OBTAINING TRUST IN AUTONOMOUS VEHICLES: TOOLS FOR FORMAL MODEL SYNTHESIS AND VALIDATION

Connie Heitmeyer
Beth Leonard

Software Engineering Section
Center for High Assurance Computer Systems
Naval Research Laboratory
Washington, DC

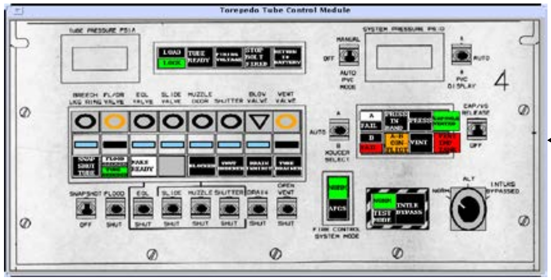
FormaliSE: FME Workshop on
Formal Methods in Software Engineering
Florence, Italy
May 18, 2015

OUTLINE

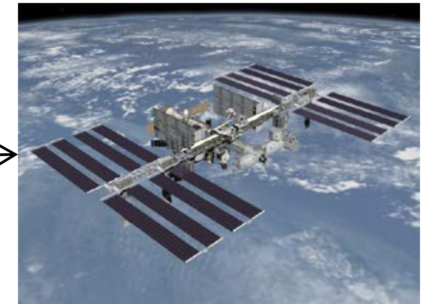
- Background
  - Formal methods: Shown to have utility in practice
  - Why software problem even harder now: Cyber Physical Systems
  - Two kinds of trust needed in developing Unmanned/Autonomous Vehicles, a special class of CPSs
- Transitioning FMs to software practice
  - Challenge 1: **How to obtain the formal system model**
    - ⇨ **Formal model synthesis from scenarios**
  - Challenge 2: **How to model/analze CPSs**
    - ⇨ **3D simulation based on a formal req. model**
- **Scenario-Based Formal Model Synthesis**
- **Formal Model-Based 3D Simulation**
- Conclusions and Future Work

BACKGROUND

UTILITY OF FORMAL METHODS IN REAL-WORLD SOFTWARE HAS BEEN SHOWN



Detect errors

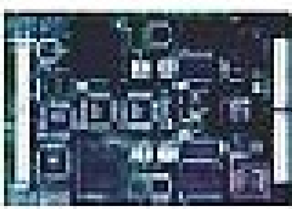


Weapons Control Panel

- Large complex program (~30KLOC)
- Contractor software req. spec: 250+ vars
- Translated into a formal model in 2 wks.
- Model checking showed that all six safety properties violated!

International Space Station

- Failure Detection, Isolation & Recovery in Thermal Radiator Rotary software module
- Translating semiformal req. documents into a formal spec exposed two serious errors!



Help verify model & code

Tools used in industry

Software-Based Crypto Device

- FMs used in certification of security
- EAL6+ Common Criteria evaluation
- Formal security model, formal verif., demo that C code satisfies formal model

Lockheed Martin

- Since 1999, SCR tools used by 3 sites
- “We currently are supporting close to 1500 models…and have found SCR Tool suite to be…invaluable…in finding requirements defects, as well as validating the functional behaviour of our software requirements.”

DEVELOPING CORRECT SOFTWARE IS BECOMING EVEN MORE CHALLENGING

- Prior focus of FMs: Embedded Systems
  - An **embedded system** is immersed in a physical system that it monitors and controls
  - Focus in development is on the embedded system only
- New Challenge for FMs: Cyber Physical Systems
  - A **cyber physical system** combines a digital system performing computation with physical processes
  - Problem: Managing the dynamics, timing and concurrency in *both* the digital system and physical processes
- Imp. Class of CPSs: (Intelligent) Unmanned/Autonomous Systems

Adapted from A. Sangiovanni-Vincentelli, “Let’s Get Physical: Adding Physical Dimensions to Cyber Systems,” Internat. Conf. on Cyber Physical Systems, 2014.

Problem for Unmanned Systems: Human Mistrust of Automation/Autonomy

- Two kinds of trust needed*
  - **System Trust**: Human confidence that system behaves as intended
  - **Operational Trust**: Human confidence that system helps him/her perform the assigned tasks
- To achieve system trust
  - Need **high assurance** that system satisfies its requirements
    - ➢ **formal modeling**, **formal verification**
- To achieve operational trust
  - Need well-designed HCI *and* **human validation** that the designed autonomy will help
    - ➢ **formal modeling**, **model-based simulation**

*Dan Zwillinger, Ratheon, S5, 2014.

A SOLID BASIS FOR OBTAINING SYSTEM & OPERATIONAL TRUST: A FORMAL MODEL

BENEFITS OF A FORMAL SYSTEM MODEL

- Can be verified to satisfy the required system properties
  ⇨ **system trust**
- Can be validated to show that it captures the intended behavior
  ⇨ **operational trust**

PROBLEM IN CURRENT SOFTWARE PRACTICE

- Formal system/requirements models are rare
  - Practitioners regard formal notations as difficult to understand and apply; don't think that formal models scale, are cost-effective*
- When they do exist, formal models are often
  - Ambiguous: Rep'd in languages w/o a formal semantics
  - Expressed at a low level of abstraction

OBTAINING A FORMAL MODEL: A PROMISING APPROACH

- Synthesize a formal model from scenarios

*C. Heitmeyer, "On the need for practical formal methods," FTRTFT, 1998.

SCENARIO-BASED FORMAL MODEL SYNTHESIS

Formal Model Synthesis from Scenarios

Already significant research on this problem

- Most research based on Message Sequence Charts (MSCs)
  - Many practitioners use MSCs to specify requirements
  - Natural therefore to develop methods which synthesize formal models from MSCs
- Why Introduce Yet Another Method?
  - The SCR notation scales, is expressive and understandable by practitioners
  - SCR tools have already been used successfully 1) to detect errors in and 2) to verify both models and source code
  - While developers have difficulty creating tabular specs, they can readily extend & modify models expressed as tables
  - A model generated from scenarios is inherently incomplete; the SCR CC automatically finds incompleteness in a model
  - SCR makes available a wide range of tools for formal model analysis and validation, test generation, code generation, etc.

