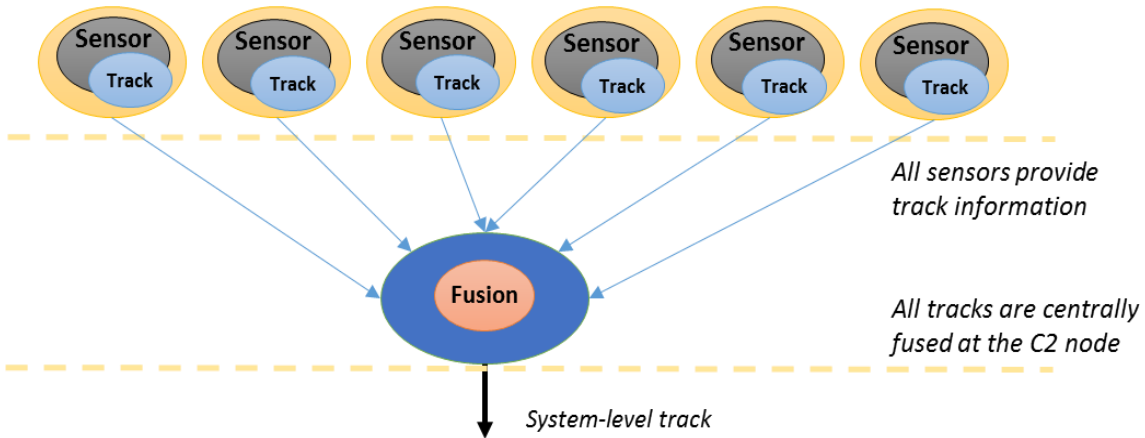
# MOTIVATING QUESTIONS

1. How can the design space of viable architectures be viewed for a scenario of interest?
2. What are the strengths and weaknesses of particular architectures?
3. What is the impact of degradation or other off-nominal operational modes?

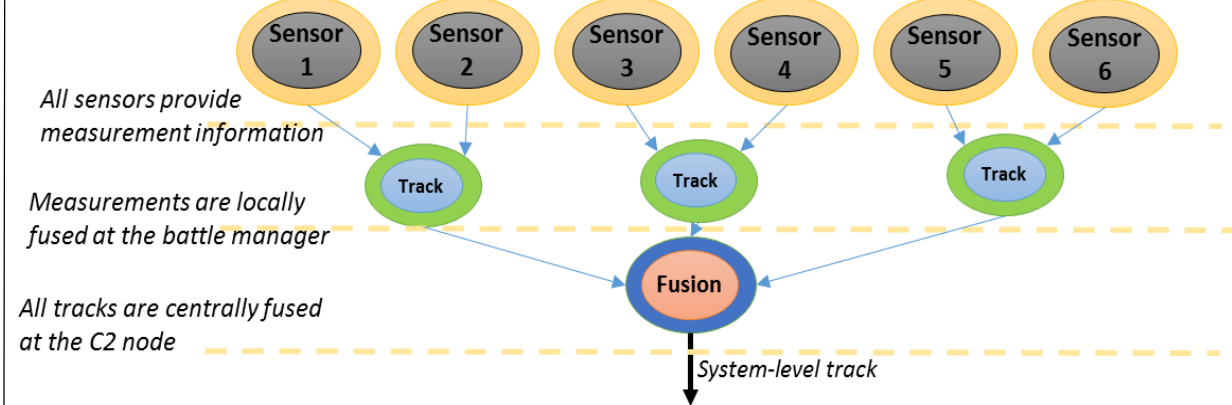


# CANDIDATE ARCHITECTURES (SYSTEMS AND CONNECTIONS)

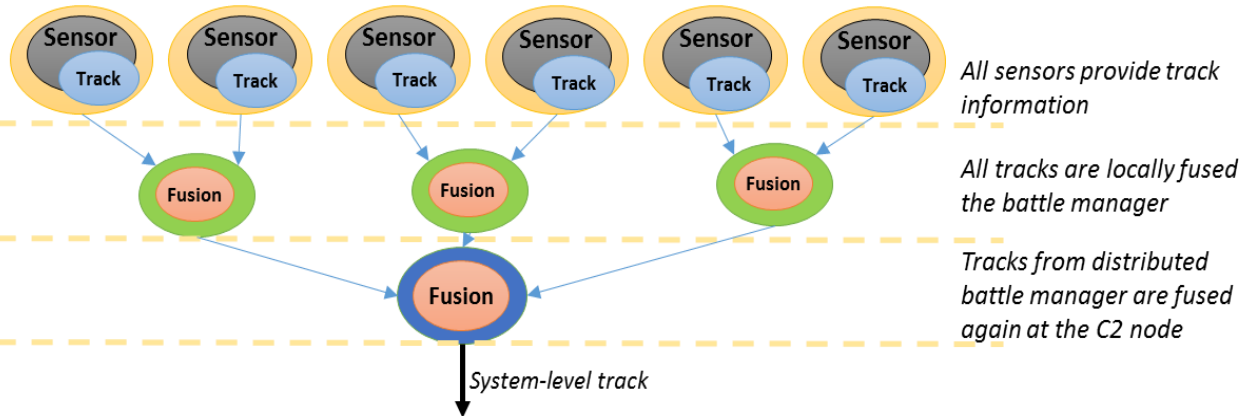
## Centralized Track Fusion (CTF)



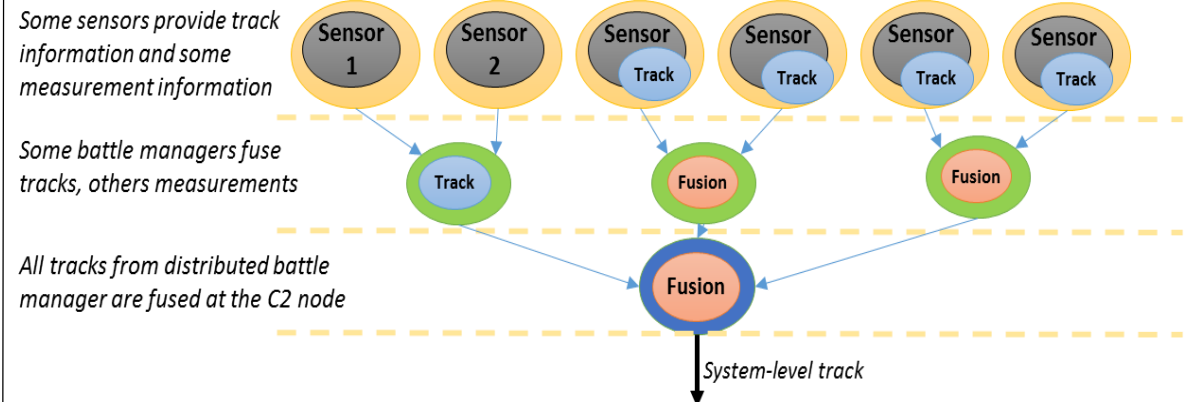
## Distributed Track Centralized Fusion (DTCF)



