SYM-AM-21-058

Excerpt from the Proceedings of the Eighteenth Annual Acquisition Research Symposium

Interdependence Analysis for Artificial Intelligence System Safety

May 11–13, 2021

Published: May 10, 2021

The research presented in this report was supported by the Acquisition Research Program of the Graduate School of Defense Management at the Naval Postgraduate School.

Interdependence Analysis for Artificial Intelligence System Safety

Bruce Nagy is a Research Engineer at the Naval Air Warfare Center, Weapons Division at China Lake. His research focuses on advanced game theory techniques, artificial intelligence, and machine learning applications for tactical decision aids. Nagy has earned four degrees: one in mathematics, two in electrical engineering, and one in biology from The Citadel and the Naval Postgraduate School. He led the development of advanced algorithms and metrics that resolved national defense issues in satellite communications for the DoD. At UCLA, during postgraduate work, he investigated modeling brain stem communication with muscle groups at the cellular level, in cooperation with the NIH. [bruce.nagy@navy.mil]

Scot Miller is a Faculty Associate Researcher at the Naval Postgraduate School. Prior to that, he served 29 years in the Navy as a P-3 pilot and Information Professional. He holds bachelor's and master's degrees in Operations Research from the U.S. Naval Academy and Naval Postgraduate School, and other graduate degrees from the University of West Florida and the Naval War College. [scot.miller@nps.edu]

Abstract

Engineers responsible for evaluating tactical and weapons systems for system safety will need a new approach for evaluating emerging artificial intelligence (AI)-enabled systems, since these systems leverage machine learning (ML) techniques. For many reasons, ML algorithms are often difficult to diagnose for safety purposes. For instance, they did not lend themselves easily to codebase inspections, thus necessitating the reduction in "autonomy" of the ML-enabled component. By modifying Interdependence Analysis (IA) techniques, a more rigorous approach to evaluating AI/ML-enabled weapons can be found. The IA process produces a rigorous exploration based on observability, predictability, and direct ability, highlighting the key requirements that encapsulate all interactions between human and machine. This paper explores using IA to define the interaction requirements for human–machine teaming, employs those results to identify key critical functions, and leverages those findings to reveal how "autonomy" reduction might be employed.

Introduction

Artificial intelligence (AI) increasingly is used in many regimens. When AI is used in weapons systems, are there new considerations the tester, certifier, and operators should consider? If so, how should they handle them? Let us define AI. AI is "the theory and development of computer systems able to perform tasks that normally require human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages" ("Artificial Intelligence," 2021).

Johnson (2021) describes two types of AI. The first is handcrafted knowledge systems that use traditional rules-based software to codify subject matter knowledge of human experts into a long series of programmed rules (Johnson, 2021). These are simply just what we have always called computer applications. Various test and certification agencies execute well-developed procedures for assessing these applications and are not considered.

The second kind of AI becoming more prevalent is machine learning (ML) and this demands investigation. In ML, test data trains an algorithm to identify something or some pattern. Once the system has learned, it is ready to identify similar or the same things or patterns from more incoming data. There are variations on exactly how this works. One important variation is that if one is trying to find a specific thing or pattern in the real world, the system must have "seen" that data in the training data set.

These types of ML capabilities have improved accuracy, often in many cases achieving 90% success in identifying the object. That is a great accomplishment for the ML algorithm, if it is designed to recommend possible work shoes when one is shopping for new shoes. However, if the algorithm is distinguishing between a church steeple, the tower of a hospital, and a medium range ballistic missile launcher, 90% does not inspire user confidence, when destroying the missile launcher is the goal. Moreover, the ML algorithms work in mysterious ways, processing a myriad of complex mathematical equations, meaning it is often unrealistic to determine how the algorithm succeeds. For ensuring the safety of weapons that include ML-enabled components, evaluating ML functions in detail is important. This paper identifies the potential challenges of employing ML and how human–machine teaming engineering design techniques contribute to a rigorous ML system safety assessment.

Safety Failure Modes of Artificial Intelligence Machine Learning

The Navy organization responsible for certifying weapons as system safe is the Naval Ordnance Safety and Security Activity (NOSSA). They employ the MIL STD 882E as their rule book (DoD, 2012), and the Joint Software System Safety Guide as their compass (Joint Services-Software Safety Authorities, 2016). As part of that review, evaluation, and certification process, they conduct hazard analyses of the weapons system under test and associated supporting elements. Their intent is to identify all possible outcome likelihoods and their consequences for every system function.

An example from the previous section highlighted a 10% likelihood that the AI ML algorithm would identify a hospital or church as the target, with the loss of many innocent lives, waste of an otherwise good weapon, and a black eye for the United States. That is just one example of possible AI ML failures. Others are listed in Table 1 (Faria, 2017). They break into three categories: system failure, human–machine interaction issues, and active sabotage. Examples of each might be: 1. ML training data set has inherited biases that skew results; 2. humans assume ML algorithm is always right and do not apply critical thinking to results; and 3. an adversary manipulates the training data set.

TABLE 1. EXAMPLES OF AI SYSTEM FAILURE MODES (Faria, 2017).

