FINAL TECHNICAL REPORT SERC-2022-TR-005

WRT-1049.8.7
Digital Engineering Enhanced T&E of Learning-Based Systems

Date: 21 June 2022

PRINCIPAL INVESTIGATOR: Dr. Peter Beling, Virginia Tech
CO-PRINCIPAL INVESTIGATOR(S): Dr. Laura Freeman, Virginia Tech
Dr. Jitesh Panchal, Purdue University

SPONSOR(S): OFFICE OF THE UNDER SECRETARY OF DEFENSE FOR RESEARCH & ENGINEERING

DISCLAIMER

The Systems Engineering Research Center (SERC) is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology.

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Office of the Assistant Secretary of Defense for Research and Engineering (ASD(R&E)) under Contract [HQ0034-19-D-0003, TO#0309].

RESEARCH TEAM

Name	Org.	Labor Category
Dr. Peter Beling	Virginia Tech	PI
Dr. Laura Freeman	Virginia Tech	Co-PI
Dr. Jitesh Panchal	Purdue University	Co-PI

Table of Contents

List of Figures

List of Tables

1 Executive Summary

The current approach to Test and Evaluation (T&E) involves treating the system in a blackbox fashion, i.e., the system is presented with sample inputs, and the corresponding outputs are observed and characterized relative to expectations. While such an approach works well for traditional static systems, test and evaluation of autonomous intelligent systems presents formidable challenges due to the dynamic environments of the agents, adaptive learning behaviors of individual agents, complex interactions between agents and the operational environment, difficulty in testing black-box machine learning (ML) models, and rapidly evolving ML models and AI algorithms [1, 2].

The objective of this research was to develop approaches to the design of test and evaluation (T&E) programs and the acquisition of data/model rights for learning-based systems. The principal objective was to understand how increasing government access to the models and learning-agents (AI algorithms) used in system design might decrease the need and expense of testing and increase confidence in results. The principal hypotheses investigated in this incubator project are that the number of samples needed to test AI/ML models to an acceptable degree of assurance can be reduced if we have access to the models themselves (in mathematics or software), reduced still further if we also have access to the algorithms and data used to train the models, and reduced further yet if we also have access to systems models and other artifacts of the digital engineering process. Therefore, the cost of acquisition can be significantly reduced if T&E programs are based on the optimal balance between the cost of acquiring the technical data/algorithm rights of AI/ML systems, and the cost of testing those systems.

This incubator project established the theory and methods for exploring how T&E requirements can and should change as a function of the test team knowledge of the technical specifications of an AI enabled systems. The incubator developed theory based in systems theory that captures changes in the systems and the state-space in which it operates through the concept of systems morphisms. The onion model describes different levels of system knowledge and a context for defining the abstraction of the system. The project experimented with two pilot scenarios to demonstrate how multiple phases of testing contribute to the evaluation of an AI enabled systems. Finally, we present the Bayesian analytical framework for combining information across the multiple phases of testing. This analytical framework also reflects the changing system configuration and context. In summary, this work essentially constitutes the building blocks for investigating the cost-benefit for test data collection on a realistic system in future phases.

A major challenge in conducting AI enabled systems research is that physical realizations are needed for T&E research. Future work could leverage the Silverfish testbed, developed under prior SERC tasks, and expand the testbed into physical implementations. Physical implementations in addition to MBSE representations would enable the direct execution of a T&E program on the Silverfish testbed. Future work should also include purposefully varying the systems knowledge (based on the onion model), the complexity of the systems and its operating environments (number of morphisms), and determine minimally adequate testing as a function of those variables.

2 Overview

Artificial intelligence and machine learning (AI/ML) has moved beyond being a research field to being an essential element of next-generation military systems. The discipline of verification and validation of AI/ML enabled complex systems, however, in its nascent stage. Little is understood about how to identify changes in operating conditions or adversarial actions that might cause the performance of an AI/ML model to deviate from design limits [1]. The challenges in this regard are amplified when considering autonomous functions that may engage in self-learning over the long life cycles seen in military systems.

The objective of this research was to develop approaches to the design of test and evaluation (T&E) programs and the acquisition of data/model rights for learning-based systems. The principal objective was to understand how increasing government access to the models and learning-agents (AI algorithms) used in system design might decrease the need and expense of testing and increase confidence in results.

The current approach to T&E involves treating the system in a blackbox fashion, i.e., the system is presented with sample inputs, and the corresponding outputs are observed and characterized relative to expectations. While such an approach works well for traditional static systems, test and evaluation of autonomous intelligent systems presents formidable challenges due to the dynamic environments of the agents, adaptive learning behaviors of individual agents, complex interactions between agents and the operational environment, difficulty in testing black-box machine learning (ML) models, and rapidly evolving ML models and AI algorithms [1, 2].

Our principal hypotheses are that the number of samples needed to test AI/ML models to an acceptable degree of assurance can be reduced if we have access to the models themselves (in mathematics or software), reduced still further if we also have access to the algorithms and data used to train the models, and reduced further yet if we also have access to systems models and other artifacts of the digital engineering process. Therefore, the cost of acquisition can be significantly reduced if T&E programs are based on the optimal balance between the cost of acquiring the technical data/algorithm rights of AI/ML systems, and the cost of testing those systems.

3 Theoretical Background

At the core of this research is systems theory [3]. Specifically, we build from the lineage of the systems theorist A. Wayne Wymore, who defined the Mathematical Theory of Systems Engineering [4] and has been credited for coining the term model-based systems engineering (e.g., [5]) for his Tricotyledon Theory of System Design [6, 7]. A mathematical mechanism used in Wymorian systems theory is the system specification morphism; where a morphism is a mathematical characterization of the preservation of equivalence between a pair of system specifications [8].

