# Verification & Validation of Cognitive Adaptive Systems – A case study: Nautilus Learning Agent Prototype

*Dr. Max Spolaor*
*The Aerospace Corporation*
*Vehicle Engineering & Autonomy Department (VE&A)*

Approved for public release. OTR 2022-01234.

# *The Nautilus Prototype*

*Objective and Goals*

- **Purpose** – Nautilus is an Aerospace-led effort to prototype a machine-learning-based mission planning capability for an autonomous spacecraft to be integrated and tested in a third-party application identified by the customer
  - *The Gryphon framework has been identified as the targeted 3rd party application*
- **Design Objective** – Nautilus is a motion model prototype that predicts future location of tracked targets based on a history of similar target movements
  - *Prototype will be integrated into the Gryphon medium-fidelity model*
- The **Goals** of this effort were to:
  - *(1) develop a deep understanding of a learning functionality with both online (i.e., in-situ) and offline (i.e., pre-training) experience*
  - *(2) gain a baseline understanding of the technical, infrastructure, integration and test (IA&T), verification and validation (V&V) and sustainment processes for integrating an adaptive capability into an existing framework*
  - *(3) leverage value gained from the DRATS basic research into an experimental software prototype*

***Nautilus is a first-of-its-kind agent to demonstrate the implications of building adaptive capabilities.***

# The Nautilus Prototype

*What approach will Nautilus use for prediction?*

- ***Given some past observations of a moving ship, where should we look to see the ship again?***

- Assumptions:
  - *Sensor provides a Field of View (FOV) much larger than the ship (i.e., 1000x1000 meter image)*
  - *A successful prediction is when the ship is somewhere within the FOV of the sensor*
    - In other words, there is no additional value to be “more” accurate (i.e., specific latitude and longitude location)


***Nautilus provides a mission manager a recommended location to point the sensor where the object is predicted to be at a specific time in the future.***

# *Verification and Validation as it Relates to Nautilus*

## *V&V Literature Review on CAS Systems and Learning Agents*

CAS intrinsic ability to evolve and adapt over time render traditional V&V methods ineffective in fully assessing the systems. How can the fundamental V&V questions still be answered?

- Will the system's software do what it is supposed to do?
- Will the system's software not do what it is not supposed to do?
- Will the system's software respond as expected under adverse conditions?

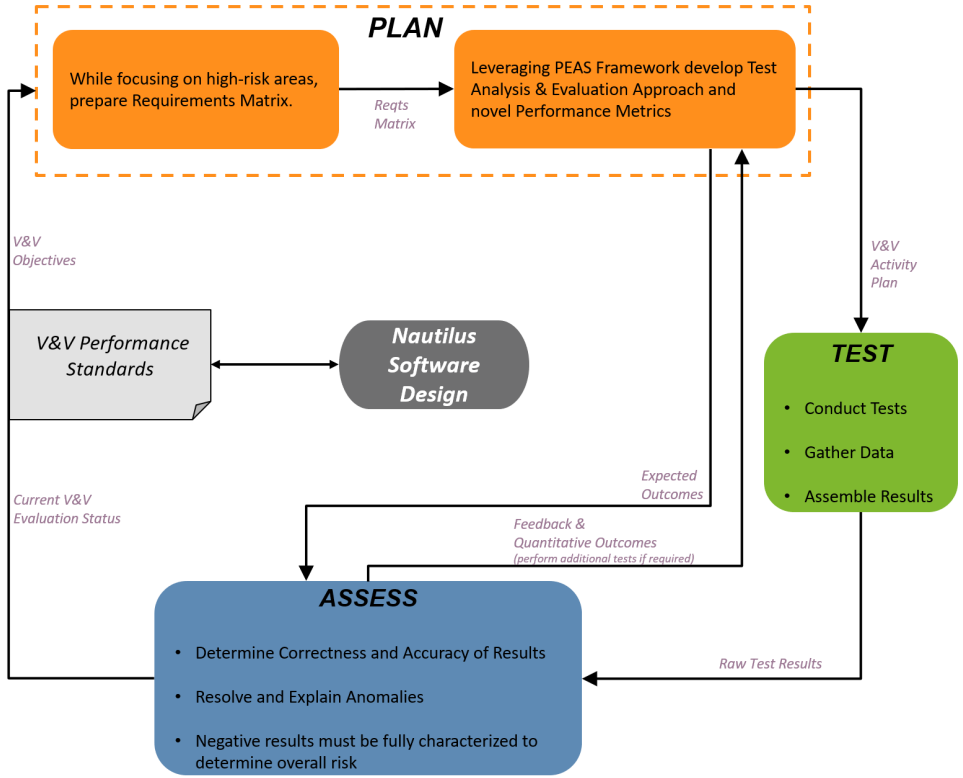
CAS introduce a paradigm shift from the traditional, one-time V&V testing of deterministic systems (e.g., spacecraft flight software) to a continuous V&V testing analysis capable of considering the non-deterministic, time-evolving learning nature of CAS:

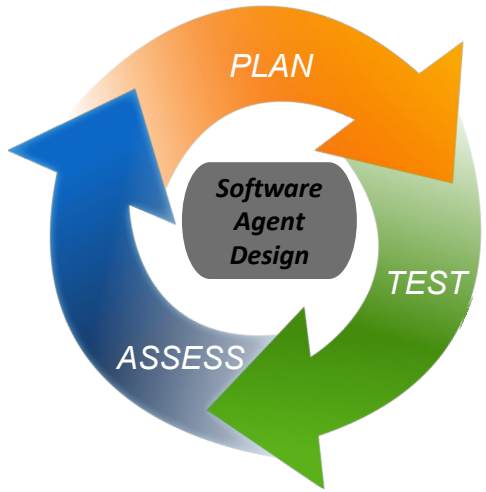
- V&V of CAS requires a deep understanding of the cognitive apparatus of the whole system.
- CAS V&V methodologies must accept and embrace uncertainty carried by the intrinsic non-deterministic nature of the systems. Novel V&V performance and testing metrics must be constructed in order to comprehensively and quantifiably capture this evolutionary modus operandi.
- To establish and sustain trust, performance metrics must be adopted from the beginning of a CAS' lifecycle. After a CAS is successfully V&V-ed and deployed, these metrics will assist with the required continuous performance monitoring of the system thus guaranteeing that performance standards met in the V&V analysis are still valid.

# A Three-Stage, Continuous, Risk-Based Approach to V&V for CAS

*An embedded, continuous V&V analysis and evaluation status*

- Stage 1 – PLAN: Identify V&V testing objectives.
- Stage 2 – TEST: Correctly perform V&V activity plan.
- Stage 3 – ASSESS: Assess V&V test results and ***report current V&V evaluation status***.

