Architecture Descriptions for HACMS

Pat Lincoln & N. Shankar

Computer Science Laboratory
SRI International
Menlo Park, CA

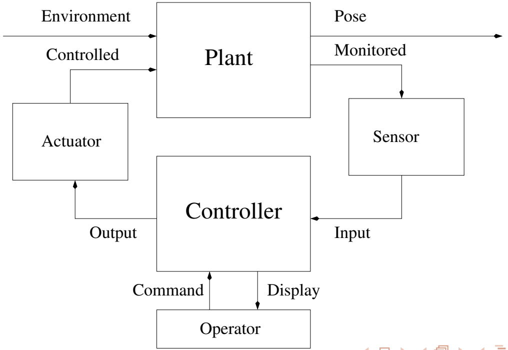
April 2, 2014

Software Architecture

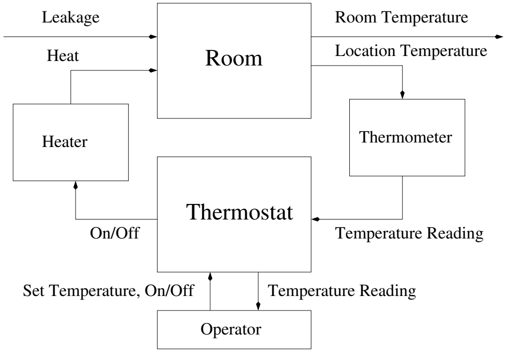
- Architecture is a salient aspect of software systems
- It captures the structure of the system and the interaction between the components and sub-systems.
- The architectural design of a software system guides
  1. Design decomposition
  2. Compositional verification
  3. Fault tolerance and resilience
  4. Security of the channels
- We present an Architecture Description Language (ADL) as a core artifact for the HACMS assurance effort.

Cyber-Physical Systems: Eight Variables Model

- These are systems composed of physical and computational components, with multiple control loops operating at multiple time scales.
- CP stems are typically distributed and consist of a network of sensors, controllers, and actuators.
- The whole system can itself be seen as a giant control loop with a plant and a controller.

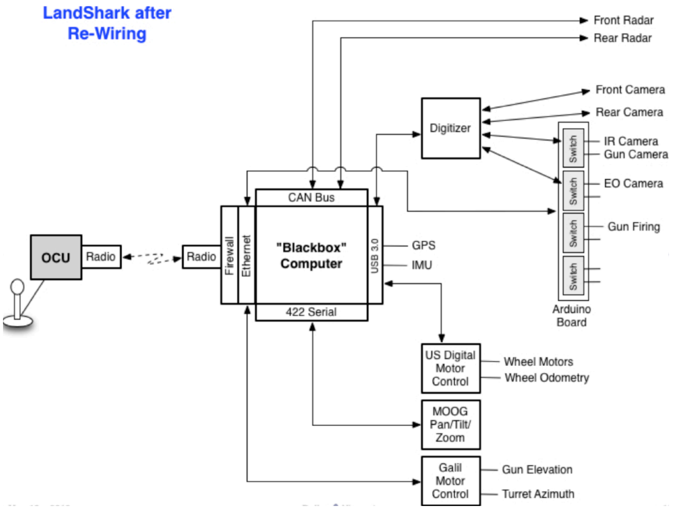


A Simple Example: Room-Heating Thermostat



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 above a specified temperature that is set by the operator.

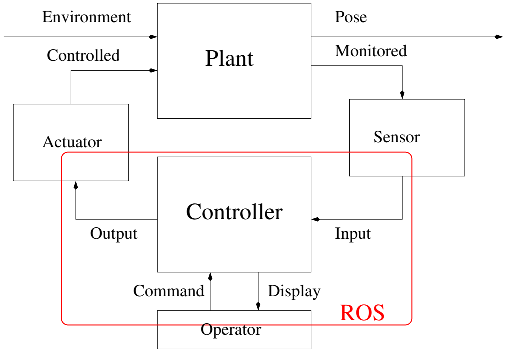
The Landshark Ground Vehicle [Bolles-Vincent]



The vehicle consists of the Operator Console Unit (OCU) and the Unmanned Ground Vehicle (UGV) running the Blackbox.

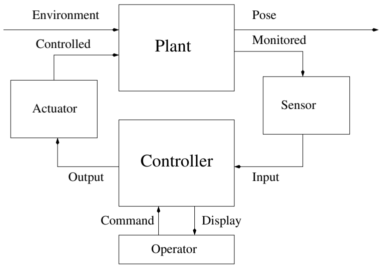
The ROS Platform

The Robot Operating System (ROS) provides a publish/subscribe architecture which is used in the HACMS Landshark.



