# High-Assurance Cyber-Physical Systems[1]

N. Shankar

Computer Science Laboratory
SRI International
Menlo Park, CA

Dec 9, 2014

[1] The talk covers work done in collaboration with Robin Larrieu, Léonard Gerard, Wenchao Li, and Sam Owre, and several other team members in the HACMS project. Supported by NASA NRA NNA13AC55C, NSF Grant CNS-0917375, and DARPA under agreement number FA8750-12-C-0284. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NASA, NSF, DARPA, or the U.S. Government.

# Background

- Cyber-physical systems are composed of physical and computational components, with multiple control loops operating at multiple time scales.
- Examples of such systems range from engine controllers, cars, and robots to factories, buildings, and power grids.
- These systems need to operate reliably even in the face of cyber-attacks — the attack surface includes the hardware, software, network, sensors, and actuators.
- The system must be accompanied by a rigorous assurance case justifying claims about system safety, security, and reliability.
- We present a framework for the *assurance-driven design* [Hall & Rapanotti, ICSEA 2008] of cyber-physical systems.
- The key ideas are inspired by the DARPA-funded project *High-Assurance Cyber-Military Systems* (HACMS) and by our partners at CMU, Kestrel, MIT, Penn, and Yale.

# Securing Cyber-Physical Systems

- A misbehaving cyber-physical systems, e.g., a nuclear power plant or a medical device, can cause physical harm.
- The *Carshark* project demonstrated that cars are vulnerable to even remote attacks.

# Motivation

- An assurance case is a *"a documented body of evidence that provides a convincing and valid argument that a specified set of critical claims about a system's properties are adequately justified for a given application in a given environment."* [Adelard]
- From the FDA Draft Guidance document *Total Product Life Cycle: Infusion Pump - Premarket Notification [510(k)] Submissions*:

  *An assurance case is a formal method for demonstrating the validity of a claim by providing a convincing argument together with supporting evidence. It is a way to structure arguments to help ensure that top-level claims are credible and supported. In an assurance case, many arguments, with their supporting evidence, may be grouped under one top-level claim. For a complex case, there may be a complex web of arguments and sub-claims.*

# Layering and Composition

- The security and safety argument is captured in the claim that the system performs its physical function under the given fault/attack assumptions.
- This argument is captured in layered *theory refinement* steps that defines the platform and controller to satisfy the axioms.
- Compositionality is addressed by using a Model of Computation (MoC) that allows components to operate independently, yet coherently.
- Each component implements a simple contract that relates its outputs to its inputs.
- Independence of processing and communication channels is realized using a hypervisor.

# Efficient Arguments

The goal of assurance-driven design is an *efficient argument*, i.e., one that is easily refuted if wrong.

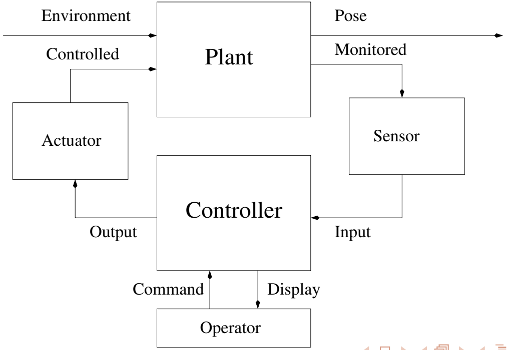
| | Efficient | Inefficient |
|---|---|---|
| Claims | Precise + Simple | Vague + Complex |
| Assumptions | Easily validated | Not easily validated |
| Architecture | Separates concerns | No separation |
| Arguments | Reusable | One-of |
| Evidence | Replayable | Irreproducible |
| Tools | Trusted | Unsound |

The point of the argument is to fail big, e.g., missing hazard, incomplete attacker model, uncheckable inference step.

