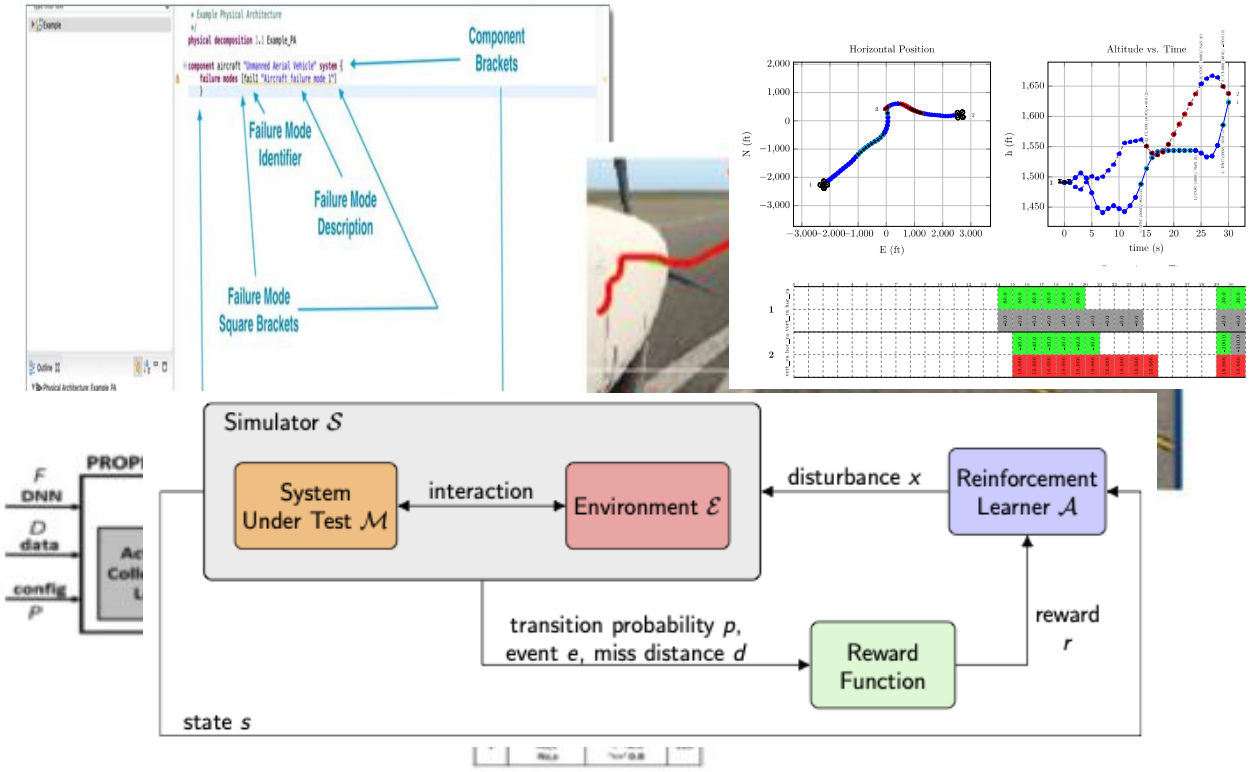
# AI Verification Tools

Adrian Agogino
Robust Software Engineering
Code TI
Ames Research Center

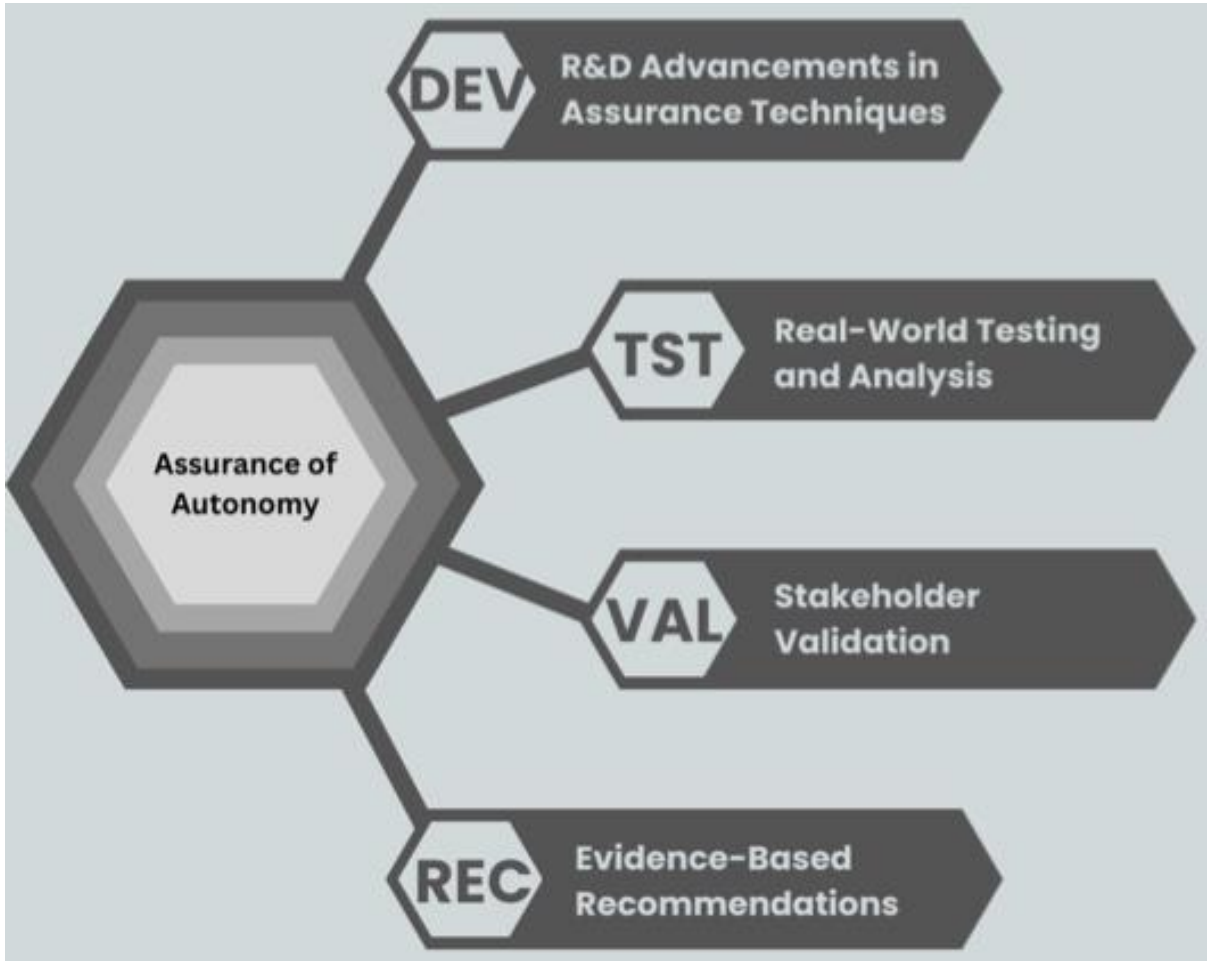
# Robust Software Engineering Group

- Leads: Guillaume Brat, Sandy Lozito




- Software testing
- Software verification and validation
- Machine learning
- Autonomy
- Multiagent systems