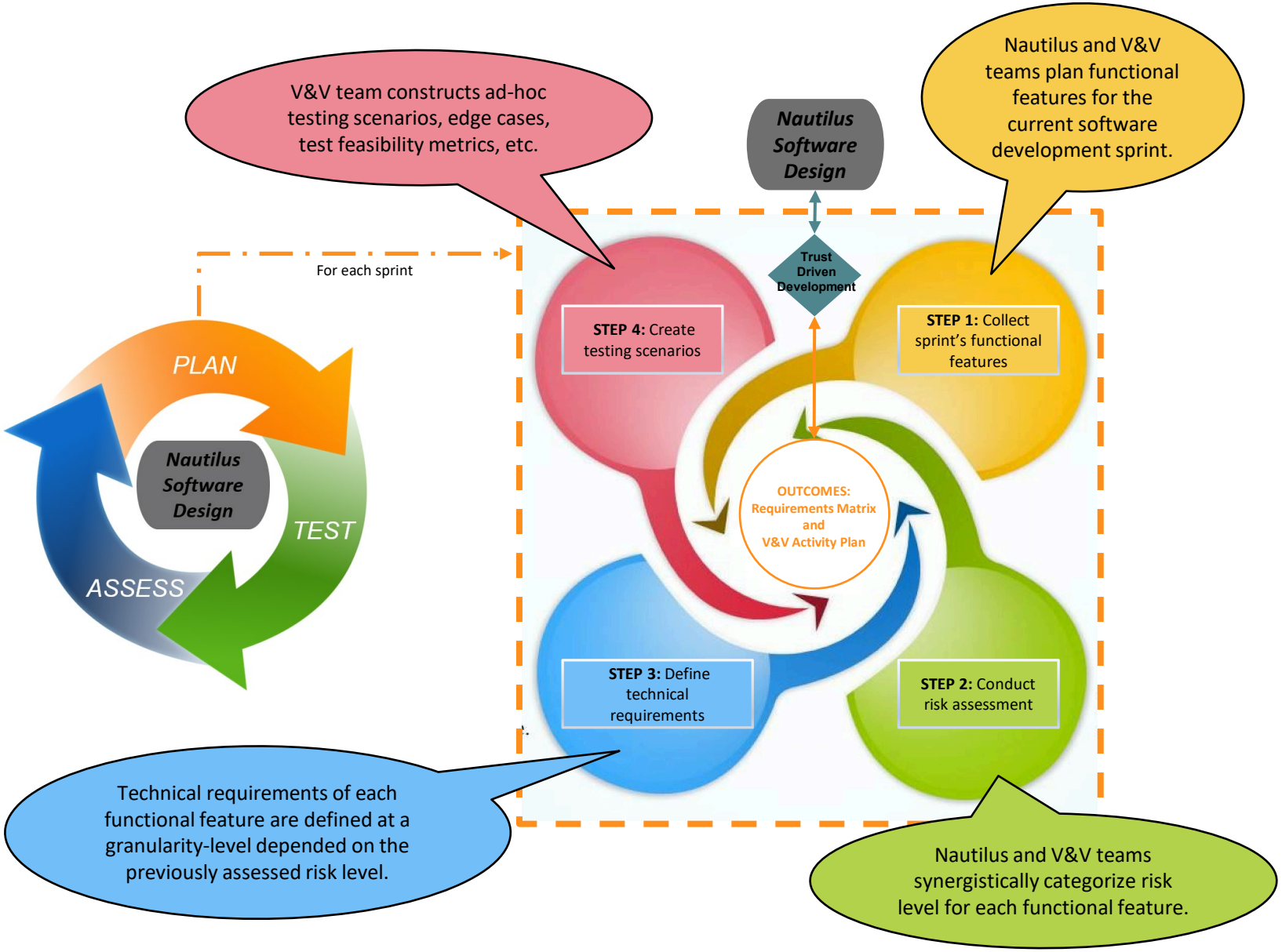



***Trust Driven Development: A Mindset shift.***
*Re-thinking of traditional V&V, quality assurance, unit testing, regression testing, etc. focused on CAS/ learning agents.*

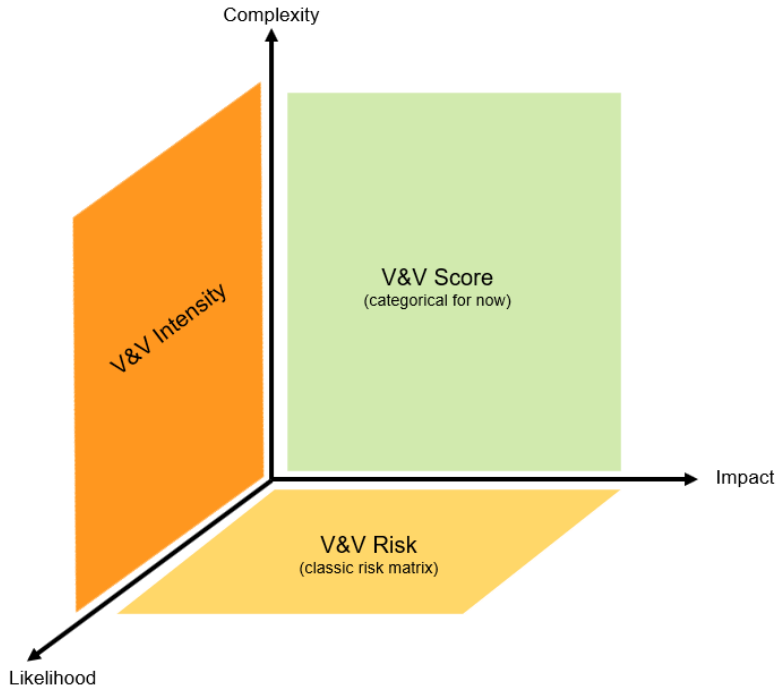
# V&V Workflow During Nautilus Development

*Nautilus follows agile software development, with functional features delivered at each sprint*



# *Implementing a Novel V&V Risk-Based Approach*

## *The CAS V&V risk cube*




***Hypothesis: Impact and Likelihood move independently in relation to Complexity for highly complex, non-deterministic systems.***

| Likelihood Level | Definition |
|---|---|
| 3 | Highly Likely; 67-100% probability of occurring. |
| 2 | Likely; 34-66% probability of occurring. |
| 1 | Not likely; 0-33% probability of occurring. |

| Impact/ Severity Level | Definition |
|---|---|
| 3 | Software element must execute correctly or catastrophic consequences (loss of life, loss of system, monetary or social loss) will occur. |
| 2 | Software element must execute correctly or the intended use (mission) of the system/software will not be realized, causing critical consequences (permanent injury, major system degradation, monetary or social impact). |
| 1 | Software element must execute correctly, or an intended function will not be realized, causing marginal or negligible consequences. |

| Complexity Level | Definition |
|---|---|
| 3 | High complexity. The system controls something or provides real-time advice to an operator to control something (e.g., turn on the computers, machines, etc.) AND its failure, delay, unintended repetition, or any other kind of malfunction could directly or indirectly cause damage or harm to anything. |
| 2 | Medium complexity. The system involves real-time processing OR any of the following: distributed processing, embedded processing, complex reasoning, interrupt-driven processing, a large number of complex interacting systems. |
| 1 | Low complexity. The system is basically a stand-alone user-driven consulting system. |