## Distributed Track Distributed Fusion (DTDF)



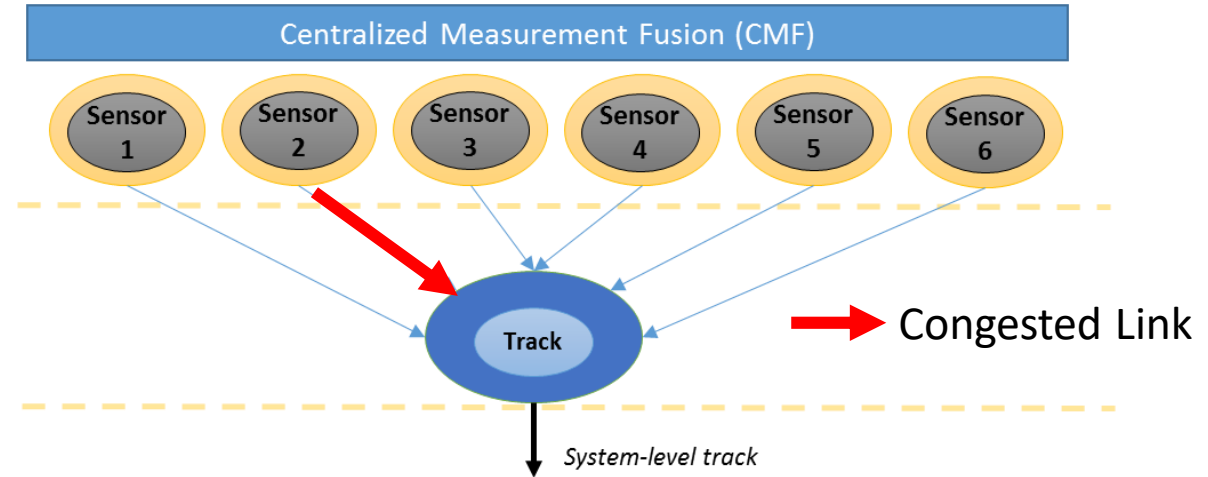
## Hybrid Tracking and Fusion (HTF)



# VARYING OPERATIONAL CONDITIONS

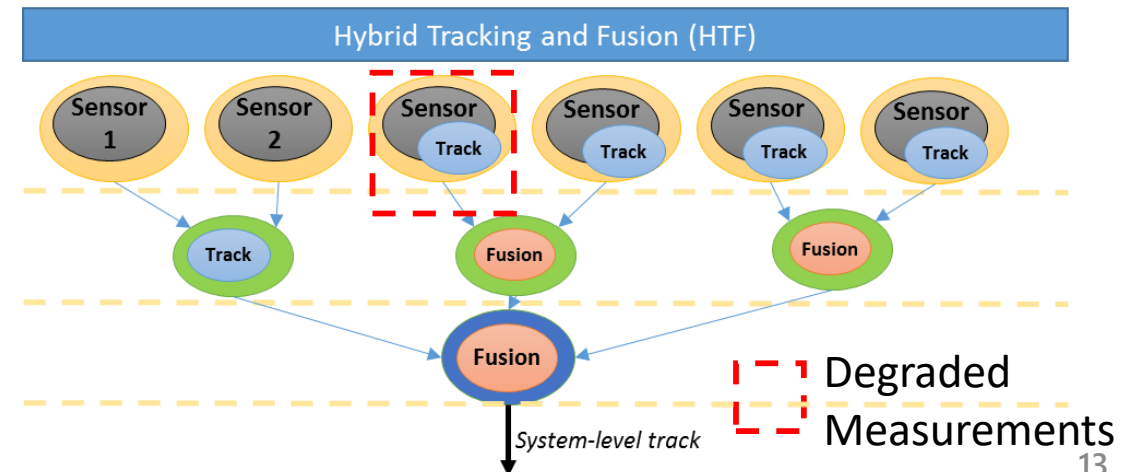
## Congested Network

- Network congestion can reduce information availability in architectures
- Type of information which may become unavailable depends on the architecture
- Network congestion in this investigation is modeled by proxy using message drop rates



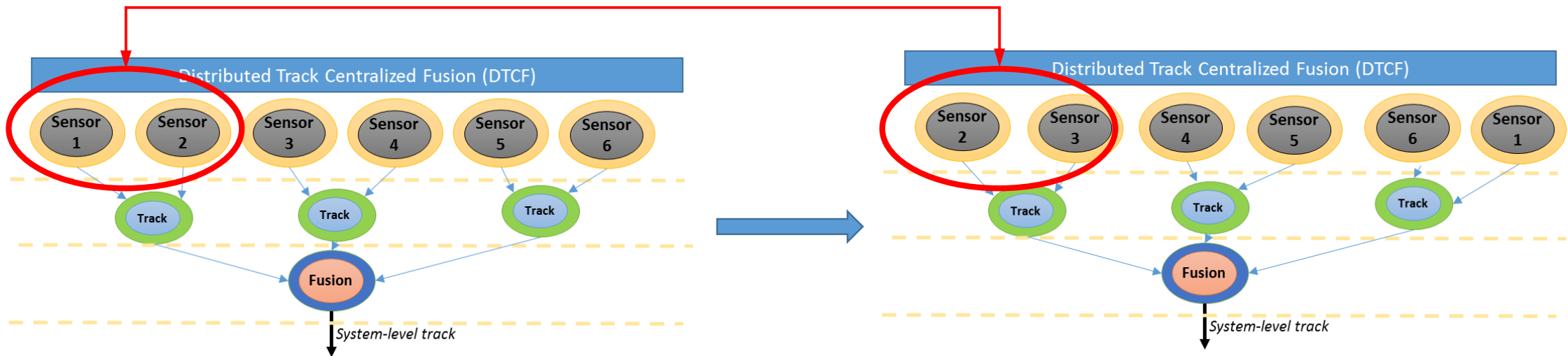
## Degraded Measurements

- Variation in measurement quality is expected to impact different architectures differently
- Gauge-able impact of bad measurements versus bad tracks will be visible by architecture



# ALTERNATE INFORMATION FLOWS

- Information flows for CMF and CTF are fixed by architecture design
- Other architectures provide opportunity to explore different information flows, i.e. sensor and battle manager pairings, while keeping the same functional allocation

