Assurance-based Learning-enabled Cyber-Physical Systems

Gabor Karsai, Vanderbilt University

Contributions from Xenofon Koutsoukos, Taylor Johnson, Ted Bapty and team

Supported by DARPA AA Program

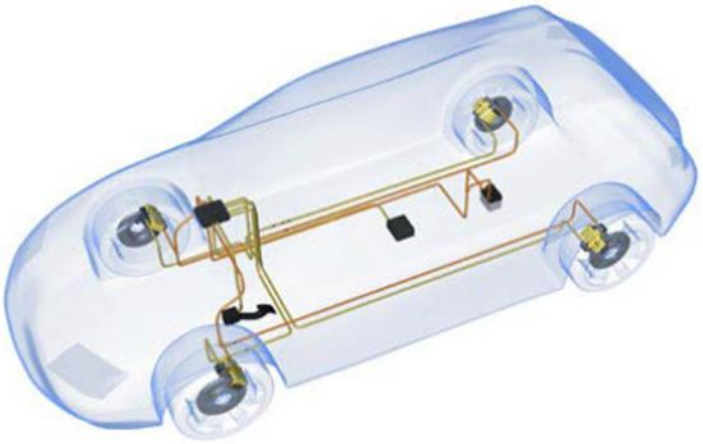
DARPA – Assured Autonomy Program Goal

Develop rigorous design and analysis technologies for continual assurance* of learning-enabled autonomous systems, in order to guarantee safety properties in adversarial environments

**assurance: a positive declaration intended to give confidence*

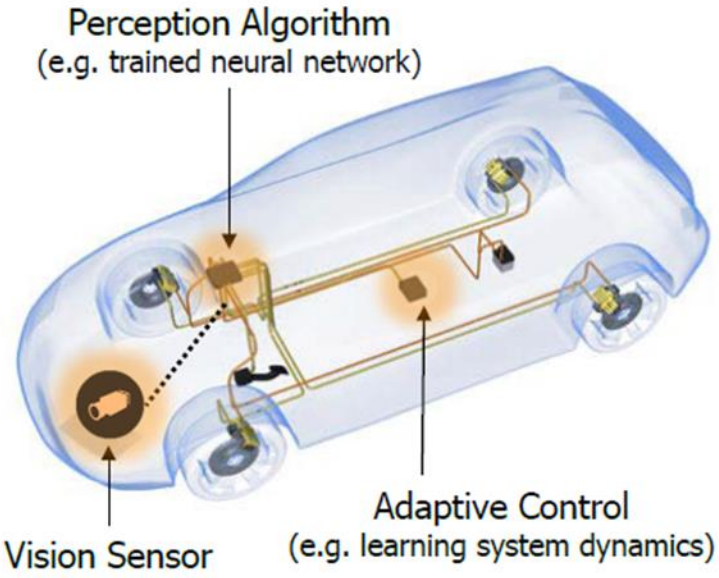
DARPA
Illustrating the challenge

Non-Learning System
(e.g. manual brake-by-wire)



Safety assurance
can be provided

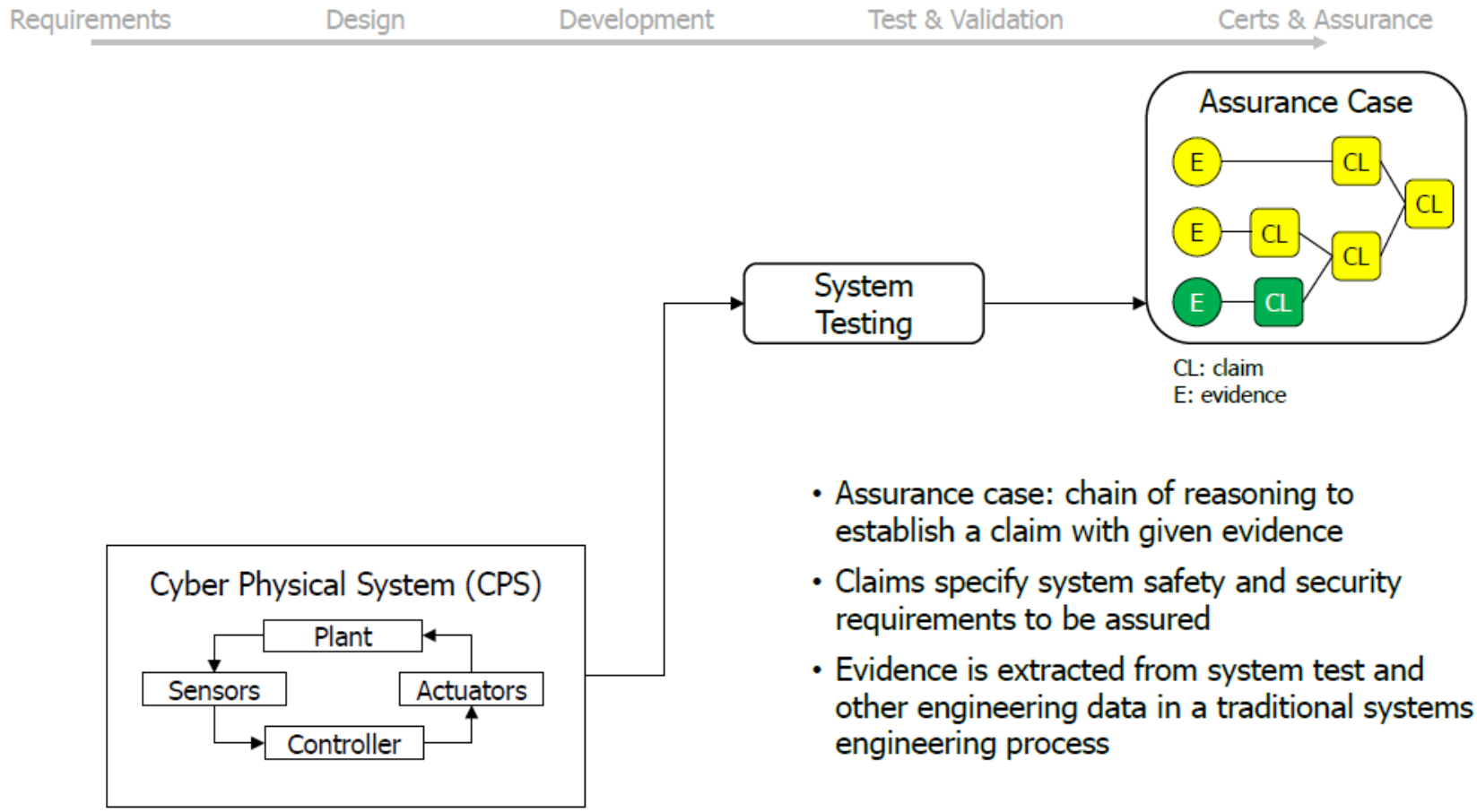
Learning-Enabled Autonomous System
(e.g. automated brake-by-wire for collision avoidance)



Safety assurance
can NOT be provided

DARPA

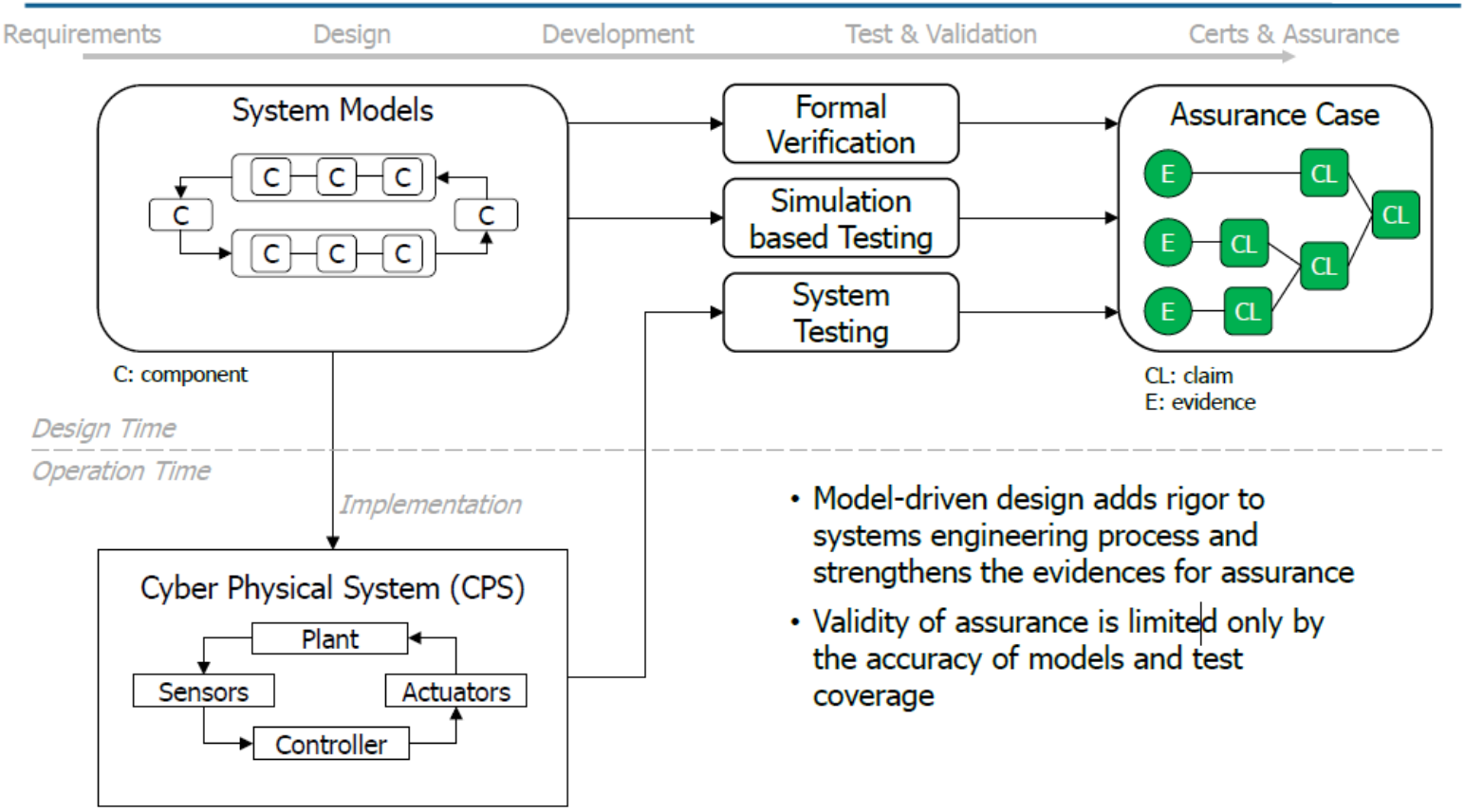
Safety Assurance for Systems -State of Practice



- Assurance case: chain of reasoning to establish a claim with given evidence
- Claims specify system safety and security requirements to be assured
- Evidence is extracted from system test and other engineering data in a traditional systems engineering process

DARPA

Model-driven Design for Safety Assurance -State of Art

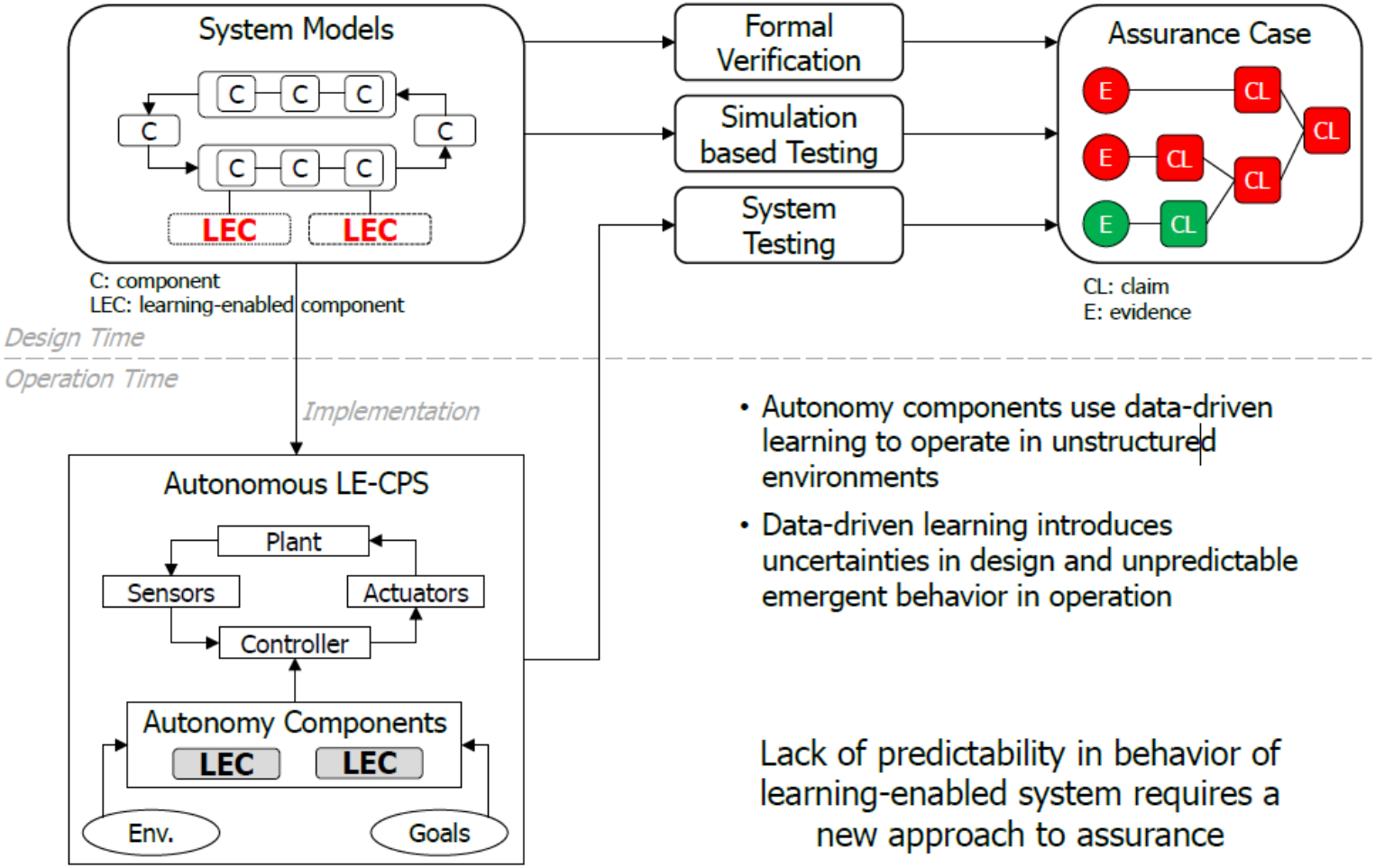


- Model-driven design adds rigor to systems engineering process and strengthens the evidences for assurance
- Validity of assurance is limited only by the accuracy of models and test coverage