Our New Scenario Language:
A Moded Scenarios Description

A Moded Scenarios Description (MSD) has three components

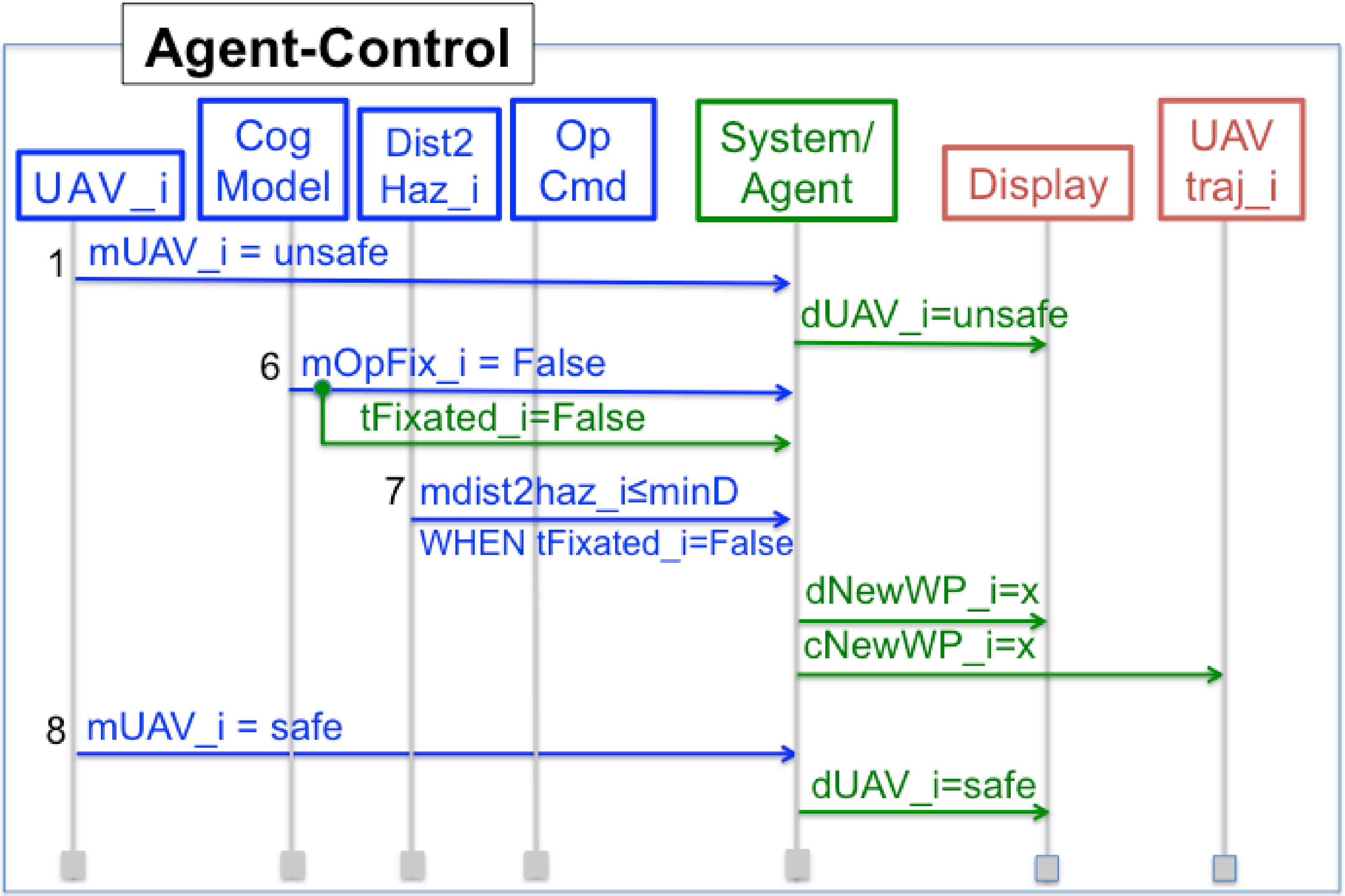
- A set of Event Sequence Charts (ESCs)
  - Inspired by MSCs
  - Look like MSCs
- A Mode Diagram
- A Scenario Constraint
  - Defines initial variable values
  - Specifies assumptions and properties (e.g., safety and security)
  - Defines constants, and state invariants

Numeric Labels link the Mode Diagram with the ESCs

Ref. [1] presents our new scenario language, a mathematical model that defines its semantics, and two algorithms for generating definitions of the dependent variables from elements of the MSD

[1]C. Heitmeyer et al., "Building Human-Centric Decision Systems," *ASE*, 2015.

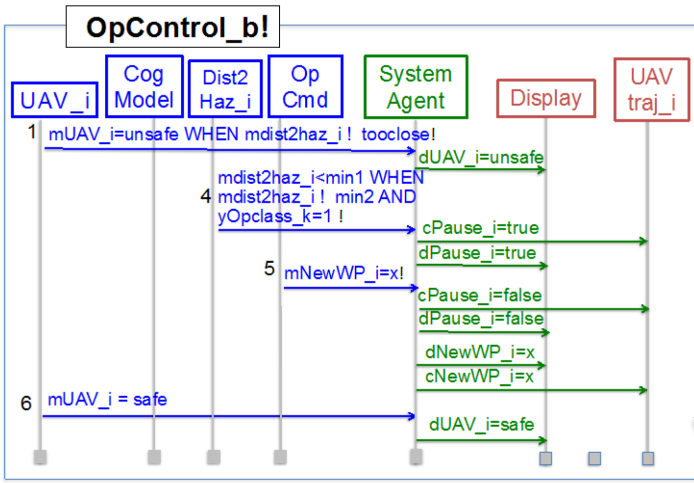
Formal Model Synthesis from a MSD: Event Sequence Chart