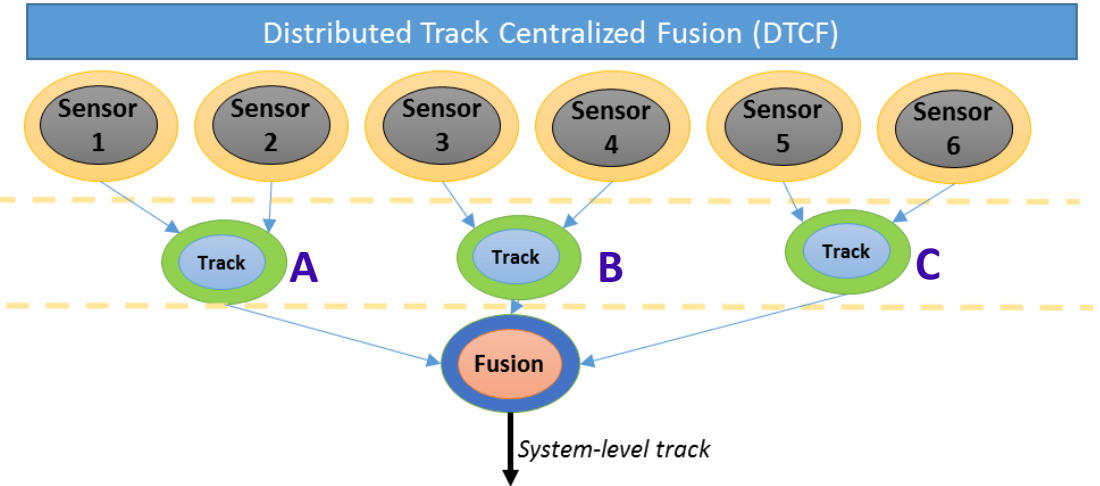


# INFORMATION FLOW PAIRINGS

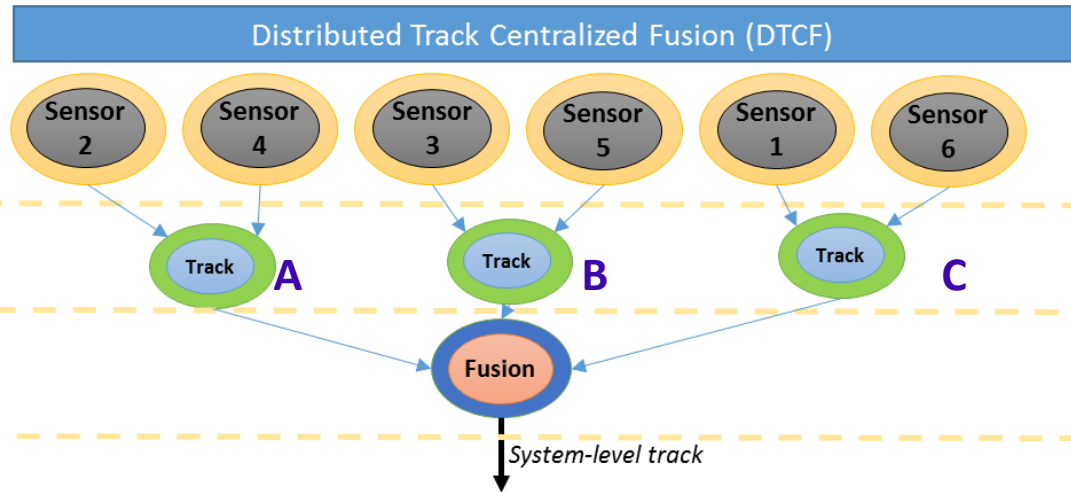
## Study Variables

Architectures	CMF	CTF	DTCF	DTDF	HTF
<b>Information Flow</b>	Baseline	Pair L1		Pair L2	
<b>Operational Conditions</b>	Baseline	Degraded Measurement		Dropped Measurements	

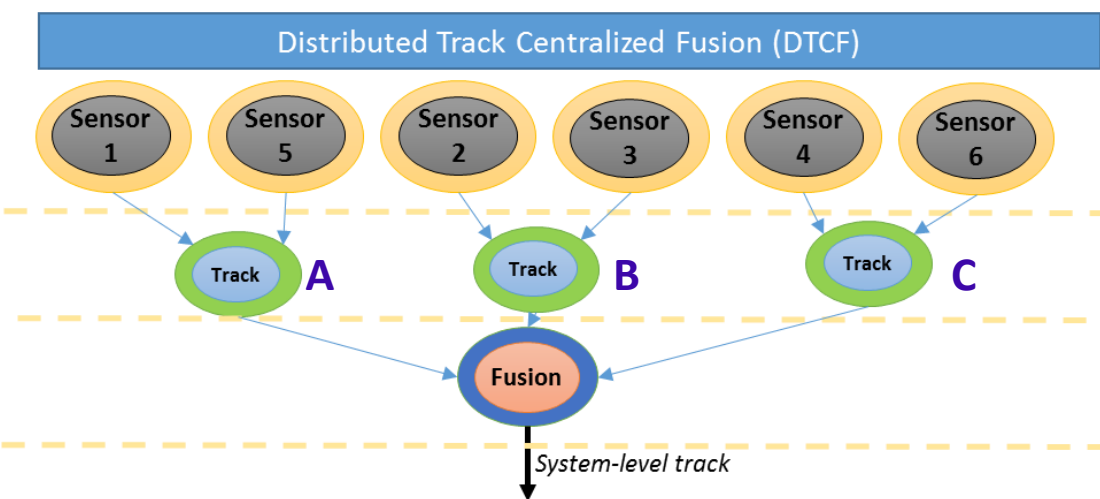
Baseline



Pair L1



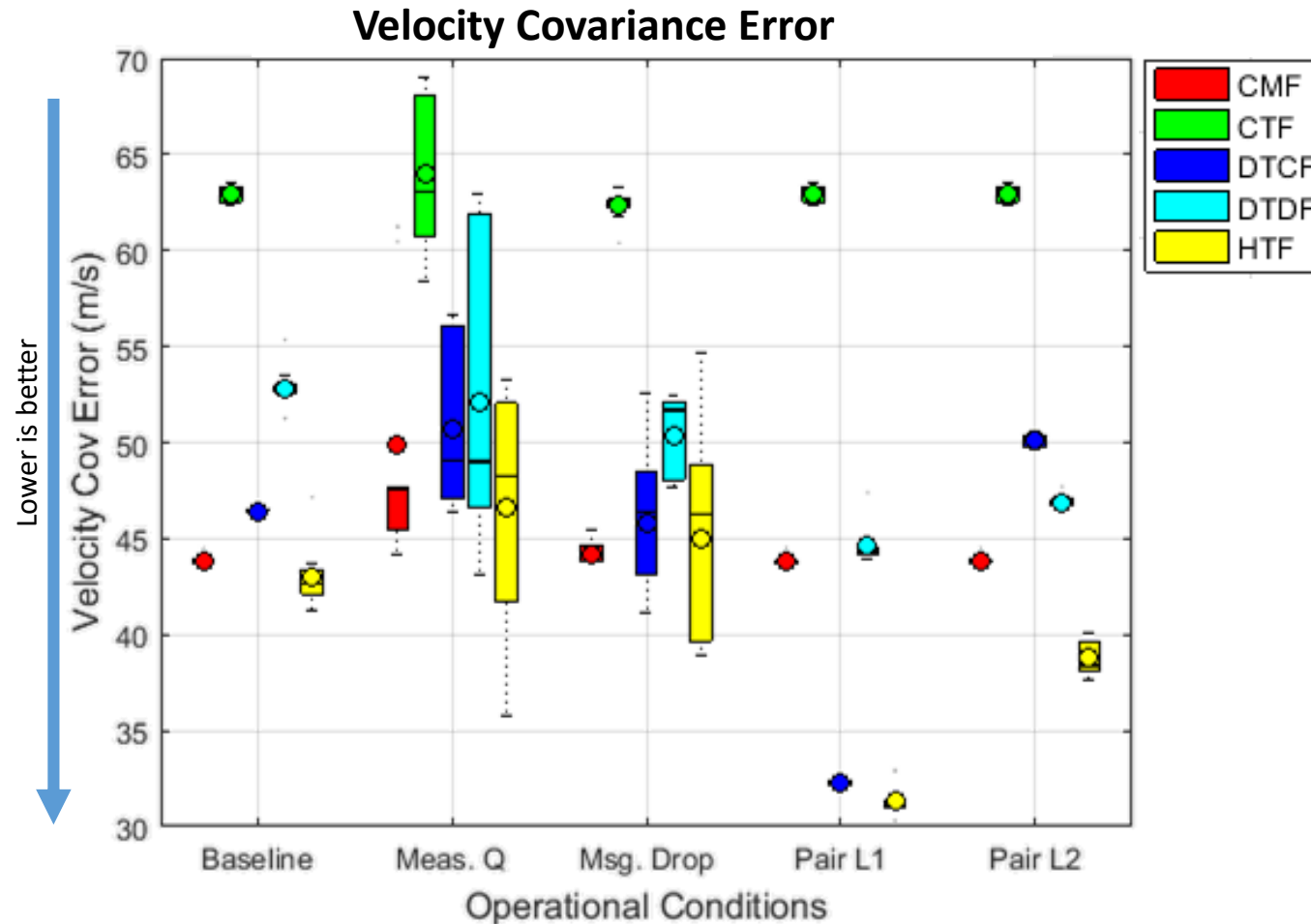
Pair L2



Note: Could also model sensor-to-sensor sharing, but omitted for this analysis

# RESULTS ANALYSIS: PERFORMANCE VARIATIONS DUE TO STUDY VARIABLES

## *Comparing architecture performance across operational conditions*



DAF 2.0 provides not only a single-run framework of composable agents, but also a robust study capability using local workstation, local server, or hybrid pool of choice (e.g., Azure, AWS, etc.)

