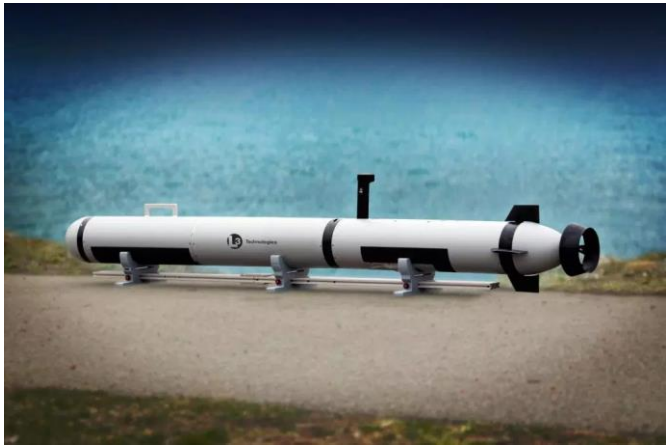
VANDERBILT UNIVERSITY
INSTITUTE FOR SOFTWARE INTEGRATED SYSTEMS

Autonomy and Safe Artificial Intelligence: Topics and Challenges

Gabor Karsai
EECS/ISIS

Autonomous Underwater Vehicles (AUV)

- 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.



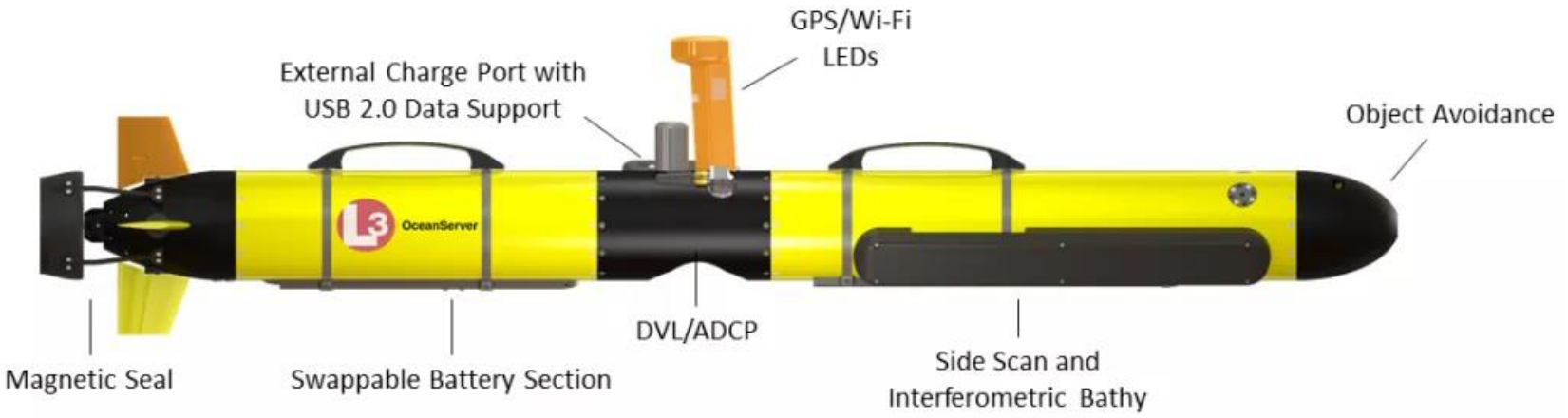
Source: https://ocean-server.com/

What makes AUVs difficult?

- Need for complete autonomy
  - Communications is problematic underwater; remote control only possible via tethers
  - Navigation is difficult – very few sensors, date is noisy
  - Highly uncertain environment: side currents, fishing nets, obstacles
  - Mission has to be adjusted on-the-fly
  - Long term operations (endurance) is desired – energy management is key
  - Failures in the system / environment must be anticipated

What is in a typical AUV?