***"Example of CAS V&V Class, Score, Risk, Intensity"***

| V&V Class | V&V Score | V&V Risk | V&V Intensity |
|---|---|---|---|
| C | 1 | Low | Least rigorous |
| B | 2 | Medium | Intermediate |
| A | 3 | High | Most rigorous |

# PEAS Onion Model

*Performance Measures, Environments, Actuators, Sensors – a test and evaluation framework for CAS*

Methodology consists of an onion model of the level of details of the system, and a procedure for hierarchically identifying the performance measures based on progressive refinement of the system from the functional architecture to the individual agents:

- Onion model is a conceptualization for the development of performance measures. It illustrates that the system development process progressively exposes different layers of details of the system. Peeling of the onion is a metaphor for both the development process, and the availability of specific implementation details that can be used to identify the performance measures.
- At each level in the onion, performance measures are specified by assuming that only the outer layers are known, and the inner layers are treated as a black box.

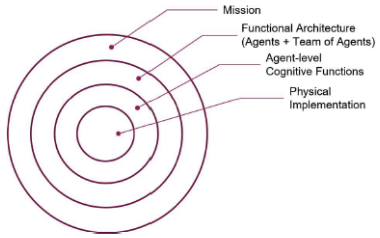



Figure 1.1: An illustration of the proposed "Onion Model" of Performance Measures

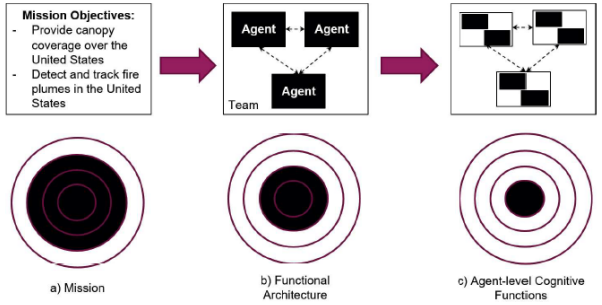



Figure 1.2: Progression in the level of detail of performance measures

***"Example of Peeling the Nautilus Onion"***

| CAS Functional Feature Goal | Agent-Level Task/Goal | Agent Sub-Task/Goal | Optional Agent Sub-Sub-Task/Goal | Associated Risks |
|---|---|---|---|---|
| Goal 1 | 1.1 | 1.1.1 | 1.1.1.1 | 1.1.1.1.1 |
| | 1.2 | 1.2.1 | 1.2.1.1 | 1.2.1.1.1 |
| Goal 2 | 2.1 | 2.1.1 | 2.1.1.1 | 2.1.1.1.1 |
| … | … | … | … | … |

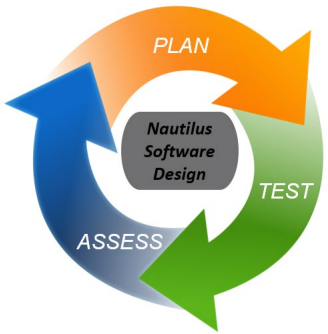
# Producing the Nautilus V&V Activity Plan

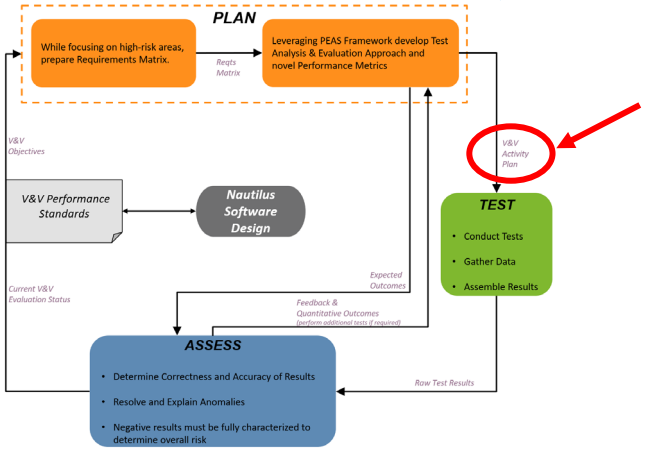
| CAS Functional Feature Goal | Agent-Level Task/Goal | Agent Sub-Task/Goal | Optional Agent Sub-Sub-Task/Goal | Associated Risks |
|---|---|---|---|---|
| Goal 1 | 1.1 | 1.1.1 | 1.1.1.1 | 1.1.1.1.1 |
| | 1.2 | 1.2.1 | 1.2.1.1 | 1.2.1.1.1 |
| Goal 2 | 2.1 | 2.1.1 | 2.1.1.1 | 2.1.1.1.1 |
| … | … | … | … | … |

***Peeling the Nautilus Onion + CAS V&V Risk Cube***

***Nautilus V&V Activity Plan***

| Task | V&V Risk Cube (axes) | | | V&V Evaluation | | | | V&V Recommendation | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Complexity | Likelihood | Impact | V&V Score | V&V Risk | V&V Intensity | V&V class | Life-Cycle Stage (i.e., risk type) | V&V Activity | V&V Methods |
| … | … | … | … | … | … | … | … | … | … | … |



# *V&V of Training Data*

- Nautilus proved to be a valuable use case for advancing research in the area of Trusted AI
  - *The extensive V&V performed on the training and test data set identified a variety of items related to data V&V attributes*
  - *The work serves as a valuable exemplar of application of Trusted AI concepts*

# Producing the V&V Activity Plan

## *Data Validation & Verification – The 13 Data V&V Attributes*

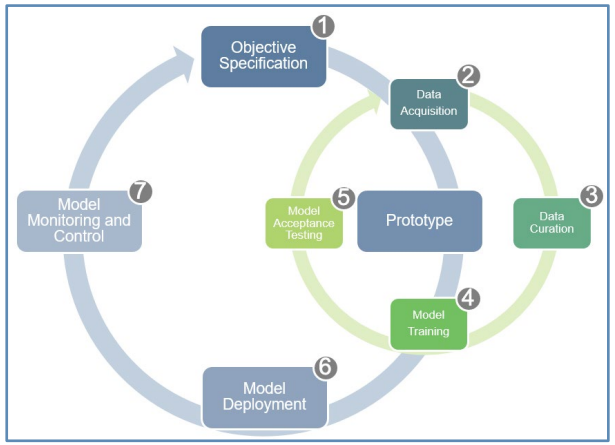
**Data Verification:** to make sure that the data is accurate.