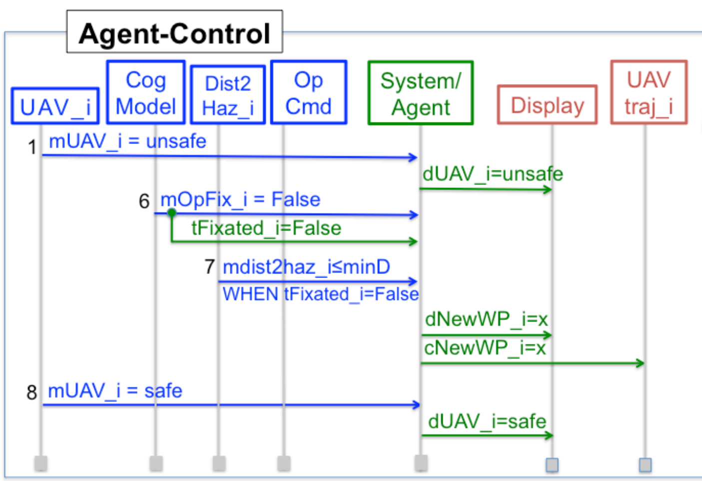


Formal Model Synthesis from a Moded Scenarios Description

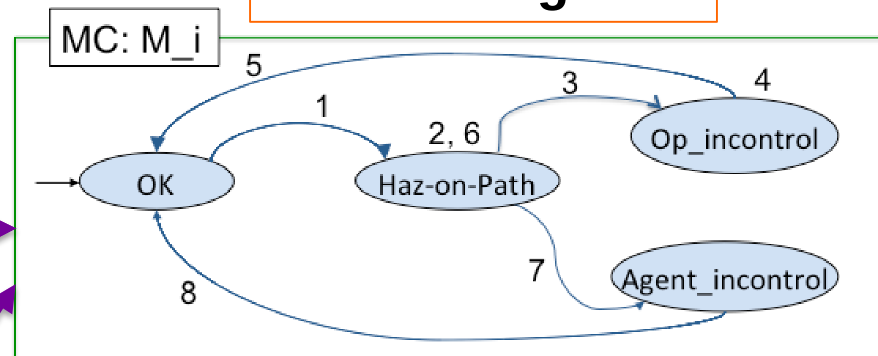
manual
automatic

Scenarios specified as ESCs





Mode Diagram

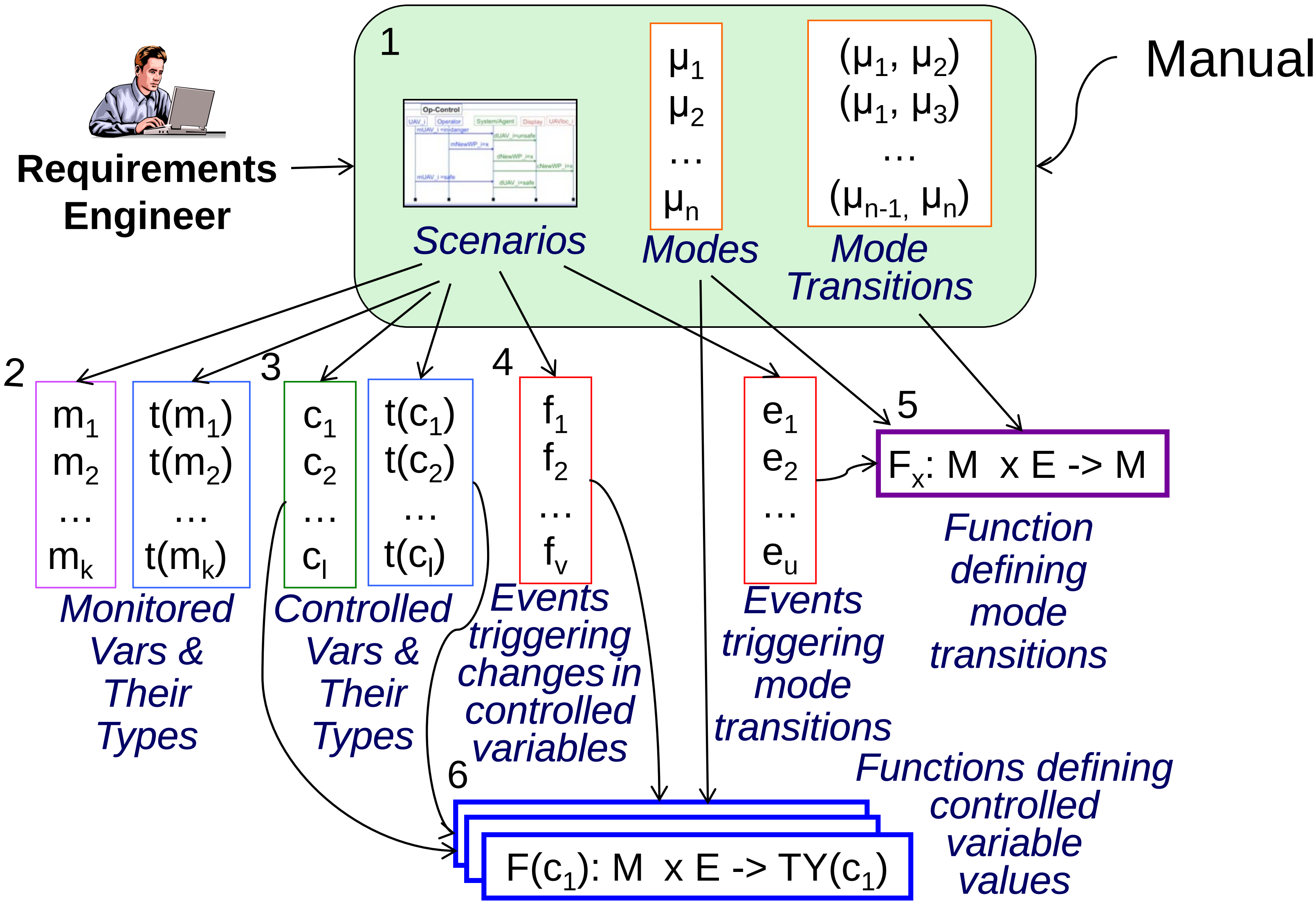


FORMAL MODEL

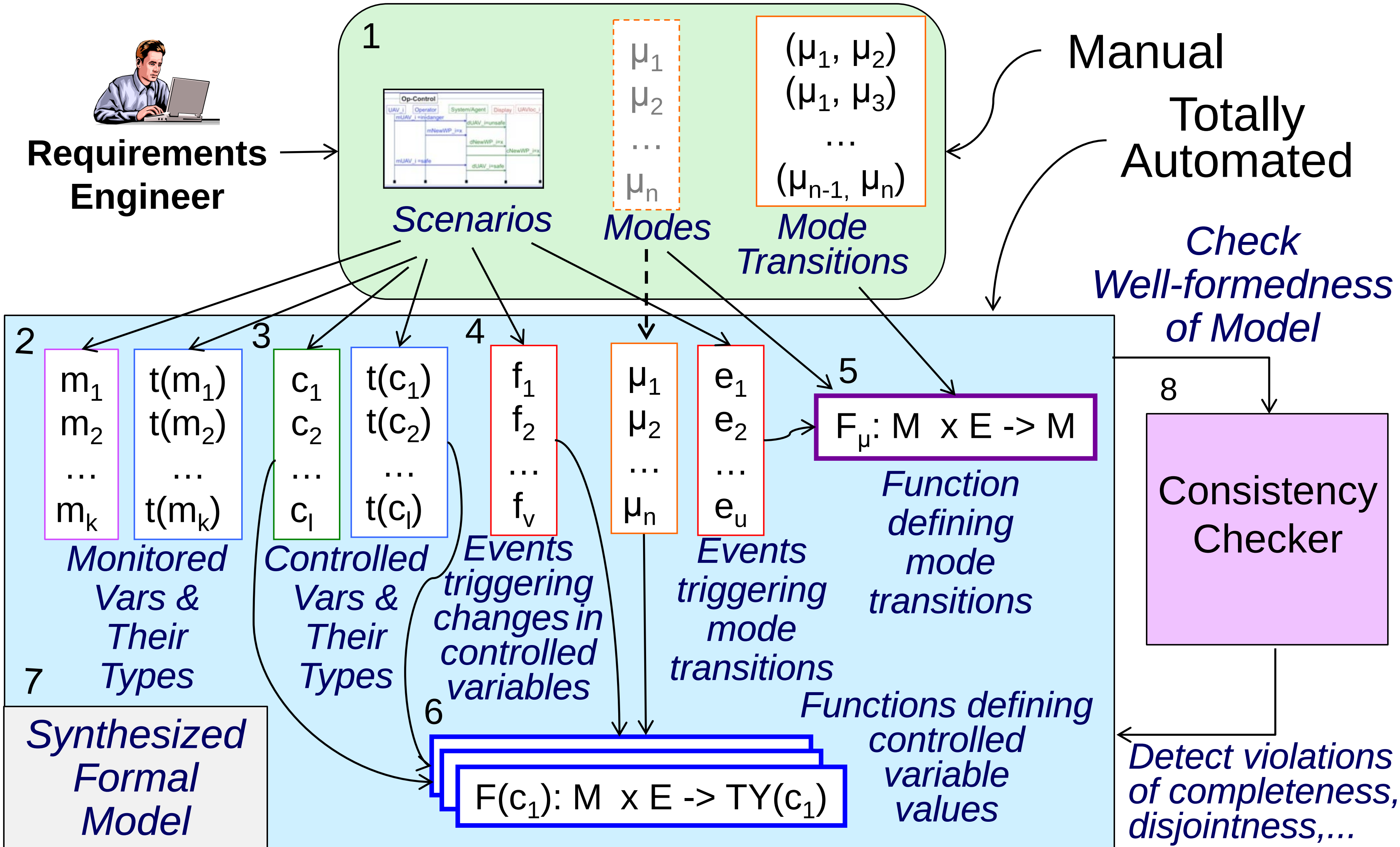
Scenario Constraint

no.	mode	event			
1	OK	@T(mUAV_i=unsafe)	FALSE	FALSE	FALSE
3	Op_Control	FALSE	@T(mNewWP_i=x)	FALSE	FALSE
4	Op_Control	FALSE	FALSE	@T(mUAV_i=safe)	FALSE
5	Agent_Control	FALSE	FALSE	FALSE	@T(mLook_i=F)
cvar	dUAV_i' =	(unsafe, -)	(-, x)	(safe, -)	(hovering, -)