### Architectures:

**CTF** Centralized Track Fusion

**CMF** Centralized Track Fusion

**DTCF** Distribution Tracking Centralized Fusion

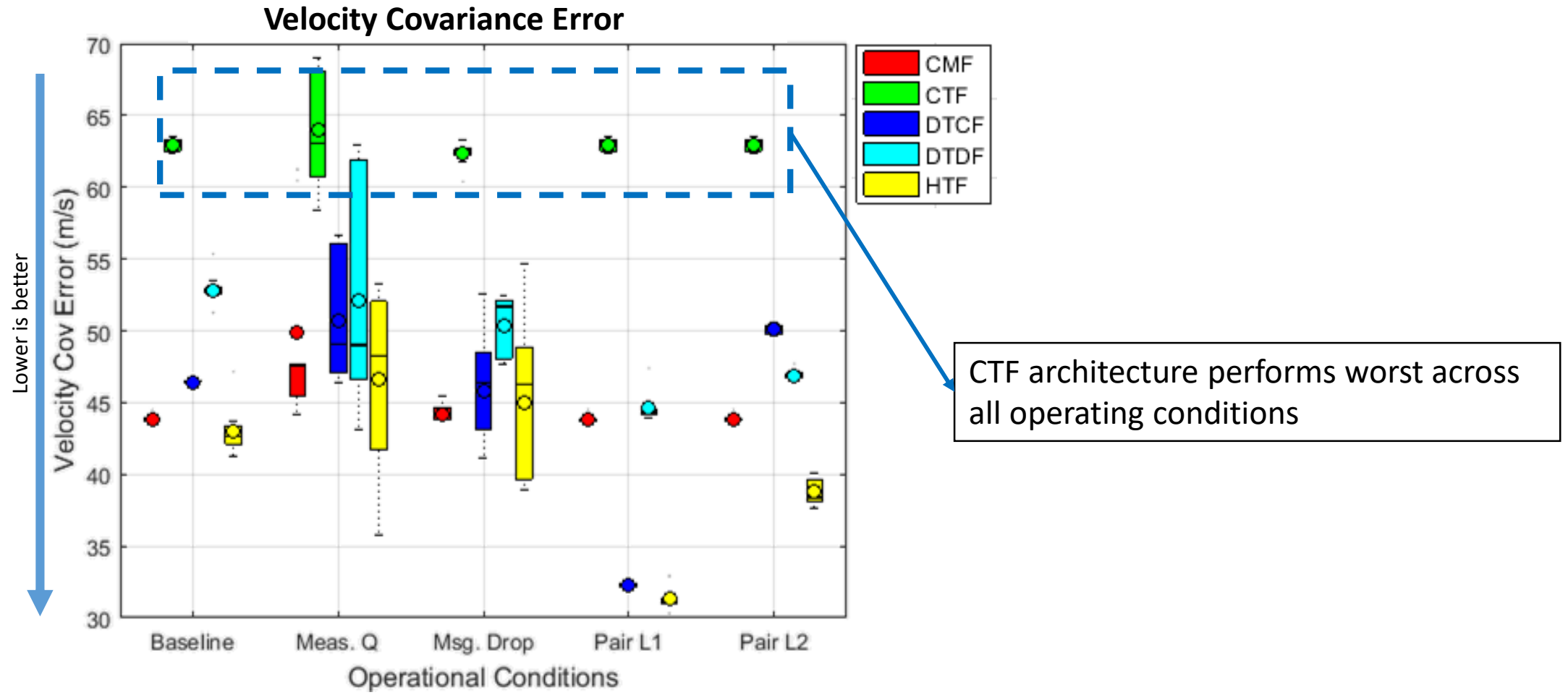
**DTDF** Distributed Tracking Distributed Fusion

**HTF** Hybrid Tracking Fusion



# PERFORMANCE VARIATIONS DUE TO STUDY VARIABLES

## *Comparing architecture performance across operational conditions*



### Architectures:

**CTF** Centralized Track Fusion

**CMF** Centralized Track Fusion

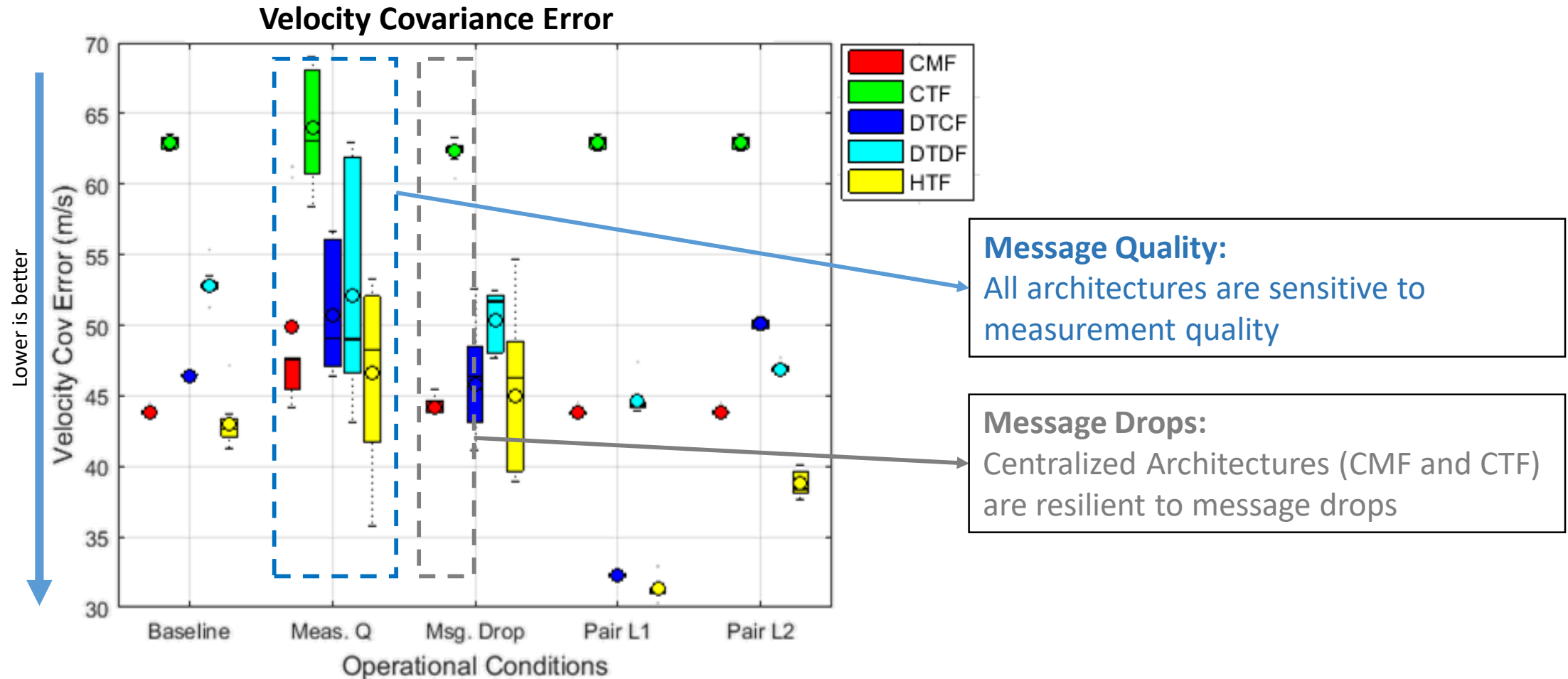
**DTCF** Distribution Tracking Centralized Fusion