**Data Validation:** to make sure that the data is correct.

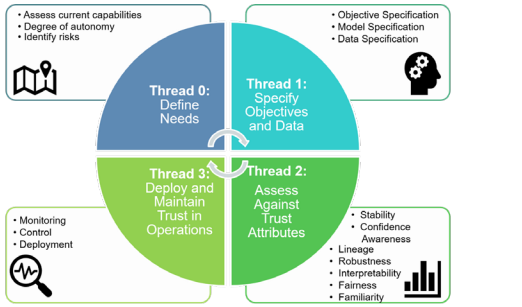
***13 Data V&V Attributes*** were identified from our literature review and through the application of our V&V Risk Cube process to the Nautilus code.

| Data V&V Attributes | |
|---|---|
| Accuracy/Validity/Correctness | ✓ |
| Currency/Timeliness/Latency | ✓ |
| Consistency/Coherence/Clarity | ✓ |
| Usability | ✓ |
| Security | X |
| Privacy | X |
| Accessibility | X |
| Availability | X |
| Scalability | X |
| Completeness/Comprehensiveness | ✓ |
| Lack of bias | ✓ |
| Coverage of the state space | ✓ |
| Relevance | ✓ |

We discovered that our data V&V attributes are closely ***related*** to the data integrity & quality concepts used in ***MLOps***.




**MLOps**




**Aerospace's Trusted AI Framework**

(From "Trust Throughout the AI Lifecycle", DATAWorks 2022, Perry, Slingerland, Spolaor, Nemerouf)

MLOps is a potential ***implementation*** of Aerospace's TAI framework.

# Data V&V Methods and Metrics

| Data V&V Attributes | Definition | Metrics | How to Calculate |
|---|---|---|---|
| **Accuracy Validity Correctness** | How accurately does each available data field represent reality? Is the information correct in every detail? | Ratio of valid data | Divide the number of invalid elements by the total number of training elements. |
| | | Number of incomplete records | Count the number of data elements that are of insufficient length for use in the system |
| | | Out of Bounds Ratio | Divide the number of data elements that are outside the field of regard by the total number of training elements |
| **Currency Timeliness Latency** | What is the probability that the data represents the values in the real world at any given time? I.e., what is the delay between an observation and its effect on the learning agent? | Data recency | Subtract current date from the training set collection data |
| | | Time distribution of data | Subtract current date from the training set collection data |
| | | Frequency | Data must be available frequently enough to meet the decision-making needs. One must define the appropriate time intervals of collecting data. |
| **Usability Consistency Coherence Clarity** | How consistent and usable is the data in format and structure within and across datasets? | Missing data ratio, NaN/NA ratio | Divide the number of blank and NaN/NA/etc. entries over the total number of entries, for each row, column, and whole dataset. Ratios over a certain predetermined threshold indicate too much missing data, rendering the dataset unusable. |
| | | Equivalency of data types across datasets | When looking for consistency across datasets, check for equivalence of datatypes between the same variables in different datasets. Divide the total number of variables that have equivalent data types by the total number of variables. |
| | | Equivalency of data within variables | When looking for consistency within a variable, check that each variable has one data format (i.e. a zip code is not saved as 0000-000 but later as 0000000). How to check for data entry correctness may depend on the specific variable. Divide the number of correctly formatted entries by the total. |
| | | Check for scaled or normalized data | Check that for numeric data, the same variables between different datasets are on the same order of magnitude or scale. Calculate and compare metrics such as mean, minimum, and maximum of the variables in question and compare; the differences should not be significant if the variables are on the same scale or order of magnitude. |

# Data V&V Methods and Metrics

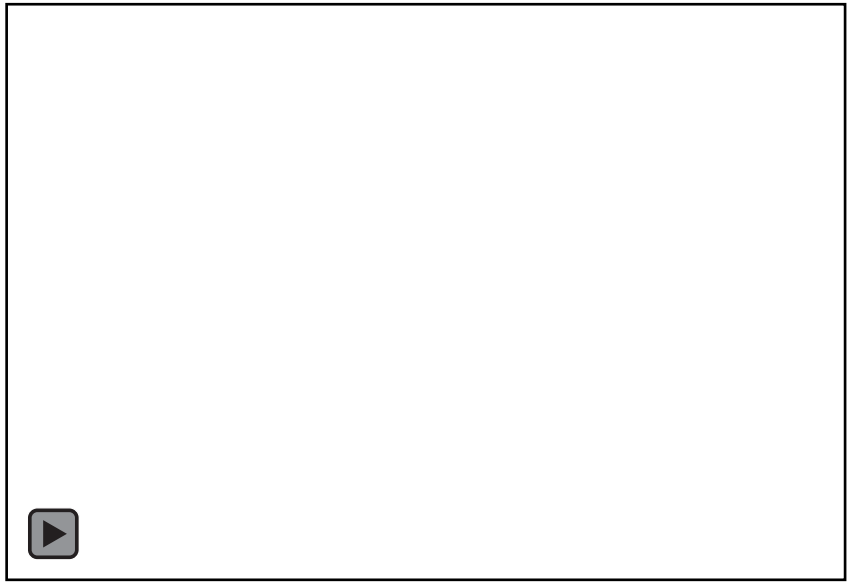
| Data V&V Attributes | Definition | Metrics | How to Calculate |
|---|---|---|---|
| **Completeness** | Of all provided records, what percentage of the available fields have a value? How comprehensive is the information? | Missing data ratio | Divide the number of blank entries over the total number of entries, for each row, column, and whole dataset |
| | | NaN/NA/etc. ratio | Divide the number of NaN/NA/etc. entries over the total number of entries, for each row, column, and whole dataset |
| **Coverage of the state space** | Is the data representative of the many possible system configurations? | Operation coverage | Determine which operations are covered in the state-space and compute how often each operation is enabled. |
| | | Min and Max values | Determine for each variable and constant a minimum and maximum value that is reached. |
| | | Range coverage | What percent of the absolute range is covered? |
| | | Value coverage for expression | This allows the user to enter an expression, which is computed in every state of the state space; all values are then displayed in a table along with the possible values, how often these values have been encountered and a witness state id, giving you one possible witness state for this value. |
| | | Number of covered values for variables | Compute for each variable, how many different values it has taken on in the state space. This is useful to diagnose state space explosion. |
| | | Precise operation coverage | Check which operations have been covered in the current state space; for those operations that have not been covered, the constraint solver is called to determine whether the operation is actually feasible given the invariant (is there a state satisfying the invariant which enables the operation). If an operation is infeasible, it can never be reached (unless we reach a state violating the invariant). |
| | | Temporal variability within data fields | Capture patterns at various time scales (daily, weekly, monthly, etc.) |