Failure Category	Failure Mode Examples
System Produces Faulty/Poor Decision Recommendation	Biased outcomes/predictions
	Skewed outcomes/predictions
	Uncertain outcomes/predictions
Human–Machine Operation Issues	Operators have lack of trust in the system
	Operators are overly trusting (overreliant) in the system
	Operators ignore the system
	Operators misunderstand the system recommendations/predictions
	Operators introduce errors into the system
System Under Attack (Cyber attack)	System is overtaken by adversary/adversary is controlling system
	System and its outcomes are corrupted by adversary
	Adversary jams or shuts down system
	Adversary gains access to system; decision information/knowledge is compromised

The consequences of these failures depend on which part of the kill process of the weapons was affected. Table 2 provides a functional view of general weapons.

TABLE 2. GENERAL WEAPONS FUNCTIONS

Functions	Details
Transport	Truck, Rail, Ship, Aircraft
Storage	Bunker, Warehouse, Environment
Load on Platform	Pier side, Unrep, Cranes, Vertrep
Maintenance	Power, BIT, Lubricants, Fueling
Readied	Power on, Check out
Placed on Launcher	Movement, Rails, Hoists, Handled
Launched	Fire, Released, Sent
Navigate	Waypoints, Terrain
Avoid	Terrain, Weather, ESM
Deceive	ECM, ESM, Cloaking, IR
Identify	Target Recognition, Human Input, Sense Laser Illumination
Fuze/Arm	Timing, Mechanical, Human Input, Sensing
Effects	Detonation, Dispersal, Virus, Programs

Observation shows that many of the functions in the table could be supported by ML tools. Navigation and target recognition could be aided by an ML computer vision tool, as well as avoiding other platforms or specific topographical entities. It stands to reason that NOSSA needs to consider all of these failure modes for each weapon function that uses a ML component.

Each failure mode is comprised of possible root causes, as seen in Table 3 (Johnson, 2021). With 31 root causes and at least 15 functions, there are at least 465 possible failure combinations to assess in NOSSA's failure analyses.

TABLE 3. EXAMPLES OF ROOT CAUSES OF AI SYSTEM FAILURES (Johnson, 2021)

Type of Root Cause	Root Cause Examples
Issues within the training datasets	Biased training datasets
	Incomplete training datasets
	Corruption in the training datasets
	Mis-labeled data
	Mis-associated data
	Lack of rare examples—data doesn't include unusual scenarios
	Unrepresentative datasets
Issues with the process of data validation	Poor data collection methods
	Poor data validation methods
	Improper data validation criteria
	Insufficient data validation
Issues with the ML algorithms	Underfitting in the model—when the model does not attain sufficiently low error on the training data
	Overfitting in the model—when the model presents very small error on the training data, but fails to generalize to new data
	Cost function algorithm errors—when the trained model is optimized to the wrong cost function
	Wrong algorithm—when the training data is fit to the wrong algorithmic approach or mathematical model
Issues with the operational datasets	Uncertainty/error in the operational datasets (Epistemic uncertainty)
	Corruption in the operational data
	Introduction of datatypes that the AI system is not designed to handle
	The pace of the situation overwhelms the human–machine decision process

Type of Root Cause	Root Cause Examples
Operational complexity	The decision space overwhelms the decision process (the number of options is too large or a viable option does not exist)
Operator trust issues	Lack of explainability
	Lack of confidence
	Overreliance
	Insufficient operator training or experience with system
Operator induced error	Inverse trust issues in which the AI system loses "trust" in the human operator or identifies operator problems
	Operator misuses the system accidentally or intentionally
	Operator fails their part in the decision process accidentally or due to being overwhelmed, negligent, or confused
Adversarial attacks	Hacking
	Deception
	Inserting false or corrupt data
	Gaining control of the AI system

A weapon uses radar computer vision to identify a turn point as part of its navigation. A possible root cause for failure might be mislabeled data. In this case, the turn point is mislabeled as 1,500 feet in elevation, when it should be labeled 1,500 meters in elevation. The likelihood that the weapons might fly into the terrain is high, with the consequence that the weapons effects are not delivered.

Here is a more insidious example. A weapon requires a system update, for instance, a new load of training data. The weapon is already loaded onboard a ship, so the ship's weapons maintenance team is assigned the task. Their communications connection to the shore base where the file will be downloaded is sketchy. Despite best efforts, the download is interrupted several times due to latency and signal jitter (common occurrences underway). Even the checksums that ensure a good transmission are misplaced, leading the ship to think it has a good download, when it really does not. Now there is an unknown likelihood the weapons could malfunction, and unknown consequences, and no way to check.

Understanding every function in the life cycle of a combat or weapons system is critical. Existing procedures already account for the vast number of functions that are not related to any ML support and are not addressed in this paper. However, there needs to be a way to determine how ML-related techniques impact the weapons and associated operators.