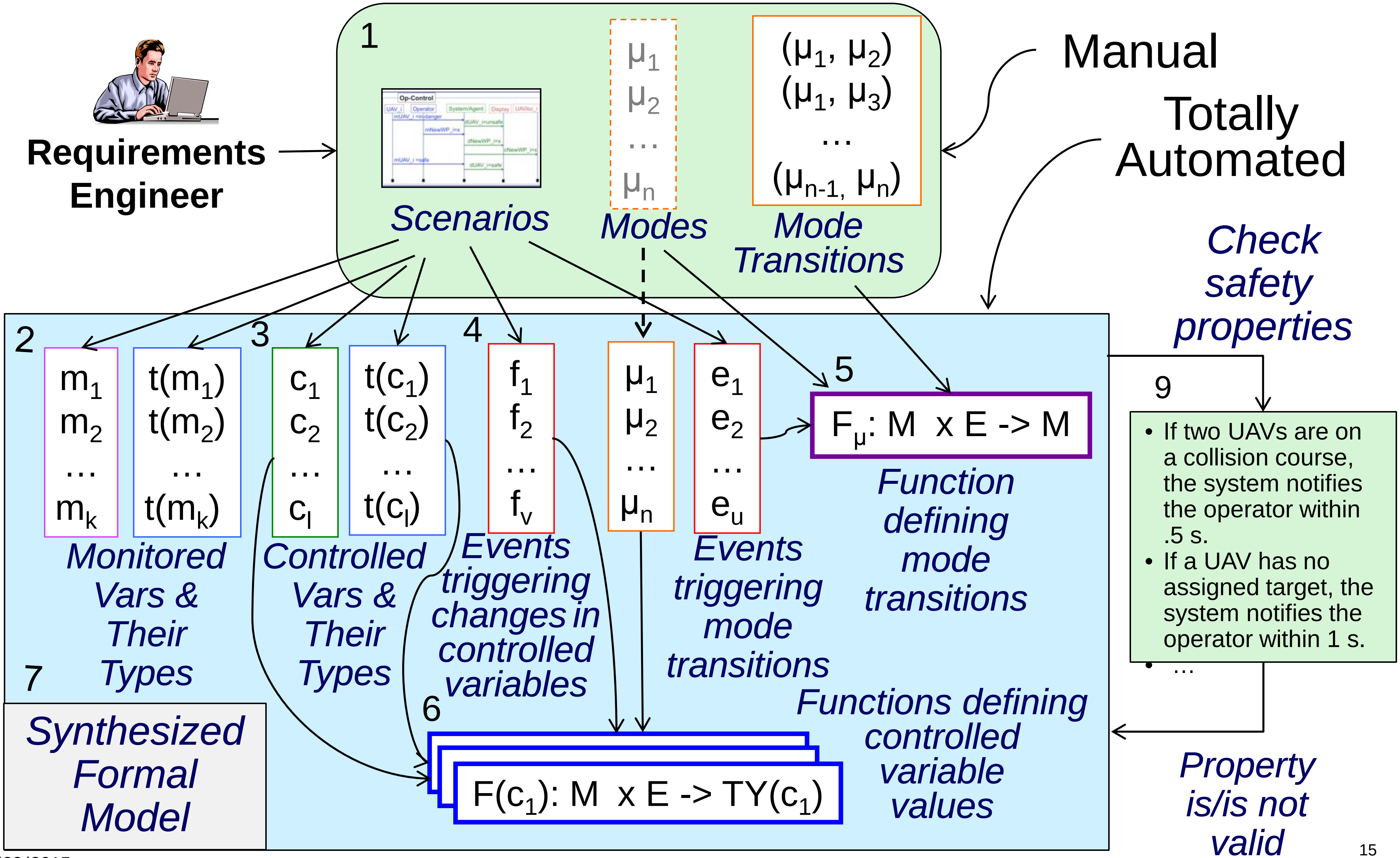
Formal System Model Synthesis: Method



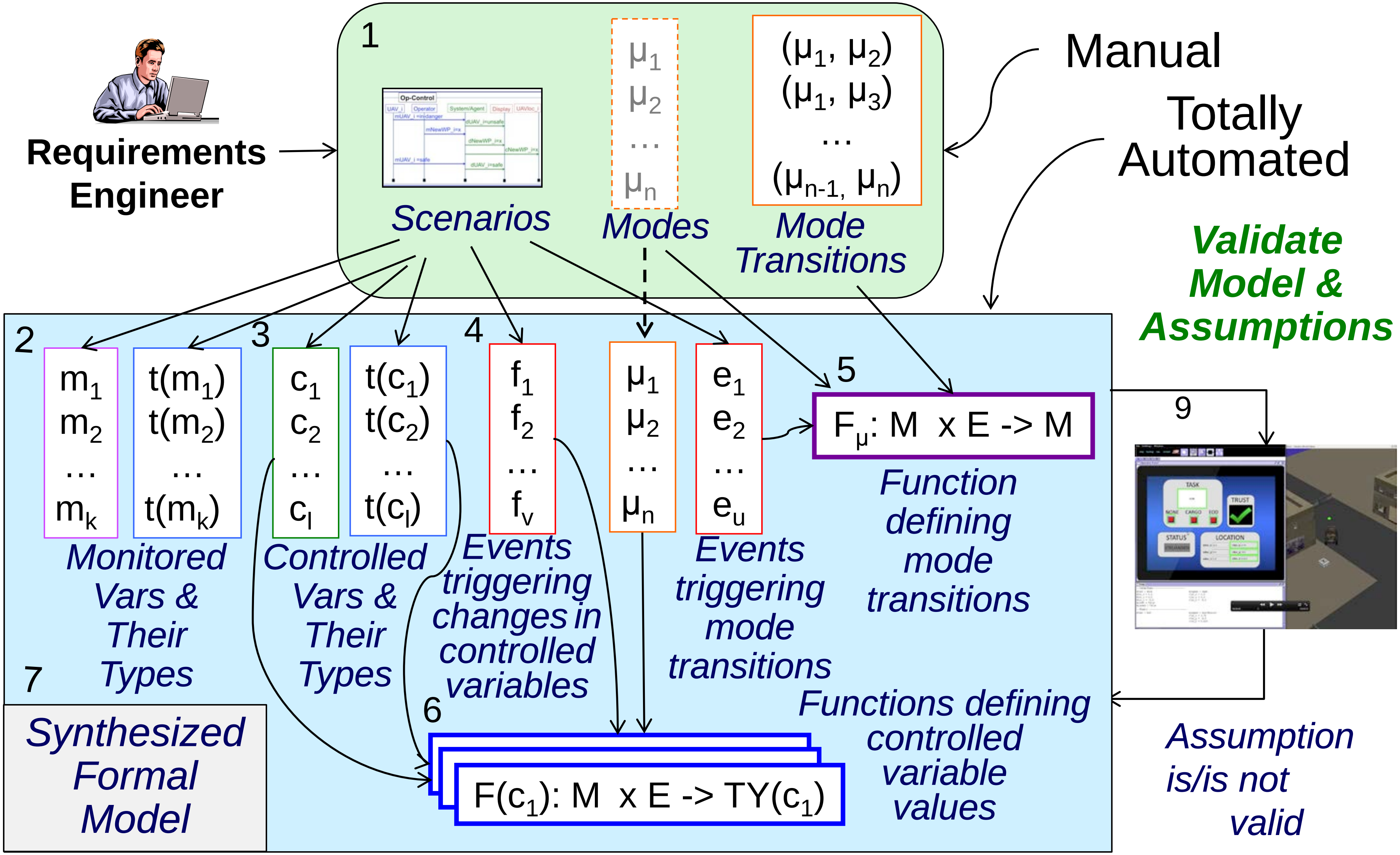
Formal System Model Synthesis: Method