The software for the sensors, controllers, actuators, and operator/display are in ROS nodes that communicate through ROS messages.

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)
```

Thermostat Requirement

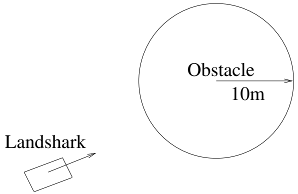
- **Environment:** We assume that the room has a leakage rate *lr*, a negative quantity, is bounded below by a negative constant $-\mu$.
- **Actuator Response:** We assume that the heater can heat the room at a rate *hr* which is 0 when the heater is off, and is bounded below by ν, with $\nu > \mu$.
- **Sensor Accuracy:** The sensed temperature $\hat{\theta}$ is always within ϵ of the actual temperature θ.
- **Plant Model:** The temperature of the room θ is such that $\dot{\theta} = hr + lr$, and $0 \leq \dot{\theta} > -\mu$ if the heater is off, and $\nu - \mu \leq \dot{\theta}$.
- **Operator:** The operator turns the thermostat *on* or *off*, and sets the desired temperature to τ.
- **Controller:** The controller simply turns the heater on as soon as the temperature is sensed to be below $\tau + \Delta$, where $\Delta > \epsilon + \mu\lambda$, where
  - ϵ is the error in sensing the room temperature with the thermometer

Thermostat Claim

- If the thermostat is on and the set temperature is τ and current sensed temperature is θ_0, then the sensed temperature θ is at least $\tau + \Delta + \epsilon$ within $(\tau + \Delta + 2\epsilon - \theta_0)/(\nu - \mu)$ time, unless the thermostat is switch off or the set temperature is changed (i.e., under operator quiescence).
- The thermostat controller may switch off the heater when the sensed temperature exceeds $\tau + \Delta$ by some bound Δ'.
- Even so, once the sensed temperature $\hat{\theta}$ exceeds $\tau + \Delta$, the temperature θ never falls below the set temperature τ under operator quiescense.

A Target Requirement: Obstacle Avoidance

- As a challenge property for end-to-end assurance, we target an obstacle avoidance (OA) maneuver.



- The obstacle is given as a GPS point.
- The main requirement is that in responding to OCU inputs, the UGV must avoid bringing the vehicle to within a ten meter radius of the obstacle.
- If the vehicle is already in violation of the obstacle, it only accepts inputs that are guaranteed to move the vehicle away from the obstacle zone.

The Refinement Layers in the Assurance Argument

- The argument is structured into three refinement layers where each layer is shown to implement the assumptions imposed by the higher layer:
  1. **The Mathematical Model:** A spatio-temporal model that captures 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:** Algorithmic/architectural models for plants and controllers/monitors, fault models for the physical components, and platform models for communication and computation.
  3. **The Computation Model:** The software for modules in the engineering model are turned into ROS nodes executing as processes within a hypervisor partition and communicating using hypervisor/network services.
- Each layer also introduces fault models and mitigations for the components relevant to it.

The Mathematical Argument

- The high-level requirement requires the vehicle to follow the operator commands without entering the obstacle zone.
  - A state estimation algorithm captures an accurate estimate of the pose of the vehicle as long as a majority of sensors are reliable to within a given bound. [Penn]
  - A obstacle avoidance controller blocks commands from the OCU that, based on the state estimate, the actuator response, and vehicle dynamics, might cause the vehicle to violate the obstacle zone. [CMU]