# Complex, Autonomous Systems Assurance




Stakeholders

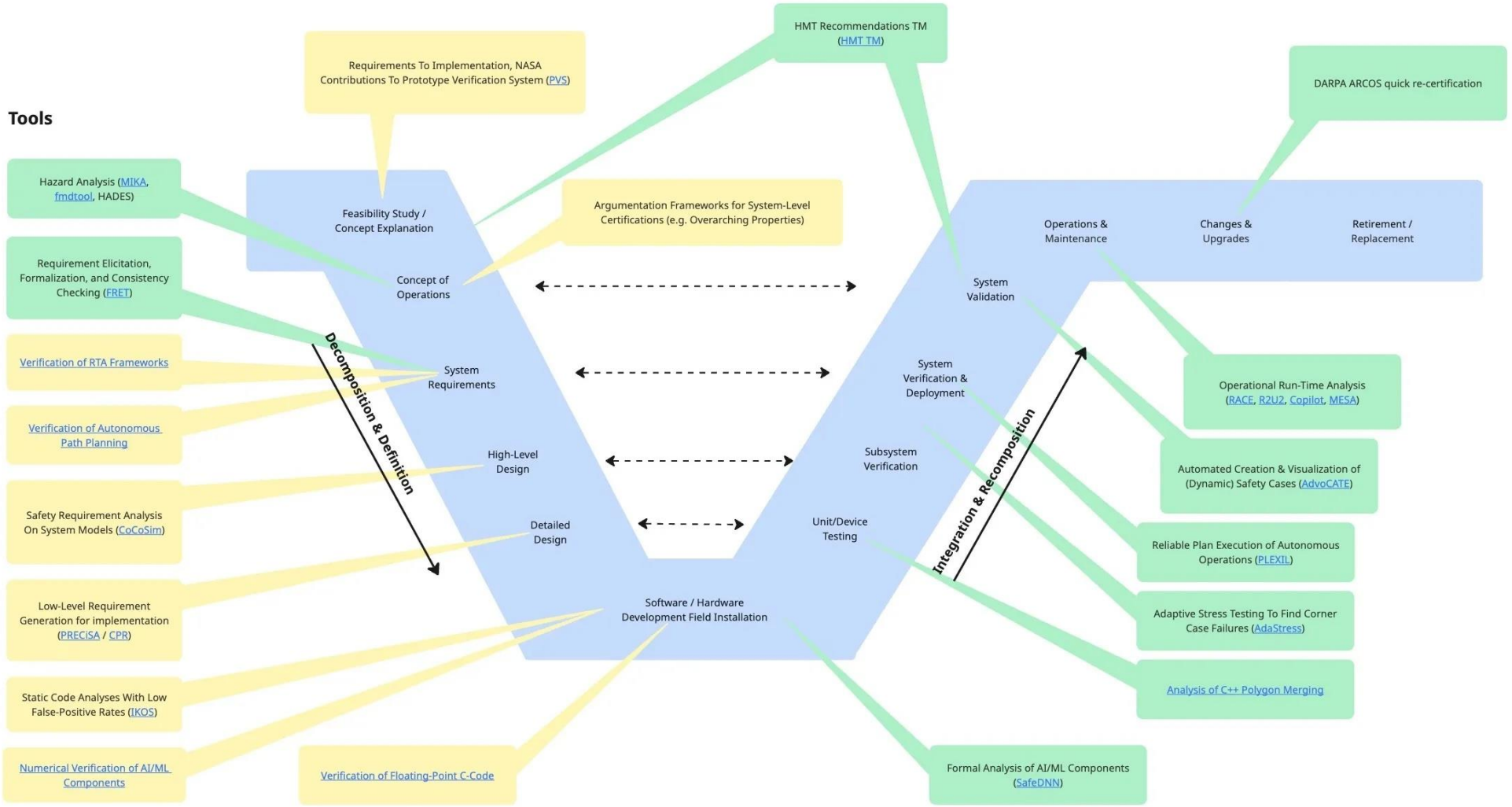
# AI and Complexity

- AI algorithms tend to be large and complex
  - Hundreds of thousands of neuron for small image processor
  - Millions of neurons for large image processor
  - Billions to trillion neurons for large language models