Applicable only to non-learning systems operating in well-characterized environments

DARPA

Learning-Enabled Autonomous Systems Lack Safety Assurance



- Autonomy components use data-driven learning to operate in unstructured environments
- Data-driven learning introduces uncertainties in design and unpredictable emergent behavior in operation

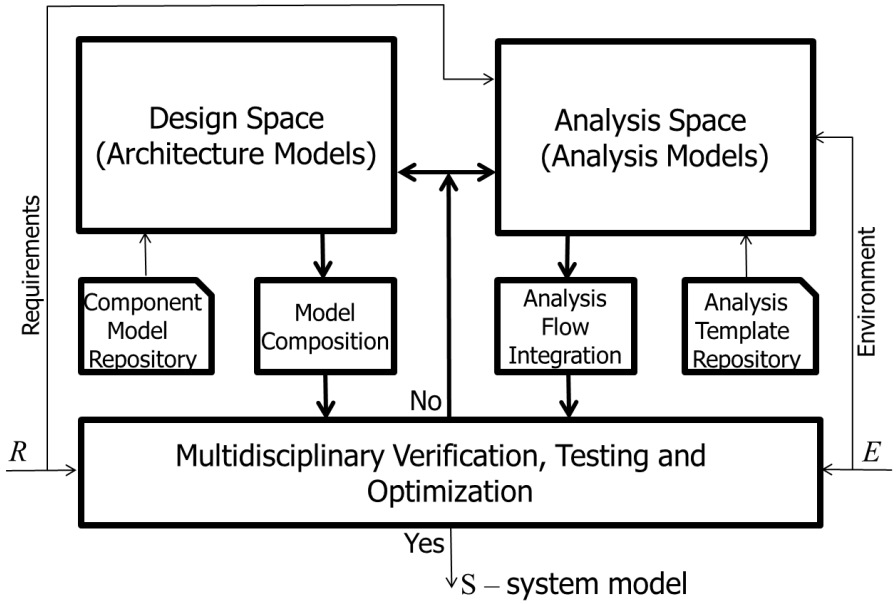
Lack of predictability in behavior of learning-enabled system requires a new approach to assurance

ALC Project

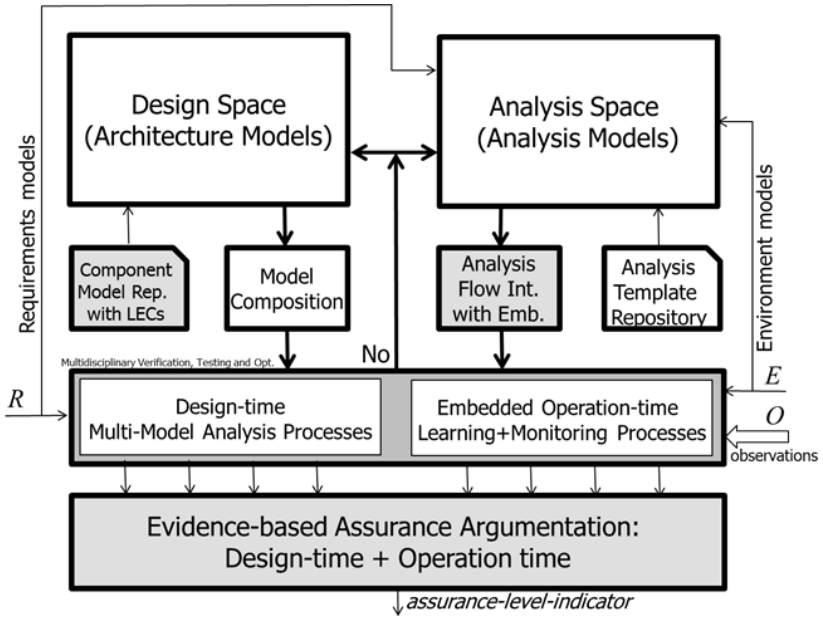
VU ALC Project vision

"The proposed research effort will address the … technical areas with overall goal of delivering an integrated design tool suite and reusable operation support components for constructing autonomous CPS including Learning Enabled Components (LECs). Our vision is to … create a new design flow that extends from design-time to operation time, re-interprets the traditional assurance argumentation to become a dynamic, operational concept. Our ultimate goal is to establish a fusion of model- and component-based methods with data-driven methods."

Model-driven design flow



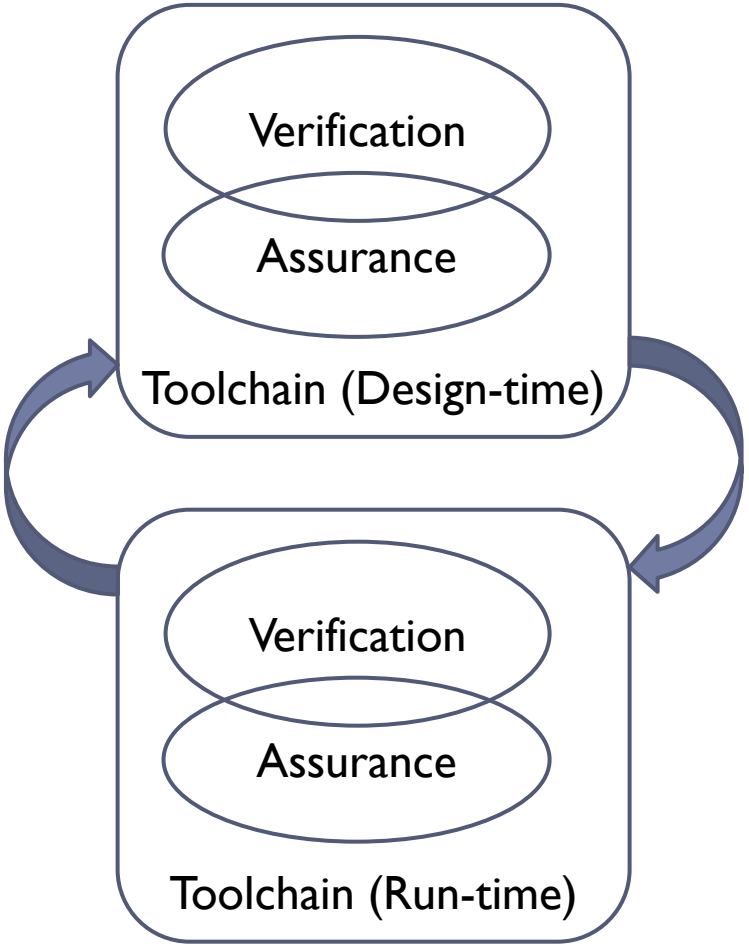
Model-driven design flow with LEC-s



Project activities

- Thrusts:
  - Verification: formal and/or coverage-driven verification of safety and liveness properties of components, subsystems, and systems, at design-time and at run-time, to provide evidence for assurance arguments
  - Assurance: construction and continuous evaluation of logical arguments that demonstrate the *truth* or *strength* of a safety claim based on available evidence
  - Toolchain: design-time and run-time software tools to implement and support the above, for real systems

- Learning (component adaptation)
  - Design-time: in design tools, while the system is not operational
  - Run-time: in the running system, on-the-fly
  - Mixed – learning from operational, 'overnight'



Challenge Problem Domain: Autonomous Underwater Vehicles ([A|U]UV)

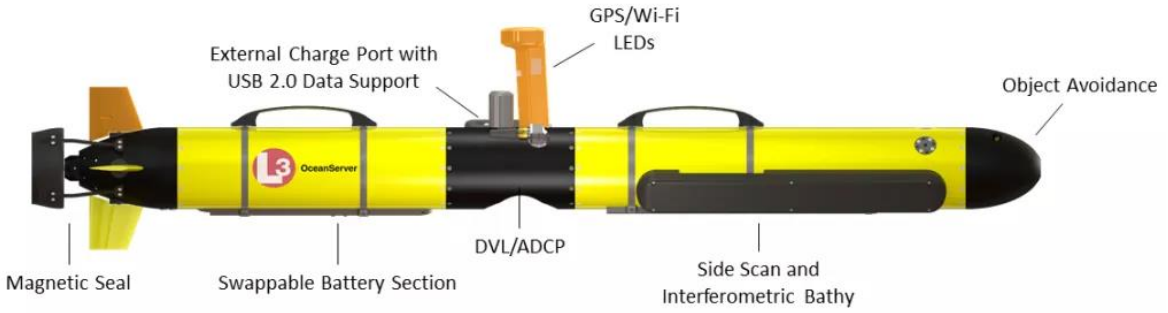
- Mobile robot that operates underwater
- Applications:
  - Commercial: pipeline inspection, renewable energy, ports monitoring, off-shore drilling, etc.
  - Research: marine biology, geology, hydrographic survey, etc.
  - Defense: mine countermeasures, reconnaissance, search, etc.

Source: https://ocean-server.com/

AUV as a CPS with LEC-s

- Learning-Enabled Components:
  - LEC Control:
    - Input: pipe distance (L/R) + orientation, heading, speed, ... (8 variables)
    - Output: Desired heading and speed
  - LEC Perception:
    - Input: sonar 'image' (2X), constructed from scanlines
    - Output: estimated pipe distance/orientation
- Control objective:
  - (1) Avoid collision, (2) Follow the pipe

L3 OceanServer IVER3



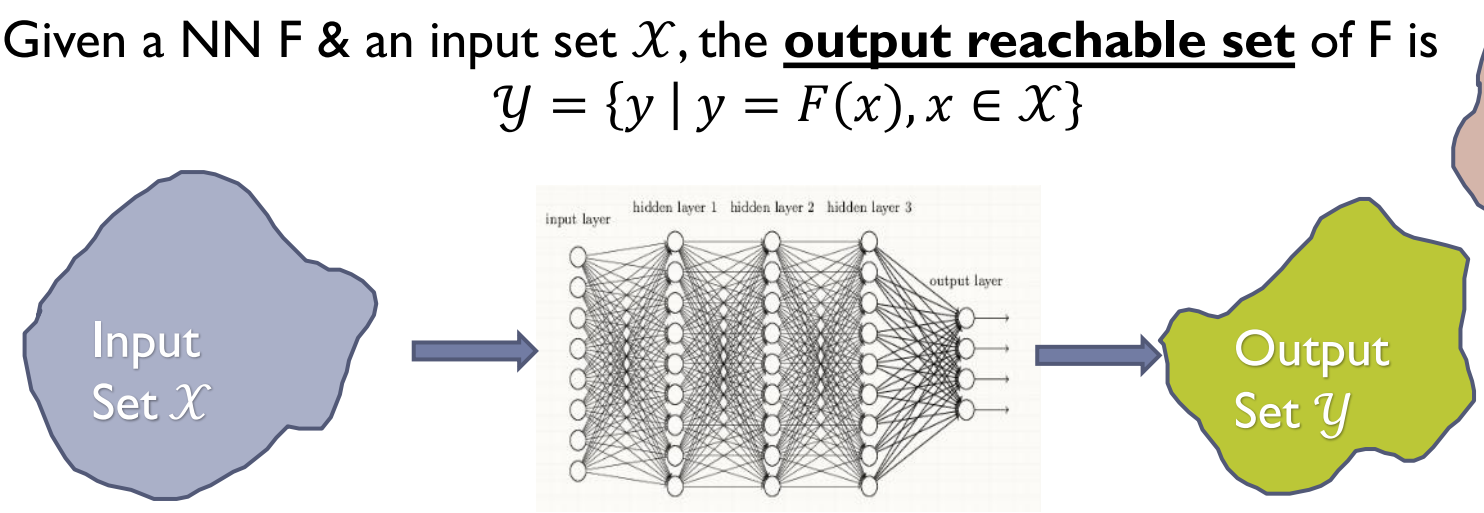
Verification Technology

Prof. Taylor Johnson and team

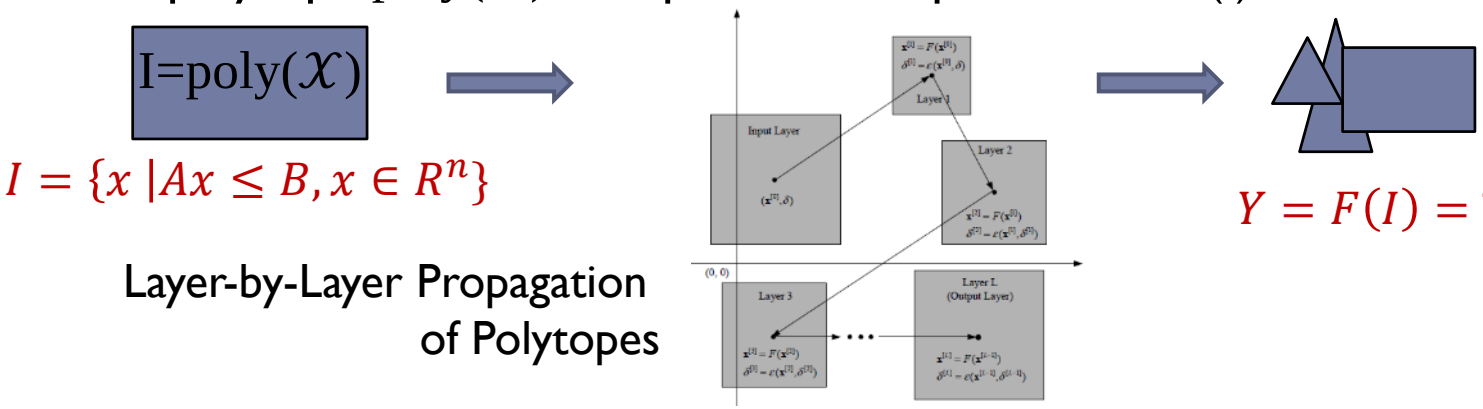
LEC Verification: Reachability Analysis of Feedforward/Convolutional Neural Networks