**DTDF** Distributed Tracking Distributed Fusion

**HTF** Hybrid Tracking Fusion

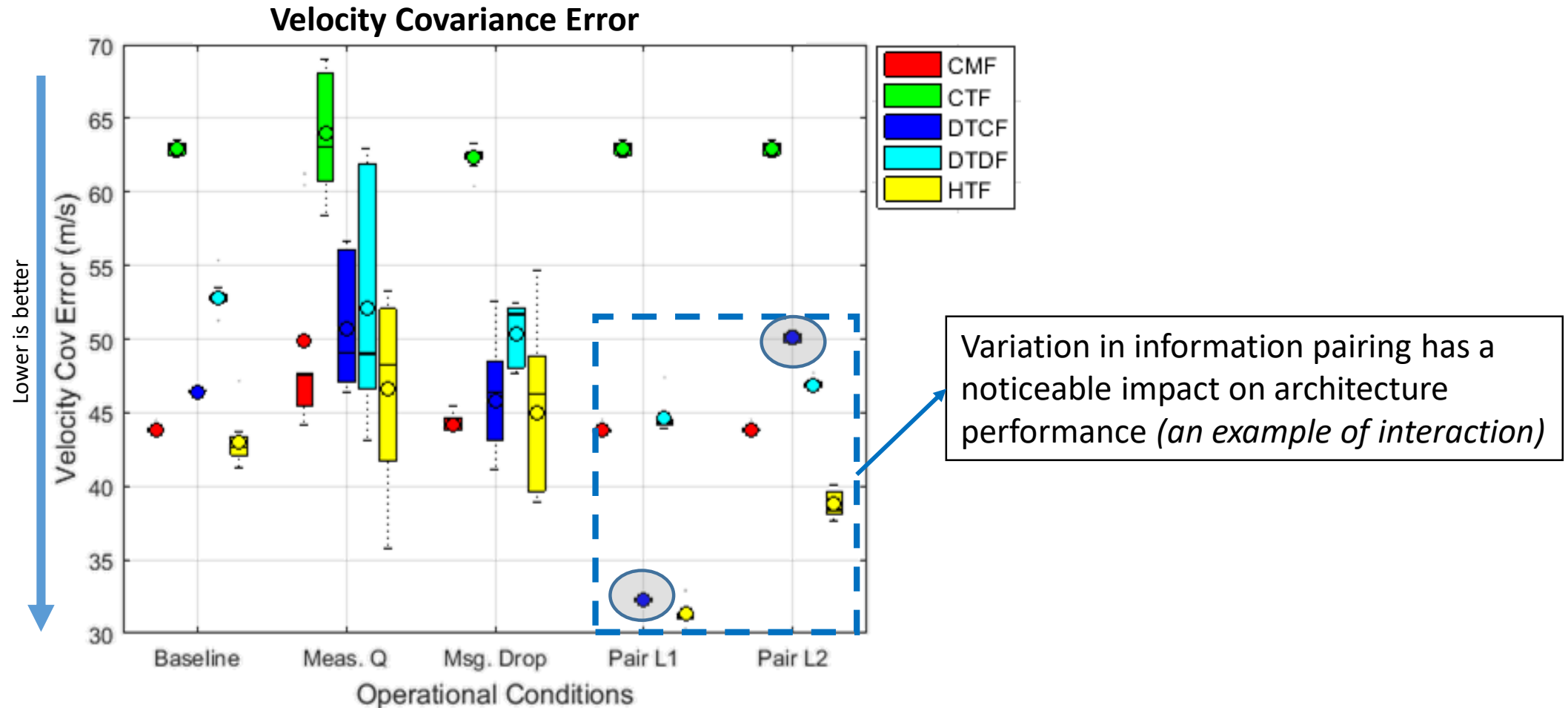
# RESULTS ANALYSIS: PERFORMANCE VARIATIONS DUE TO STUDY VARIABLES

## *Comparing architecture performance across operational conditions*



# PERFORMANCE VARIATIONS DUE TO STUDY VARIABLES

## *Comparing architecture performance across operational conditions*



### Architectures:

**CTF** Centralized Track Fusion

**CMF** Centralized Track Fusion

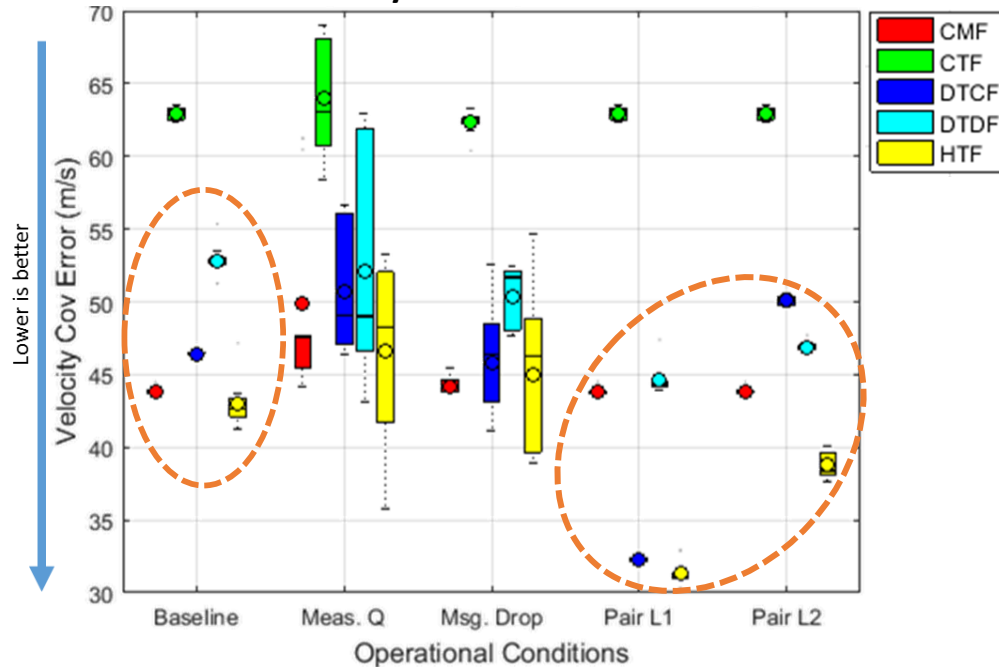
**DTCF** Distribution Tracking Centralized Fusion