- What we are asking from AI is complex
  - Perception
  - Understanding world model
  - Language understanding
  - End-to-end control

# Need Tools for AI Lifecycle

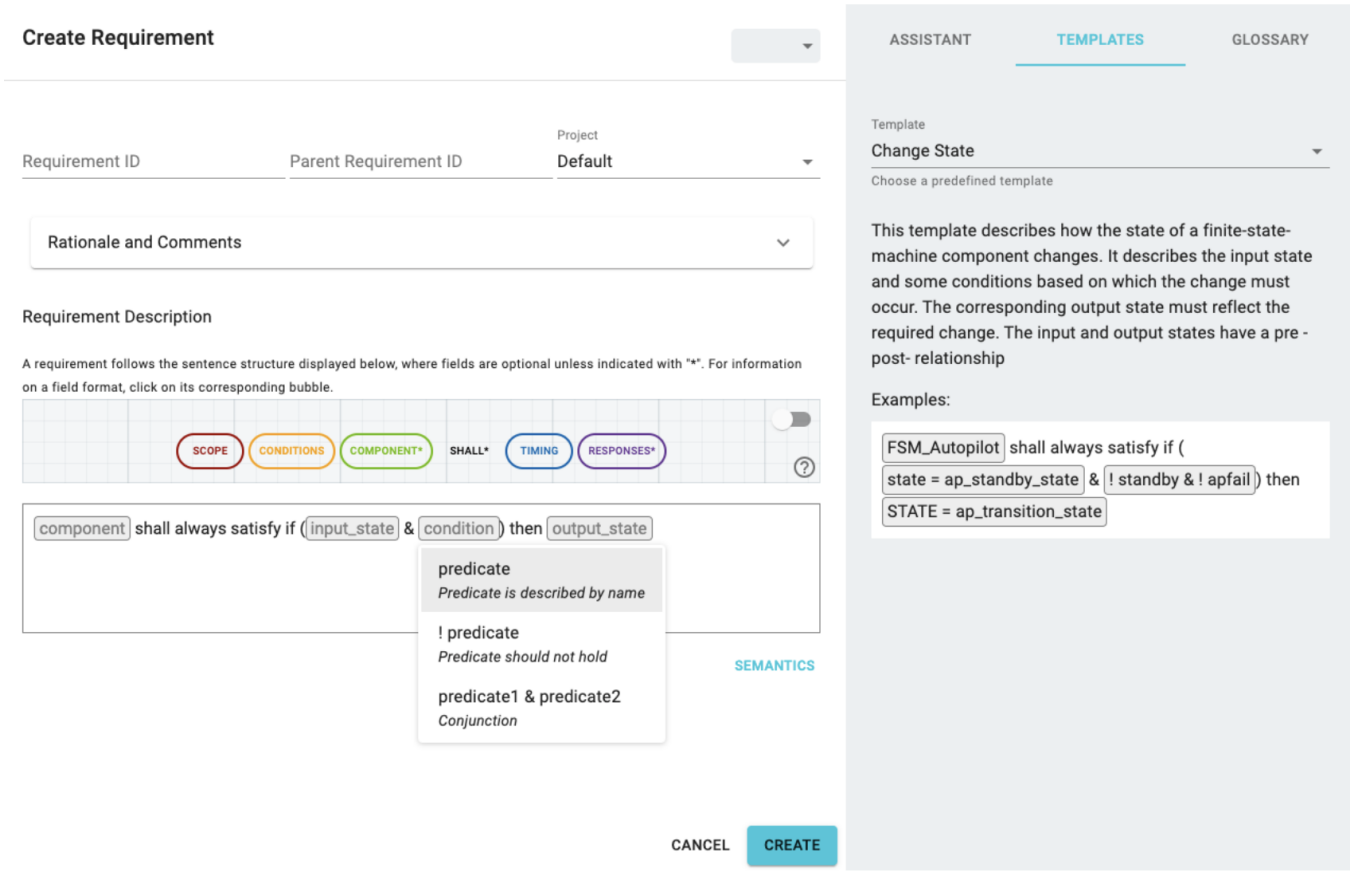
- Architecture of complex systems
- Requirements for complex systems
- Explainability of complex systems
- Testing and verification of complex systems