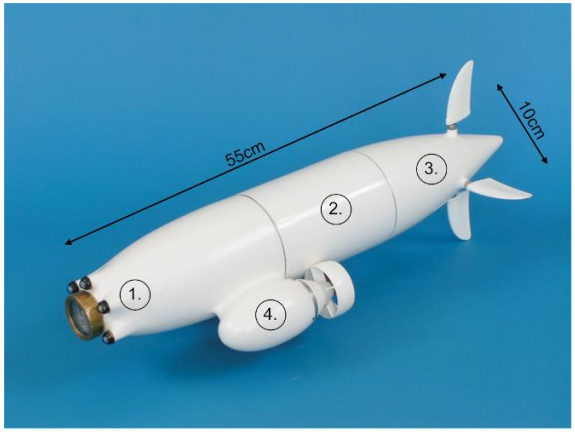
- L3 OceanServer IVER3

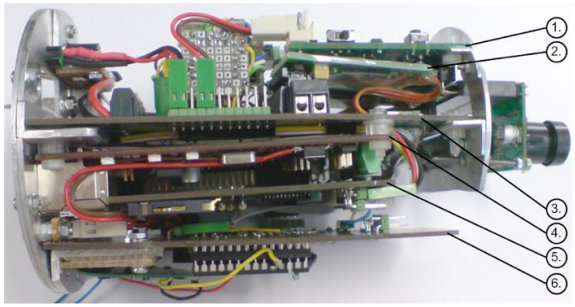


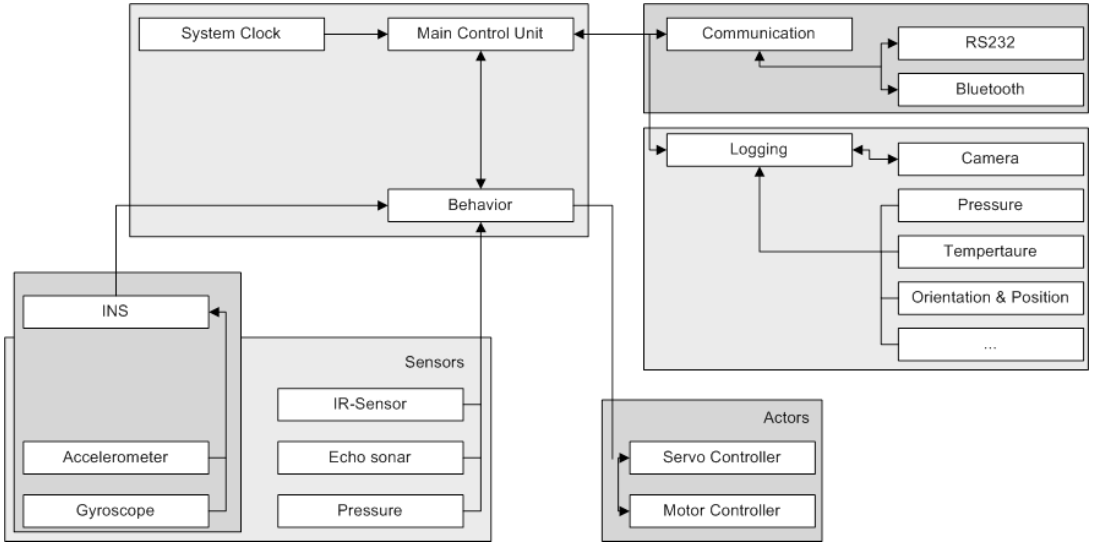
What is in a typical AUV?

- Dimensions: Standard Length: 60-85 in. Tube Diameter 5.8 in, Weight 59-85 lb.
- Depth Rating: 100 meters
- Endurance: 8 to 14 hours at speed of 2.5 knots, configuration dependent
- Speed Range: 1 to 4 knots (0.5 to 2.0 m/s)
- Communications: Wireless 802.11n Ethernet standard (Iridium optional)
- Antenna Mast: Navigation Lights with IR and visible LEDs (programmable strobe)
- Navigation: Surface: GPS (WAAS corrected) Subsurface: RPI Doppler Velocity Log (DVL), 81 m range, depth sensor and corrected compass
- Energy: 800 W hrs. of rechargeable Lithium-Ion batteries (Swappable section)
- Onboard Electronics: Intel Dual-Core 1.6 GHz N2600 processor with MS Windows embedded; Up to 512 GB solid-state drive for data storage
- Propulsion System: 48 V Servo Controlled DC Motor with three-blade cast bronze propeller
- Control: Four independent control planes (Pitch/Yaw Fins)
- Charging: 24 V External Connector with USB 2.0 Support

AUV Hardware/Software Architecture







- Osterloh, Christoph, Marek Litza, and Erik Maehle. "Hard-and Software Architecture of a Small Autonomous Underwater Vehicle for Environmental Monitoring Tasks." In Advances in Robotics Research, pp. 347-356. Springer, Berlin, Heidelberg, 2009.