System specifications may be defined at many levels within a hierarchy. The hierarchy of system specification is a prominent aspect of a branch of Wymorian systems theory commonly referred as computational systems theory, or formally known as the Theory of Modeling and Simulation [7, 8]. Each level of the hierarchy of system specification reveals further detail as to the knowledge of the structure from external interfaces and interactions to internal component and coupling knowledge. Furthermore, within each level of system specification, a morphism essentially characterizes

abstraction and elaboration of detail.

To concisely articulate the concepts, we have selected a subset of Wymorian systems specification equations and associated morphisms within the hierarchy of system specification, which we have adapted from [8].

Here we define a system specification and associated morphism at the level of least knowledge of structure, in which we only have knowledge external interactions and interfaces.

$$IO = (T, X, Y) \tag{1}$$

Where IO is the *Input/Output (I/O) Observation Frame*, T is the *time base*, X is the *input values set*, and Y is the *output values set*.

For the associated I/O Observation Frame Morphism, we relate two systems $IO = (T, X, Y)$ and $IO' = (T', X', Y')$ at the I/O frame level by defining functions to relate the input and outputs interfaces. Let $g : (X', T') \rightarrow (X, T)$ be a function to derive a segment over X given a segment over $X'$and $k : (Y, T) \longmapsto (Y', T')$ be a function to derive an output segment over output set over Y' of the little system given an output segment over Y of the big system S.

Here we define a system specification and associated morphism at the level of most knowledge of structure, in which we have full knowledge of internal components and their coupling (i.e., internal interfaces).

$$N = (T, X_N, Y_N, D, \{M_d | d \in D\}, \{I_d | d \in D \cup \{N\}\}, \{Z_d | d \in D \cup \{N\}\}), \tag{2}$$

where N is a *Coupled System (Network of Systems)*, X_N is the *set of external input of the network*, Y_N is the *set of external outputs of the network*, D is the *set of component references* with $d \in D$, M_d is an *I/O System*, $I_d \subseteq D \cup \{N\}$ is the *set of influencers* of d, and $Z_d : x_{i \in I_d} Y X_i \rightarrow X Y_d$ is the *interface map* for d, with

$$Y X_i = \begin{cases} X_i \text{ if } i = N \\ Y_i \text{ if } i \neq N \end{cases} \tag{3}$$

and

$$X Y_d = \begin{cases} Y_d \text{ if } i = N \\ X_d \text{ if } i \neq N \end{cases} \tag{4}$$

For the associated Network of Systems Morphism (Coupling Morphism), let N and N' be represented by $(T, X_N, Y_N, D, \{M_d\}, \{I_d\}, \{Z_d\})$ and $(T', X'_{N'}, Y'_{N'}, D', \{M'_{d'}\}, \{I'_{d'}\}, \{Z'_{d'}\})$. We define a *network of systems morphism* from N onto N' as a structure $\langle \text{coord}, \{k_{d'}\}, \{g_{d'}\}, \{h_{d'}\} \rangle$ such that $\{k_{d'}\}$, $\{g_{d'}\}$, and $\{h_{d'}\}$ satisfy the coupling, state transition, and input/output preservation conditions. Where coord : $D \rightarrow_{\text{onto}} D'$ be a mapping from the set of components D of N onto the set of components D' of N'; $h_{d'}$ is a mapping of states; $k_{d'}$ is a mapping of outputs; and $g_{d'}$ is a mapping of inputs.

The parameter morphism, in conjunction with the above specifications within the hierarchy, is a mapping of parameter space along with state space. The parameter morphism is an explicit

documentation of allowable deviations (approximations) from exact morphisms, as is the expectation with the IO observation frame and network of systems morphisms, relative to changes in parameter sets. A simple example of a parameter morphism is the selection of the mean versus a distribution as a parameter test set.

Furthermore, although equations (1) and (2) are deterministic, the concepts can be morphically or inherently defined stochastically in forms such as with Markov models and neural networks. This is not shown for simplicity of articulation of the concepts. For further detail, refer to [8].

Lastly, the framing of the hierarchy and associated morphisms is important to understand the systems theoretical context as a whole. First, the relationship between the IO observation frame and the network of systems is a one-to-many specification relationship, meaning that one system specification at the IO observation frame can lead to specification of many (maybe infinite) network of system specifications. However, each network of system specification can map to only one specification at the IO observation frame level. Second, a morphism at the IO observation frame level does not guarantee a morphism at the network of systems level. Third, however, a morphism at a network of systems level implies a morphism at the IO observation frame level. These are systems theoretic concepts we use to underpin our methodology for T&E of LBS.

3.1 Methodology

The practice of engineering systems is reliant on use of surrogate analogies for T&E. In some cases, we may not have access to the fielded system until late in the program and, therefore, select a surrogate as an analogous representation of the current (phase appropriate) design of the system of interest. In other cases, the system of interest may be fielded and we want to understand observed behavior, for which we may use a surrogate, analogous environment for testing the fielded system (or analogous test system). These activities are typically thought of as necessary risk reduction, for which we characterize the validity of the analogies through the use of systems theoretic morphisms.

Consider the following example to provide further context: The IO observation frame morphism could be used to characterize the change in operational conditions and change in adversarial actions. The network of systems morphism can be used to characterize the changes in implementation of a LBS subsequent to changes in operational conditions and adversarial actions. Furthermore, from the last paragraph of the previous section, a morphism between system implementations (i.e., network of systems morphism) implies a morphism at the mission level, which, therefore, is indicative of mission success.