# AI Tools for Design Lifecycle



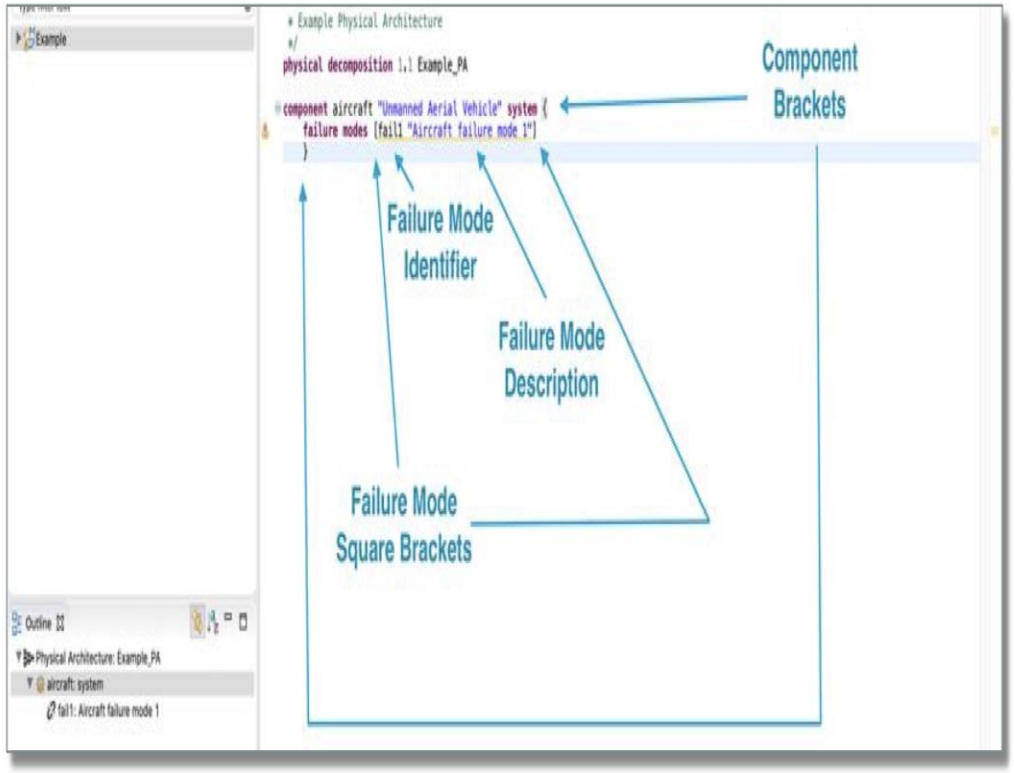
# FRET: Creating Formal Requirements

- FRET allows natural language to be made into formal requirements



# AdvoCATE

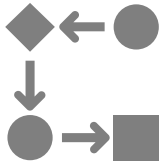
- Safety risk management
- hazard analysis
- linking of hazards to requirement logs
- risk modeling using bow tie diagrams
- structured arguments
- evidence logs