Questions

- **Safety**: How do we know that the vehicle is safe to operate?
  - The vehicle must not take an unsafe control action…
- **Reliability**: How do we know that the vehicle is reliable?
  - The vehicle must execute its mission…

… even if components or subsystems degrade or fail.

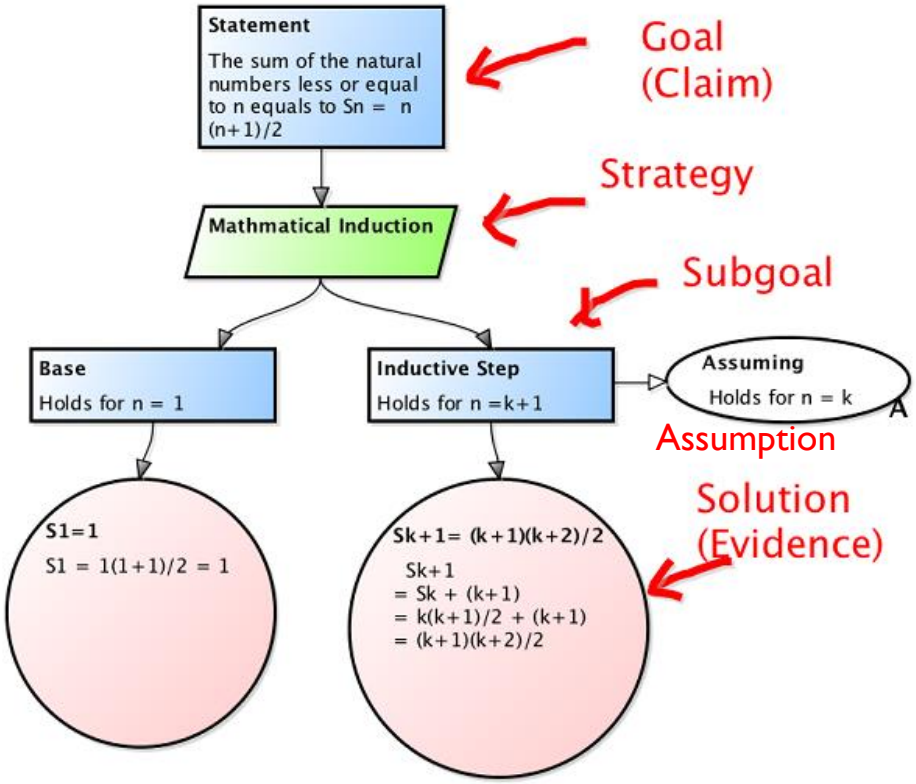
Central question of 'Safe AI':

'AI-based systems' are often created using machine learning techniques – training based on data. Learning is never perfect because the data set is never complete or perfect.

How do we design and prove that the AI-based system is (1) safe and (2) remains operational – even if its 'AI' is imperfect?

Goal Structuring Notation (GSN)

- A graphical tool to represent a logical argument



Source: http://www.goalstructuringnotation.info/

4+1 types of nodes:

Goal: What we want to prove ('safety claim')