**DTDF** Distributed Tracking Distributed Fusion

**HTF** Hybrid Tracking Fusion

# ARCHITECTURE PERFORMANCE ACROSS OPERATIONAL CONDITIONS

Velocity Covariance Error



**Architectures:**

**CTF** Centralized Track Fusion

**CMF** Centralized Track Fusion

**DTCF** Distribution Tracking Centralized Fusion

**DTDF** Distributed Tracking Distributed Fusion

**HTF** Hybrid Tracking Fusion

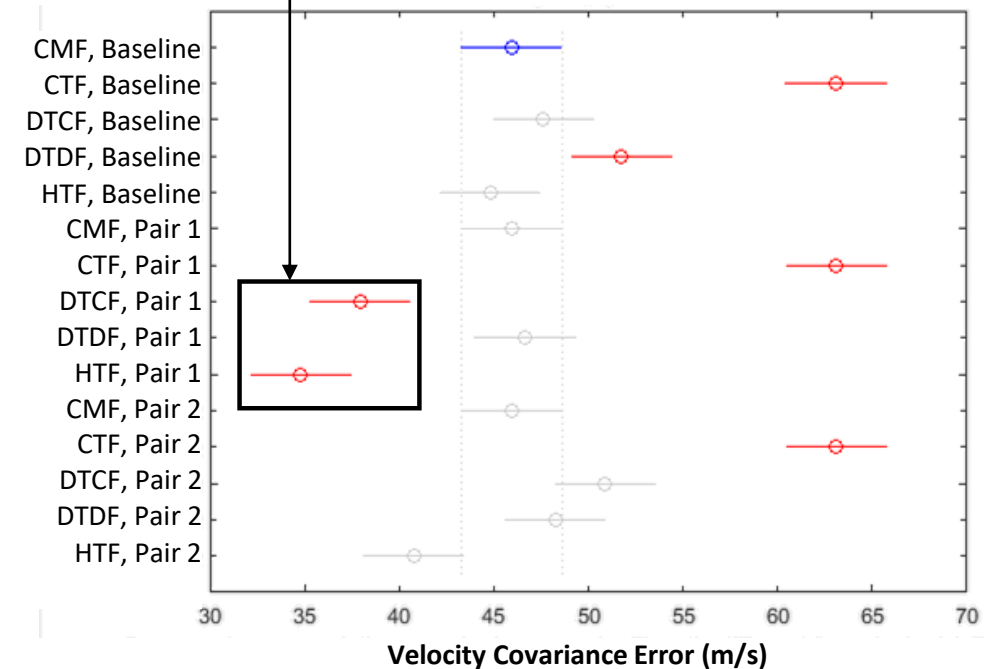
Interactions are important between operational conditions!

BUT

Not feasible to visually inspect all interactions in the design space

Two of the six statistically distinguishable alternatives provide improvement in tracking performance over baseline CMF (blue)

Interaction Plot

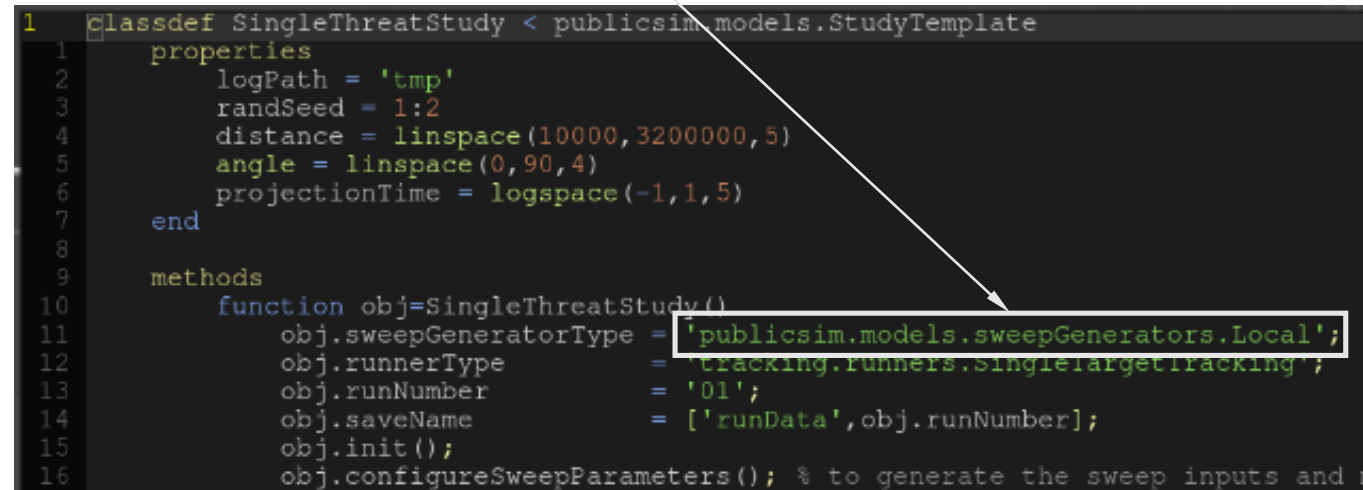


Design of Experiments (DoE) provides statistical tools to identify and quantify interactions

# DAF MODEL EXTENSIBILITY AND ENABLING TOOLS

- A core extensibility feature of the DAF framework is the Study Runner
  - Specifies a parameter set (model parameters and random seed) then builds a full-factorial combination of runners
  - Specifies a SweepGenerator structure with a tailored run file and submit scripts
    - Current options include local, torque/pbs/qsub, and SLURM (under development)
- Readily extensible to tools built in the cloud or cloud services themselves

Outputs generated for a variety of sweep structures



```
1 classdef SingleThreatStudy < publicsim.models.StudyTemplate
2     properties
3         logPath = 'tmp'
4         randSeed = 1:2
5         distance = linspace(10000,3200000,5)
6         angle = linspace(0,90,4)
7         projectionTime = logspace(-1,1,5)
8     end
9     methods
10        function obj=SingleThreatStudy()
11            obj.sweepGeneratorType = 'publicsim.models.sweepGenerators.Local';
12            obj.runnerType = 'tracking.runners.SingleTargetTracking';
13            obj.runNumber = '01';
14            obj.saveName = ['runData',obj.runNumber];
15            obj.init();
16            obj.configureSweepParameters(); % to generate the sweep inputs and t
```

The image shows a MATLAB code snippet for a class named `SingleThreatStudy` that inherits from `publicsim.models.StudyTemplate`. The code is displayed in a dark-themed editor with line numbers 1 through 16. It defines properties for `logPath`, `randSeed`, `distance`, `angle`, and `projectionTime`. The `methods` section includes an initialization function `obj=SingleThreatStudy()` which sets `obj.sweepGeneratorType` to `'publicsim.models.sweepGenerators.Local'`, `obj.runnerType` to `'tracking.runners.SingleTargetTracking'`, `obj.runNumber` to `'01'`, and `obj.saveName` to `['runData',obj.runNumber]`. It also calls `obj.init()` and `obj.configureSweepParameters()` with a comment indicating its purpose is to generate sweep inputs and outputs. A white arrow points from the text "Outputs generated for a variety of sweep structures" to the `obj.sweepGeneratorType` assignment on line 11.