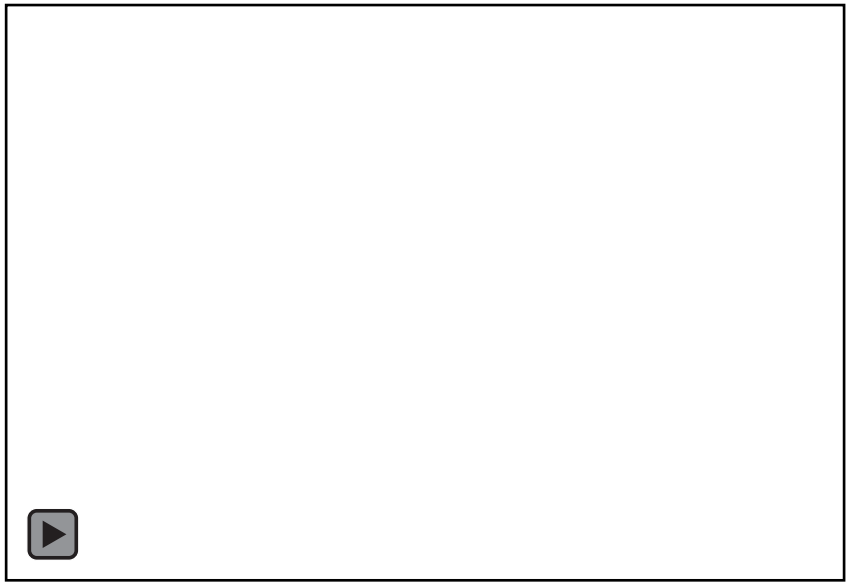
# Data V&V Methods and Metrics

| Data V&V Attributes | Definition | Metrics | How to Calculate |
|---|---|---|---|
| **Relevance** | The level of consistency between the content of data and the user's areas of interest. In other words: the extent to which data answers of gives insight into the question of the individual user. | Do the data meet the fundamental needs for which they were collected? Can the data be used for additional purposes? | Percent of entries distinguished by usage. |
| **Lack of bias** | Sample bias occurs when a dataset does not reflect the realities of the environment in which a model will run. In other words, a bias in which a sample is collected in such a way that some members of the intended population have a lower or higher sampling probability than others. | Sample bias | T-test. Test for appropriate random sampling of population subgroups. |
| | Exclusion bias is most common at the data preprocessing stage. Most often it's a case of deleting valuable data thought to be unimportant. However, it can also occur due to the systematic exclusion of certain information. | Exclusion bias | Ratio of features used vs. features available. |
| | This type of bias occurs when the data collected for training differs from that collected in the real world, or when faulty measurements result in data distortion. Measurement bias happens from the way we choose, utilize, and measure a particular feature. | Measurement bias | Diagnostic plots of features available, features extracted, features used for training. |
| | This is a kind of measurement bias and is common at the data labeling stage of a project. Recall bias arises when you label similar types of data inconsistently. This results in lower accuracy | Recall bias | Diagnostic plots of data vs. extracted features. |
| | Selection biases occur when looking at samples that are not representative of the population. In other words, is the bias introduced by the selection of data for analysis in such a way that proper randomization is not achieved, thereby failing to ensure that the sample obtained is representative of the population intended to be analyzed. | Selection bias | Diagnostic plots identifying representative populations categories vs. amount of data distribution/histograms/density distribution. |

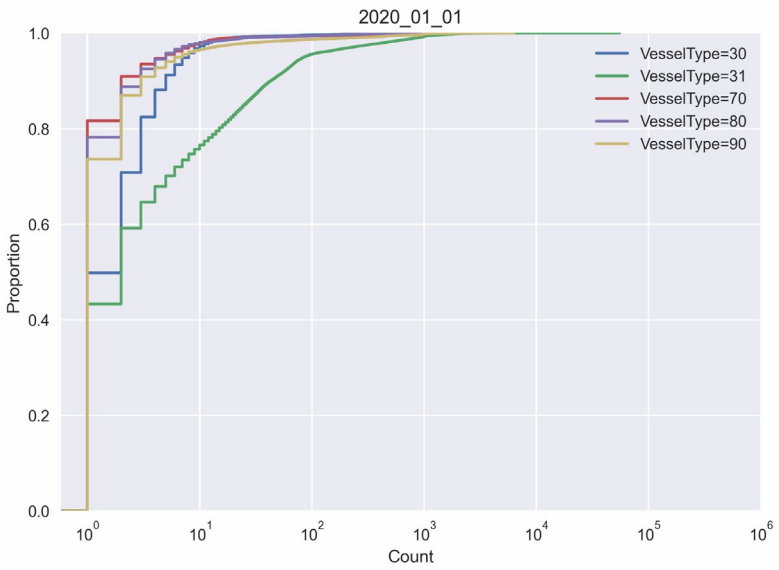
# Nautilus V&V Results

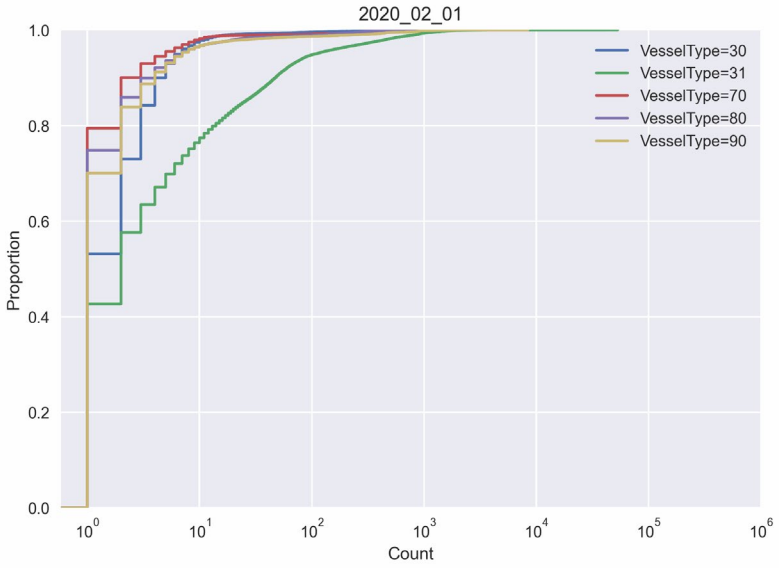
*Count and Variation Analysis of Vessels of Interest & Cumulative Geographic Density Analysis*