Commonly associated with LBS is the onion model shown in Figure 1. In the outer layer, we have minimum knowledge of the system context, which we categorize as mission knowledge. In the second to outer layer, we begin to have knowledge of the interior structure in the form of a functional architecture. In the third to outer layer (second to inner layer), we have knowledge of the agent cognitive functions. In the inner most layer, we have maximum knowledge in the form of knowledge of the physical implementation of the system of interest. From a systems theory perspective, we can provide a view of validity of analogies relative to the onion model.

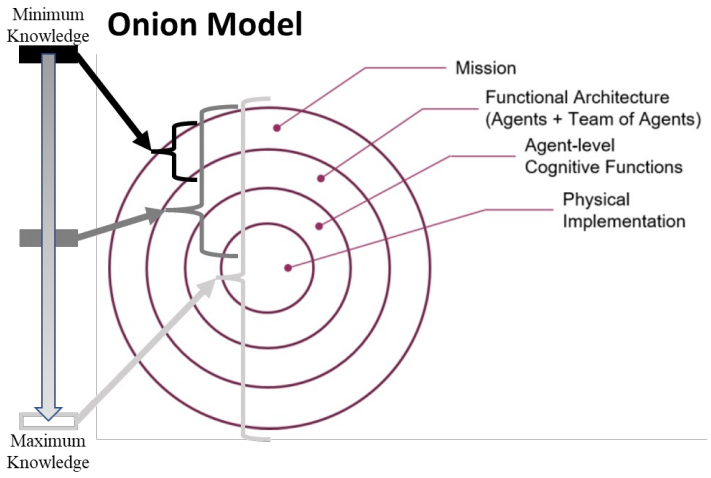

Figure 1: Onion model typically used to understand layers of LBS.

In Figure 2, we provide a systems theoretic context. We show the real mission to the top left and the preservation of equivalence to surrogate T&E context shown in the top right. We propose characterization of this equivalence through systems theoretic mechanisms, such as the IO observation frame and associated morphism. We also show the field system to the bottom left and the preservation of equivalence to surrogate model shown in the bottom right. We propose characterization of this equivalence through systems theoretic mechanisms, such as the network of systems and associated morphism.

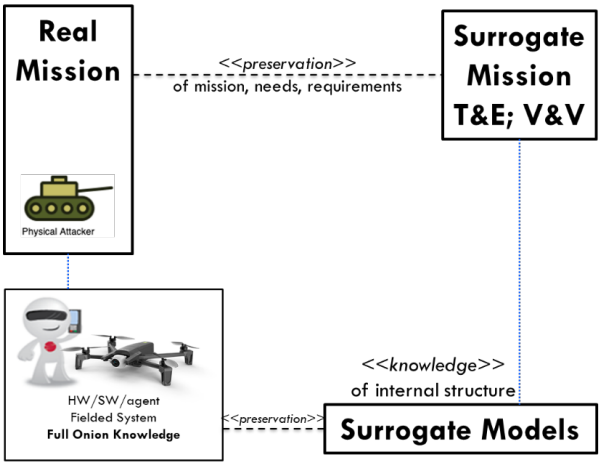

Figure 2: High-level view of systems theory perspective of the onion model.

For this project we used the mission context of detection of a potential attacker, consistent with Silverfish [9]. Rather than focus on the full system of systems of Silverfish, we selected to focus on the unmanned aerial vehicle (UAV) component as the system of interest for this research project. In Figure 3, we provide further explanation to our set of experiments within the context of systems theory and the onion model.

First, we have used the You Only Look Once (Agent YOLO) algorithm as our agent, which has an unknown T&E context conducted prior to our acquisition of the agent. Therefore, we cannot determine its morphic equivalence to the real mission and must conduct further testing. The new T&E mission analogies are expected to be characterized through systems theoretic morphisms. For this project, we used a series of T&E surrogate mission contexts of the potential attacker in the form of a soccer match (i.e., red versus blue) and automobile detection (truck versus other type of vehicle). While the soccer match was a simulation (video from the internet), the automobile surrogate mission context was both a simulation and physical test.

Second, we were not able to acquire the physical hardware expected for the fielded system at the onset of the project. Therefore, we relied on surrogate models, for T&E, that we believe to be analogous to the fielded system. Each surrogate model is expected to be morphically characterized to determine its equivalence relative to the fielded system. For this project we have selected a series of surrogate models for the UAV. First, the initial Agent YOLO may only be analogous as far as the cognitive function is concerned. Second, we used a surrogate drone, which has lower cost and quality of hardware than the fielded systems. Last, it should also be noted that even when we have access to the expected fielded system, the morphic validity of the analogies must also be confirmed. For this, we suggest that a digital twin (i.e., simulation) and final product (or physical twin) from low-rate initial production (LRIP) be used for an initial operational test and evaluation (IOT&E), both of which should be morphically characterized for its equivalence to the expected (or measured) reality.