Azure Platform as a Service or Infrastructure as a Service options are flexible candidates for conducting this type of analysis

# AZURE SERVICES AND LESSONS LEARNED

- Azure offers a Batch Service for specifying a job function, a data set (a block of runs), and a pool of virtual machines to tackle large jobs
  - Pros: Easy set up for simple jobs like ffmpeg across video files
  - Cons: Difficult to configure if you require, for example MATLAB.
- Azure offers VM Scale Sets for pre-conditioning a set of nodes for a job
  - Pros: Can be done programmatically, and can be configured for images with custom software
  - Cons: Up-front cost for configuring VMs
- Azure offers VMs of varying sizes to adjust to varying project needs
  - Pros: Easy custom configuration and spin up
  - Cons: Higher incurred cost with higher capability machines

Depending on needs, cost, and skillset, Azure may or may not be the right solution for a given job

# CONCLUSIONS

---

- DAF 2.0 is a capable infrastructure for multi-target tracking and other problems of interest
- DAF 2.0 is well-suited to the generation and analysis of diverse SoS data sets in an extensible and reproducible way
- Azure provides a powerful set of tools to enable Monte Carlo analysis or even local development BUT is not necessarily a “magic bullet” when considering implementation difficulties, cost, and deployment times if other alternatives are available

# QUESTIONS?

---



# BACKUP

---

# ANALYTICS BY THREAT

## Notes

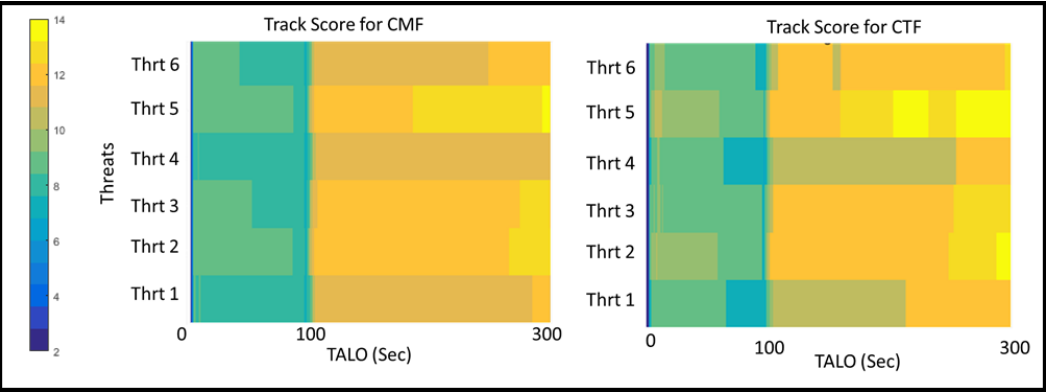
- 1. Scenario redacted for distribution
- 2. Track Score defined by MIL-STD-6016e

Study Variables					
Architectures	CMF	CTF	DTCF	DTDF	HTF
Information Flow	Baseline	Pair L1		Pair L2	
Operational Conditions	Baseline	Degraded Measurement		Dropped Measurements	

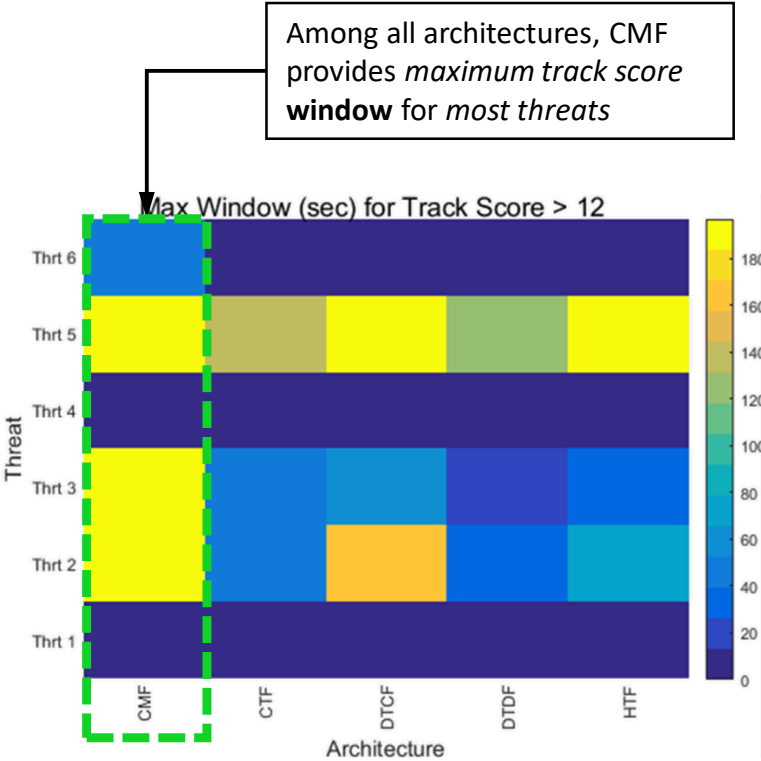
**Architectures:**

**CTF** Centralized Track Fusion  
**CMF** Centralized Track Fusion  
**DTCF** Distribution Tracking Centralized Fusion  
**DTDF** Distributed Tracking Distributed Fusion  
**HTF** Hybrid Tracking Fusion