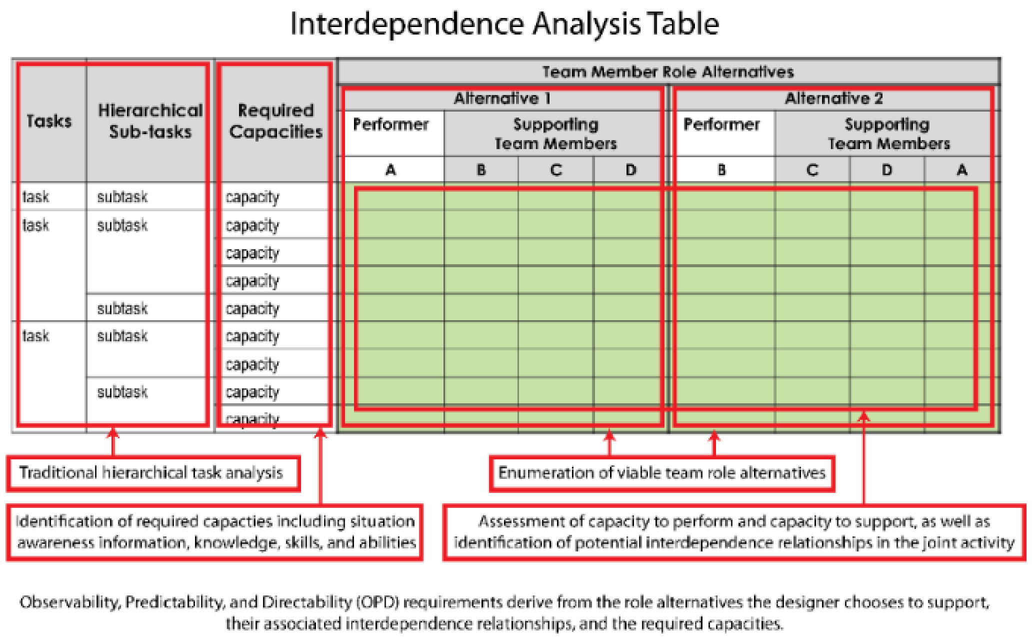
Tables 1–3 list many ways that ML functions cause safety issues. If their likelihoods and consequences are significant, these functions can be designated critical, which by the MIL STD requires applying a scrupulous code check process. This is impractical for ML functions, so reducing the criticality of the ML functions remains paramount.

Criticality is associated with "autonomy," which in this case means that the function will operate automatically without human intervention. Thus, if the function can be modified to include human involvement, then the criticality is not rated as high, and no code check is required, which then resolves the evaluation conundrum of critical ML functions. One proven way to investigate human–machine interaction is through a system engineering technique called Co-Active Design, especially a key component called interdependence analysis (IA). Does applying IA to the ML functional evaluation help?

Co-Active Design

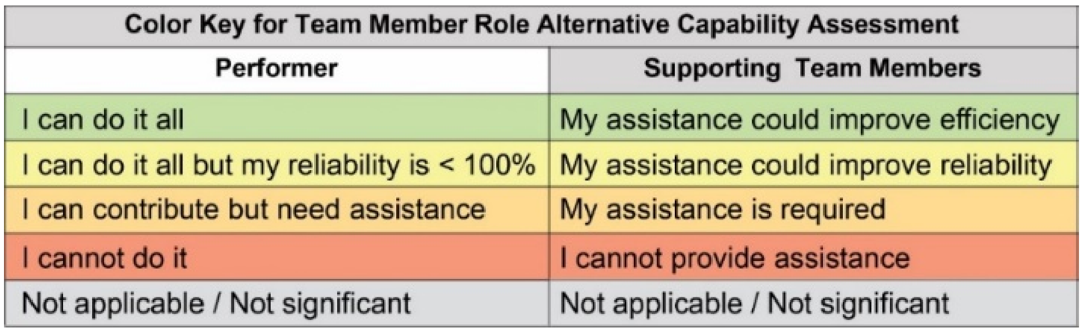
The purpose of IA is understanding how people and an agent (in our case the ML function) can effectively team by identifying and providing insight into the potential interdependent relationships used to support one another throughout an activity (Johnson et al., 2014). One of the IA tool's strengths is that it can guide the initial design and anticipate how the tool or system will be used by the warfighter. In this specific case, though, NOSSA is most interested in how humans can be involved in the application of a ML algorithm so as to ensure that it is not deemed a critical function. As we have seen, critical functions require a software code review, which is problematic, since advanced ML algorithms are complex, especially in the mathematics involved, and a software code review is not practical.

The IA tool is in the form of a table, as shown in Figure 1. IA breaks the planning process down hierarchically into specific tasks. Those tasks are analyzed to determine the necessary capacities needed to conduct each function, using cognitive task analysis techniques (Annett, 2003; Bolte et al., 2003; Chapman et al., 2009; Crandall et al., 2006). Once capacities are identified, the next step is assessing each performer's (ML component or human) ability to provide that capacity. The assessment process uses a color-coding scheme, as shown in Figure 2. The color scheme is dependent on the type of column being assessed. The color assessment is subjective, but the process compels the analyst to carefully consider the algorithm and human interactions. Under the "performer" column, the colors are used to assess the individual's capacity to perform the activity specified by the row. The green color in the "performer" column indicates that the performer can do the task. Yellow indicates less than perfect reliability. Orange indicates some capacity, but not enough for the task. Red indicates no capacity.