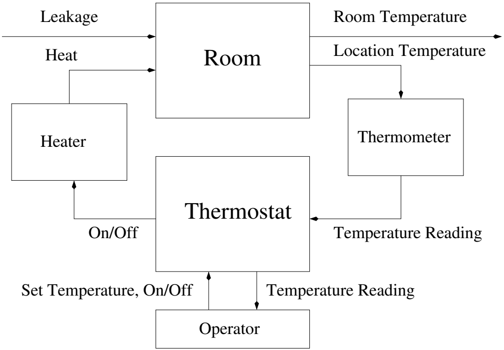
# Overview

- Assurance for cyber-physical systems
- The Landshark robot architecture
- Robot Architecture Definition Language (RADL)
- The quasi-synchronous (QS) model of computation (MoC).
- The Landshark robot software architecture
- The Evidential Tool Bus (ETB) framework for assurance workflows
- A compositional assurance case for the Landshark
- Key principle: *Keep the design static (schedules, addresses, memory partition, buffers) and decoupled.*

- Cyber-physical systems are composed of physical and computational components, with multiple control loops operating at multiple time scales.
- CP systems are typically distributed and consist of a network of sensors, controllers, and actuators.
- The system can be structured as an "eight variables" control loop with a plant and a controller.

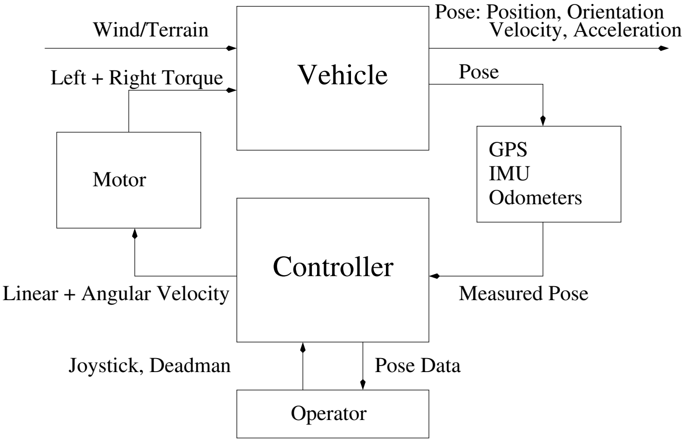


# A Simple Example: Room-Heating Thermostat

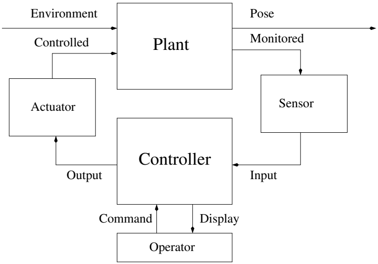



1. The *plant* consists of the room whose temperature is being maintained, the *actuator* is the heater, and the *environment* is the energy leakage from the room.
2. The goal *requirement* is to maintain the average temperature across the room between two bounds set by the operator.

# The Landshark Eight Variables Model



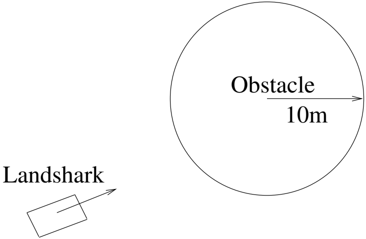
# Top Assurance Claim



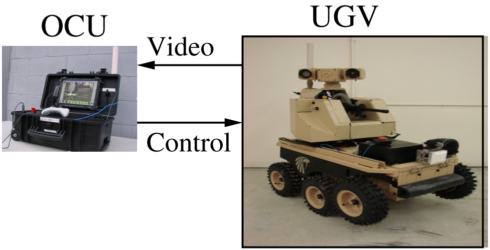

```
  EnvironmentAssumption(environment) AND
  PlantModel(environment, control, pose, monitor) AND
  SensorAccuracy(monitor, input) AND
  ActuatorResponse(output, control) AND
  ControllerOutput(input, command, output, display) AND
  OperatorModel(display, command)
IMPLIES
 Requirement(command, environment, pose, display)
```

# Landshark Requirements

- The Landshark starts, stops, and moves as directed from the joystick.
- The operator commands include switching control between the CMU Path Planner (PP), the Penn Constant-speed Cruise Controller (CCC), and the joystick.
- *Emergency stop* must bring the vehicle to a halt
  1. To avoid hitting a pre-specified obstacle, or
  2. When no one is in control, e.g., when communication is jammed or if the deadman is not engaged.