# Fault Model Design Tools (FMD)

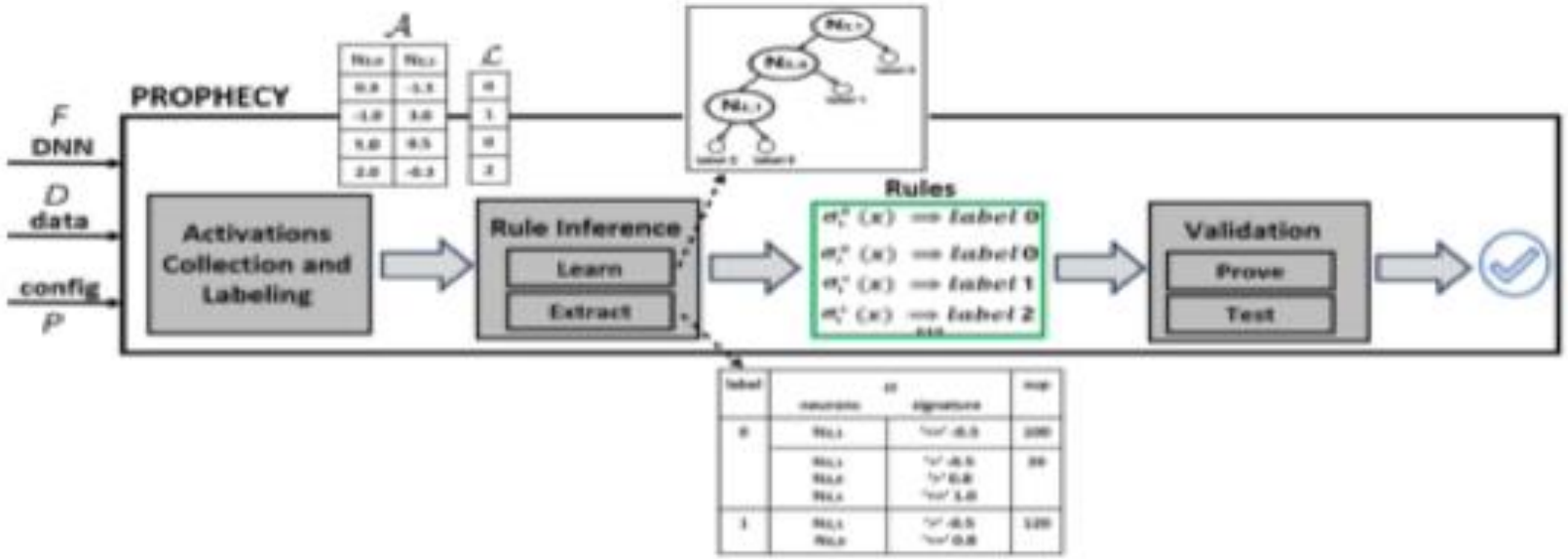
- Simulation-based (rather than document-based) hazard analysis process
  - Flexible Modelling Paradigm
  - Powerful Simulation Techniques
  - Efficient Analysis Process





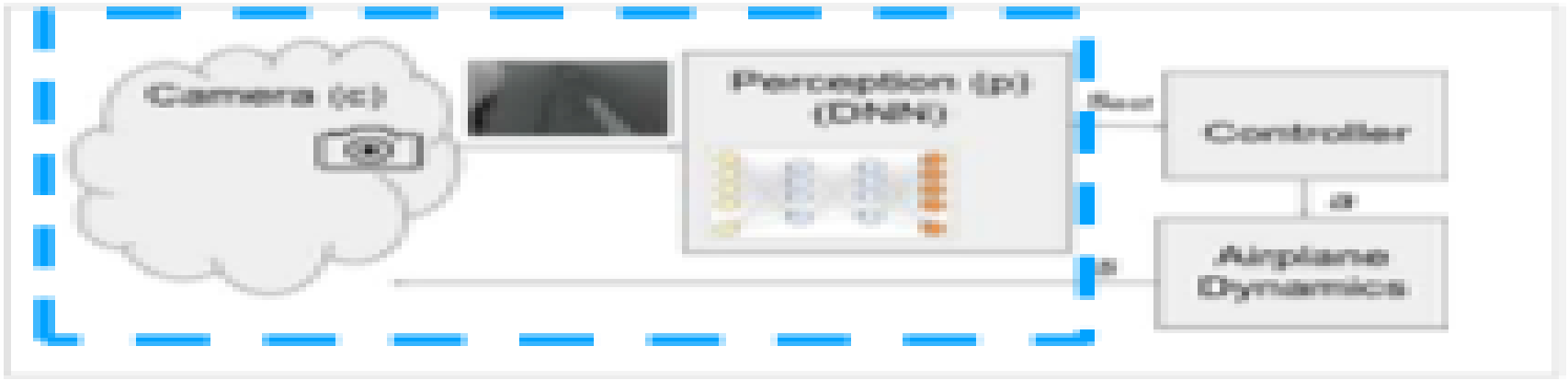
# SafeDNN: Understanding and Verifying Neural Networks

- Extract rules from neural network
- Verify properties from these rules
  - Formal proofs, Runtime monitoring, Specification-driven Testing