Formal System Model Synthesis: Method



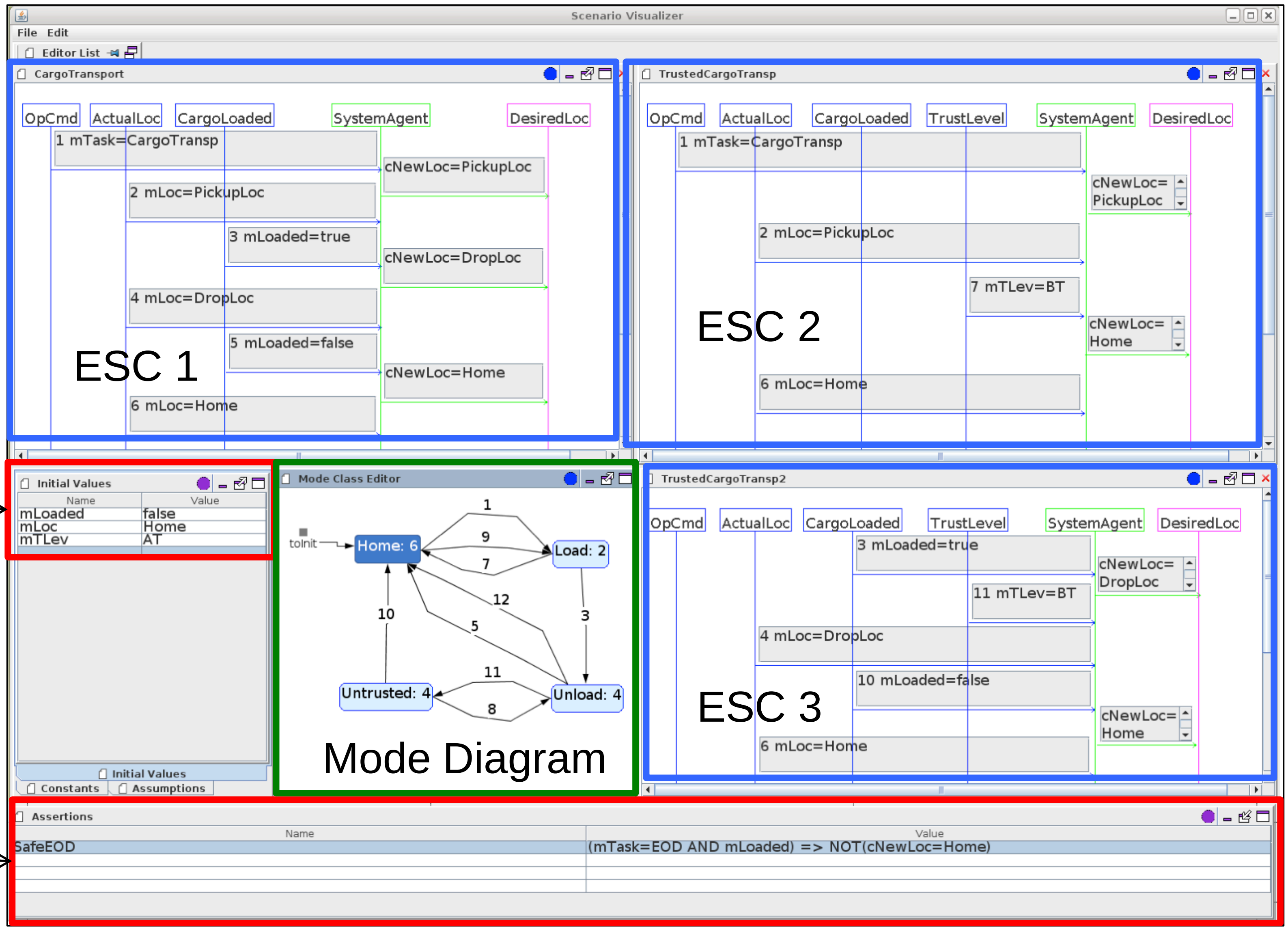
Synthesized Formal Model: Provides Basis for Validation