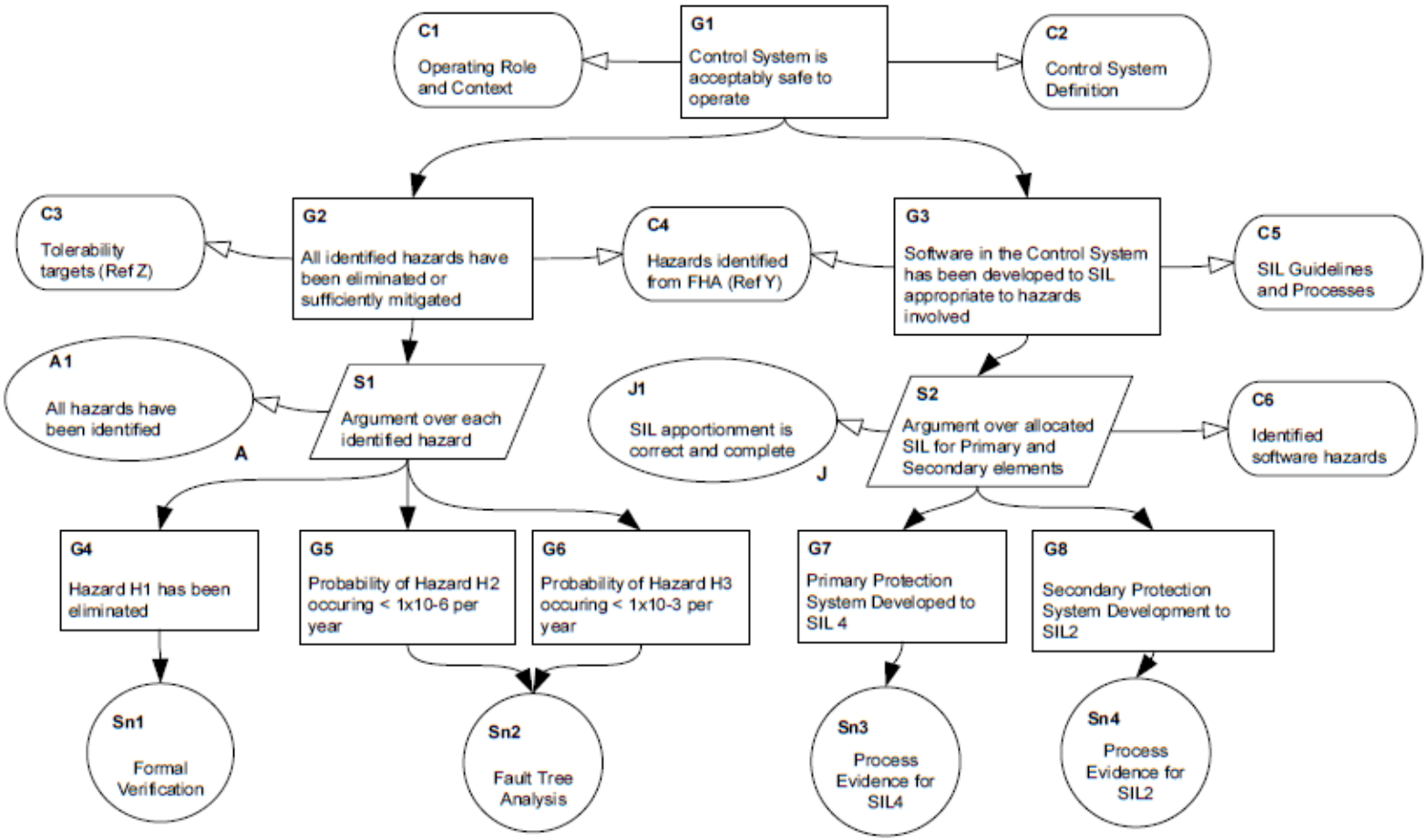
Assumption/Context: Under what circumstances