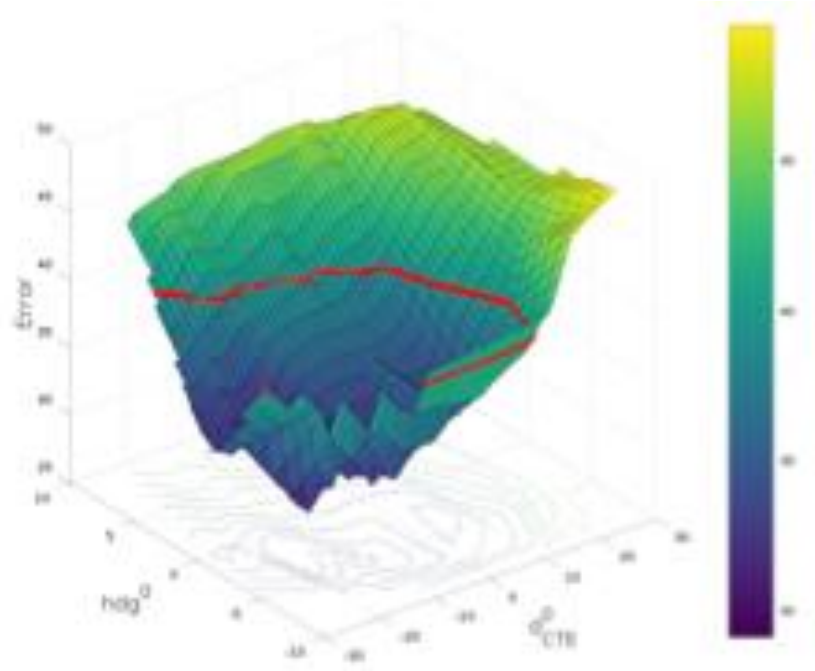
# Closed Loop AI Analysis

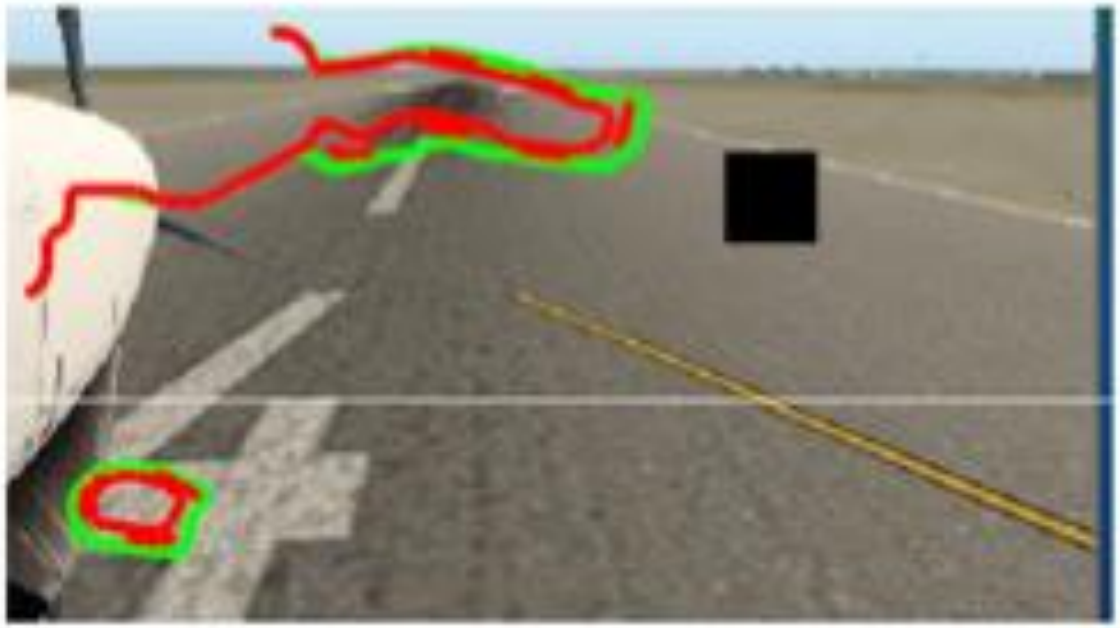
- Can define requirements for outer loop system even if perception component is complex
- Enables
  - Probabilistic analysis,
  - Worst-case analysis,
  - Scenario-based Compositional Verification