Interdependence Analysis Table
Tasks
Hierarchical Sub-tasks
Required Capacities
Team Member Role Alternatives
Alternative 1
Alternative 2
Performer
Supporting Team Members
A
B
C
D
task
subtask
capacity
Traditional hierarchical task analysis
Enumeration of viable team role alternatives
Identification of required capacties including situation awareness information, knowledge, skills, and abilities
Assessment of capacity to perform and capacity to support, as well as identification of potential interdependence relationships in the joint activity

Observability, Predictability, and Directability (OPD) requirements derive from the role alternatives the designer chooses to support, their associated interdependence relationships, and the required capacities.

Figure 1. Explanation of the Different Areas of the Interdependence Analysis (IA) Table (Johnson, 2014)



Color Key for Team Member Role Alternative Capability Assessment	
Performer	Supporting Team Members
I can do it all	My assistance could improve efficiency
I can do it all but my reliability is < 100%	My assistance could improve reliability
I can contribute but need assistance	My assistance is required
I cannot do it	I cannot provide assistance
Not applicable / Not significant	Not applicable / Not significant

Figure 2. Color Dey for Team Member Role Alternative Capability Assessment (Johnson, 2014).

Once completed, careful analysis of the results displays or infers several key design inputs for the human–machine teaming involved in the robot delivery process:

1. From the color schemes, analysts determine how many possible paths exist between the users and the systems to accomplish the tasks. This identifies areas where multiple paths are availability (improving reliability), and where single points of failure may compromise success (and require further design attention). And indeed, may be showstoppers to the NOSSA evaluators.
2. Analysis shows where the system is required to either ensure effectiveness or improve efficiencies. Teaming is achieved by practicing three actions between the performer and the supporting team members: staying observable, predictable, and directable (OPD). Where there are opportunities for a team member to assist the performer, and vice versa, engineers and designers can brainstorm ways to achieve that teaming through the lens of OPD. In this case, NOSSA evaluators seek ways for the AI-enabled function to be either supported by or monitored by humans. This reduces "autonomy" and the criticality of the system function.
3. Combining these first two results helps designers and developers prioritize efforts on the machine's development. Which capacity executed by the performer adds the most value? This establishes a context for organizing work efforts and resources. From a NOSSA perspective this analysis helps focus on potential ML-enabled trouble areas. As described in Table 1, three areas of potential issues exist: system failure, human–machine interaction issues, and active sabotage. As evaluators use the IA, when they evaluate the OPD between the ML algorithm and the humans, they should also consider these three possible failure mode areas, associated possible root causes from Table 3, and which general functional area is impacted from Table 2.

Applying IA to ML Safety Use Case

Nagy (2021) created an unclassified use case for exploring system safety aspects of ML-enabled systems (see Figure 3). We use that scenario as a use case for the IA. The results are depicted in Table 4. In this scenario, a truck is loaded with two robots, which are then driven to a point, then unloaded. For simplicity, assume predecessor functions such as storage and maintenance (from Table 2) have no ML components. The robots proceed via another route to deliver packages to one or two recipients. As will be introduced in the analysis, at several stages in the process ML-enabled techniques are used to conduct activities or make decisions. Understanding how the human–machine teaming is associated with those activities and decisions is critical to the evaluation process that NOSSA uses. Many of the other functions normally considered in an IA were skipped as not relevant to the AI safety discussion.