- Given a NN F & an input set $\mathcal{X}$, the **output reachable set** of F is

$$\mathcal{Y} = \{y \mid y = F(x), x \in \mathcal{X}\}$$

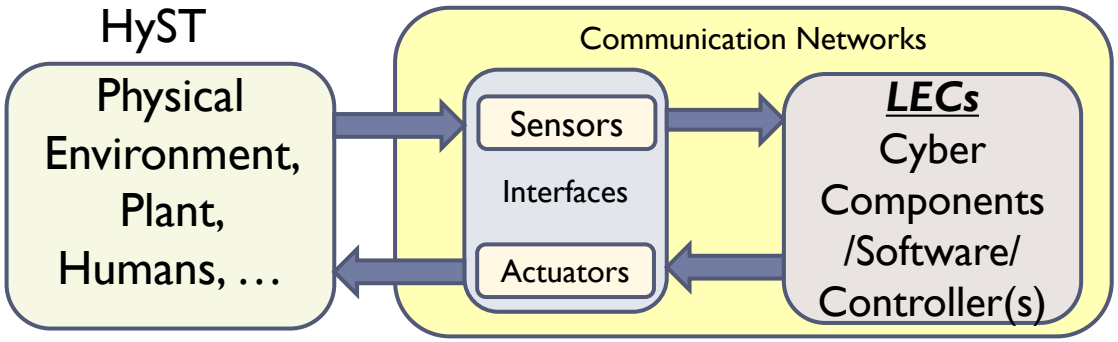


- Computationally: Given a NN F, a convex initial set of inputs I represented as a polytope $\text{poly}(\mathcal{X})$, compute the output set Y = F(I) of the network



Layer-by-Layer Propagation of Polytopes

Closed-Loop CPS with LECs: Verification Flow and Tools

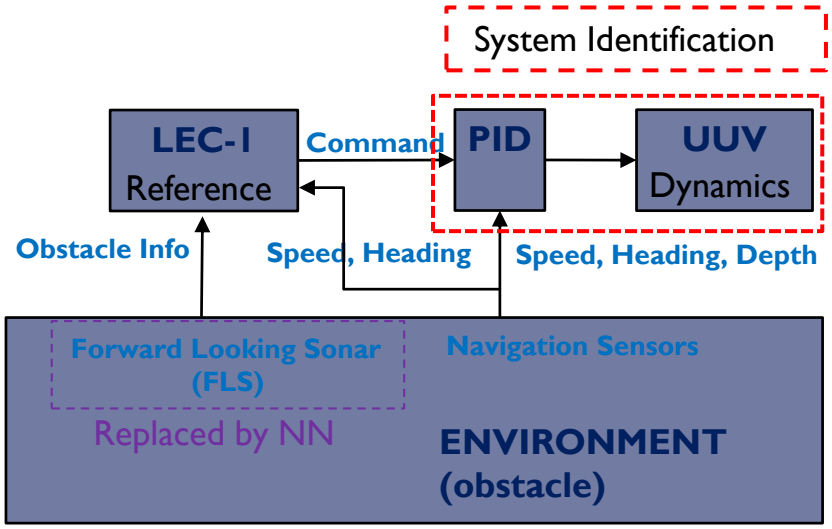


- Plant models: **hybrid automata**, or networks thereof, represented in HyST/SpaceEx/CIF formats
  - Hybrid automaton: **finite state machine** + set of real-valued variables that evolve continuously over intervals of real time according to **ordinary differential equations (ODEs)**
  - **Hybrid** behaviors: discrete transitions and continuous trajectories over real time
  - Plant dynamics: linear, nonlinear, hybrid, continuous-time, discrete-time, …
- LEC and cyber models: for now, feedforward **neural networks**, represented in **ONNX** format (compatible with Keras, Tensorflow, Matlab, etc.)
  - ReLUs, CNNs (max pool, etc.), tanh, sigmoid, …
- Specifications: primarily **safety properties** for now, some **reachability properties**
- Verification: composed LEC and plant analysis: autonomous closed-loop CPS
  - **Bounded model checking**: k control periods, alternating reachability analysis of controller and plant

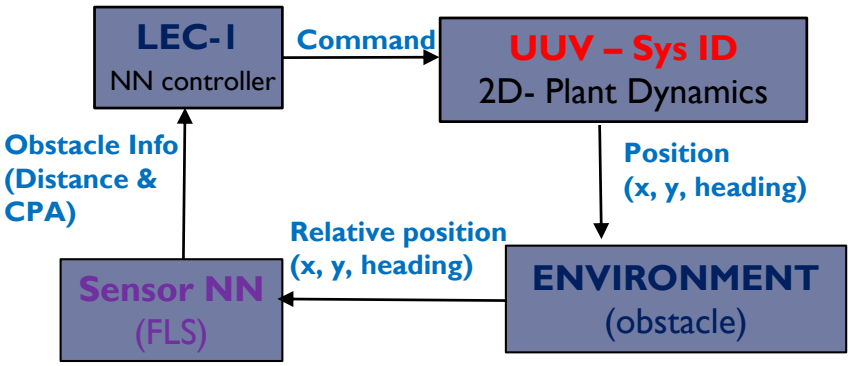
UUV Closed-Loop Verification [WAAS'20]

Verification Setup for Obstacle avoidance

Simulation (ROS-Gazebo)



Verification (NNV)

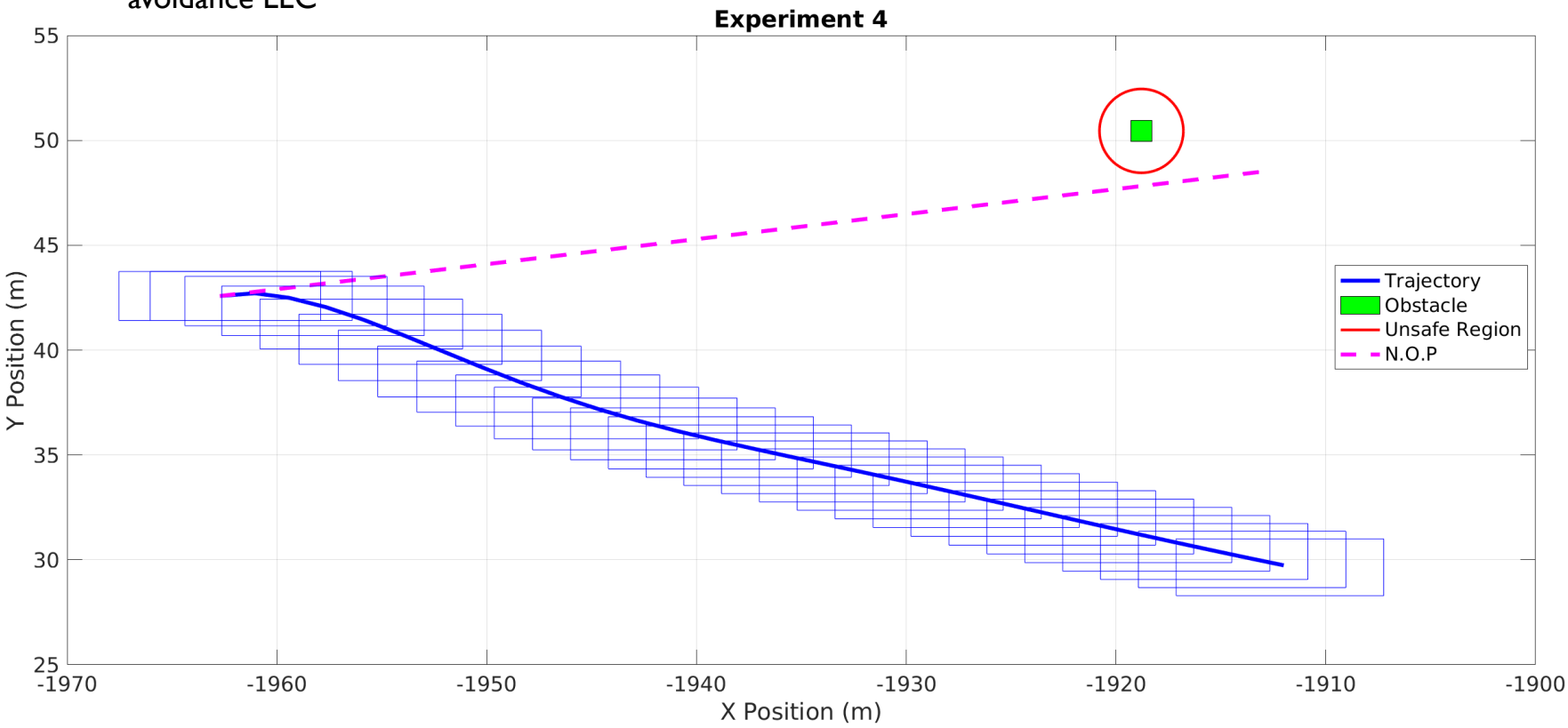


UUV Closed-Loop Verification Results

Evaluate 4 Scenarios
- 3 Safe
- 1 Unknown
- N.O.P.: without collision avoidance LEC

Safety property
Distance from the vehicle to the center of obstacle > 2m

Experiment 4
- Safe
- 24 seconds
- Exact + over-approx methods

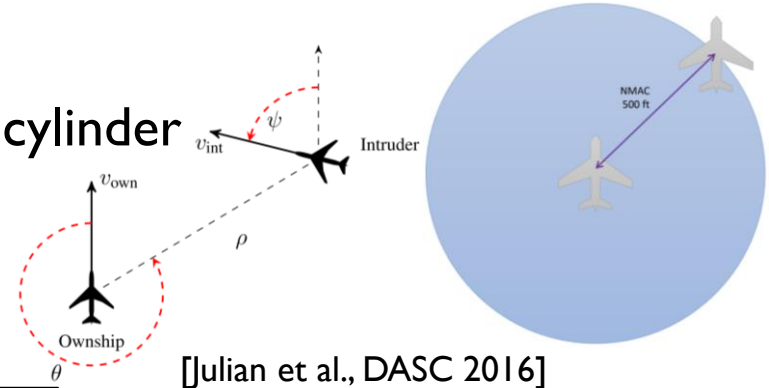


ACAS-Xu Closed-Loop Scenario

Maintain safe separation between aircraft

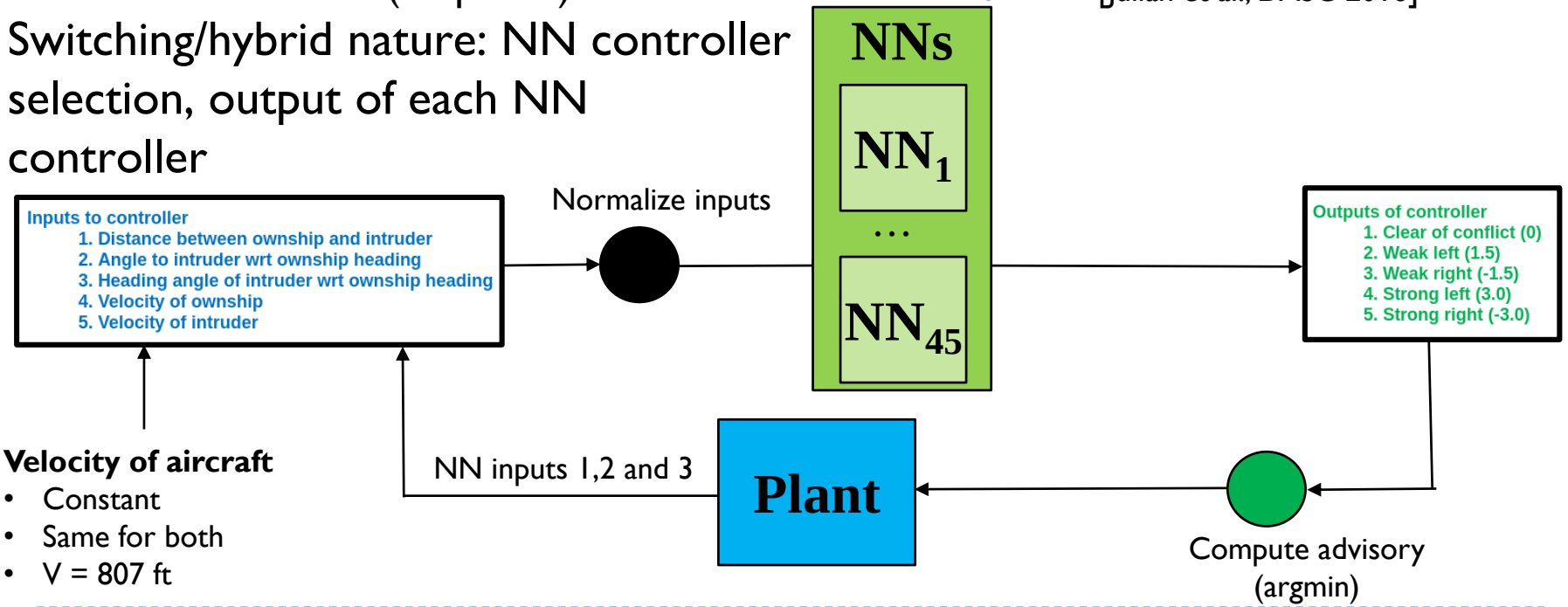
Do not violate Near Midair Collision (NMAC) cylinder

- 100 ft above or below (200 ft cylinder height)
- 500 ft radius (1000 ft diameter)
- Plant model: Dubins dynamics, constant velocity, constant altitude (co-planar)

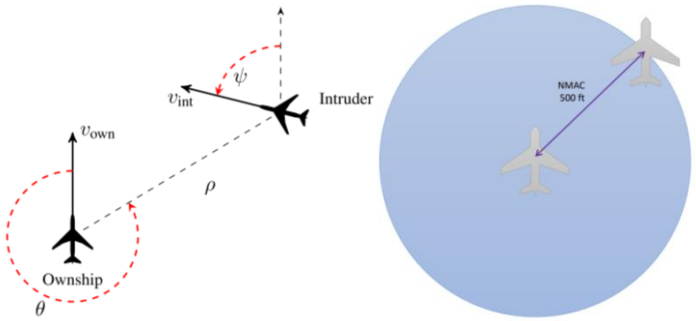


[Julian et al., DASC 2016]

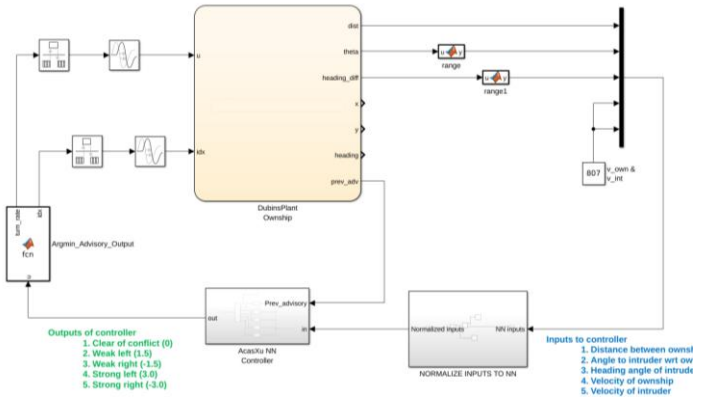
Switching/hybrid nature: NN controller selection, output of each NN controller