Strategy: How we go about proving the goal

Solution: Evidence to support a goal

Sub-goals: decomposition of a higher level goals

GSN Example



Source: http://www.goalstructuringnotation.info/

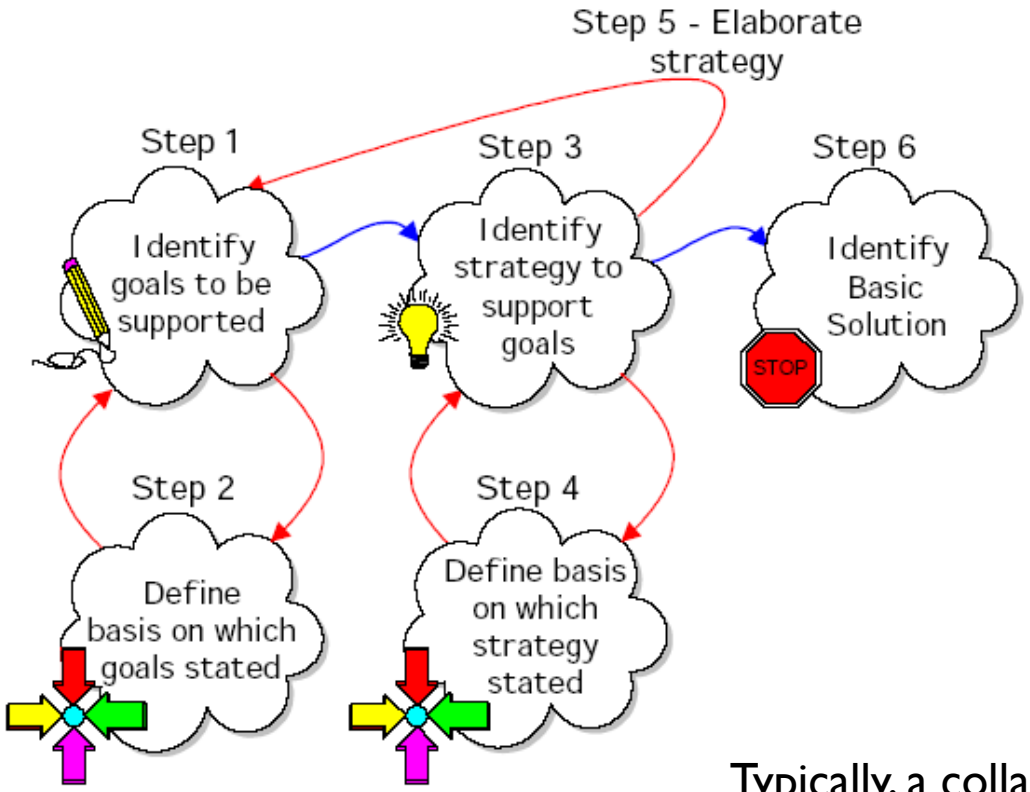
GSN Example - Vocabulary

- Hazard: a potentially dangerous occurrence
- FHA: Functional Hazard Analysis – an engineering process "to identify and classify the system functions and safety hazards, environmental, and health-related consequences associated with functional failure or malfunction"
- SIL: Safety Integrity Level – a relative level of risk-reduction provided by a safety function, or to specify a target level of risk reduction.
  - PFD: Probability of dangerous failure
- Formal verification: mathematical proof for software
- Fault-tree analysis – an engineering process to identify how (low-level) faults could get combined and cause (system-level) failures

SIL	PFD
1	0.1–0.01
2	0.01–0.001
3	0.001–0.0001
4	0.0001–0.00001

How to build a GSN for system?