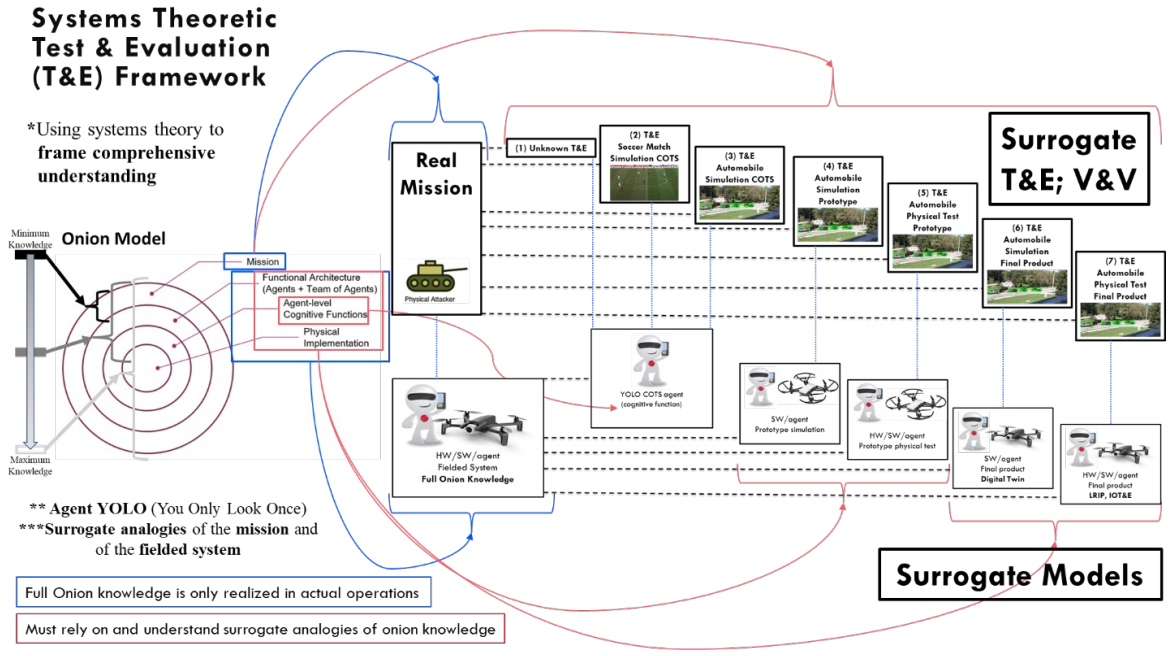

Figure 3: Proposed Systems Theoretic Test and Evaluation Framework.

Because the full knowledge relative to the onion model can only be known once the final product design becomes the fielded system and is placed in its real mission context, we must rely on and understand surrogate analogies to provide confidence in mission success. We have selected to

use Bayesian methods, such as discussed in [10], to characterize confidence in mission success relative to knowledge on the morphic equivalence. Further detail on the use of Bayesian methods is provided in another section.

4 Experimental Testbed

The broader objective for creating the experimental testbed is to assist in validating the Testing and Evaluation (T&E) framework for learning-based systems. For this incubator project, the specific goal is to demonstrate how the T&E framework can be utilized for a specific scenario where the goal is to detect the presence of enemies, tracking them, and sending a signal for silverfish protected field.

4.1 Overview of the Experimenal Testbed

The testbed consists of (a) scenarios, (b) hardware, and (c) software. The details are described next.

4.1.1 Scenarios

Two scenarios are created as surrogate problems as a part of creating experimental testbeds.

Scenario 1: Person Identification and Tracking in a Soccer Game: This scenario is based on a soccer game as a surrogate problem using stock video to detect players belonging to different teams and their location in the field. The players, shown in Figure 4, are classified into two different teams based on their apparel colors and patterns. The idea here is to showcase the different teams as allies and enemies. In addition to this, the location co-ordinates of the players are continually tracked.

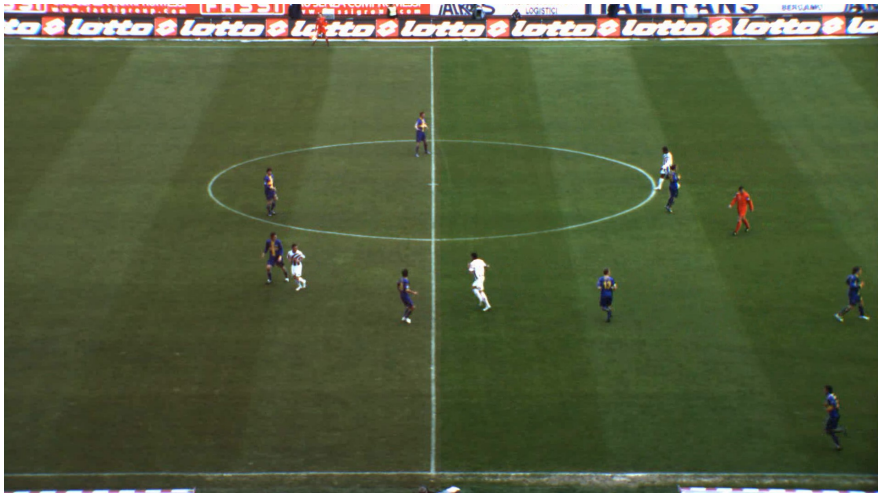

Figure 4: Scenario 1: Person Identification and Tracking in a Soccer Game

Scenario 2: Vehicle Detection and Tracking: This scenario is based on automobile detection and tracking as a surrogate problem to detect vehicular traffic, location co-ordinates and their velocities (see Figure 5). The vehicles are categorized based on their sizes, i.e., small vehicles rep-

resent allies (friends) and large vehicles represent enemies. Similar to Scenario 1, the location co-ordinates of the vehicles are detected along with their velocities.

Figure 5: Scenario 2: Vehicle Detection and Tracking

The coordinates obtained from the two scenarios are mapped and visualized on a grid shown in Figure 6.