# Attacker Model




The attacker can initiate an attack on the Unmanned Ground Vehicle (UGV) either by:

1. Inserting malware into a separate partition
2. Spoofing or damaging a sensor
3. Gaining access to the network partition — the red team can log into a partition.
4. Spoofing or jamming external communication

The Operator Console Unit (OCU) is out-of-bounds, mainly because the real controller will be a hardware device.

# Environment Assumptions

- The environment consists of the terrain with its *grade* and *slippage*, the *wind resistance* and *friction*, and the location of any obstacles.
- Bounded grade: The operating surface is reasonably level so that there is no vertical component to the torque.
- Bounded resistance: The external forces on the vehicle, e.g., friction, wind resistance, is bounded by $R(s)$ for speed $s$.
- Bounded slippage: The vehicle pose should reflect the control inputs given the resistance and the grade.
- In Phase 2, the obstacle is located at a fixed location $O$, but in general, the obstacle could be part of the environment.

# Plant Assumptions

- The vehicle dynamics are captured by the kinetic model used in developing
  1. Resilient state estimator (RSE)
  2. Constant-speed cruise controller (CCC)
  3. Obstacle avoidance controller (OAC) and Path Planner (PP)
- Specifically,
  1. The torque control input (braking or acceleration) is bounded: $-B \leq F \leq M$
  2. The speed of the vehicle is bounded from above: $|v| \leq V$.
  3. The vehicle responds to the torque input while respecting the above bounds: $F/m - R(|v|) \leq a \leq F/m$, if $|v| \leq V$, and $a = 0$, otherwise.
- The monitored quantities are position $p$ and speed $|v|$.

# Sensor/Actuator Assumptions

- The three sensors are the GPS, and the left and right wheel odometry.
- The Phase 2 Landshark will have an improved IMU and a LIDAR.
- At most one sensor is faulty.
- The other sensors estimate position and speed with bounded error.
- The vehicle knows its starting position up to some error.
- The actuator primarily converts the controller's acceleration and emergency stop outputs to torque $F$ with a bounded error.

# Controller Claims [Penn + CMU + SRI]

- Assuming that the computations are "timely" ...
- The Resilient State Estimator (RSE) estimates the speed within a bound (10%) of the true vehicle speed even when one of the sensors is faulty or compromised.
- The absolute difference between the actual and desired speed remains bounded after the Constant-Speed Cruise Controller (CCC) has been in control for at least $T$ time units.
- When the CCC is given control, the difference between the actual and desired speed converges exponentially to the bound above within $T$ time units.
- The Obstacle Avoidance Controller (OAC) ensures safe braking to avoid contact with the obstacle.

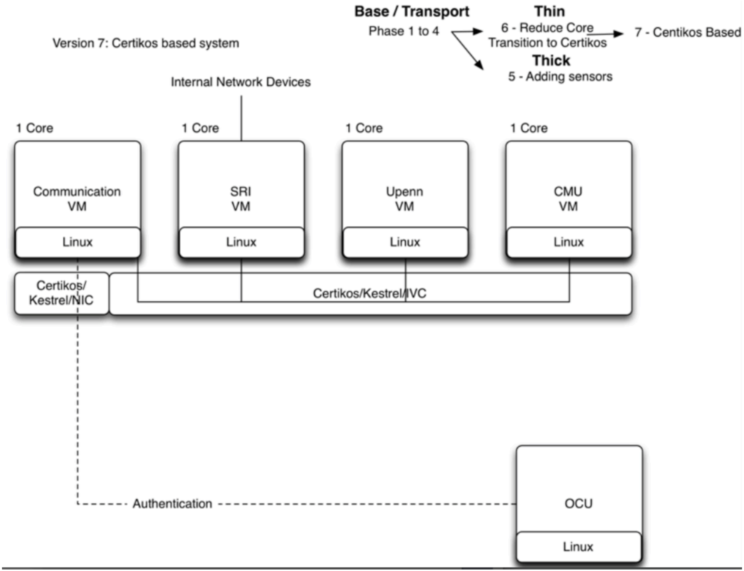
# Refinement Layers in the Assurance Argument

The argument is structured into three layers. Each layer implements the assumptions imposed by the higher one:

1. **The Mathematical Model:** Spatio-temporal models of the physics of the vehicle, the environment assumptions, the system-level requirements, and the mathematical designs of the controllers and monitors.
2. **The Engineering Model:** Architectural models for plants and controllers/monitors, fault models for the physical components, with a model of computation (MoC) for communication and computation.
3. **The Computation Model:** Platform model implementing the MoC and discharging the platform assumptions.