Our Tool's Representation of a Moded Scenario Description

Template defining initial values of variables

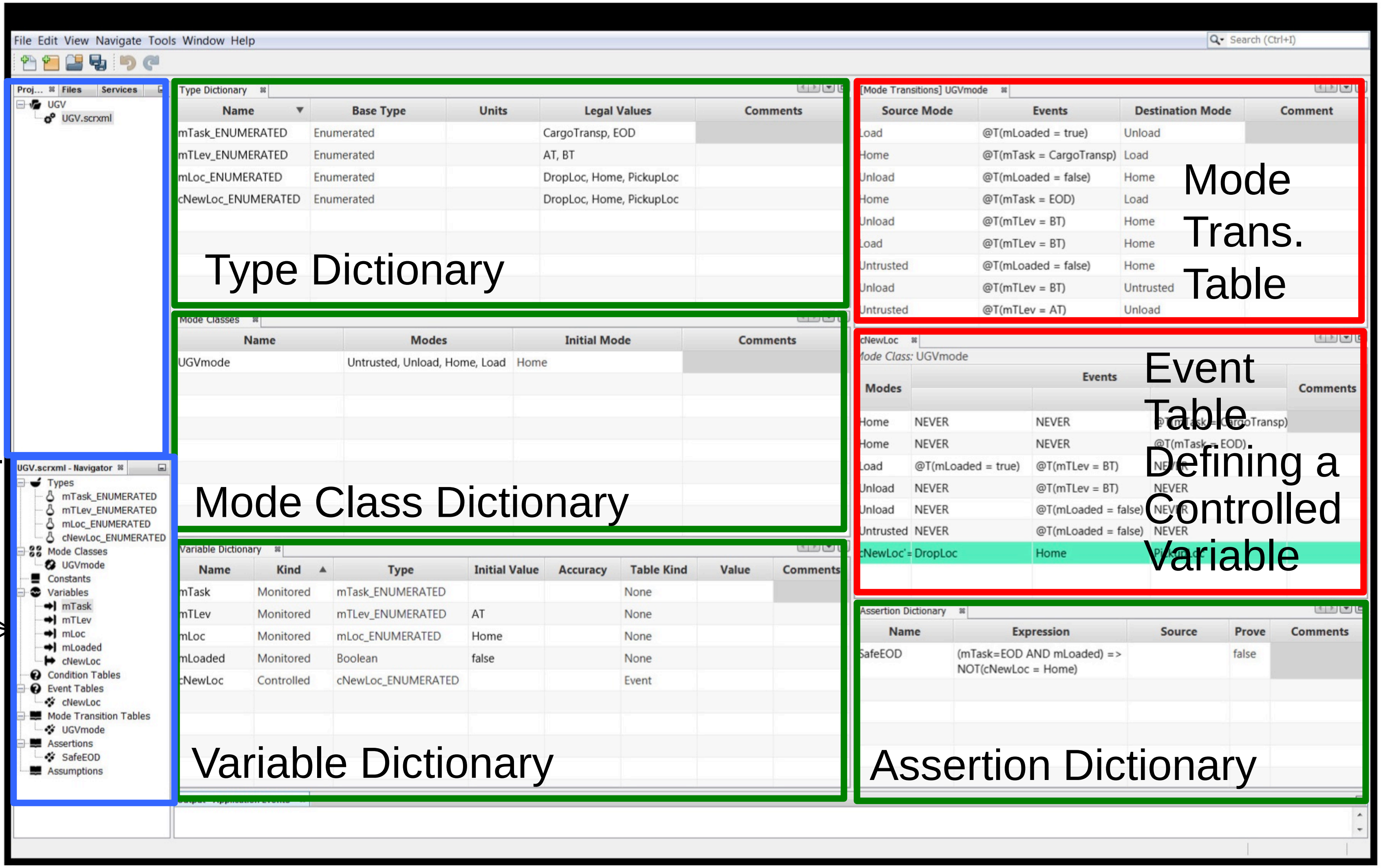
Template containing a single assertion



The Formal Model Synthesized from the MSD

List of Models

List of all Model Compo-nents

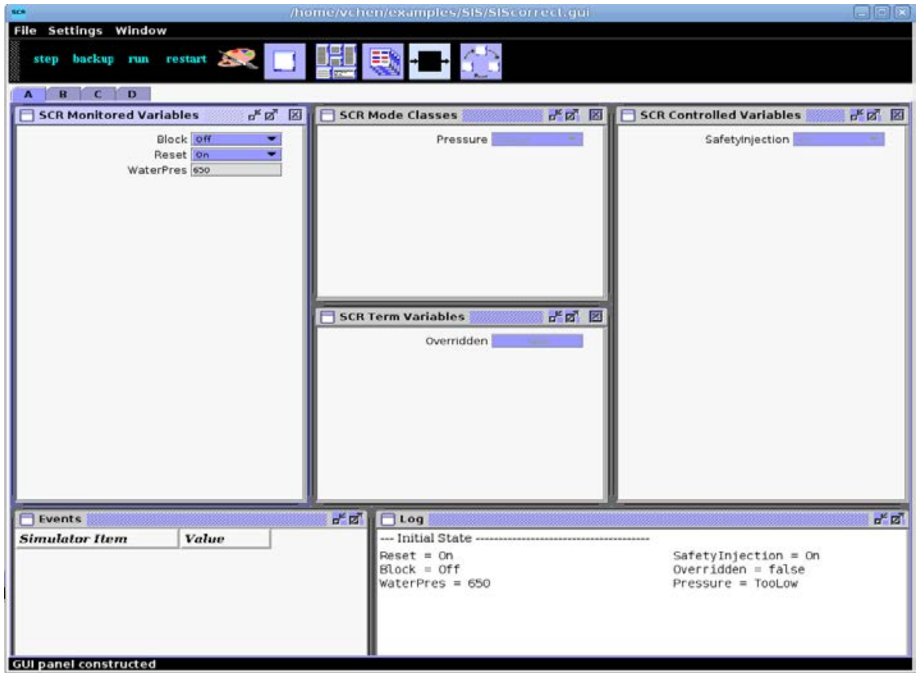


3-D Simulator

Simulators Based on a Formal Model

Many just have textual displays

A few (e.g., SCR, Statemate) allow creation of custom 2D GUIs



Logs each state change and notifies user when violations of assumptions or specified properties occur

Simple features such as buttons, switches, and dials

Limitations

No 3D, discrete computation only, no continuous movement

Two Types of Simulators: Formal Model Based vs Application-Specific

Approach: Integrate a formal model based simulator with an application-specific simulator

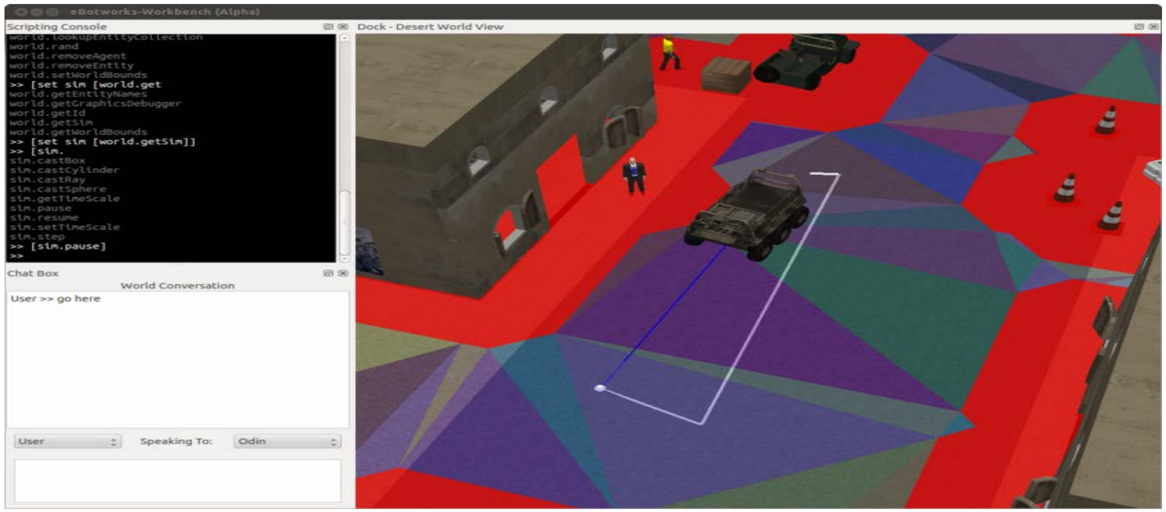
Process

1. Choose an appropriate application/domain simulator: Represents system's physical aspects and its operational environment