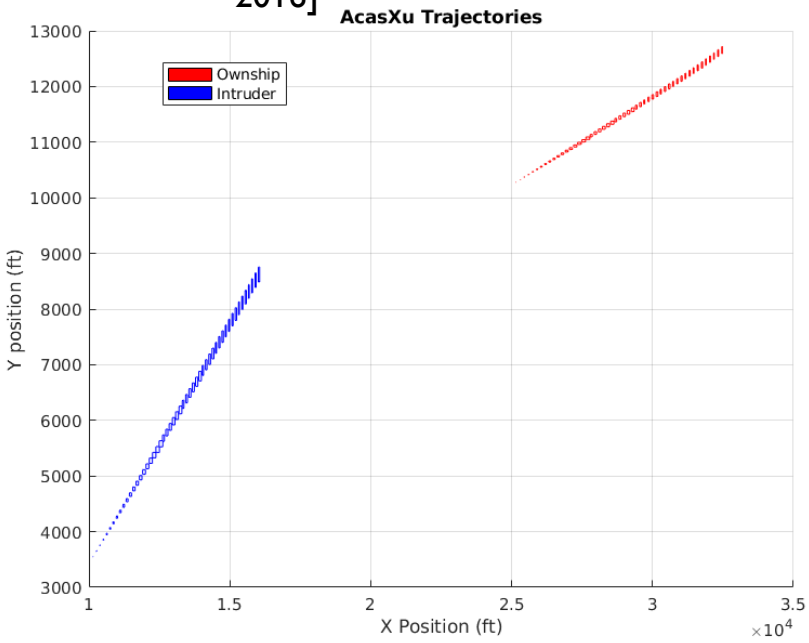


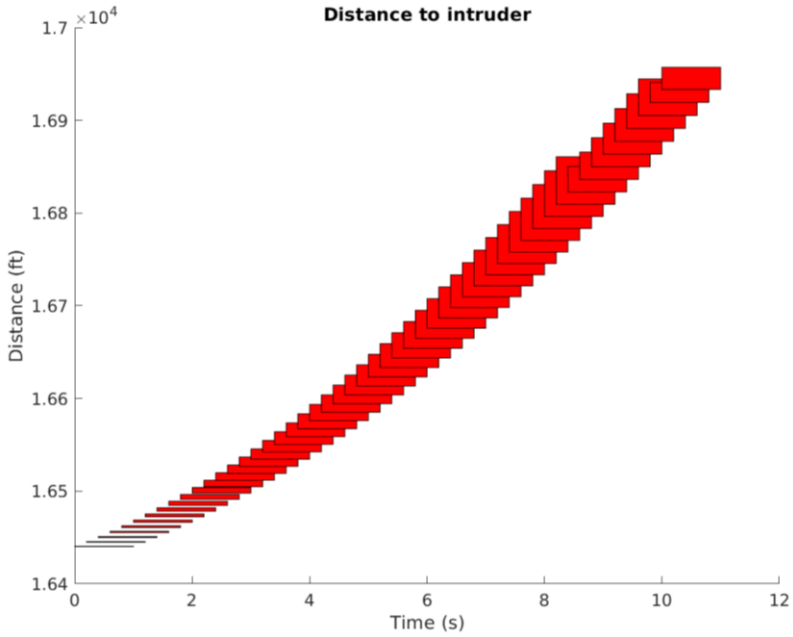
ACAS-Xu Closed-Loop Verification Results



[Julian et al., DASC 2016]

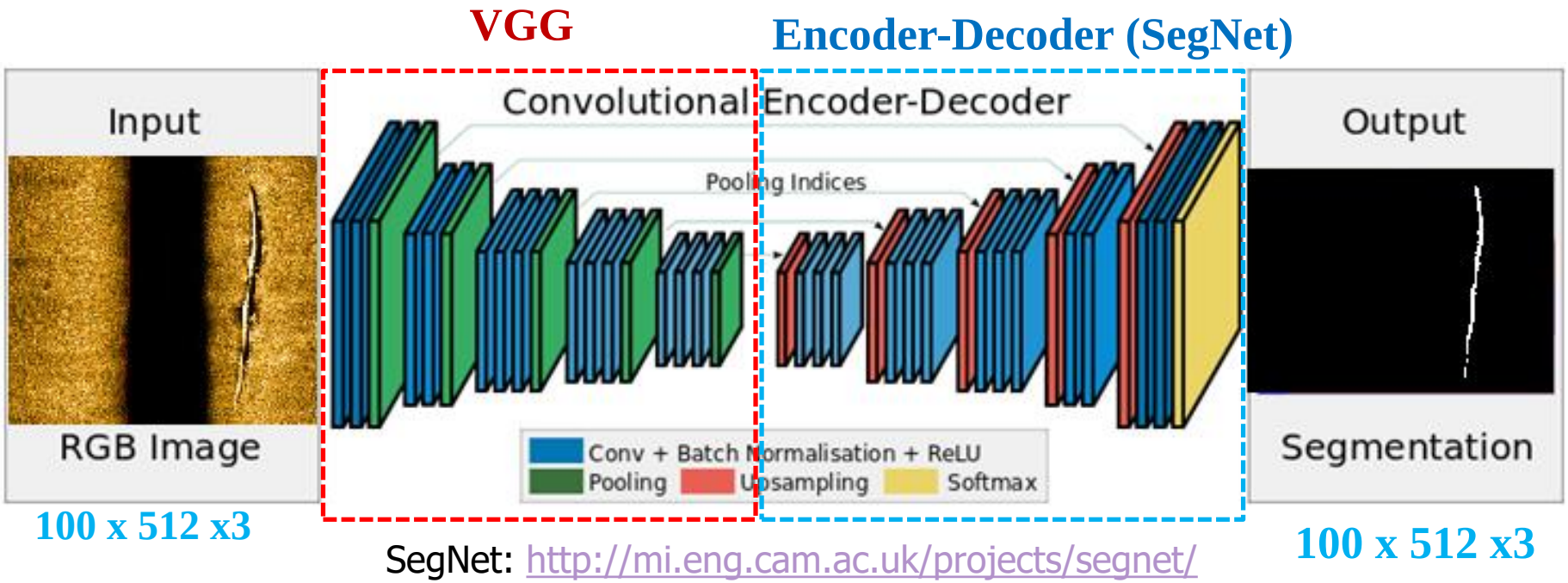






Perception Robustness Verification

- Target application: Perception LE Component



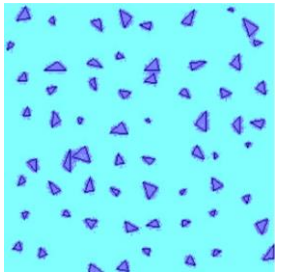
SegNet: http://mi.eng.cam.ac.uk/projects/segnet/

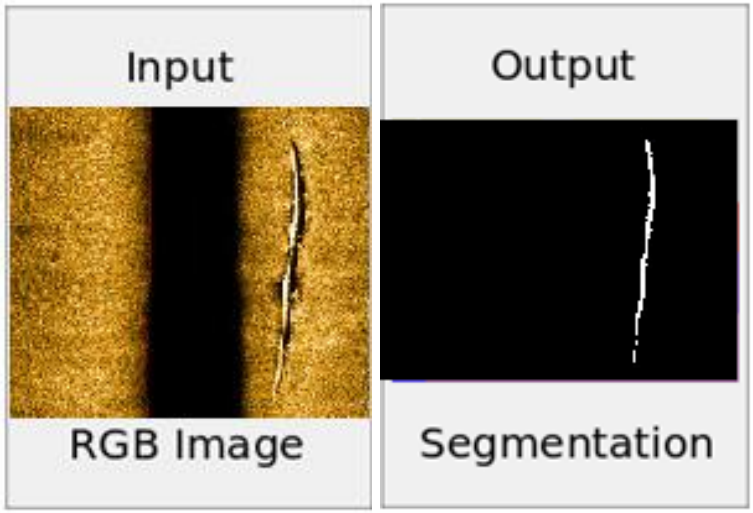
Representative robustness question:

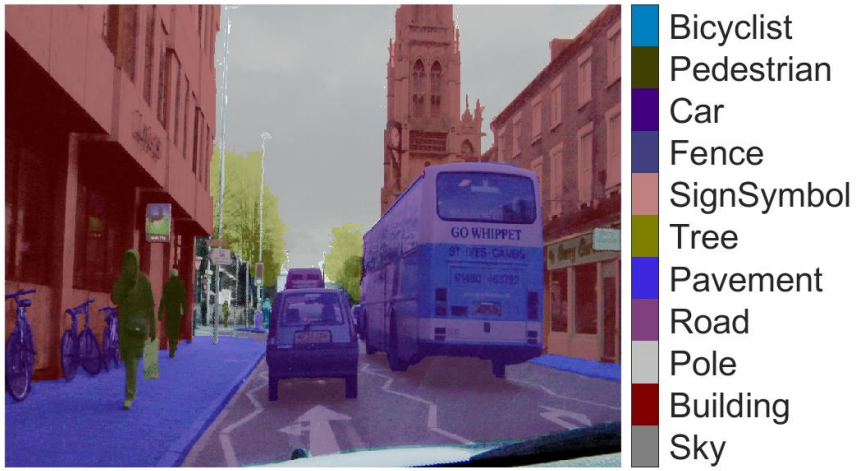
Under an adversarial attack, what is the percentage of pixels that are correctly (and robustly) classified?

Semantic Segmentation Robustness

- Representative data sets:
  - M2NIST: multi-digit MNIST
  - CamVid
  - UUV pipe segmentation







Semantic Segmentation Robustness

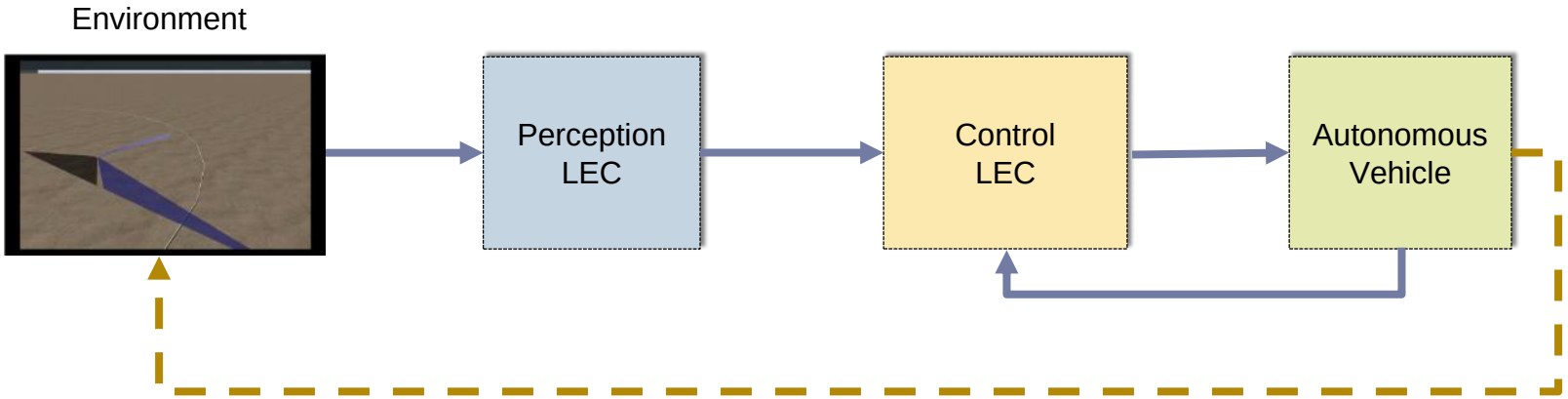
- Semantic segmentation
  - Input: image of width W pixels, height H pixels, and N color channels
  - Output: image specifying class $c \in C$ of every pixel
  - $F: \mathbb{R}^{W \times H \times N} \rightarrow C^{W \times H}$

- Robustness for semantic segmentation
  - Extended from classification robustness
  - For a small perturbation of an input image, does the class c of any pixel change (and if so, which ones and how many)?
  - Builds on set-based reachability analysis: again consider a set of input images as an ImageStar, and determine the output reachable set of F
    - More precisely: $F: \mathbb{R}^{W \times H \times N} \rightarrow \mathbb{R}^{W \times H \times C}$ and then take argmax (softmax) over C
  - Output space is significantly more complicated, as for classification, it was just C^1 (or more precisely, $\mathbb{R}^C$ prior to taking the argmax/softmax to determine the class)
  - Major challenge to overcome is handling this upsampling process from the latent space, accomplished e.g. through dilated/transposed convolutions/deconvolutions or unpooling layers

Assurance Monitoring Technology

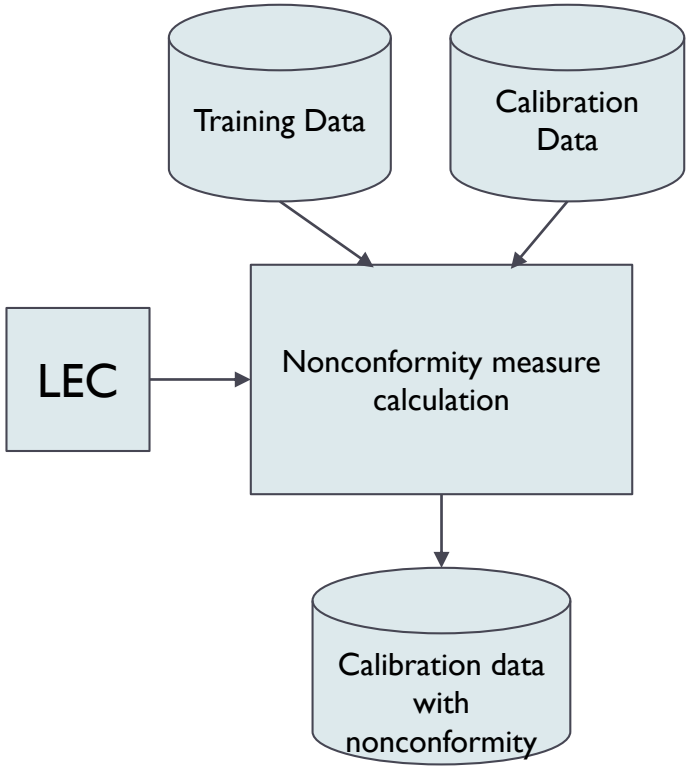
Prof. Xenofon Koutsoukos and team

Assurance Monitoring



- **Assurance Monitoring Based on Inductive Conformal Prediction**
  - Characterize how close the LEC behavior is to a model that represents the *expected safe behavior* obtained during the training phase.
  - Compute measures of confidence associated with predictions from LECs
  - *Nonconformity measure* is used to evaluate the degree to which a new example disagrees from a set of examples
  - Confidence is computed based how different is a test example compared to a set of calibration examples

Inductive Conformal Prediction (ICP)



- **Nonconformity measure**: A function that measures the disagreement between the actual label and the prediction using the neural network

1. Split the training set into
   - The proper training set
   - The calibration set
2. Use the proper training set to train the neural network
3. For each example in the calibration set:
   - Supply the input to the trained neural network to obtain the prediction
   - Calculate the nonconformity scores
   - Sort the calibration examples using descending order of the **nonconformity scores** in the set A
4. For each new example, compute the fraction of examples that are equally or more nonconforming (p-values)
5. Compute a predictor with a given confidence based on the p-values

Shafer, G., & Vovk, V. (2008). A tutorial on conformal prediction. *Journal of Machine Learning Research*, *9*(Mar), 371-421.