## Example Median Track Score (Baseline)



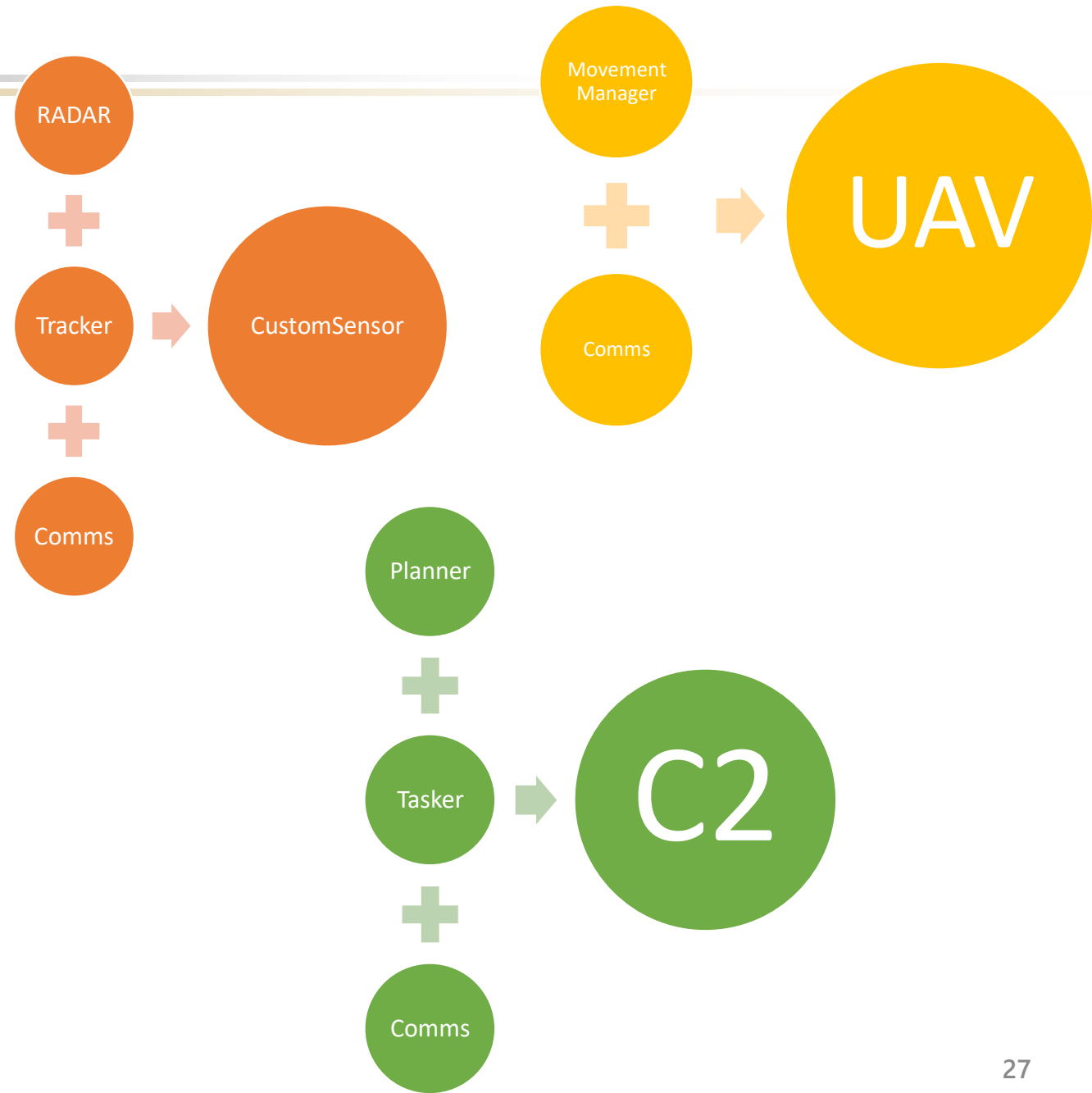
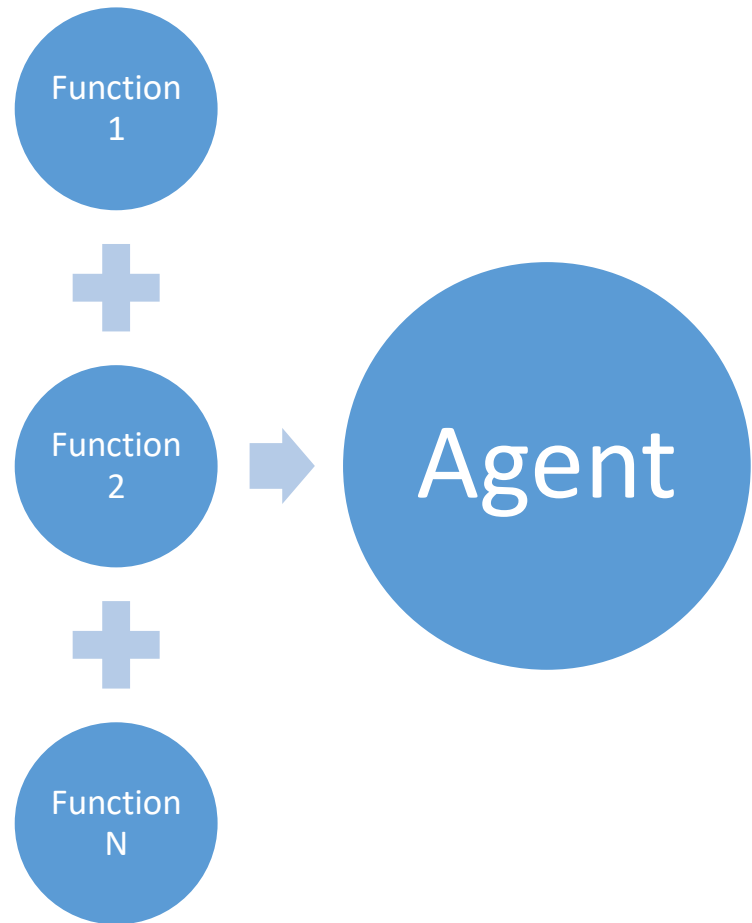
## Metric Design



**Problem:** Difficult to compare different architectures based on per threat, over time basis.

**Solution:** Develop Aggregate Track Score Metric (Right)

# FUNCTIONS & AGENTS



# WHAT IS A SYSTEM OF SYSTEMS (SoS) DESIGN PROBLEM?

SoS- A collection of (often heterogeneous, distributed) systems that intentionally interact for some novel purpose yet each retains some operational & managerial independence.

Maier's SoS Design Heuristics\*:

- *Stable Intermediate Forms*
  - Complex systems evolve faster with SIFs; Establish early (valued) capabilities, then build out
- *Policy Triage*
  - Choose carefully what you try to control; Standards; Be willing to cut out or add
- *Leverage at the Interfaces*
  - The greatest leverage in system architecting (and greatest danger) is at the interfaces.
- *Ensuring Cooperation*
  - Since systems can choose to participate (or not), incentives must be designed in

\* Maier, M. W., "Architecting Principles for System-of-Systems," *Systems Engineering*, Vol. 1, No. 4, 1998, pp. 267-284.

# DAF 2.0 MOTIVATION

1. Licensing/IP restrictions
  - Re-use of common elements difficult in context of previous contracts
  - Lower computational overheads (ITAR/Protected but not CUI when possible)
2. Partitioning for controlled methods or data
  - Separation of generic, basic science from defense topics
  - Non-defense academic publications (modeling/sim, communications, etc.)
3. Software lessons learned and refactoring
  - Organization of logic and models
  - Logging and communication streamlining
  - Removal of global and persistent variables
4. Proper model organization
  - Cross-project extensibility
  - Rectifying modeling principles with implementation

Purdue adjusted its approach to model/sim based on prior multi-year experience (DAF 1.0)

# RESULTS ANALYSIS: UNDERSTANDING DESIGN SPACE USING DoE (1)

Interactions are important between operational conditions!

- Architecture performance changes with different information flows
- Not feasible to visually inspect all interactions in the design space

Design of Experiments (DoE) provides statistical inference to identify and quantify interactions

Analysis of Variance					
Source	Sum Sq.	d.f.	Mean Sq.	F	Prob>F
Arch	2688.74	4	672.184	244.67	0
Pair	210.54	2	105.268	38.32	0
Ops	222.06	2	111.03	40.41	0
Arch*Pair	254.11	8	31.763	11.56	0
Arch*Ops	47.73	8	5.967	2.17	0.0889
Pair*Ops	33.55	4	8.387	3.05	0.0478
Error	43.96	16	2.747		
Total	3500.68	44			

Significant Interactions

Insignificant Interactions

Significant Interactions

## Architectures:

**CTF** Centralized Track Fusion

**CMF** Centralized Track Fusion

**DTCF** Distribution Tracking Centralized Fusion

**DTDF** Distributed Tracking Distributed Fusion

**HTF** Hybrid Tracking Fusion

# CHARACTERISTICS

- Flexible, streamlined core simulation infrastructures
  - Discrete event-based simulation core
  - Broad model exploration capabilities
- Extensive use of inheritance
  - Modularity through superclasses, programming practices
  - Rapid prototyping across domains (ABT, Ballistic, regional vs global)
  - Reusable functions and orchestration code
- Multiple package structure
  - Open-source “Public” package
  - Cross-project “ITAR” package
  - Closed, proprietary packages specific to each project

Current, ground-up DAF implementation supports greater development flexibility, pre-empts questions about intellectual property, and permits broader leveraging of Purdue expertise