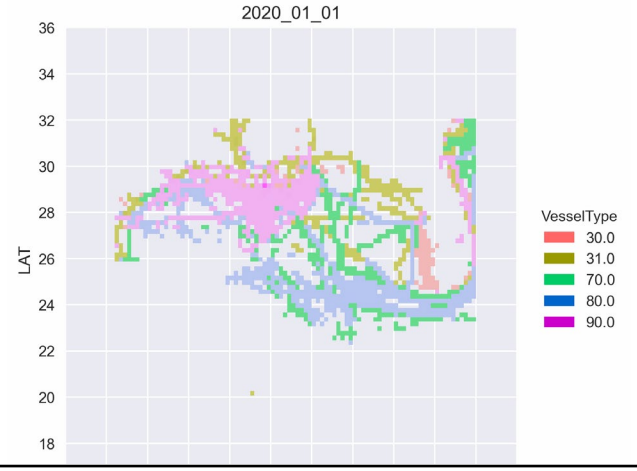
| Vessel Code | Vessel Type |
|---|---|
| 30 | Fishing |
| 31 | Towing |
| 70 | Cargo |
| 80 | Tanker |
| 90 | Other |



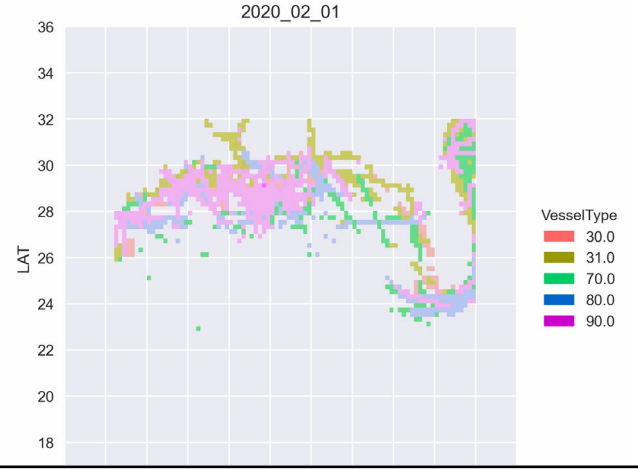



# Nautilus V&V Results

*Geographic Density Analysis & Spatial Coverage*




| Vessel Code | Vessel Type |
|---|---|
| 30 | Fishing |
| 31 | Towing |
| 70 | Cargo |
| 80 | Tanker |
| 90 | Other |



# *Novel Metrics for Measuring Adaptivity*

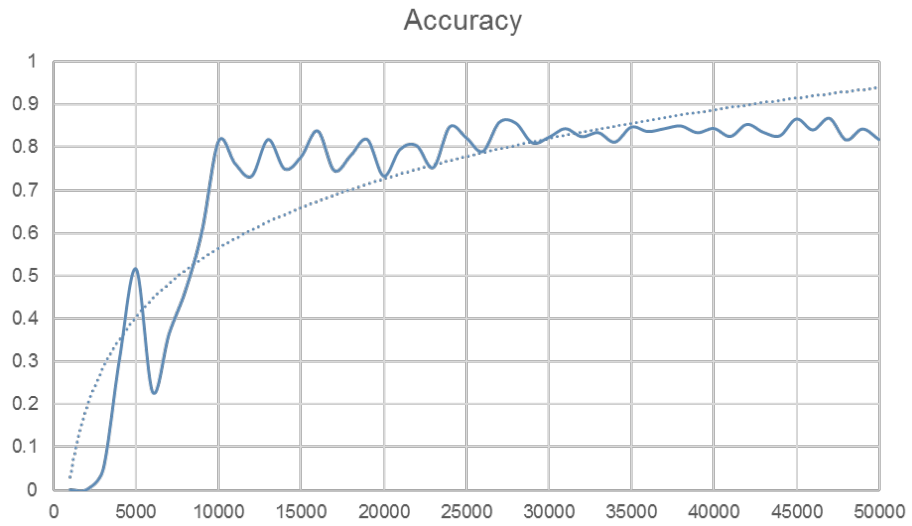
Experiments conducted to assess Nautilus' adaptivity under static and in-situ learning conditions provide novel key metrics for assessing future intelligent agents and demonstrate a quantifiable measure of an agent's capacity to learn to respond to environmental changes

# *Learning Functionality*

*Measuring the impact of in-situ learning – Experiment 1*

- Although the Gryphon interface does not currently support in-situ learning for Nautilus, experiments were done to quantify the impact in a standalone framework

  - Process:
    - *Start with an empty model*
    - *Predict each new data point*
    - *Measure and store performance result*
    - *Train on the new data point*
    - *Repeat*




- Lesson Learned:
  - *This experiment achieved 90% of Nautilus' peak performance after ~10K training examples*
  - *This is about 3.5 (!) minutes of data from just after midnight 1/1/2020*
  - *Faster than logarithmic improvement*

***An In-situ learning capability can rapidly bootstrap a model to improve its performance***

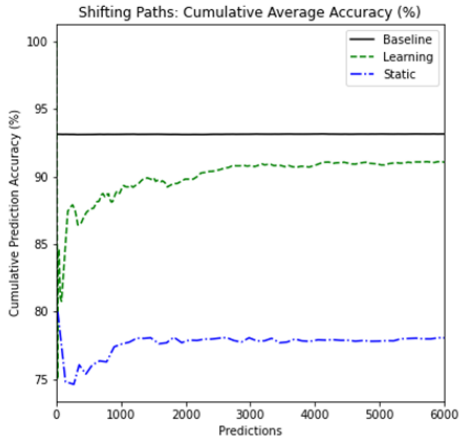
# Learning Functionality

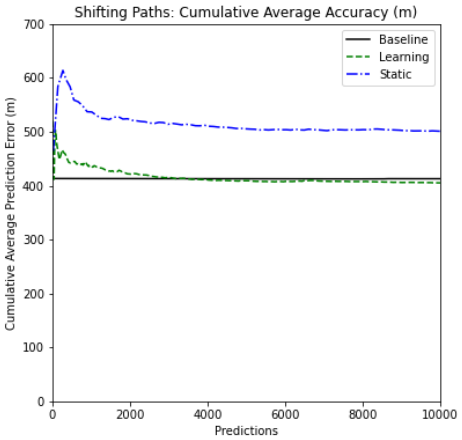
*Measuring the impact of in-situ learning – Experiment 2*

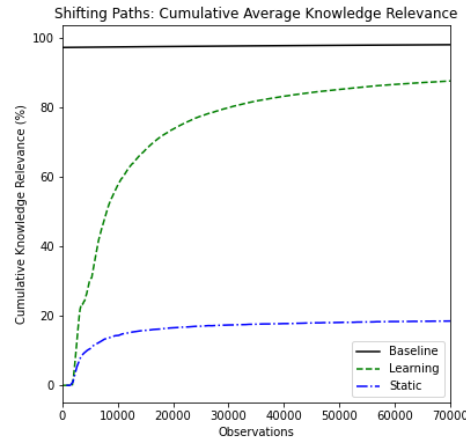
- In addition to measuring the impact of an agent in continuous training mode (Experiment 1), the Nautilus team devised an experiment to measure Nautilus' ability to respond to ships moving in different tracks