2. Use two simulators: E.g.,
   - a customized formal model based simulator as the system controller and
   - the application-specific simulator to represent the dynamic behavior of the system environment
3. Integrate the two simulators: Allows communications between the two at appropriate points during execution

Benefits of Integration

- From application-specific simulator: more realistic simulation
- From formal model tools (including simulator): formal foundation that allows notification of property violations during simulation

eBotworks*: An Application-Specific Simulator for UGVs (Unmanned Ground Vehicles)

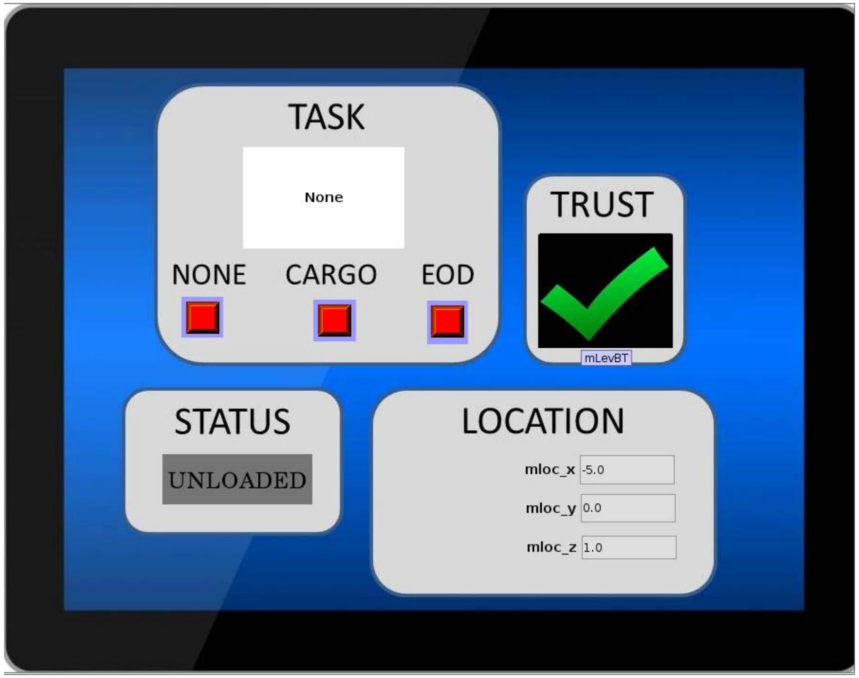


- Simulator and testbed for autonomy software for command and control of unmanned systems
- Built to support locomotion and path planning
- Wheeled UGV is the choice of vehicle we selected
- Using eBotworks, we built a simulated world containing landmarks (e.g., roads) and objects (e.g., packages, vehicles)