1						
2						
3						
4						
5						
6						
	A	B	C	D	E	F

Figure 6: Grid used for visualization and mapping

4.1.2 Hardware

For hardware implementation, two drones namely, **Ryze Tello** (lower fidelity prototype drone) and **Parrot ANAFI** (higher fidelity prototype drone) are used. The specifications of the Ryze Tello drone and the Parrot ANAFI drone are shown in Figure 8.

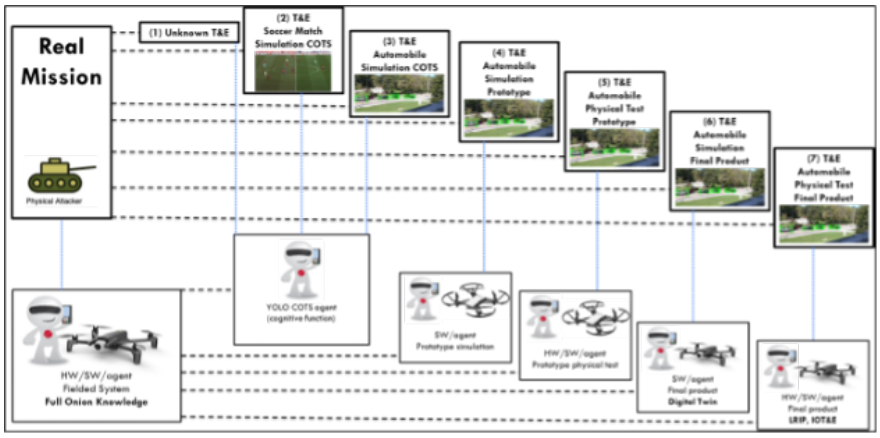

Figure 7: Various Testing Scenarios

- **Parrot ANAFI (Higher Fidelity Drone)**

320 gms | 21 MP Camera, 4K Video, Gimbal stabilization | 180° tilt camera 26 min flight time

- **Ryze Tello(Lower Fidelity Drone)**

80 gms | 5 MP Camera | 13 mins flight time

Figure 8: Comparison of the specifications of the higher and lower fidelity drones

Lower fidelity prototypes are used to test whether the high-level design concepts can be translated into tangible outputs. On the other hand, higher fidelity prototypes provide outputs that are as similar as possible to the desired requirements defined initially.

The drones capture videos which are then segmented into images frame by frame. The differences in the images from the two drones can be clearly seen in terms of the resolution, field of view and stability.

4.1.3 SOFTWARE

The primary goal of the software implementation is to identify the location coordinates of the allies and enemies and track them in real time. To do so, videos captured from the drones are used as input, and the output being series of location coordinates. This implementation broadly consists of four steps: image preprocessing, object detection & classification, object tracking, and mapping.

Image Preprocessing: The goal of this step is to retrieve a series of clean images from the videos to prepare them for the further steps and to reduce computation time.

The steps in image preprocessing are as follows:

Figure 9: Drone Image Comparision (Above: Higher Fidelity, Below: Lower Fidelity)

1. Raw videos obtained from the hardware are segmented frame by frame into a series of images.
2. Images are resized to a lower size to increase computation speed.
3. Gaussian Blur is used to smoothen the images and to reduce unwanted noise.
4. Images are cropped to obtain the region of interest.

To simulate the different qualities of video camera from different hardware, i.e., a lower fidelity and higher fidelity input in Scenario 1:Person Identification and Tracking in a Soccer Game, the original video is used as a higher fidelity input and the blurred version of the original video is used as a lower fidelity input.

Object Detection & Classification: The goal of this step is the detection, classification and localization of the objects present in the frame. Here, the preprocessed images are used as the input, and passed through a trained or pre-trained object detection model to receive object location and classes. For this purpose, YOLOv3 (You Only Look Once, version 3) is used, which is an object detection algorithm that identifies specific objects in videos or images. A custom object detection model is trained for the soccer game scenario to detect and classify the players into different teams. Whereas, for the vehicle scenario, to detect and classify allies (friends) and enemies, a pre-