Anomaly Detection

- The nonconformity measure can be used to evaluate the degree to which a new example disagrees from a set of examples
- For test examples, we compute the fraction of nonconformity scores for the calibration data that are larger than the nonconformity score of the test input (**empirical *p*-value**)
- If the empirical p-value < ε the example is classified as a conformal anomaly
- There are at least three explanations for a conformal anomaly
  - A rare or previously unseen example from the same probability distribution as the training set
  - A true novelty not generated from the same probability distribution as the training set
  - The training examples are *not* IID

Laxhammar, R. & Falkman, G. , Inductive conformal anomaly detection for sequential detection of anomalous subtrajectories, *Annals of Mathematics and Artificial Intelligence*, (2015) 74: 67.

Online Out-of-Distribution Detection in Multidimensional Time Series

- In CPS, examples arrive one by one and after observing each new example, we would like to quantify the degree to which the examples disagree with the training data
- If the examples are IID, the inductive conformal anomaly detection algorithm produces p-values that are independent and uniformly distributed in $[0,1]$
- Out-of-distribution detection can be performed by testing the hypothesis that p-values that are independent and uniformly distributed in $[0,1]$

V. Fedorova, A. Gammerman, I. Nouretdinov, and V. Vovk. 2012. Plug-in martingales for testing exchangeability on-line. In Proceedings of the 29th International Conference on International Conference on Machine Learning (ICML'12).

Exchangeability Martingales

- Given the sequence of p-values, a martingale is calculated as a function of the p-values
  - Power martingale

$$M_n^\varepsilon = \prod_{i=1}^{n} \varepsilon p_i^{\varepsilon-1}$$

  - Simple mixture martingale

$$M_n = \int_0^1 M_n^\varepsilon \, d\varepsilon$$

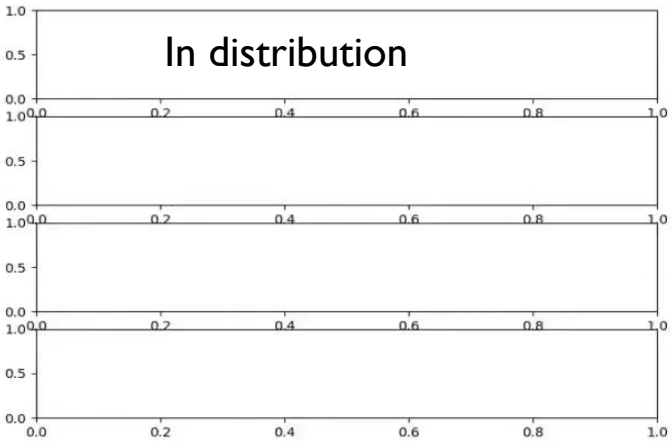
- The value of the martingale reflects the strength of evidence *against the exchangeability assumption*, i.e. that the examples are generated from the same probability distribution independently
- Such a martingale will grow only if there are many small p-values in the sequence
- If the generated p-values concentrate in any other part of the unit interval, we cannot expect the martingale to grow

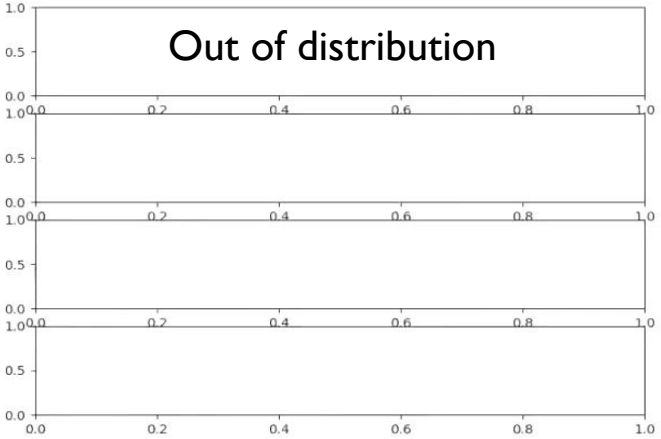
V. Fedorova, A. Gammerman, I. Nouretdinov, and V. Vovk. 2012. Plug-in martingales for testing exchangeability on-line. In Proceedings of the 29th International Conference on International Conference on Machine Learning (ICML'12).

Nonconformity Measure (NCM)

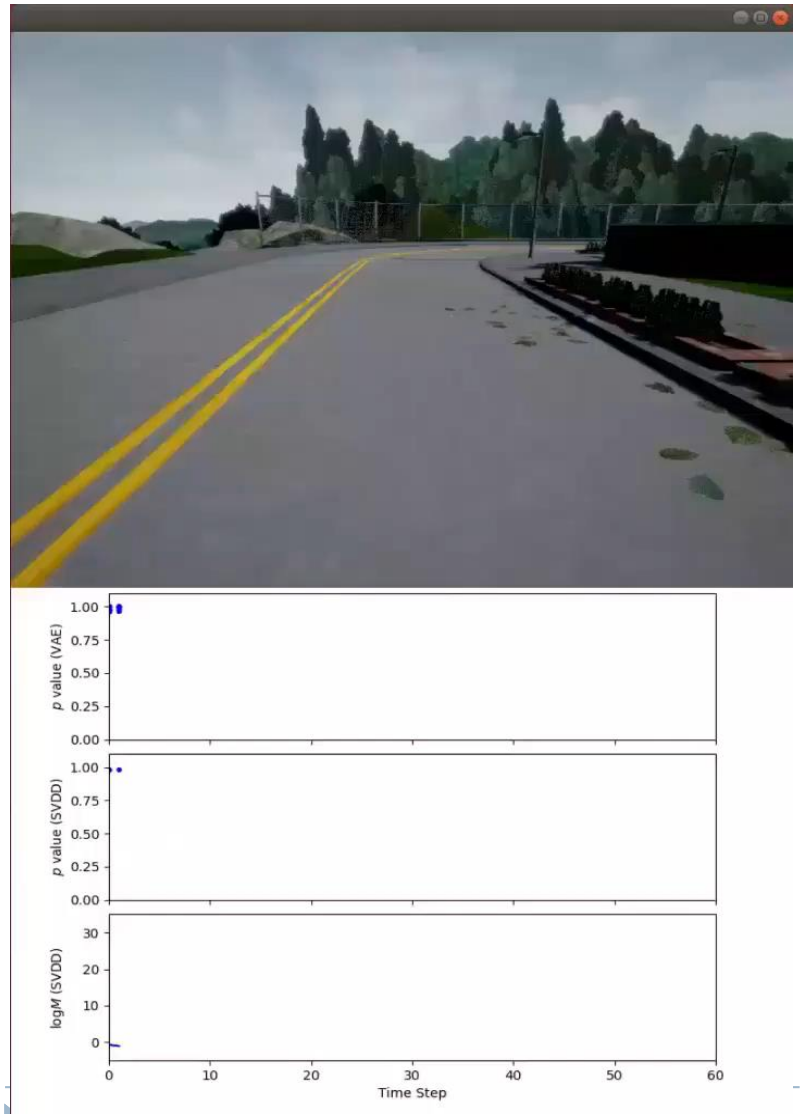
- Computing the NCM using k-nearest neighbors requires storing the training data which may be infeasible for autonomous CPS
  - Need methods for reducing the memory and time requirements
- Train an appropriate neural network architecture which can be used to compute efficiently the NCM
- 1. Autoencoders
  - Use the reconstruction error as the NCM
  - Based on current experiments, the method is not robust
- 2. Variational autoencoders
  - Use the generative model to sample multiple IID examples for the input of the current time step
  - Use the reconstruction error (probability) as the NCM
- 3. Deep One-Class Classification
  - Deep Support Vector Description (SVDD)

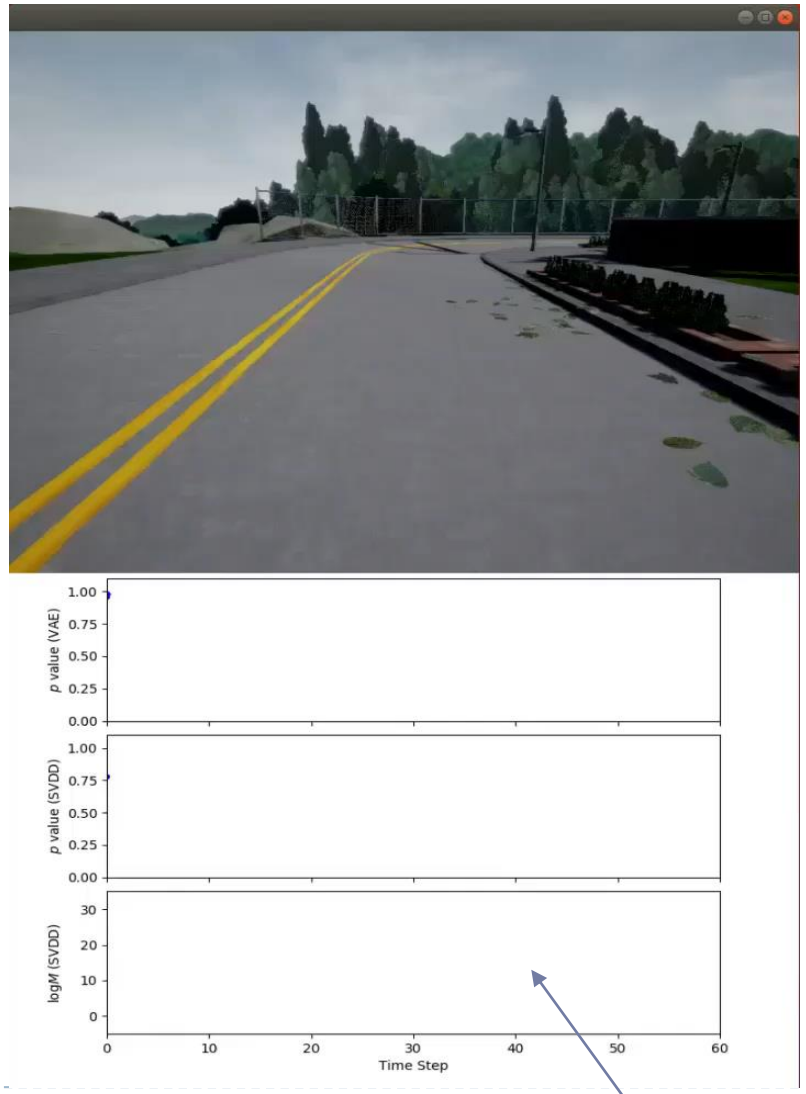
Assurance Monitoring
Distribution Shift Detection Example





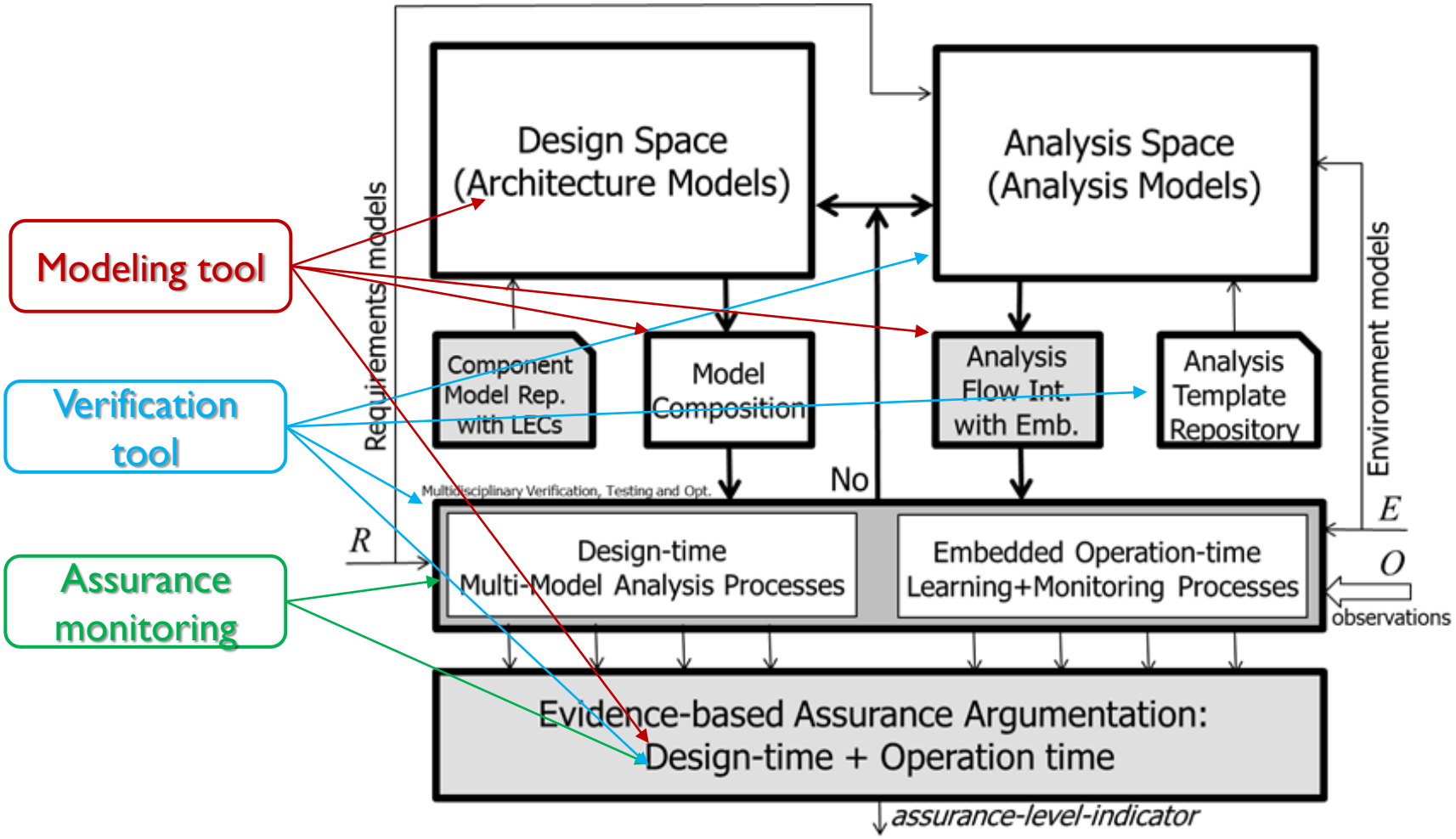
Assurance Monitoring: Distribution Shift Detection in adversarial scenarios