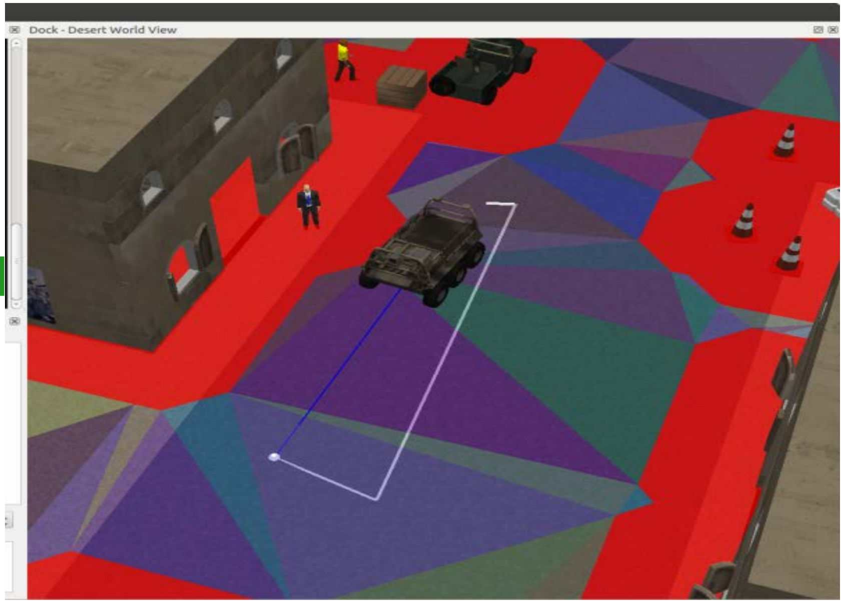
*http://www.knexusresearch.com/products/ebotworks.php

Integrating eBotworks with the SCR Simulator

System Controller: Customized GUI Front-End for the SCR Simulator



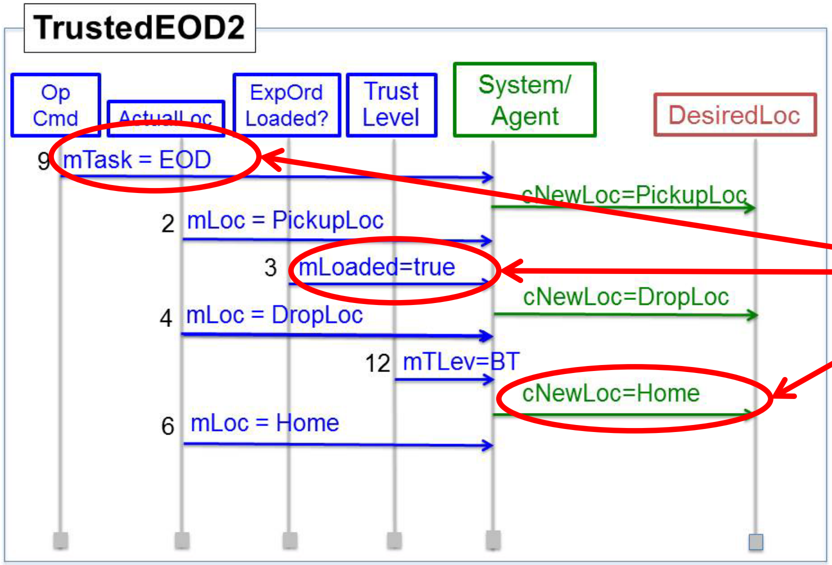
eBotworks: Displays system environment, vehicle location & motion, path planning



- User inputs (e.g., commands to perform a task and changes in trust measure) given via SCR simulator and passed to eBotworks
- eBotworks performs actions associated with commands, sending information about vehicle status and location back to SCR
- Integration via shared files

Validation of UGV Model: Property Checking During Simulation Exposed an Error

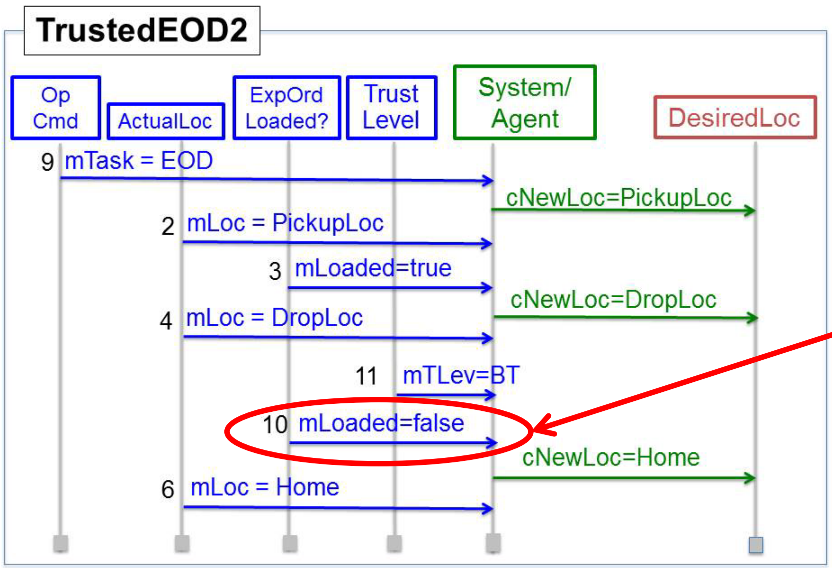
Task: Explosive Ordnance Disposal (EOD)



Bringing explosive ordnance home

=

UNWANTED SYSTEM BEHAVIOR



Unloading explosive ordnance before coming home

=

INTENDED SYSTEM BEHAVIOR

SUMMARY AND FUTURE WORK

- Benefit of Formal Methods Tools: High Assurance
- Two Important Gaps in Formal Methods Tools
  1. Getting an initial model
     - Addressed by synthesizing model from scenarios
  2. Simulating 3D, motion, continuous behavior
     - Addressed by integrating formal methods simulator with application-specific simulator
- Future Work:
  - Improved tool support for specifying scenarios and model synthesis
  - Develop SCR simulator interface to facilitate future integrations
  - Integrate SCR simulator with other application-specific simulators with more capabilities
    - AV2 Ground Vehicle
    - Unmanned Cargo Transport Helicopter

Role of Formal Methods in Developing "Intelligent" Autonomous Systems[1, 2]

- Needed research "ranges from economics, law, and philosophy to computer security [and] **formal methods**"
- "As autonomous systems become more prevalent in society, it becomes increasingly important that they robustly behave as intended. The development of autonomous vehicles, …autonomous weapons, etc., has therefore stoked interest in **high-assurance systems** where strong robustness guarantees can be made"
- "…society will reject autonomous agents unless we have some credible means of making them safe"
- **Formal verification and validation are critical…**

[1]"Research priorities for robust and beneficial artificial intelligence," Future of Life Institute, Jan. 2015

[2]"Benefits and risks of artificial intelligence," T. G. Dietterich, President, AAAI, Jan. 2015