- Six-step process

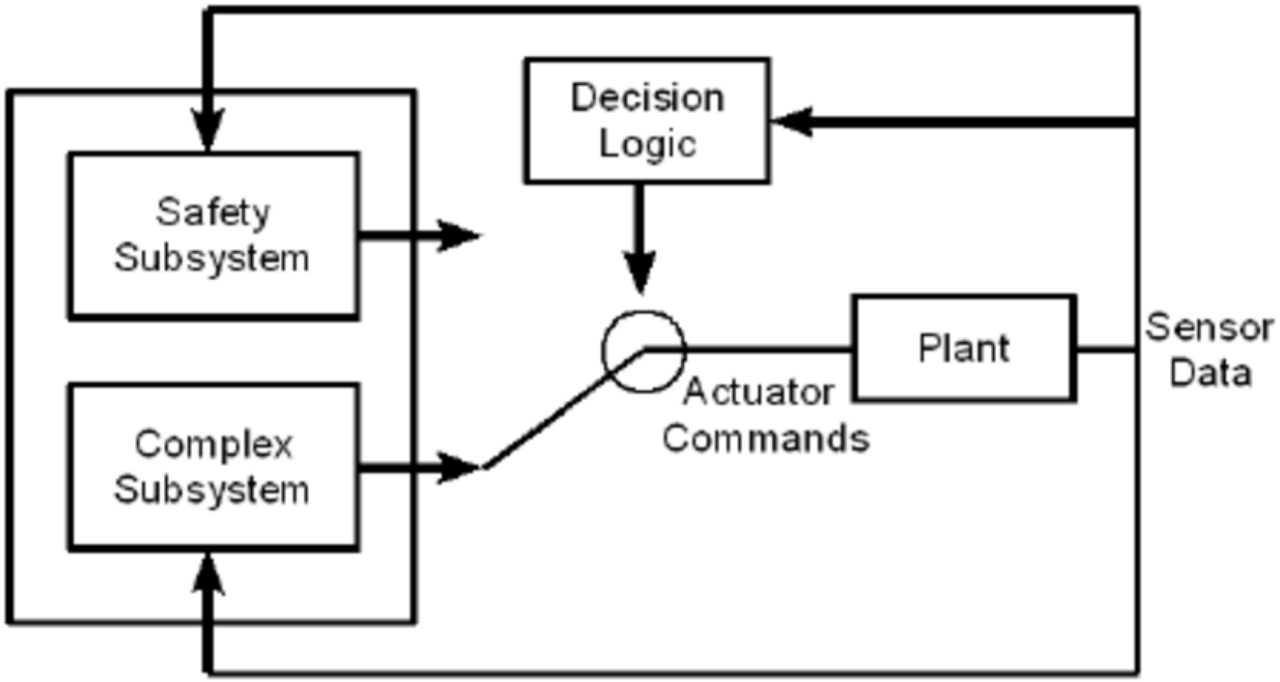


Typically, a collaborative process, exercised by a group of safety engineers

Source: http://www.goalstructuringnotation.info/

How to ensure safety of an AI system?

- An old idea: Simplex Architecture

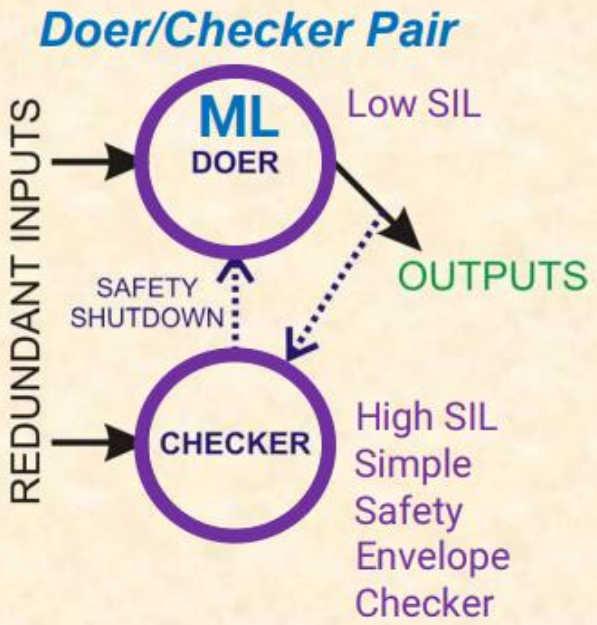


Rivera. Jose, Danylyszyn. Alejandro, Weinstock. Charles, Sha. Lui, and Gagliardi. Michael, "An Architectural Description of the Simplex Architecture," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Report CMU/SEI-96-TR-006, 1996.

Simplex for AI/ML-based systems

- Doer/Checker

- **"Doer" subsystem**
  - Implements normal, untrusted functionality
- **"Checker" subsystem – Traditional SW**
  - Implements failsafes (safety functions)
- **Checker entirely responsible for safety**
  - Doer can be at low Safety Integrity Level
  - Checker must be at higher SIL



Source: https://users.ece.cmu.edu/~koopman/pubs/koopman18_waise_keynote_slides.pdf

The problems with AI in autonomy

- Perception is a very difficult component to verify
- Real-life testing does not scale – math does not add up.
- Simulation-based testing only for cases we thought of
- Edge cases are very hard to anticipate (but we must)
- Adversary examples for perception show how easy it is to 'fool' it
- Autonomous systems (e.g. vehicles) must operate in a heterogeneous environment (fishing boats, human drivers, etc.)

Regulations can set the requirements but not the solutions.

To think about…