ALC Toolchain

Tool architecture - coverage



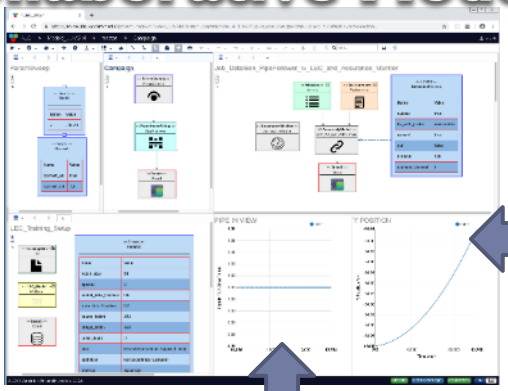
ALC Toolchain
Our approach

ALC Workflows

Collaborative Modeling

System Integrator

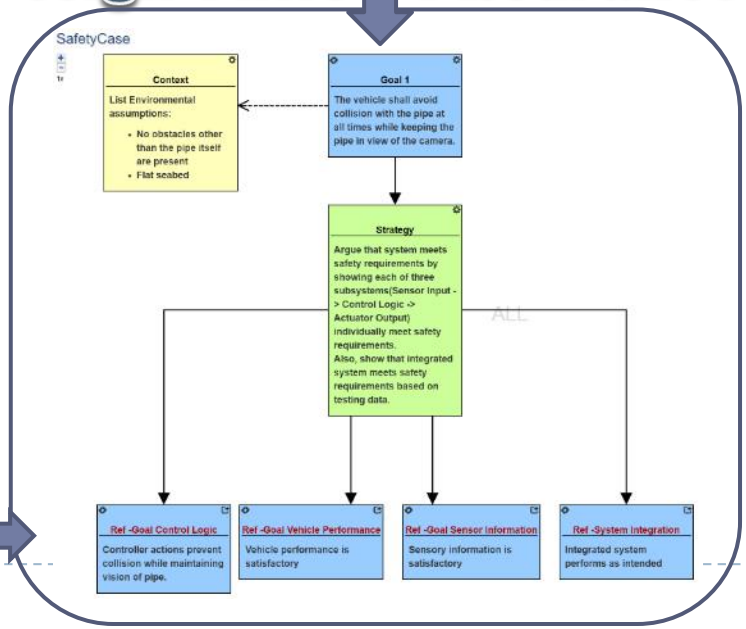
LEC Developer



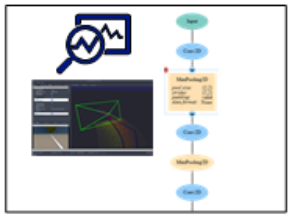
Design Time Assurance

- *The model driven toolchain supports training, verification and design-time assurance of learning enabled components.*
- *Toolchain helps with developing safety assurance cases for the system using collected evidence.*
- *Complete provenance tracking of experimental runs and data collection is supported.*

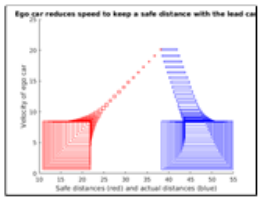
Assurance Engineer



Execution, Training, Data Collection, Verification

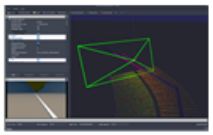


{d} Run System with LEC



{c} Verify Closed-Loop System With LEC

{b} Train LECs

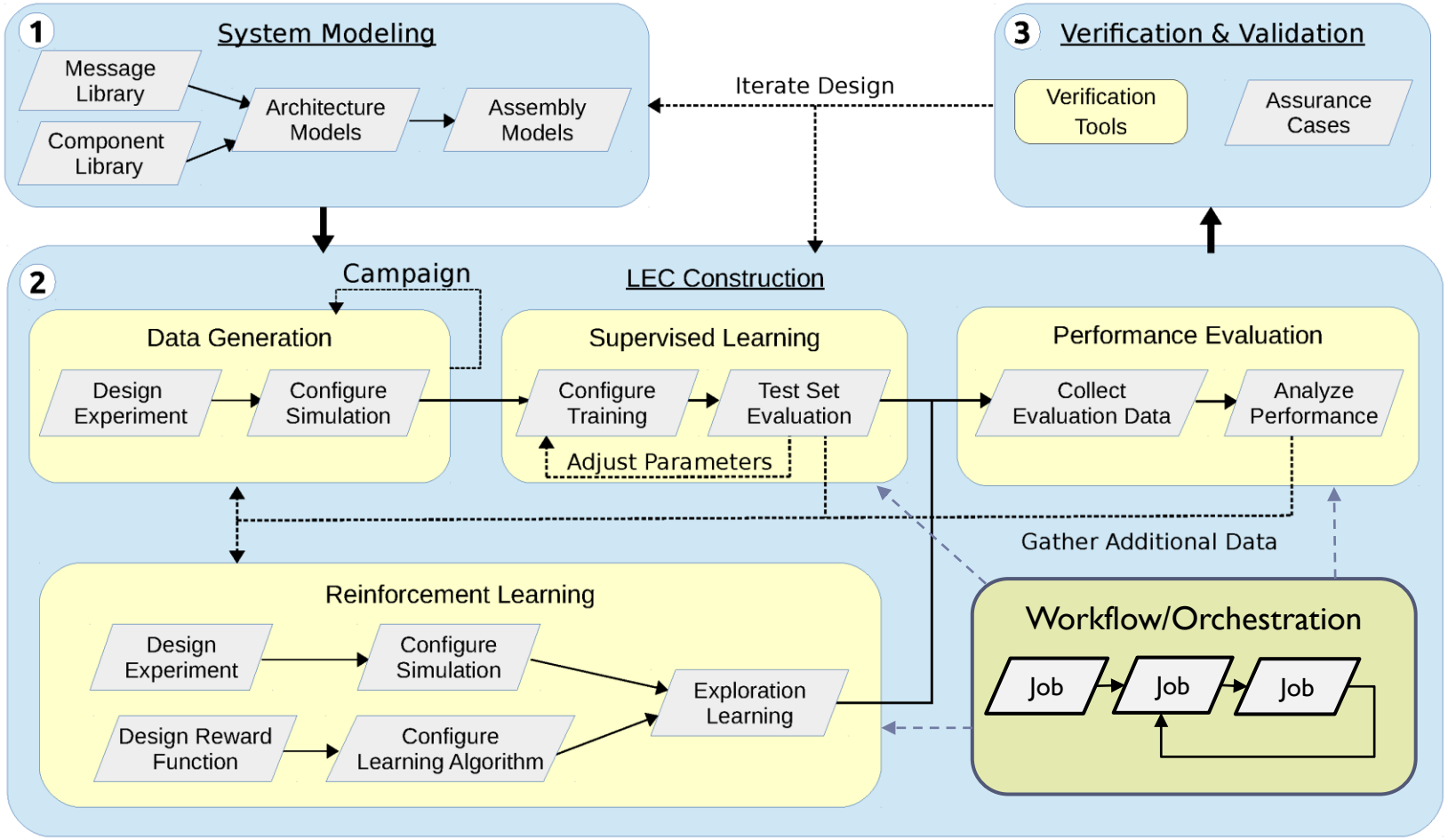


{a} Run Scenarios To Collect Data

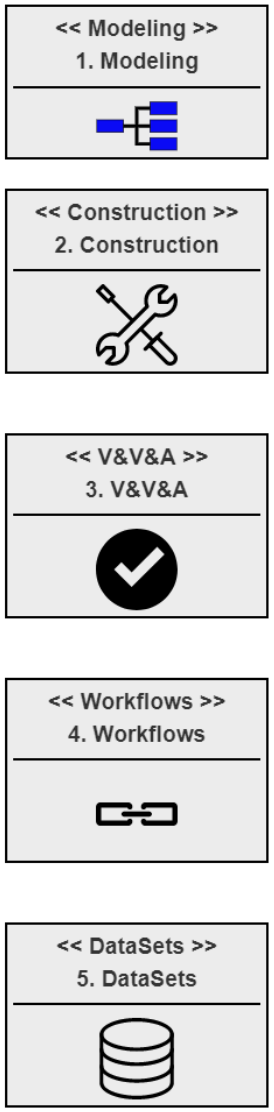
Typical Workflow Sequence

ALC Design Workflow

- Specialized for LEC development



Modeling Blocks, Systems, Training, & Execution



Model Systems
- Block Library
- Messages/Datatypes for Software
- System Structure

Construct Experiments
- Data Collection
- LEC Training
- Assurance

Verification, Validation, and Assurance
- Formal System Verification
- LEC Validation
- Assurance Argument Modeling

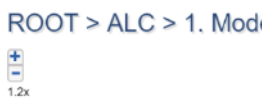
Workflows:
- Create/Execute Sequences of Operations

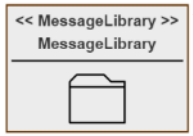
DataSets:
- Maintain Data Created via Experiment Constructions and Workflows
- Track Data Provenance
- Launch analysis of data