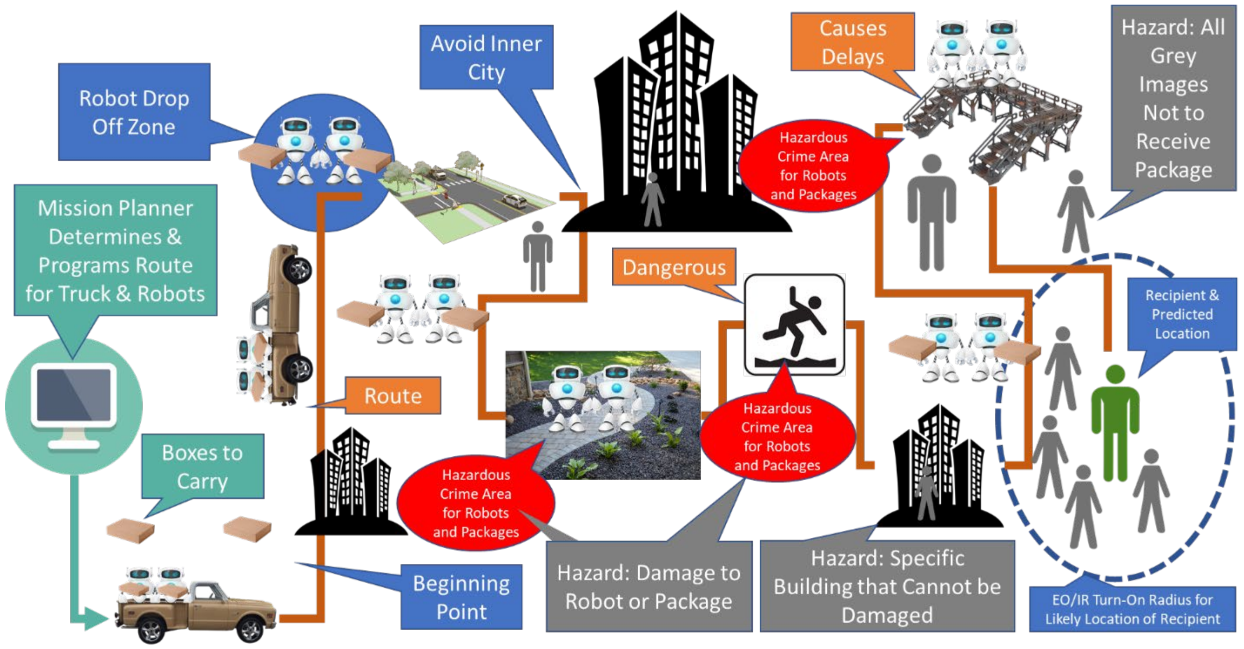

Robot Drop Off Zone
Avoid Inner City
Causes Delays
Hazard: All Grey Images Not to Receive Package
Hazardous Crime Area for Robots and Packages
Mission Planner Determines & Programs Route for Truck & Robots
Dangerous
Recipient & Predicted Location
Route
Hazardous Crime Area for Robots and Packages
Boxes to Carry
Hazardous Crime Area for Robots and Packages
Beginning Point
Hazard: Damage to Robot or Package
Hazard: Specific Building that Cannot be Damaged
EO/IR Turn-On Radius for Likely Location of Recipient

FIGURE 3. AI-ENABLED ROBOT DELIVERY SAFETY USE CASE (Nagy, 2021)

The above use case supports two functions. The first involves mission planning, defining the route and robots to use. The second function involves the management of the robot. There are two performers of interest in the scenario, the robot and the user. For the first system, the user must agree to both the route and robots to use. In the second system, the user interfaces with the robot through a graphical user interface (GUI). In most IA, two options are available. One where the unmanned system is chosen to perform functions, and humans provide support, and vice versa. In this case there is just one option, that the robot traverses a route and delivers the packages. This simplifies the IA.

In column D, we designate which part of the robot is executing that capacity. For example, route movement is the role of the stabilizer and propulsion mechanisms, while identify customer is a segment of the computer vision ML module. These distinctions are crucial, since if an ML module is involved, NOSSA evaluators need to up their awareness. The supporting performer is either the mission user, but in NOSSA's case, the evaluator. Column F describes the interactions between the robot and human, described by the three OPD characteristics. In this column we evaluate how the human uses those techniques to work with the robot. As one can see by examining the Column F, OPD in this scenario is accomplished through the GUI. The GUI is well designed and enables specific windows for observing location and robot state, provides a route details window which predicts the robots next moves, and specific robot and route approval buttons to provide direction. An additional geography map provides back up situation awareness for users. A final feature, crucial for NOSSA evaluators, is that the geographic map on the GUI can also be toggled to a statistical display.

Details associated with the GUI's capabilities are relevant to the interdependence possibilities for the ML-enabled components of the overall system. They enable the GUI to reduce the autonomy ML related functions within the delivery robot. These specific GUI functions are listed in Table 5.

TABLE 5. SCENARIO GUI FUNCTIONS

Number	Scenario GUI Function Details
1	Enables manual take over control, e.g., direction, speed, avoidance
2	Provides awareness of autonomous system's actions—and change through "Affiliation" designation—who receives package
3	Option to cancel delivery actions
4	Verify recipient visually
5	Abort operations if needed
6	Change the success threshold (a statistic identifying the likelihood of success) when selecting recipient and related package delivery approach
7	Modify methods for recipient recognition or navigation method, from a ML function to traditional approach

Figure 4 pictures the GUI and highlights its Markov Decision Process (MDP) State design. A MDP is a way to model discrete, stochastic-based actions, popularly used in engineering and data science disciplines. In the Figure 4 example, the GUI allows the user to observe each state of the robot (and the actions associated with the state), displays the route prediction of where the robot is planning to go (actions that move to a different state), and ways to move from state to state by directing the robot to make task changes, if necessary.

IA examines the process defined between the human user and the automated system for analysis. The MDP GUI in Figure 4 shows how a user can track the progress of the robots (following a process), as they take various actions from state to state (the process flow). The list of seven ways to reduce autonomy, described above, emphasizes how IA can analyze process states and their related actions, as well as defining where user intersession (monitoring and decision override, etc.) can occur to ensure autonomy reduction of the robots' critical functions during mission execution.