**Experiment Methodology:**

1. Train Nautilus on one month of data (January 2020)
2. Test Nautilus on one month of data (January 2020).
   a. These results will serve as the baseline for comparison – Use the last four results to create a line labeled *Baseline*
3. Apply the 1Km shift to the dataset and save (January 2020 Shifted)
4. Test Nautilus using shifted dataset
   a. Output measurements for every 1k samples
   b. These results will be called the *Static Model* results
5. Test Nautilus using shift dataset again, but leave the agent in training mode to allow in-situ learning
   a. Output measurements for every 100 samples
   b. These results will be called the *Learning Model* results



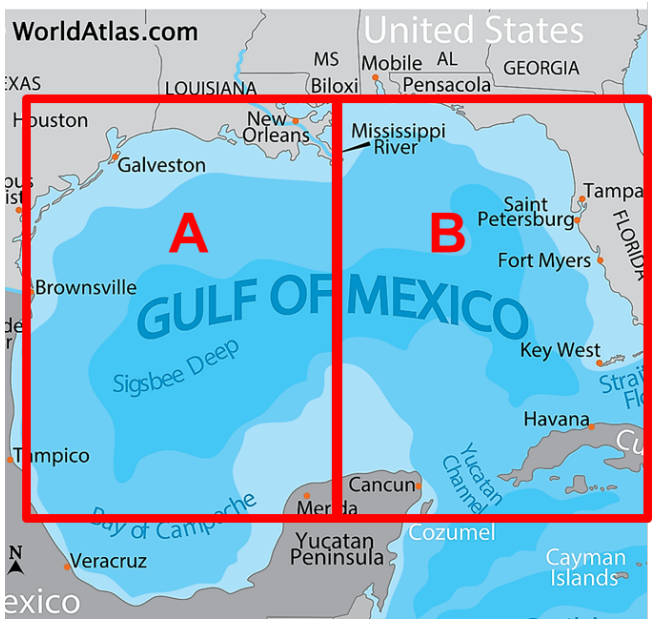




***The learning model demonstrated a resiliency in its ability to adapt to changing target behaviors***

# Learning Functionality

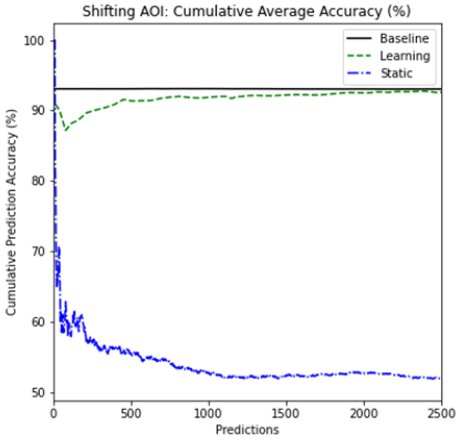
*Measuring the impact of in-situ learning – Experiment 3*

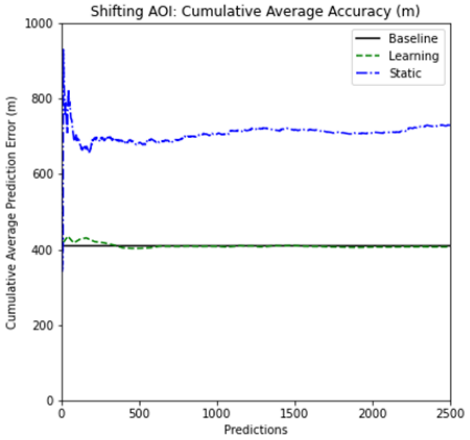
- In addition to measuring the ability of the agent to adapt to target behavior (Experiment 2), the Nautilus team devised an experiment to measure Nautilus' ability to operate in different areas of interest (AOI)

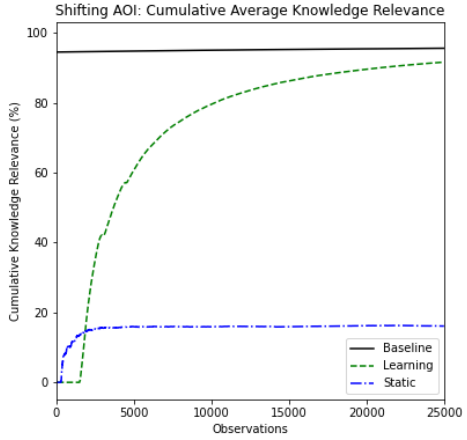



**Experiment Methodology:**

1. Train Nautilus on one month of data in Area A
2. Test Nautilus on one month of data in Area A
   a. These results will serve as the baseline for comparison – Use the last four results to create a line labeled *Baseline*
3. Test Nautilus using one month of data in Area B
   a. Output measurements for every 100 samples
   b. These results will be called the *Static Model* results
4. Test Nautilus using one month of data in Area B, but leave the agent in training mode to allow in-situ learning
   a. Output measurements for every 100 samples
   b. These results will be called the *Learning Model* results



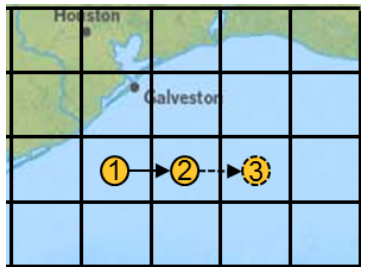




***As expected, the model had a slower recovery in this experiment, but the learning model demonstrated a resiliency in its ability to adapt to changing AOI's***