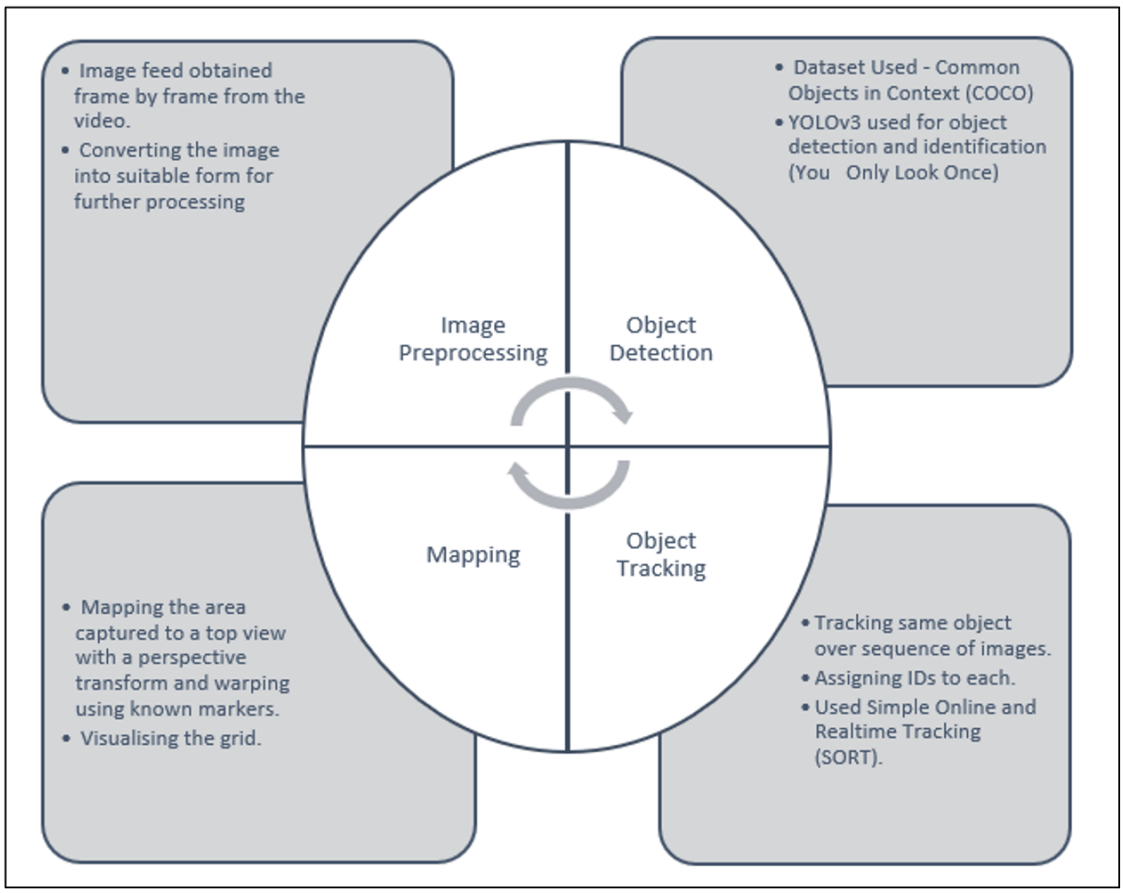

Figure 10: High level view of the process

trained model is customized using the COCO (Common Objects in Context) [11] dataset, which is a large-scale object detection, segmentation, and classification dataset. The COCO dataset has more than 2,00,000 labelled images and more than 100 categories.

Object Tracking: The goal of this step is to track the movement of an object, which involves tracking of the detected objects frame-by-frame and storing its location coordinates along with some relevant information. A unique identification number is assigned to each detected object for the duration of which it is continuously tracked. There are several challenges associated with object tracking such as occlusion, discontinuity in detections, etc. To tackle these issues, the Simple Online Real-time Tracking (SORT) algorithm is used. We are successfully able to perform tracking of each object along with finding its approximate velocity.

Mapping: The output obtained from the object tracking step is utilized to map and visualize the allies and enemies on a grid. The visualization is useful for sending a signal to silverfish protected field. This is accomplished using warping techniques and perspective transformations of a known field or using markers to a visualization grid. Figure 11 is a representation of the soccer scenario in the grid format with exact location coordinates. The blue and white dots depict players in the teams whereas the black dot is the soccer ball.

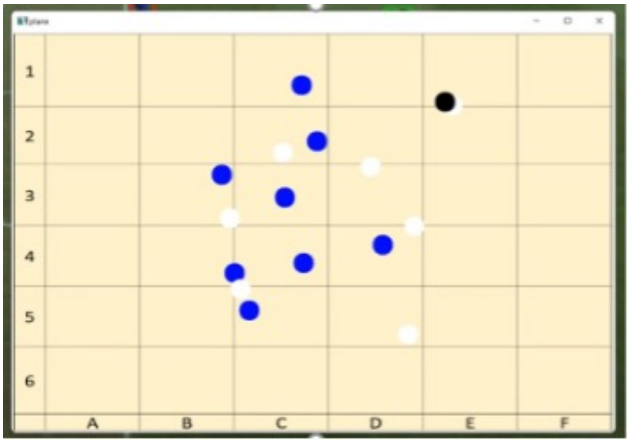

Figure 11: Grid Visualization

4.2 Preliminary Results

The results obtained from above steps, i.e., the location coordinates of the detected objects are used to evaluate the detection accuracy of identified objects to be used in the Bayesian Framework, described next.

5 Bayesian Framework

We can characterize the relationships between different models – simulation environments vs. low-fidelity systems vs. higher-fidelity systems – via a Bayesian network. A Bayesian network is a graph model that describes the relationship between nodes via probabilities. To illustrate the concept, we consider a detection system with two outcomes – either the target is detected or it is not detected – and two true states – either the target is present or it is not – with four possible combinations. These cases are summarized in Table 1.

Case #	Case	Target Present?	Target Detected?
1	True Positive	Yes	Yes
2	False Negative	Yes	No
3	False Positive	No	Yes
4	True Negative	No	No

Table 1: Cases for a target detection problem.

One might imagine each of these cases having a different "cost" from a T&E perspective – i.e., a false negative (target is present but not detected) may have more operational cost than a false positive (target is falsely detected). The goal of T&E is ultimately to characterize that cost, e.g., to compute its *expected value*, i.e., the cost of each case (C_i) times the probability of each case ($\mathbb{P}_i$):

$$\mathbb{E}(C) = \sum_{i=1}^{4} C_i \mathbb{P}_i. \tag{5}$$

The Bayesian framework considers each of the probabilities $\mathbb{P}_i$ for the final fielded system as a function of the probabilities for the analogous systems. That is, if the probabilities for the simulated environment and a lower-fidelity prototype are $\mathbb{P}_i^{sim}$ and $\mathbb{P}_i^{low}$, respectively, then the final probability $\mathbb{P}_i$ can be written in terms of conditional probabilities:

$$\mathbb{P}_i := \mathbb{P}(x \in S_i) = \sum_{j=1}^{4} \mathbb{P}(x \in S_i | x \in S_j^{low}) \mathbb{P}(x \in S_j^{low}). \tag{6}$$