# SysAI

- Find safety envelopes
- Find regions of failure



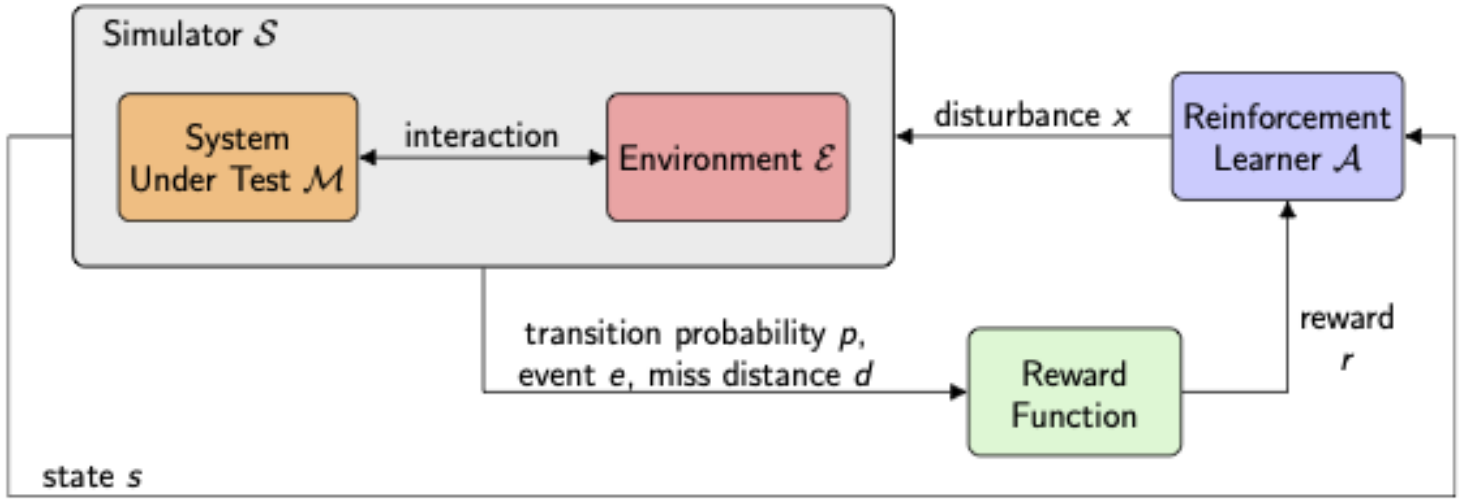



# AdaStress

- Many systems too complex to verify formally
- Many systems have too large of input space to perform Monte Carlo (random) testing
- AdaStress learns to test regions that are most likely to fail to find failures even when input space in large
  - Finds failures with few samples
  - Can be used on blackbox systems such as large neural networks