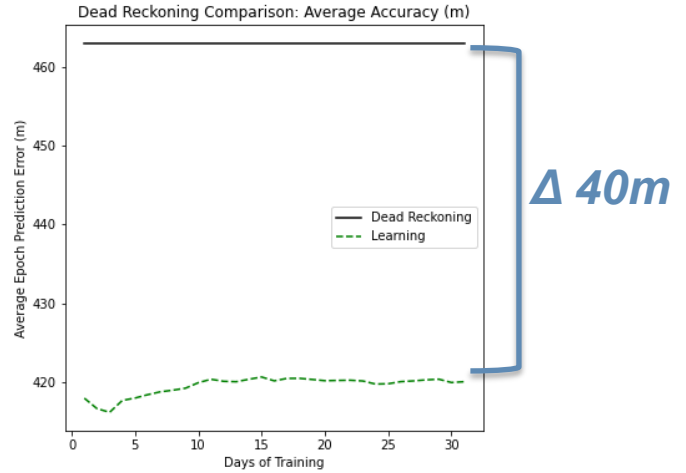
# Learning Functionality

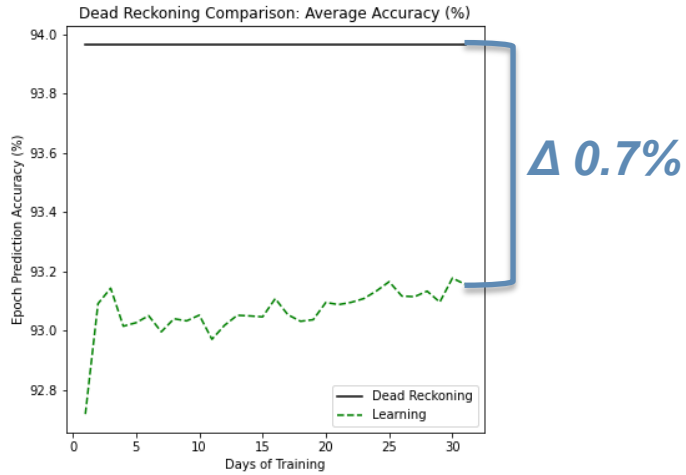
## *Comparing Nautilus to a Dead Reckoning Model*

- Given 2 prior observations, the dead reckoning algorithm returns a prediction that assumes the vessel will continue at its calculated heading and velocity for a given prediction interval (i.e., 50 seconds)
  - ***Lesson's Learned:***
    - A 50 second prediction interval was not long enough to overcome the impact of the grid size implemented in the Nautilus algorithm.
      - *Note, the performance difference is not significant between the two models*
    - Water vehicle maneuvers are more gradual than ground vehicle maneuvers. The movement of ocean-going ships in the Gulf of Mexico is very predictable, not dynamic and largely driven by physics.
    - An RL-base algorithm can learn physics-based movement to a high degree of accuracy, without the limitation of only predicting inertia-driven movement




- Given a different testing scenario, dead reckoning would meet with limitations:
  - Would not work well for agile, rapidly moving vehicles
  - Longer prediction intervals experience poorer accuracy
  - Unable to respond to changing conditions or behaviors




***Nautilus demonstrated nearly identical performance to a simple dead reckoning model and would likely perform better given longer prediction intervals***

# *Conclusions*

- The V&V effort served to support the acceptance of the Nautilus prototype as built and delivered to the Government. The V&V report verified that all the identified goals and requirements of Nautilus were met, and it validated that the right algorithm was chosen in concert with valid training data.

- CAS introduce a paradigm shift from the traditional, one-time V&V testing of deterministic systems (e.g., spacecraft flight software) to a continuous V&V testing analysis capable of considering the non-deterministic, time-evolving learning nature of CAS.
  - *Trust Driven Development: A Mindset shift. Re-thinking of traditional V&V, quality assurance, unit testing, regression testing, etc. focused on CAS/ learning agents.*

- Nautilus proved to be a valuable use case for advancing research in the area of Trusted AI
  - *The extensive V&V performed on the training and test data set identified a variety of items related to data V&V attributes*
  - *The work serves as a valuable exemplar of application of Trusted AI concepts*

- The experiments conducted to assess Nautilus' adaptivity under static and in-situ learning conditions provide novel key metrics for assessing future intelligent agents and demonstrate a quantifiable measure of an agent's capacity to learn to respond to environmental changes.


Questions?

# Backup

# *Status Summary - Prototype QA Related Activities*

*Code Analysis – Peer Review*

- In software development, peer review is a best practice where the code author's colleagues review the code in detail to evaluate technical content, accuracy and clarity.
  - *According to the Capability Maturity Model, conducting a peer review aimed at detecting and correcting software defects prevents larger costs and downstream defects in field operations.*

- The peer review of the Nautilus code was conducted by subject matter experts within the lead software developer's department. All comments were documented in a comment review matrix by the review team for the Nautilus developers to review and adjudicate.

| Comment Category | Category Description | Count of Comments |
|---|---|---|
| Readability | Comments in this category address items associated with a 3rd party's ability to read and understand the processes within the code. | 83 |
| Usability | Comments in this category address items associated with a 3rd party's ability to effectively operate the code | 9 |
| Accuracy | Comments in this category address items that need addressed or verified to ensure the code is operating correctly. | 2 |

***The volumes noted by category in the table are at a reasonable level given the size of the software and its rapid prototype nature***

# Status Summary - Prototype QA Related Activities

## Code Analysis – Static Code Analysis

- The purpose of a static code analysis is to identify any issues that may increase the risk of future maintenance, development or upgrades associated with the code
  - *Code analysis has proven to be an effective tool for communicating risk due to software issues and assesses risks for maintainability, understandability, reliability, and testability.*

| Metrics | Description |
|---|---|
| Program Reliability Risk | Based on Cyclomatic Complexity |
| Probability of a Bad Fix Risk | Based on Cyclomatic Complexity |
| Maintainability Risk | Based on Essential Complexity and Module Length |
| Maintainability/Extensibility Risk | Based on Class Coupling |
| Maintainability/Reliability Risk | Based on Class Cohesion |
| Maintainability/ Understandability Risk | Based on Comment-to-Code Ratio and Code Nesting Level |
| Runtime Stability Risk | Based on Code-Check Items and Error Handling |
| Code Quality | Based on Code-Check items, Code Smells, & Technical Debt |
| Design Vulnerabilities | Based on Public Declarations |
| Architectural Degradation | Based on Dependency Structure Matrix |
| Releasability Risk | Based on Intellectual Property Markings |

**Static Code Analysis Results:**

- ***Essential Complexity: Predictor of Maintainability Risk***
  - Findings: The essential complexity metric analysis showed approximately 49% of the code and 20% of the modules have a high maintainability risk
  - V&V Assessment: Given that Nautilus is considered a research prototype, the results of this risk are of minimal concern.
- ***Code Quality Analysis: Code Smells as a Maintenance Risk***
  - Findings: A small number of code smells were found, indicating that it may be more difficult than optimal to facilitate updates. Although some smells were found, the overall rating for code smells is an A.
  - V&V Assessment: The volume and estimated impact of the code smell finding are minimal and not of significant concern
- ***Intellectual Property Scan: Releasability Risk***
  - Finding: One proprietary marking was found in the code.
  - V&V Assessment: This risk is of no concern.
- ***Code Quality Analysis: Technical Debt as a Maintenance Risk***
  - Finding: The Technical Debt for Nautilus is estimated to be ~3 hours and the Reliability Remediation is estimated to be 0 hours. Overall code quality is rated as an A.
  - V&V Assessment: The technical debt and maintenance risk is considered very low for Nautilus and no significant concerns are present.

***The static code analysis of Nautilus found no significant concerns***