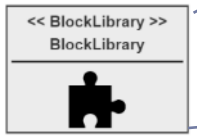
System Modeling

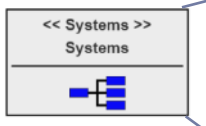
Data Models, Messages

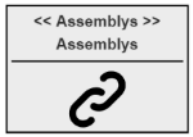
Components:
Hardware, Software/LEC

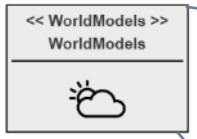


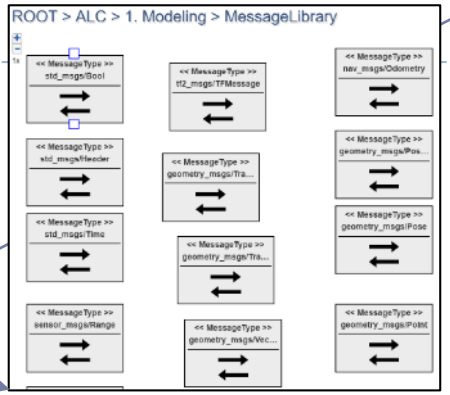


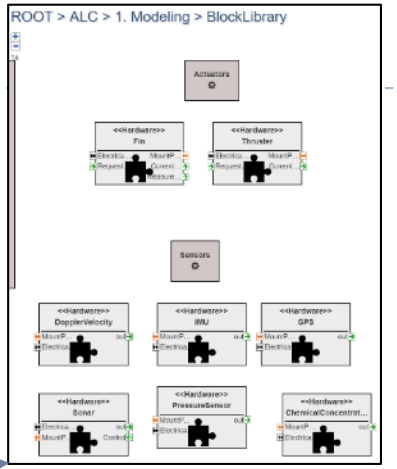


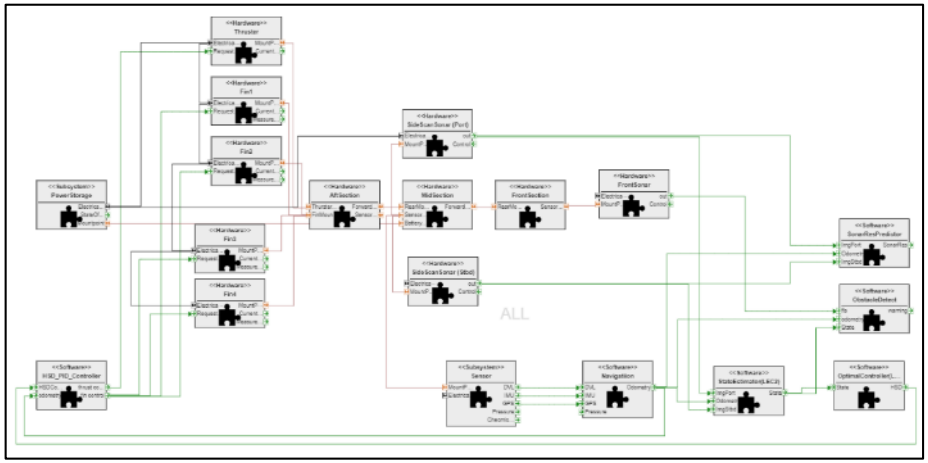


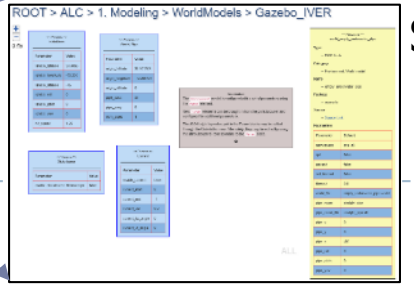








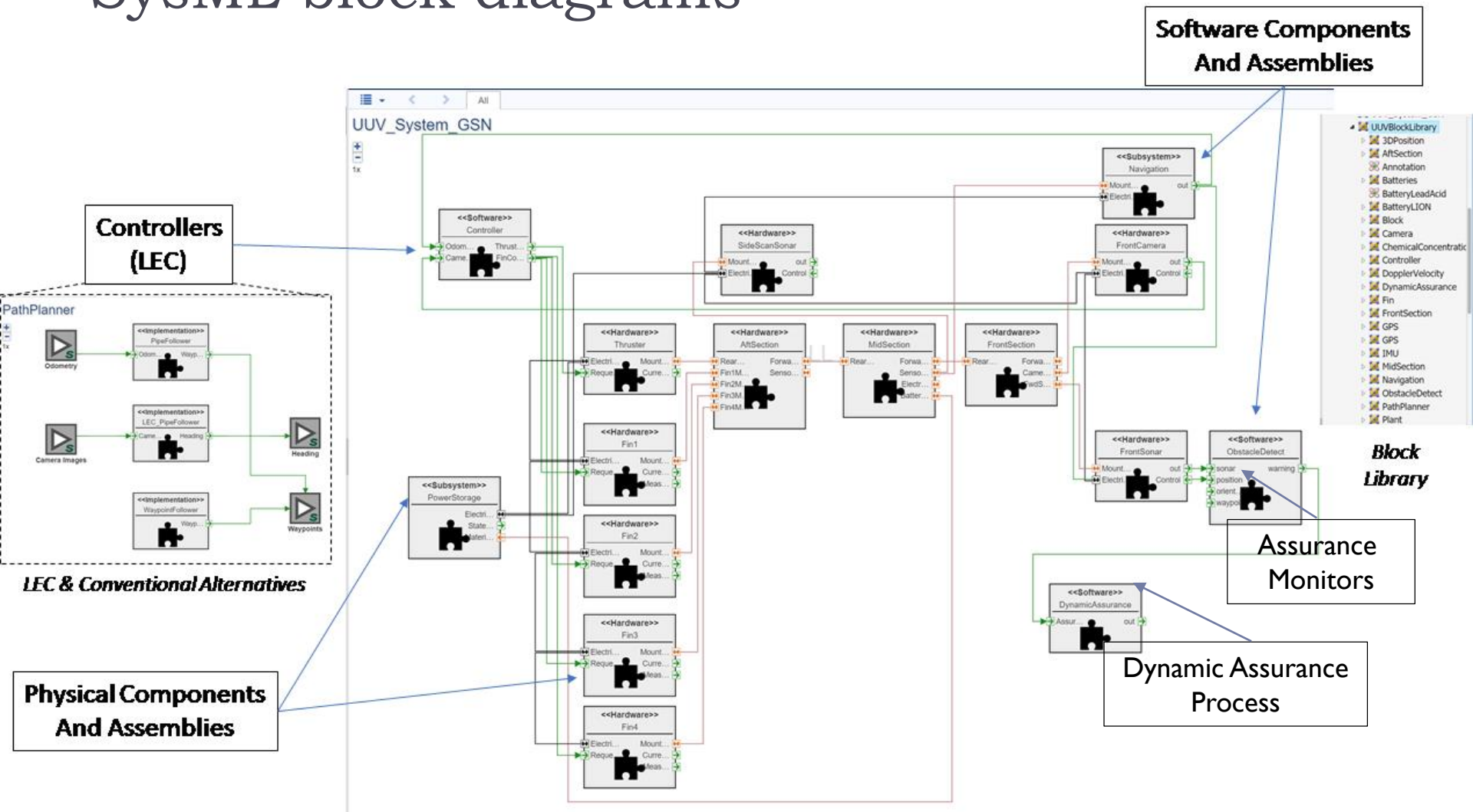




Systems: Components/ Subsystems; Parameters,...

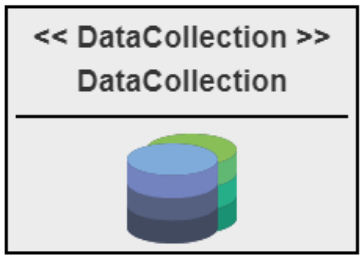
World models: Scenarios, Environments, Parameters

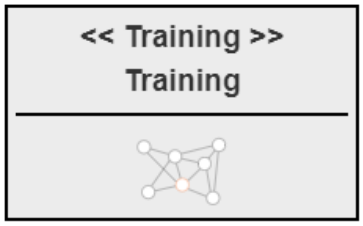
System architecture
SysML block diagrams

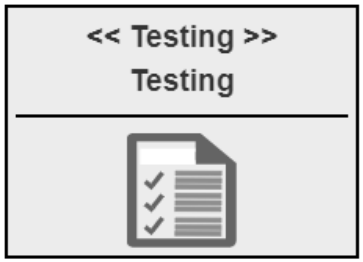


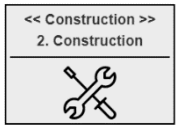
Subset of SysML Blocks, IBD to model all blocks, implementation alternatives for flexibility

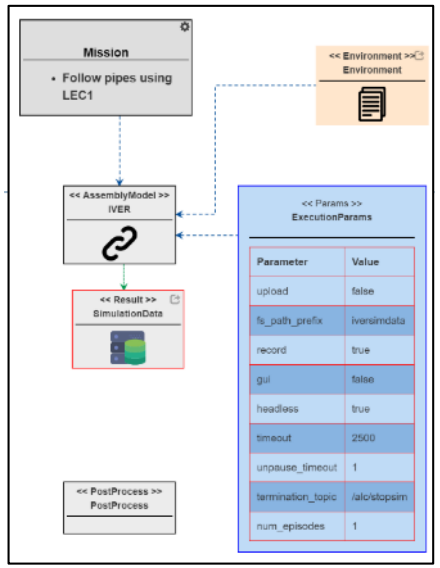
Construction



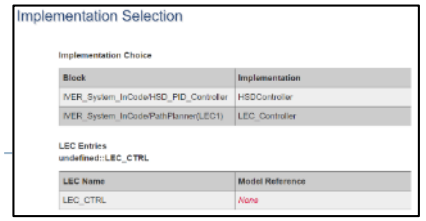




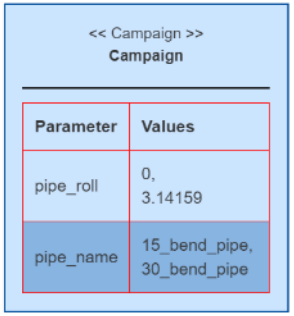


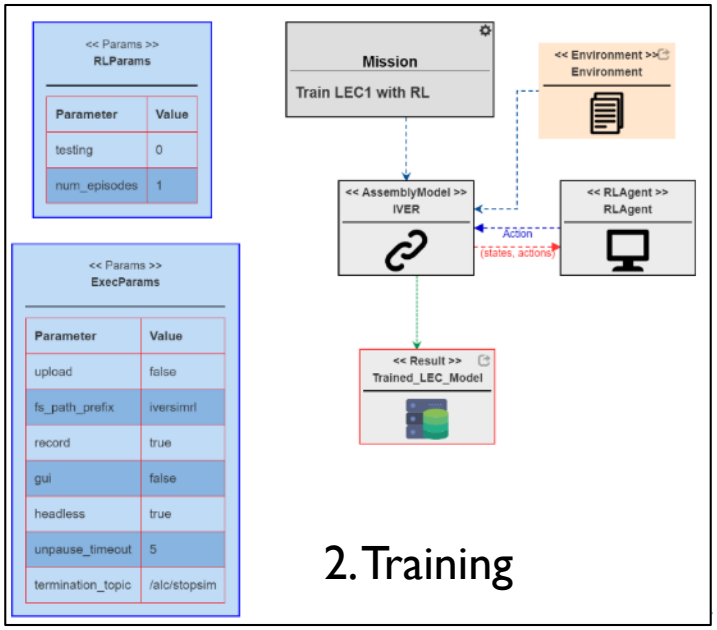


1. Data Collection

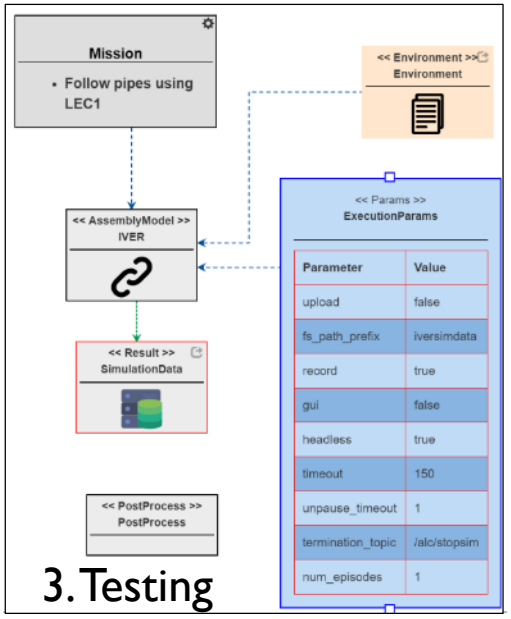


Select Configuration



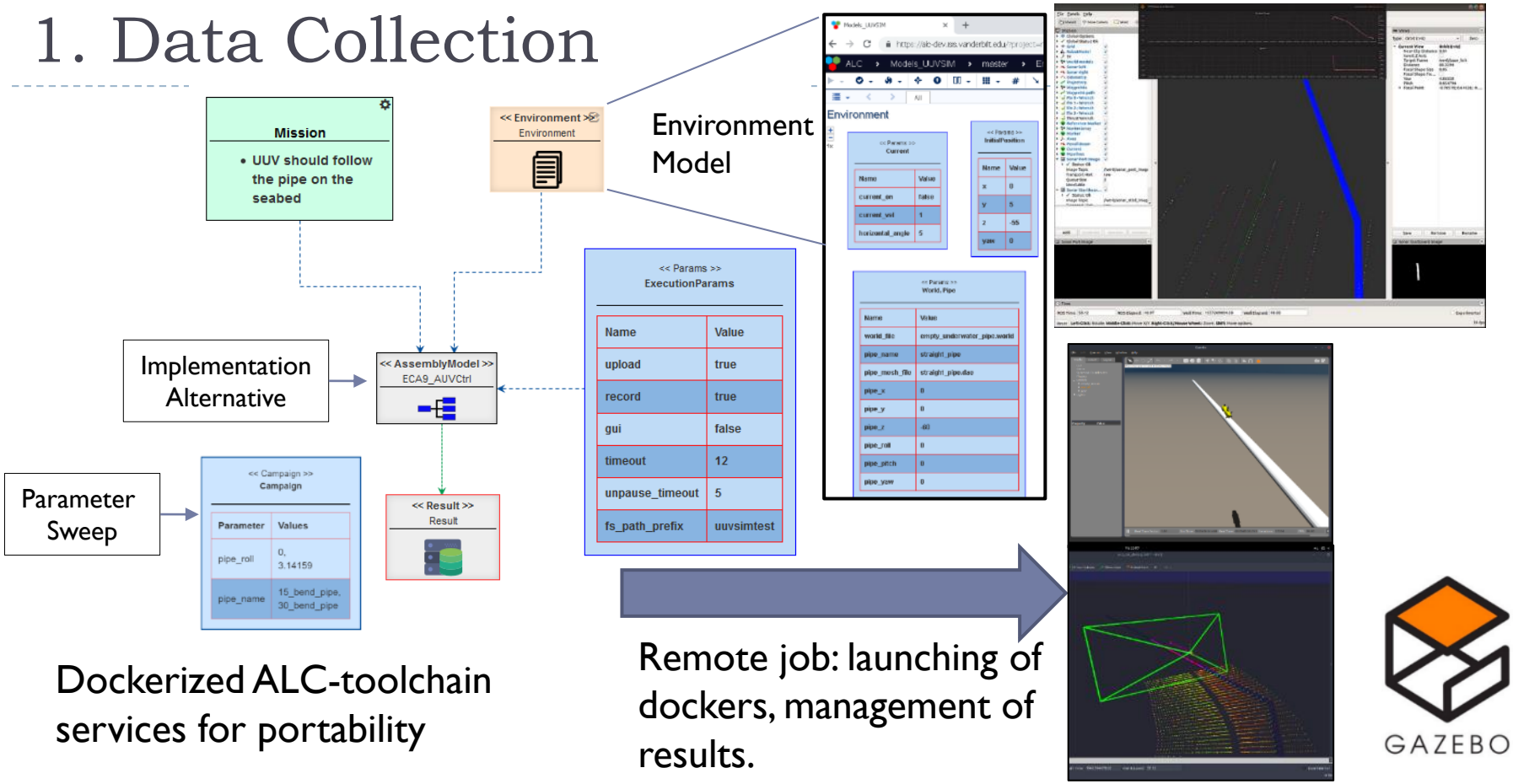


2. Training



3. Testing

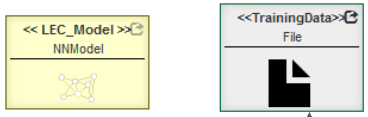
Construction:
1. Data Collection

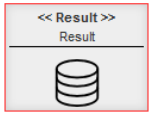


- *Assembly model* selects a specific implementation variant of a system architecture
- *Mission*, *Environment*, and *Execution parameters* set up the experiment scenario
- *Campaigns* across parameters a configurations related to system and environments
- Tool generates configuration file for running the simulation, captures results +meta data for all trials

Construction: 2. Training

- Neural Net model and parameters specified in "LEC Model"
- "Training Data" links to data generated from previous experiments
- Training job is dispatched to worker machine (typically with GPUs)
- Results and metadata are saved from the training sessions



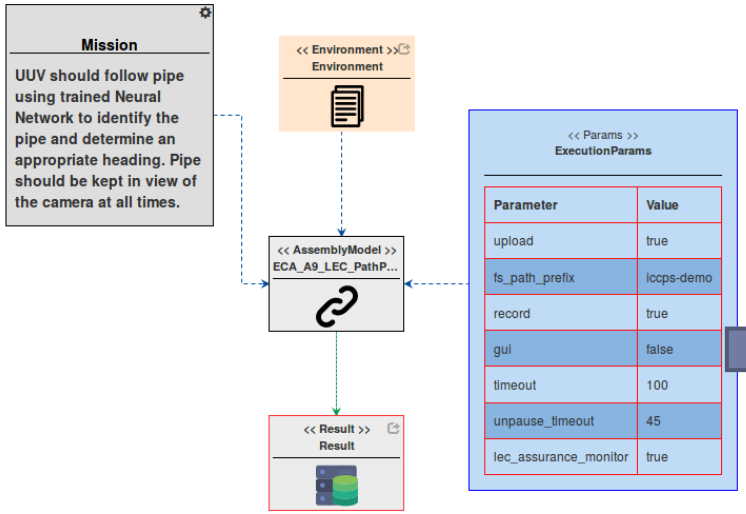


<< Params >>
Params

Name	Value
batch_size	64
epochs	5
useful_data_fraction	0.5
train_data_fraction	0.7
image_height	492
image_width	768
color_depth	3
loss	keras.losses.mean_squared_error
optimizer	keras.optimizers.Adam()
metrics	accuracy
upload	true

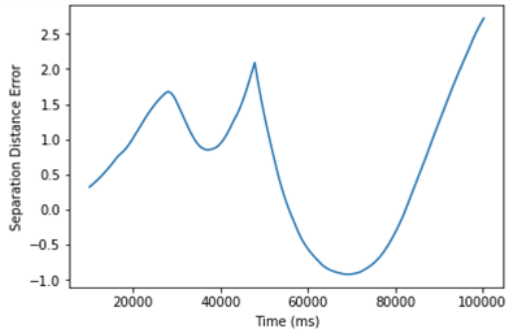
Name	Type
result-DataGen_PipeFollower-1542125401110	*metadata.json*
result-DataGen_PipeFollower-1542129035690	*metadata.json*
result-DataGen_PipeFollower-1542140169160	*metadata.json*