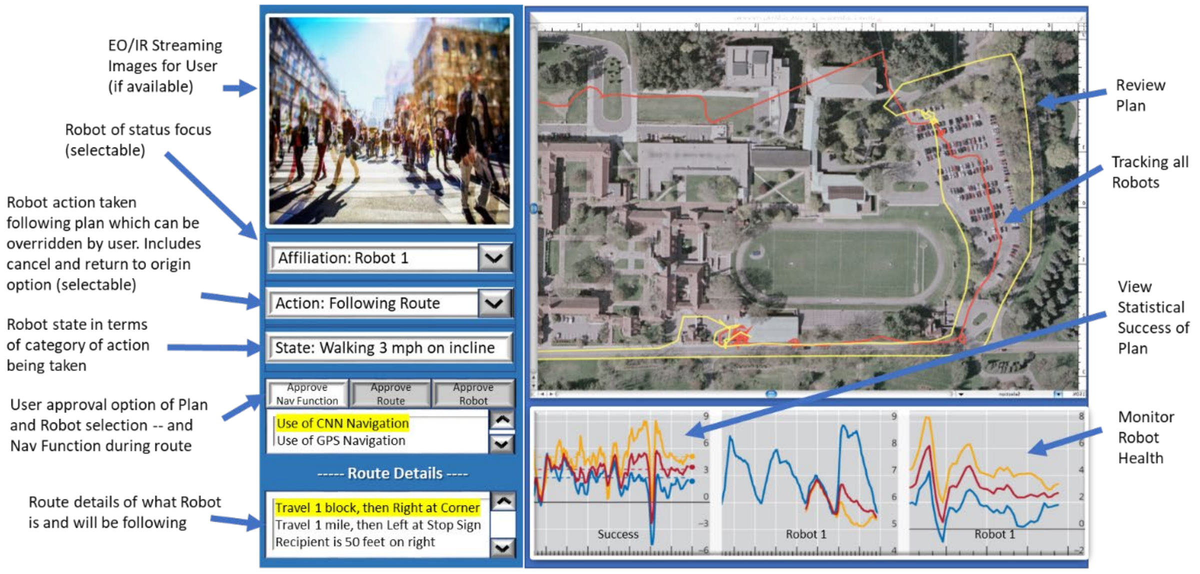

EO/IR Streaming Images for User (if available)
Robot of status focus (selectable)
Robot action taken following plan which can be overridden by user. Includes cancel and return to origin option (selectable)
Robot state in terms of category of action being taken
User approval option of Plan and Robot selection -- and Nav Function during route
Route details of what Robot is and will be following
Affiliation: Robot 1
Action: Following Route
State: Walking 3 mph on incline
Approve Nav Function
Approve Route
Approve Robot
Use of CNN Navigation
Use of GPS Navigation
----- Route Details ----
Travel 1 block, then Right at Corner
Travel 1 mile, then Left at Stop Sign
Recipient is 50 feet on right
Success
Robot 1
Robot 1
Review Plan
Tracking all Robots
View Statistical Success of Plan
Monitor Robot Health

FIGURE 4. MARKOV DECISION STATE GUI DESIGN (Nagy, 2021)

To understand this analysis process, let us examine several capacities and the associated analysis. In capacity row 1 in Table 4, the evaluator learns that OPD is executed through the GUI and is straight forward. The function reflects a standard database look up call, and it is clear that it is not ML-related. Normal NOSSA procedures will apply. In capacity row 4, the capacity is the application of a naive Bayes algorithm to determine best input attributes. This sounds very ML-like, so the analyst should enlist two sets of questions. First, what are the OPD interactions between this algorithm and the humans? The GUI provides a view into the statistical data. This enables evaluators to ascertain whether the algorithm is working properly. The second set of questions addresses the three areas of ML failure modes. From Table 1, the first is "Does the algorithm produce faulty/poor decision recommendations?" The second is "Are there human–machine operation issues?" and the last is "Is the system under (cyber) attack?" Our goal is not to resolve these questions directly, though at some point that will be required. Rather, are there human interventions that prevent these issues that by reducing the "autonomy" of the algorithm? Because if there are, then the function is not so critical as to require a code base check.