Each layer also introduces fault models and mitigations for the components relevant to it.

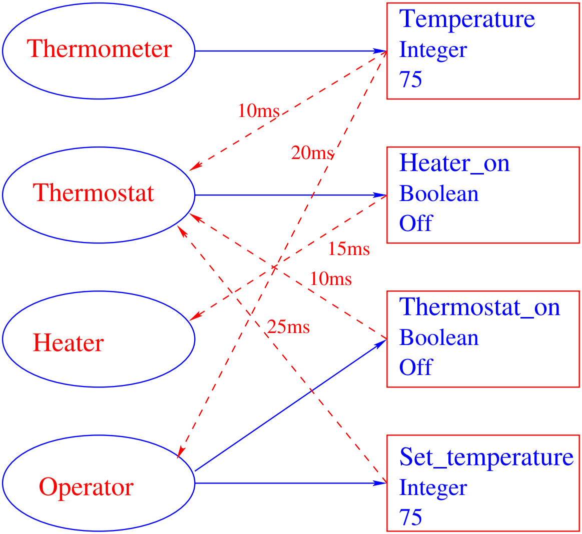
# The Landshark Platform [from Regis Vincent]



# Robot Architecture Definition Language (RADL)

- The Robot ADL bridges the gap between the engineering and computation models.
- The ADL is inspired by the popular Robot Operating System (ROS) middleware for building cyber-physical systems.
- The architecture definition captures
  1. Message types
  2. Nodes, with their period, and steady-state computation steps, published topics, received topics (with latency bounds), and devices
  3. Topics, with a message type, period, and authentication
  4. Mapping of nodes to partitions within processors and associated devices
  5. Mapping of channels to buses with firewalls and authentication

# Thermostat in RADL



# Robot Architecture Definition Language (RADL)

- A *logical* architecture (setting up the platform assumptions) consisting of
  1. Nodes with
     1. Publications with defaults
     2. Subscriptions with defaults and maximum latencies
     3. Period
     4. Worst-case execution time
     5. Step function
     6. Device interfaces
  2. Topics with message types and default values
- A *physical* architecture (discharging platform assumptions) consisting of
  1. Processors
  2. Partitions assigned to processors, containing nodes
  3. Buses with devices and partitions as endpoints
  4. Devices
- A certified build process that produces the image matching the assurance case.

# Basic Assurance Case

- We want to prove that the system satisfies its requirement.
- RADL Logical Architecture (LA) + Abstract components + RADL Theorems + 8-Var Assumptions implies Requirement
- RADL MoC Semantics implies RADL Theorems (this talk)
- Each Concrete component implies Abstract component
- RADL Physical Architecture (PA) implements RADL Logical Architecture + RADL Semantics
- Hence, RADL PA + Concrete components + 8-Var Assumptions implies Requirement

# The Quasi-Synchronous Model of Computation

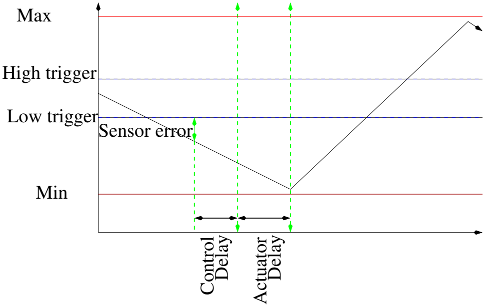
- Synchronous model has been heavily studied, but many typical distributed control systems applications don't need synchrony.
- The quasi-synchronous (QS) model consists of locally clocked periodic processes, with a bounded clock drift and bounded message latencies.[2]
- The QS model entails the following RADL theorems (verified in PVS; 195 lemmas):
  1. Bounded processing latency for message
  2. No overtaking, with timing assumptions
  3. Bounded consecutive message loss
  4. Bounded queue length to eliminate message loss
  5. Bounded age $MA(m)$ of message inputs $m$ used by subscriber
- These theorems are used to prove physical system properties, e.g., room-heating thermostat and obstacle avoidance controller.