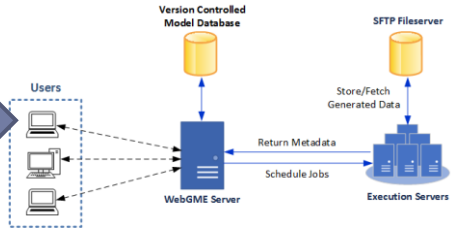
Construction:
3. Testing



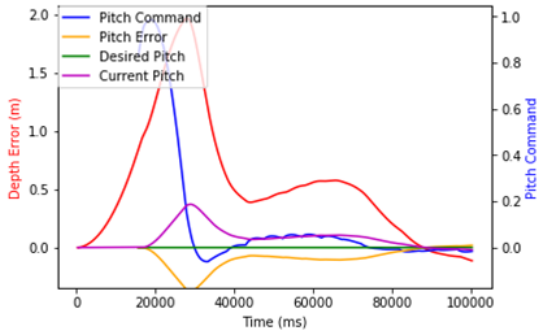
Analysis in Jupyter Notebook

Also, "single step" the process for debugging



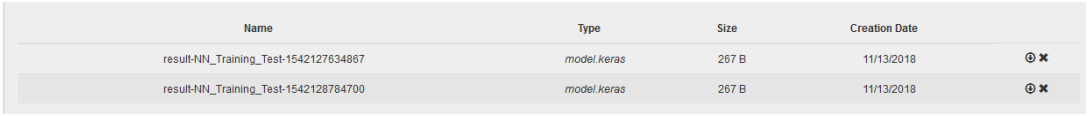


Execute on Remote Server/s



Training Model Data Managed on GitLab

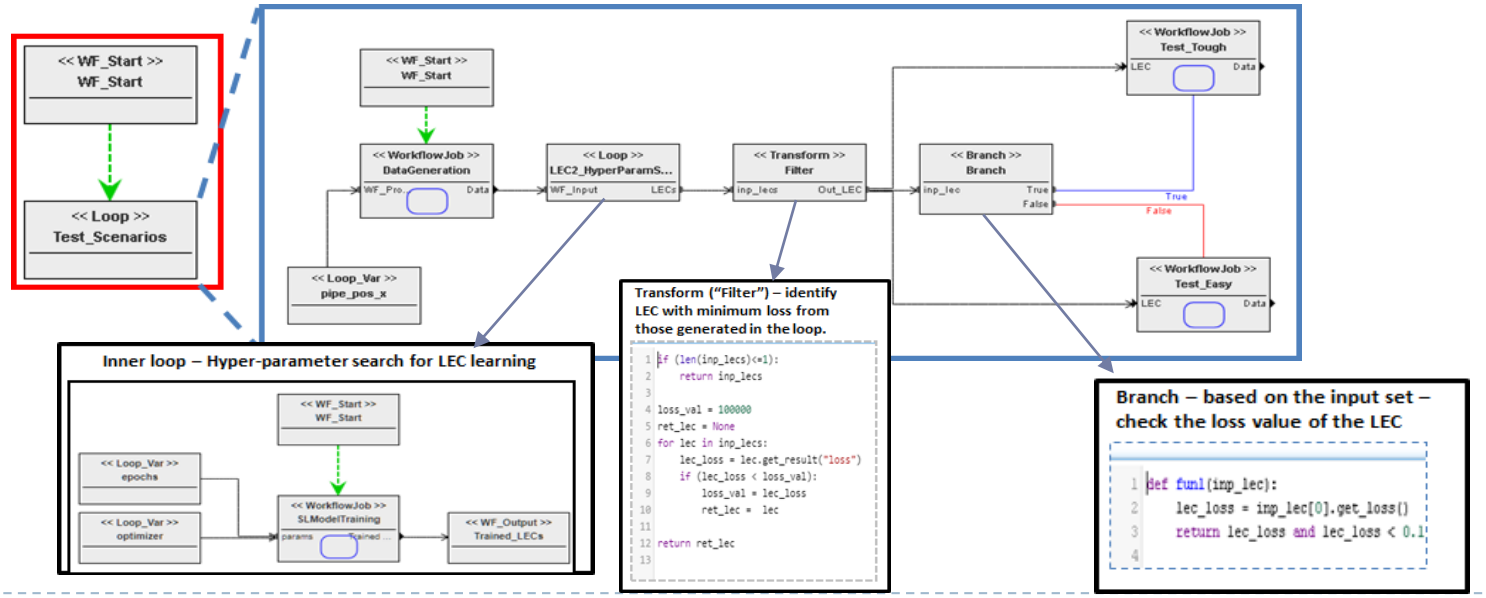
- Trained Neural Net can be tested in the simulator with another Experiment model
- Performance metrics are recorded for LEC evaluation, e.g.:
  - Distance from ideal path
  - Pipe within camera field of view



Results in file store + git, cross-linked for data provenance

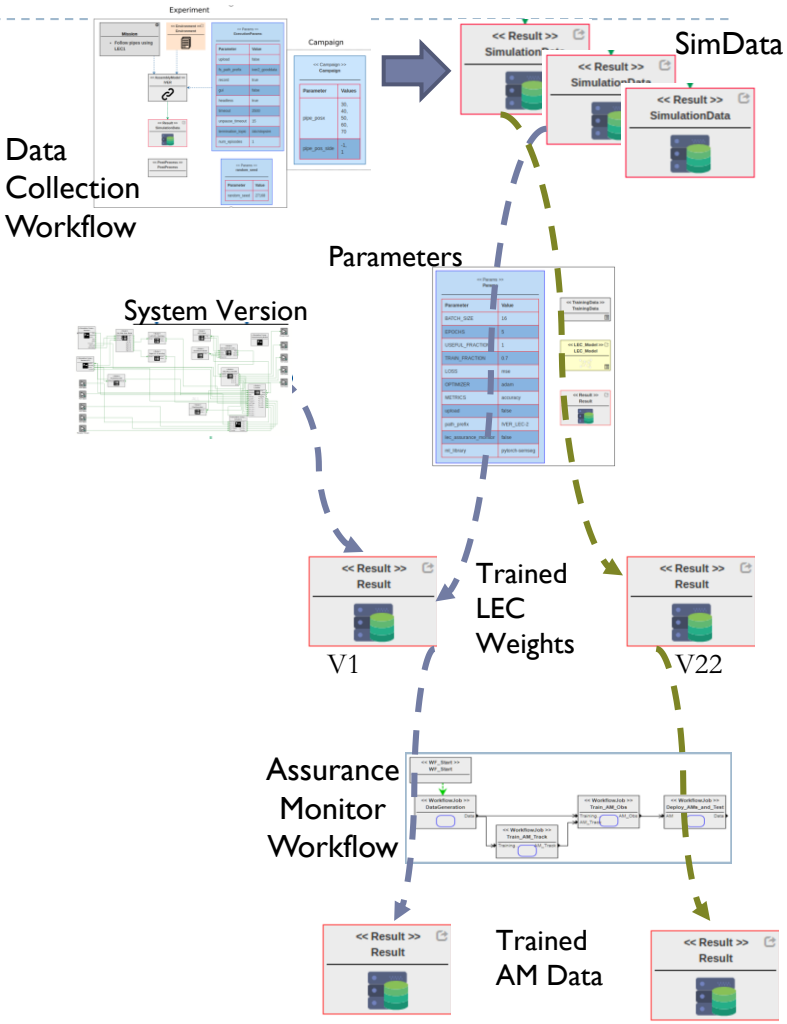
Automation: Workflow Models

- Workflow models allow specification and execution of job graphs
  - Each workflow job specifies execution of one or more activity models
  - Data dependencies between jobs are handled automatically
- Workflow supports
  - Loops – For (parallel), while/ do-while (sequential)
  - Transforms - Filter / Join (subset or aggregation of results)
  - Branch – execution path based on user-specified condition
- Example Workflow to train and optimize LEC2

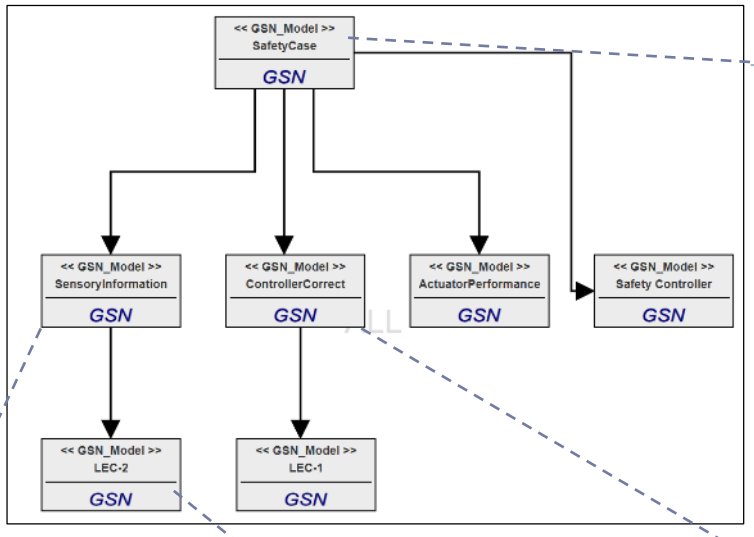


Toolchain Support for Data Provenance

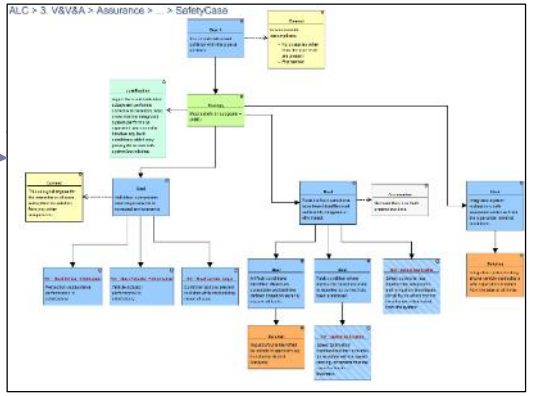
- All artifacts – generated during data collection, training, evaluation
- Recorded for each execution:
  - Parameter settings
  - LEC(s) Models (Deployed/Initial)
  - Data used in training, validation and evaluation
- Allows re-execution of any step/workflow
- Track the evolution of data/ LECs/ Assurance
- Maintain traceability links at each stage to:
  - Data used in training LECs
  - Initial trained model used in training LECs
  - LECs used in generating data sets/Assurance

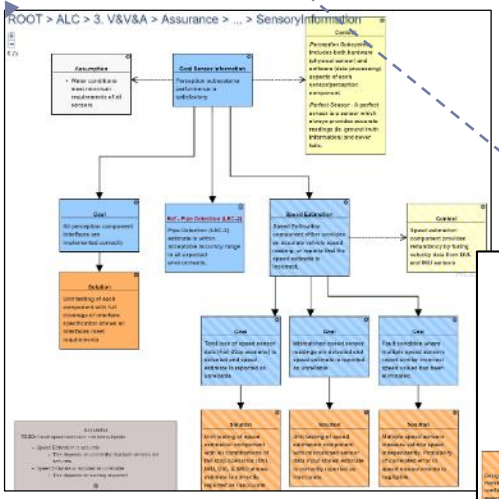


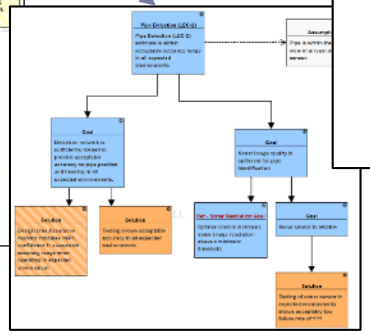
System Assurance Case: GSN

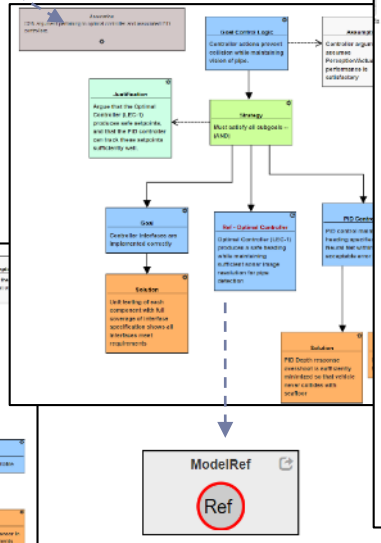


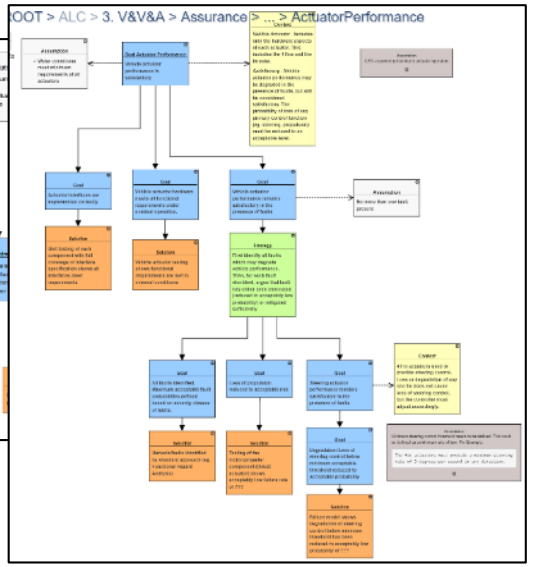
- Top-level goals correspond to high level safety claims
- Leaf goals correspond to claims which can be directly supported by evidence/solutions
- Evaluation metrics from LEC experiments can be used as evidence for leaf goals
- User Defined Combination Logic (E.g. M-of-N, etc.)

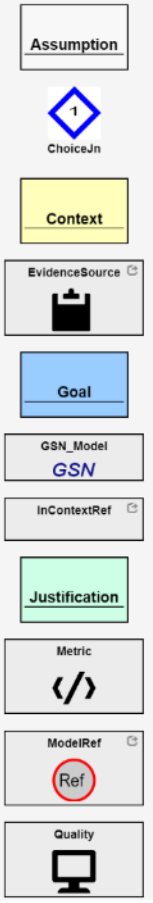












GSN Model for UUV

Cross-Referencing Components, Datasets, for Context/Evidence

System-level Assurance Technology

- Dynamic assurance:
  - LECs are continuously checked by Assurance Monitors to detect potential issues
  - Assurance Level is dynamically calculated based on system mode/state, environment, etc.
  - Autonomy System acts as a supervisor to take contingency action if needed
- Static assurance:
  - For certification
  - Comprehensive A/C
  - Covers autonomy + LEC +...