TABLE 4. IA FOR SYSTEM SAFETY USE CASE

. TASKS	. SUB TASKS	C. CAPACITIES	D. PERFORMING ROBOT COMPONENT	E . SUPPORTING HUMAN	F. OBSERVABILITY, PREDICTABILITY, AND DIRECTABILITY ASSESSMENT WRT NOSSA EVALUATIONS, PLUS SPECIFIC GUI FUNCTIONS FROM ABOVE
-thru GUI	ap obstacle s	1. Use leg route & obstacle DB	Data Loader Manager	U ser	OPD-thru GUI and MDP (1, 7)
		2. Use wx DB	Data Loader Manager	U ser	OPD-thru GUI
		3. Use police intel DB	Data Loader Manager	U ser	OPD-thru GUI
	haracteri ze legs	4. Use naive Bayes (nB) to determine best input attributes	nB, DB Farm, DB Manager	E valuator	OPD-thru GUI Leverage statistical output part of GUI to verify inputs for attributes make sense. (2, 3, 6)
		5. User Random Forest (RF) to estimate probability and missing attributes	RF, DB Farm, DB Manager	E valuator	OPD-thru GUI While RF is a black box to evaluators, in this case techniques exist to prove that the results are useful. Evaluators need to understand this proof and how to apply. (2, 3, 6)
	elect robot/rou te pairs	6. Apply temporal greedy search (TGS) to create robot /route candidates	TGS, Business Rule Manager, DB Farm, DB Manager	E valuator	OPD-thru GUI While TGS is an algorithm, it is not ML, no special attention required (1, 2, 3, 6, 7)
		7. Use non-linear optimization (NLO) to determine combos that provide highest likelihood of mission success			OPD-thru GUI While NLO is an algorithm, it is not ML, no special attention required (1, 2, 3, 6, 7)
nload robot in delivery Zone	emove from truck	8. Activate robots	Processor, Power Regulator, and Power Supply	T ruck Driver	OPD-thru GUI (1)
obot navigation	etermine lead	9. Select robot as lead	Main Navigation and Guidance Controller	U ser	OPD-thru GUI (1)
	avigate	10, Access planned waypoint DB	Main Navigation and Guidance Controller	U ser	OPD-thru GUI and MDP (1, 2, 7)
		Update status	Main Navigation and Guidance Controller	U ser	OPD-thru GUI and MDP (1, 2, 3, 5, 6, 7)
elivery	nter delivery zone	11. Compare up date to plan	Main Navigation and Guidance Controller	U ser	OPD-thru GUI and MDP (1, 2, 3, 5, 6, 7)
		12. Adjust location as necessary	Main Navigation and Guidance Controller	U ser	OPD-thru GUI and MDP (1, 3, 7)
	dentify customer	13. Use computer vision (CV) to identify customer	Image DB and CV	U ser, Recipient	OPD-thru GUI and MDP; CV is ML, so a human on the loop checking the identity as seen by the robot reduces "Autonomy." In other systems, this may not be feasible. May have to assume risk here. (1, 4)
		14. Check time so delivery can be synchronous	GPS Signal & SATCOM Transceiver, GPS Translator	U ser	OPD-thru GUI and MDP (2, 3, 5)
		15. Deliver package	Robot arms	R ecipient	(1, 4, 7)

If we review these questions from that lens, what do we learn? First, because the evaluator employs the seven GUI functions to explore the algorithm outputs, that means the evaluator or user could be involved in directing the rerunning of the algorithm so that it could succeed (see functions 2, 6, and 7 in Table 5). Alternatively, the user may be able to insert data directly (functions 1 and 7). Because the user, using the GUI, can modify robot data and inputs, question number two comes into view. Can the human introduce errors into the statistics so that the system is not functioning properly? Yes, this is possible, but can be addressed through proper training, something Navy systems scrupulously employ. Finally, the fact that humans can view the algorithm output means that they could detect adversary or insider activity disrupted the data (in this case, all the GUI functions apply). Yes, there are procedural implications for using this part of the system. One can also surmise, though, that because of the human ability to address all three types of failure modes, that an evaluator could conclude that the software criticality of this particular algorithm is relatively low. That is, the algorithm's "autonomy" is not high.

Capacity row 5 is similar to row 4, but there are interesting differences with regard to "explainable" AI—explaining the AI decision. For example, when using a Random Forest (RF) algorithm (an advanced ML technique based on many variations of a single decision tree), the user interface might consider a design that supports the strength of the algorithm—this is referred to as a symbiosis. For example, the GUI might be designed to show the user when there is a gap in the existing input data to the RF (a potential "real world issue") and how this data gap is being estimated/compensated by the RF algorithm, along with providing the user with a statistical explanation of the algorithm's estimation approach and solution. Further, Nagy (2021) shows that an RF algorithm can be used to estimate missing statistical data, including statistical data describing highest success in supporting a final decision (e.g., routes and robot selection). This type of statistical explanation for critical decision-making data provides the user with increased confidence as to whether to accept or reject the estimations of an RF algorithm.

This raises another key point. If possible, GUI design and algorithm selection should be symbiotically determined, facilitated by the IA results. For the RF example, selection of an ML algorithm's capability to compensate for "real world issues" through statistical understanding of how the data was estimated should influence the GUI design with the end goal of reducing the autonomy of an ML algorithm. The design supporting the expandability strength of the ML should provide visibility into potential "real world issues" and how the selected algorithm compensates with statistical insight for the user. This IA driven, symbiotic GUI to ML algorithm design ensures that the user has the knowledge needed to make a final approval regarding any data estimated, thereby assisting in answering the first question, "Does the algorithm produce faulty results?" When symbiosis of design and algorithm occur, as provided in the RF example, there is greater confidence that the answer will be "no."

When symbiosis does not occur, there is a greater chance that system safety risks increase. Again, using the RF algorithm as an example, through its estimation process and statistical explanation, the user is provided with the necessary data to have confidence in approving or rejecting the route and robot selection during a "real world issue" lack of available data. Providing the necessary "knowledge" for a user to have confidence in the final approval becomes a key requirement necessary to maintain overall system performance and adequately reduce autonomy of an ML algorithm. Without the user having confidence in how well the algorithm is addressing "real world issues," final approval authority is meaningless. This is because the man-in-the-loop is not achieving the designed result of autonomous reduction of the ML algorithm. When things go wrong during deployment, it is about making sure that the user has the necessary "knowledge" provided by the ML algorithm (or other means) to make the final approval decision.

In this example we use the three general fault areas for ML. Table 3 adds root causes. In this particular example, the possible root causes that might require deeper examination would include training dataset corruption, mis-validation of data set, overfitting of the ML model, operator overreliance, or adversarial deception. These are just two examples of how to use IA to support NOSSA critical function analysis for ML-enabled functions. A complete IA would yield far more insights into potential problem areas and root causes.