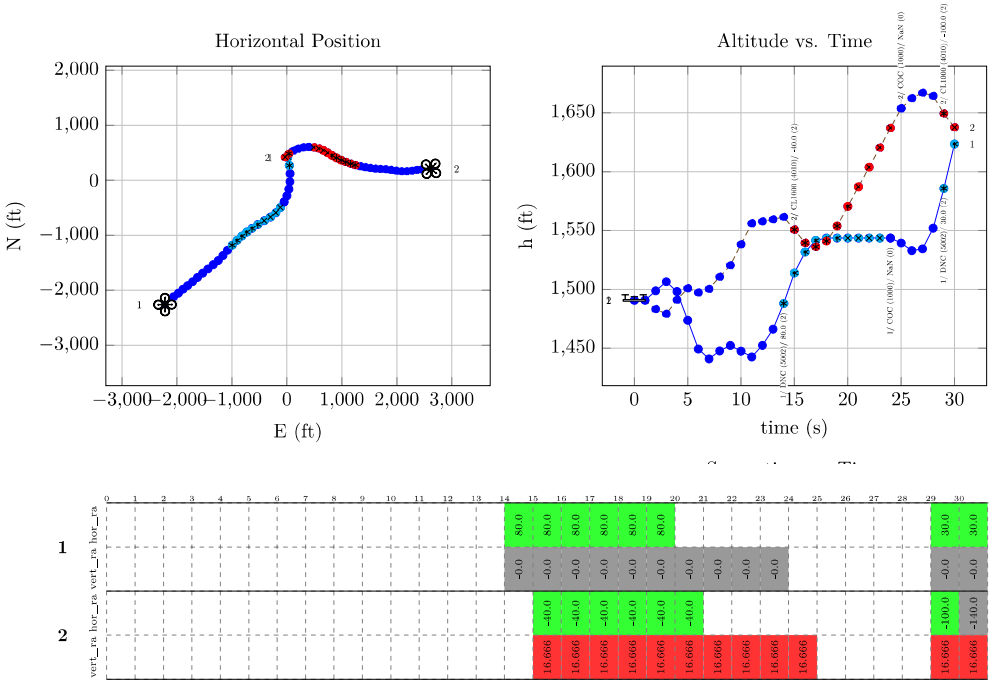
# AdaStress

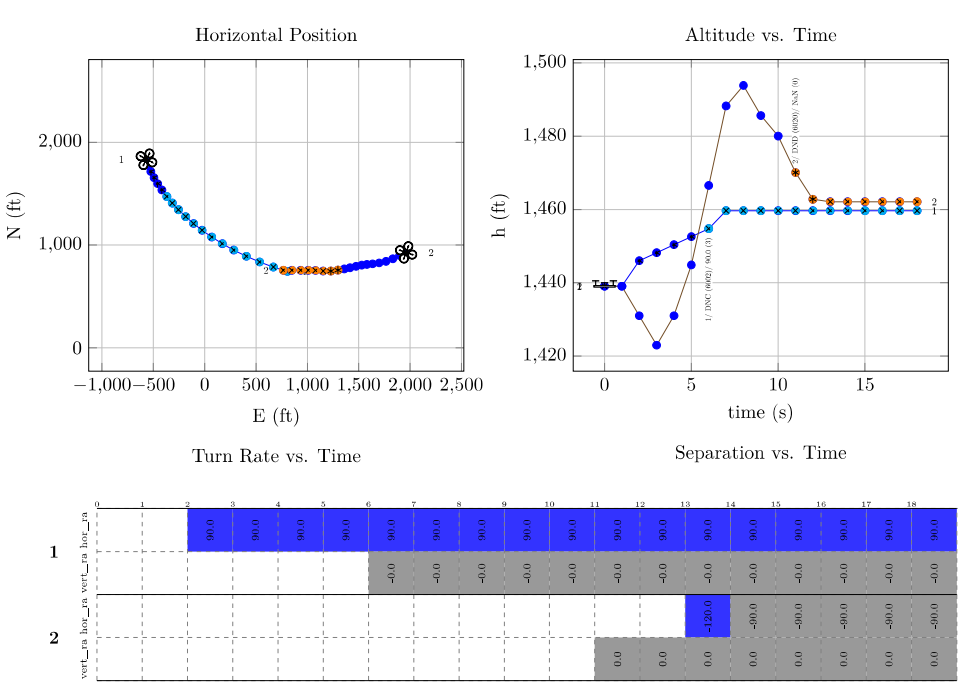
- Perform adaptive stress testing with:
  1. Reinforcement Learning
  2. Likelihoods
  3. Blackbox Manipulation (optional)



# AdaStress Case Study Case Study: xSu

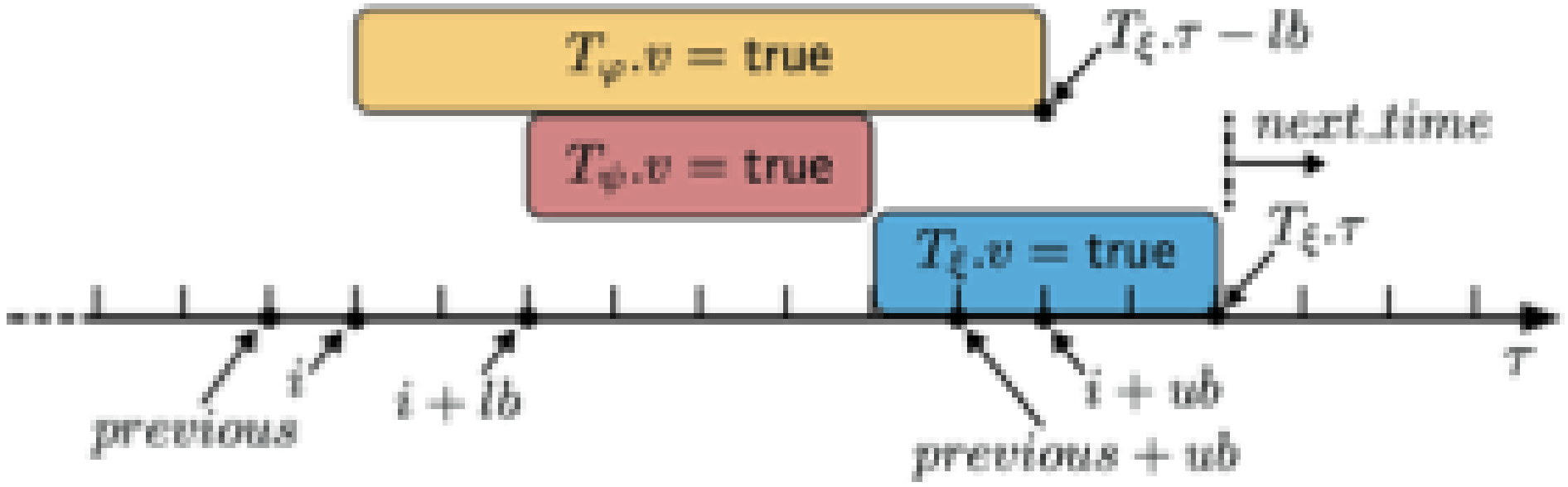
- AdaStress can find previous unknown failure conditions



# R2U2: Realizable, Responsive, Unobtrusive Unit

- Stream based runtime monitoring
  - Use temporal logic



# Example of Tool Workflow in “V” Lifecycle

English Req[ts] → *disambiguate* → Formal Req[ts]
FRET

DO-178 certifiable code

AI/ML enabled code

Run-time Monitoring
R2U2
OGMA/CoPilot

ConOp

*refine* w/
AdaStress
SysAI

Req[t]-based testing
Adaptive stress testing
Statistical-based

Formal Analysis
SafeDNN
Prophecy

Design Time
PREVENT
AdvoCATE

Hazard Realization

Run Time
MITIGATE
Assurance Case

System Resilience

*Risk Analysis*
MIKA & FmdTool

Hazards