Here the more complicated equation has necessitated more complicated notation: $\mathbb{P}(x \in S_i)$ is the probability of Case i and $\mathbb{P}(A|B)$ is the probability of A given B. So (6) means that the probability of, for example, getting a true positive (Case 1) in the fielded system is the probability of getting Case j in the low fidelity system multiplied by the probability of getting Case 1 in the fielded system given that we got Case j in the low-fidelity system, summed across j. This may seem like – and indeed is – a more complicated way of writing the same thing. However, if we can accurately estimate the conditional probabilities in (6), it allows us estimate the probabilities $\mathbb{P}_i$ and ultimately the cost via (5) by mostly running lower-fidelity tests. The same mechanism can then be used to capture the relationship between the lower-fidelity test and the simulated environment:

$$\mathbb{P}_i = \sum_{j=1}^{4} \sum_{k=1}^{4} \mathbb{P}(x \in S_i | x \in S_j^{low}) \mathbb{P}(x \in S_j^{low} | x \in S_k^{sim}) \mathbb{P}(x \in S_k^{sim}). \tag{7}$$

The Bayesian network summarizing the relationship between these conditional probabilities is shown in Figure 12.

5.1 Estimating the Network

In this subsection, we describe briefly how to estimate the probabilities in the previous section. In this case, we actually use a different kind of Bayesian procedure known as Bayesian inference. We begin with an estimate of the probability distribution called the *prior*, and then update that estimate as we test. For the detection case, we can model the outcomes with a binomial distribution with unknown success probability p. There is a fairly standard approach in the statistics community to estimating p. First, the prior is typically chosen to be a beta distribution $B(\alpha, \beta)$ where α, β are parameters that can be tuned to the problem. For example, one might given the prior a weight N_{prior} and start with a guess for p which we denote p_{prior}; we then would set $\alpha = p_{prior} N_{prior}$ and $\beta = N_{prior} - \alpha$. Then if tests yield s successes and f failures, we would update our estimate to be $B(\alpha + s, \beta + f)$. This procedure is illustrated in Figure 13. Here $p_{prior} = 0.4$ but we see the inference procedure closing in on the true value of $p = 0.7$ as more tests are taken.

6 Silverfish

As a testbed for the methodologies, the team used a hypothetical weapons system developed through past SERC projects under funding from OUSD (RE) and DEVCOM AC [12, 13]. The system, known as Silverfish, is a networked munition system designed to deny ground to the enemy using ground-based weapons, known as obstacles, that can engage unauthorized persons or ground

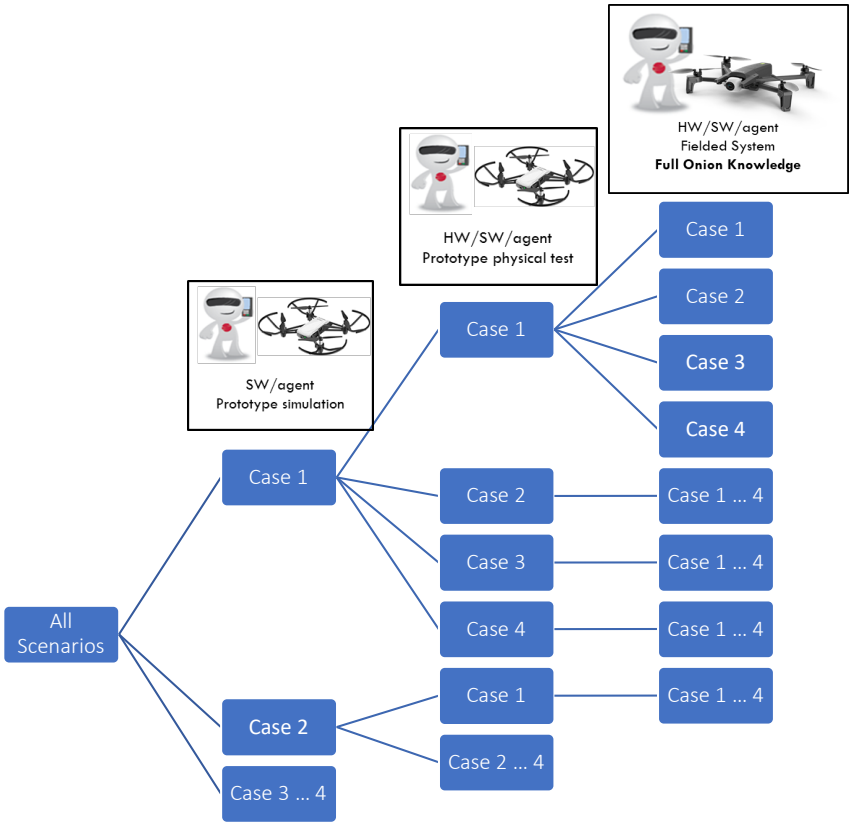

Figure 12: A Bayesian network for the detection case. Scenarios are divided into four cases (true/-false positive/negative) across three test cases (software environment, protoype/low-fidelity, fielded system).

vehicles within the denied area. Surveillance sensors including static infrared and video cameras and target characterization sensors, such as acoustic and seismic sensors, monitor the area to provide the operator with situational awareness regarding persons and vehicles. An unmanned aerial vehicle also provides surveillance and early warning information. Silverfish exists as a hybrid simulation/hardware emulation characterized in model-based systems engineering (MBSE) terms by a set of SysML models describing its architecture and functions from several perspectives. It also includes AI/ML models for detecting cyber attacks on the UAV.