Next Steps and Recommendations

After considering the analysis conducted above, one asks, "Does IA serve to solve the challenges of evaluating ML based components in weapon systems?" "Not completely," the authors argue. However, what IA does do is to add detail to those functional areas that need to be evaluated. IA helps identify specific ML components, which humans might be involved in the process to reduce "autonomy" and suggest which mechanisms amongst the OPD triad might be best employed, such as GUIs and their design. Moreover, not all designers appreciate the interdependence that should exist between user and the algorithm and therefore build no OPD connections. This makes reducing "autonomy" infinitely harder. Conducting IA rapidly speeds that discovery, which might suggest necessary rework by the original developer.

By adding the three main fault areas of ML use to the IA, very specific evaluation details and questions are raised. While it may not solve the emerging ML evaluation conundrum, it does add considerable detail to the kinds of discussions that system developers and NOSSA ought to consider, especially when evaluators use the root cause details to inform their questions. Further, the authors recommend adding a seventh column to the IA table, which would include what root causes were suspect and why. Here are several specific considerations that also emerged from developing this work.

As a nascent requirement for the NOSSA evaluators, recommend careful consideration of each function using these techniques, plus frequent review of deployed system performance. Specifying a special follow up with operational users may be required for the first years of use. While Nagy's scenario benefits from a very capable GUI already informed by a knowledge of IA, NOSSA evaluators should expect that not all systems submitted for such review will be as well developed. In fact, NOSSA may want to consider making an IA a requirement for submission of a system for certification.

We recommend not updating training data sets to the deployed edge, which is to platforms deployed in actual operations. At this point, the processes that support such updates are not well understood. Any updates to training data sets ought to be reexamined by NOSSA. However, as the use of ML devices becomes prevalent, NOSSA will not be able to maintain the pace to evaluate these changes. New procedures will need to be considered, developed, evaluated, and then adopted. We recommend tasking the Naval Postgraduate School, the Joint AI Center, or the Navy's Digital Software Engineering Transformation Working Group to start this work.

In many respects, updating training data sets is like the Navy's development security operations (DEVSECOPS) efforts to update patches to the Fleet in hours, not weeks. NOSSA should learn from those lessons, though recognizing that the size of most training data sets makes over the air updates challenging and unreliable. NOSSA and training data set updaters should consider leveraging standard storage device deliveries. This would include ways to secure the data on those storage devices.

Introducing ML techniques into systems, because of the importance of the training data sets, means that changes to the system engineering processes are necessary. In the past, the SE "Vee" diagram is a set of procedures that are executed to an end state, normally at the end of a variety of test and evaluation events. This includes operational test, security and weapons certification, and in NOSSA's case, safety certification. According to Johnson (2021), this new SE "Vee" may change to be continuous for the entire life cycle of a system. This means, in theory, that NOSSA has a continuous responsibility to monitor system safety. That is a significant change, and worth thinking about. It may be that IA provides at least a way to wrap one's head around this potentially new responsibility. IA could be used to identify those functions that do require continuous evaluation.

References

Annett, J. (2003). Hierarchical task analysis. *Handbook of cognitive task design* (pp. 17–35). Lawrence Erlbaum Associates.

Artificial Intelligence. (2021). In *Oxford online dictionary*. https://en.oxforddictionaries.com/definition/artificial_intelligence

Bolté, B., Endsley, M., & , Jones, D. (2003). *Designing for situation awareness: An approach to user-centered design*. Taylor & Francis.

Chipman, S., Schraagen, J., & Shalin, V. (2009). *Cognitive task analysis*. Lawrence Erlbaum Associates, Inc.

Crandall, B., & Klein, G. (2006). *Working minds: A practitioner's guide to cognitive task Analysis*. Bradford Books.

DoD. (2012). *Department of Defense standard practice: System Safety* (MIL-STD-882E).

Faria, J. (2017, October 23–26). Non-determinism and failure modes in machine learning. *Proceedings of IEEE 28th International Symposium on Software Reliability Engineering Workshops*, 310–316.

Johnson, B. W. (2021, March 3). *Metacognition for artificial intelligence system safety* [Online conference presentation]. Developing Artificial Intelligence in Defense Programs Acquisition Research Symposium, Monterey, CA, United States.

Johnson, M., Bradshaw, J. M., Feltovich, P. J., Jonker, C. M., van Riemsdijk, B. M., & Sierhuis, M. (2014). Coactive design: Designing support for interdependence in joint activity. *Journal of Human-Robot Interaction*, *3*(1), 43–69.

Joint Services-Software Safety Authorities. (2016). *Software system safety: Implementation process and tasks supporting MIL-STD-882E.* Joint Staff.

Nagy, B. N. (2021, March 25). *Using event-verb-event (EVE) constructs to train algorithms to recommend a complex mix of tactical actions that can be statistically analyzed* [Online conference presentation]. Fifth Annual Naval Application of Machine Learning, San Diego, CA, United States.

Varshney, K. (2016, January 31–February 5). Engineering safety in machine learning. *Proceedings of Information Theory and Applications Workshop*, 1–5