[2] The QS model subsumes the loosely time-triggered architecture (LTTA) which uses shared memory broadcasts — all LTTA theorems apply to QS.

# On Quasi-synchrony

*. . . in practice, at least in the domain of critical control systems, the use of clock synchronization is not so frequent . . . We believe there are historical reasons for this fact, which can be found in the evolution of these systems: control systems formerly used to be implemented with analog techniques, that is to say without any clock at all. Then, these implementations evolved toward discrete digital ones, and then toward computing ones. When a computer replaced an analog board, it was usually based on period sampling according to a real-time clock. For the sake of modularity, when several computers replaced analog boards, each one came with its own clock.* *Paul Caspi*

- Characteristic time
  $\lambda = MA(Input) + MA(Output) + maxT(Control)$, where $maxT$ gives the maximum period of a node.




- Stability: When the thermostat is on, once the temperature is strongly safe (roughly between Low and High Trigger), it remains safe (between Min and Max).
- Convergence: When the thermostat is turned on, the temperature is strongly safe within a time bound.

# A Quick Overview of ETB

- The Evidential Tool Bus (ETB) is a distributed tool integration framework for constructing and maintaining claims supported by arguments based on evidence.
- ETB uses Datalog as a metalanguage for defining workflows and constructing assurance claims with supporting arguments and evidence.
- Interpreted Datalog predicates invoke external tool services.
- Arguments are Datalog derivations covering claims about files (file hashes) in a `git` repository.
- Formal semantics defined for interpreted predicates with abstract machine for tabled evaluation augmented with a novel mechanism for termination detection.
- Assurance cases are built with two workflows: artifact construction and artifact validation.

# An Example ETB Workflow: AllSAT

The defined predicates `sat` and `unsat` invoke the interpreted `yices` predicate on the given file `F`.

```
sat(F, M) :- yices(F, S, M),
             equal(S, 'sat').
unsat(F) :- yices(F, S, M),
            equal(S, 'unsat').

allsat(F, Answers) :- sat(F, M),
                      negateModel(F, M, NewF),
                      allsat(NewF, T),
                      cons(M, T, Answers).
allsat(F, Answers) :- unsat(F),
                      nil(Answers).
```

Though `allsat` calls `sat` and `unsat`, the `yices` wrapper is only executed once on the file `F` since the resulting claim is tabled.

# DO-178C A4: Verification of Outputs of Software Design Process

```
a4verify(AA, A4Evidence) :-
          a4artifacts(AA, HLR, LLR, SWA, A4Checklist, ReviewDocs),
          a4checklist_reviewed(A4Checklist, ReviewResult),
          a4verify_checklist(HLR, LLR, SWA, A4Checklist,
                                           ..., A4Evidence).
```

`a4verify_against_checklist` is actually a wrapper that generates the clause below, which triggers the tool wrappers.

```
a4verify_checklist(HLR, LLR, SWA, A4Checklist, ..., A4Evidence) :-
    design_llr_hlr_compliance(LLR, HLR, VerificationReport_1),
    design_model_trace_anchor(LLR, HLR, VerificationReport_2),
    ...,
    design_llr_conforms_to_standards(LLR, VerificationReport_5)
    ...
    package_a4evidence(..., VerificationReport_5, ..., A4Evidence)
```

# Conclusions

- Cyber-physical systems rely on software and networking.
- The safety and security of these systems is a critical challenge.
- High assurance requires static configuration and aggressive decoupling.
- Quasi-synchrony is a simple, reusable model of computation for high assurance.
- The RADL architecture definition cleanly separates logical from physical architecture.
- Assurance-Driven Design of cyber-physical systems has the goal of producing an assurance argument for the design.
- Layering and composition make the argument *efficient*, i.e., easily refutable if wrong.
- The Evidential Tool Bus is used to develop and maintain assurance arguments and artifacts.