7 Future Opportunities

This incubator project established the theory and methods for exploring how T&E requirements can and should change as a function of the test team knowledge of the technical specifications of an AI enabled systems. The incubator developed theory based in systems theory that captures changes in the systems and the state-space in which it operates through the concept of systems morphisms. The onion model describes different levels of system knowledge and a context for defining the abstraction of the system. The project experimented with two pilot scenarios to demonstrate how multiple phases of testing contribute to the evaluation of an AI enabled sys-

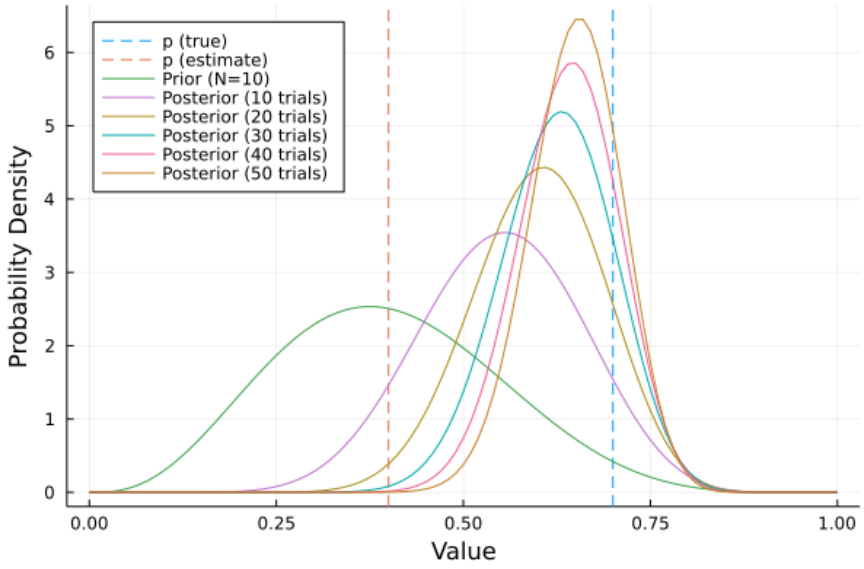

Figure 13: A Bayesian approach to estimating the probability of success in a binomial distribution; here the prior estimate of the probability of success is 0.4 and the true value is 0.7. The inference procedure updates the estimates as tests are conducted, closing in on the true value.

tems. Finally, we present the Bayesian analytical framework for combining information across the multiple phases of testing. This analytical framework also reflects the changing system configuration and context. In summary, this work essentially constitutes the building blocks for investigating the cost-benefit for test data collection on a realistic system in future phases.

A major challenge in conducting AI enabled systems research is that physical realizations are needed for T&E research. Future work could leverage the Silverfish testbed directly and expand the testbed into physical implementations. Physical implementations in addition to MBSE representations would enable the direct execution of a T&E program on the Silverfish testbed. Future work should also include purposefully varying the systems knowledge (based on the onion model), the complexity of the systems and its operating environments (number of morphisms), and determine minimally adequate testing as a function of those variables.

References

[1] T. A. McDermott, M. R. Blackburn, and P. A. Beling, "Artificial intelligence and future of systems engineering," in *Systems Engineering and Artificial Intelligence*. Springer, 2021, pp. 47–59.

[2] T. Cody, S. Adams, and P. A. Beling, "A systems theoretic perspective on transfer learning," in *2019 IEEE International Systems Conference (SysCon)*. IEEE, 2019, pp. 1–7.

[3] L. Von Bertalanffy and J. W. Sutherland, "General systems theory: Foundations, developments, applications," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 6, pp. 592–592, 1974.

[4] A. W. Wymore, *A mathematical theory of systems engineering: the elements*. Wiley, 1967.

[5] E. A. Bjorkman, S. Sarkani, and T. A. Mazzuchi, "Using model-based systems engineering as a framework for improving test and evaluation activities," *Systems Engineering*, vol. 16, no. 3, pp. 346–362, 2013.

[6] A. W. Wymore, *Model-based systems engineering*. CRC press, 2018.

[7] P. Wach, B. P. Zeigler, and A. Salado, "Conjoining wymore's systems theoretic framework and the devs modeling formalism: Toward scientific foundations for mbse," *Applied Sciences*, vol. 11, no. 11, p. 4936, 2021.

[8] B. P. Zeigler, *Theory of Modeling and Simulation: Discrete Event & Iterative System Computational Foundations Ed. 3*. Elsevier Science, 2018.

[9] C. B, "A preliminary design-phase security methodology for cyber-physical systems," vol. 7, p. 21, 2019.

[10] A. Salado and H. Kannan, "A mathematical model of verification strategies," *Systems Engineering*, vol. 21, no. 6, pp. 593–608, 2018.

[11] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[12] C. H. Fleming, C. Elks, G. Bakirtzis, S. Adams, B. Carter, P. Beling, and B. Horowitz, "Cyber-physical security through resiliency: A systems-centric approach," *Computer*, vol. 54, no. 6, pp. 36–45, 2021.

[13] P. Beling, B. Horowitz, C. Fleming, S. Adams, G. Bakirtzis, B. Carter, T. Sherburne, C. Elks, A. Collins, and B. Simon, "Model-based engineering for functional risk assessment and design of cyber resilient systems," University of Virginia Charlottesville United States, Tech. Rep., 2019.