  - The OCU operator commands are delivered to the controller in a timely manner
- The mathematical argument is formalized in PVS with claims verified using HybridSAL and KeyMaera.

The Engineering Model

- The engineering model introduces two controller components:
  1. A Resilient State Estimator (RSE) that computes an estimate of the position and velocity of the vehicle while allowing for a minority of faulty sensors.
  2. An Obstacle Detector (OD) that uses the state estimate from the RSE to ignore or respond to incoming commands
- The platform model assumes a globally periodic locally synchronous model of computation with
  1. Periodically scheduled processes with a bounded jitter
  2. Communicating through a timely and reliable (integrity + authentication + bounded latency) channels
- The engineering models employ Simulink and claims establishing correspondence to the mathematical model are verified using PVS and SimCheck.

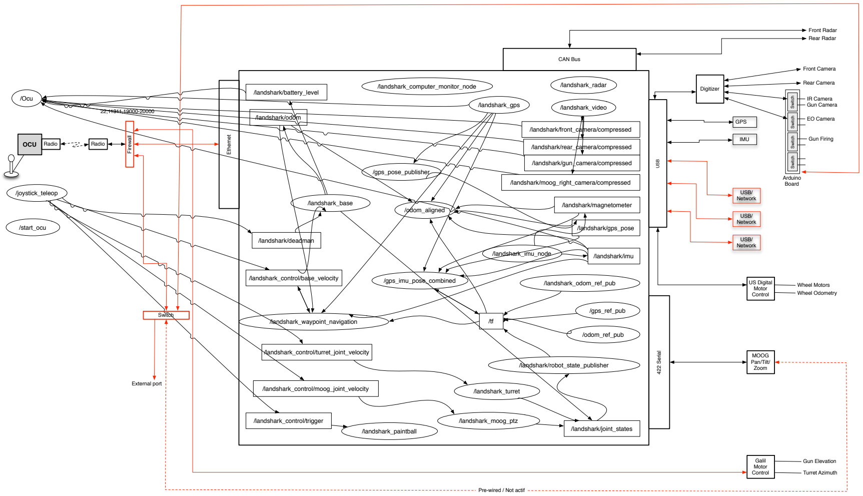
Computation Model

- The main obligation here is to ensure that the platform has been correctly realized.
- The computation model uses the Robot Operating System (ROS) platform with
    1. A start-up procedure that places the Landshark system in a stable state where the partitions, nodes, devices, communication links have been initialized. [MIT]
    2. A scheduler that coordinates the execution of the tasks to satisfy the platform assumptions of the engineering model [SRI, Yale]
    3. Authenticated/monitored communication between ROS nodes [SRI, Princeton, Kestrel, UIUC]
- The ROS nodes on the vehicle run in separate hypervisor partitions using inter-partition communication. [Yale]
- All message traffic from external nodes are filtered through a firewall (to ensure that no internal addresses are spoofed). [SRI, Kestrel]

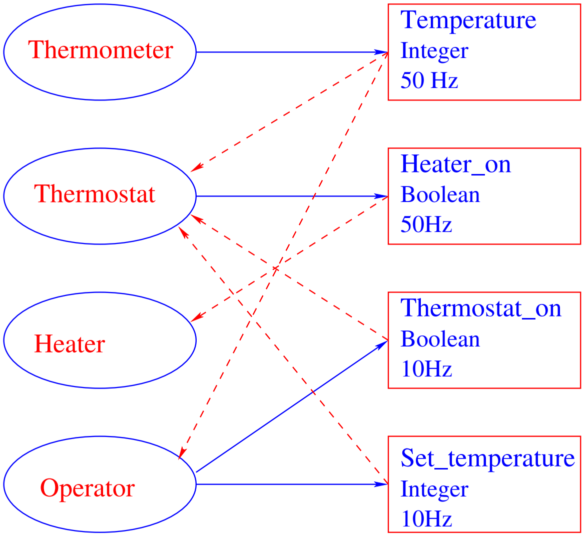
Architecture Description Language (ADL) for CPS

- The ‘D’ in ADL can stand for Definition, Design, or Description.
- There is very little clarity about
  1. What is a software architecture?
  2. What is the point of an architecture definition?
- We want the ADL to capture the flow of information in space/time:
  1. What are components?
  2. What is the model of interaction between components?
  3. How does architecture constrain the behavior to guarantee useful properties of the software subsystem and the cyber-physical system
  4. How is the virtual component-interaction model realized on a concrete platform?
  5. Can the architecture description help to structure the assurance argument?

Landshark on ROS



Thermostat on ROS



Robot ADL

- The Robot ADL bridges the gap between the engineering and computation models.
- The ADL for ROS-based systems consists of
  1. Message types
  2. Nodes, with their period, initialization, 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

Robot ADL: Nodes

```
thermostat: NODE {
 PACKAGES
   ...
 VARS
   readTemp: integer
   setTemp: integer
   on: boolean
 PUBLISH
   Heater_on: boolean
 SUBSCRIBE
   Temperature[2]: integer MAXLATENCY l1
   Thermostat_on: boolean MAXLATENCY l2
   Set_temperature: integer MAXLATENCY l3
 INIT
    ...
 NEXT
    ...
 RATE 50
```

Robot ADL: Topics

```
Temperature: TOPIC {
  TYPE integer
  RATE 50
  AUTHENTICATED YES
}
```

Robot ADL: Platform

```
blackbox: PROCESSOR {
  DEVICE
   thermometer1: thermometer
   heater1: heater
   radio1: radio
  HYPERVISOR
   CertiKOS
  PARTITIONS
   controllerPartition
   heaterPartition
   thermometerPartition
  BUSES
   CertiKOS_IPC
   USB

  }
```

Robot ADL: Processors and Devices

```
ocu: PROCESSOR {
  DEVICE
    radio2: radio
  PARTITIONS
    operatorPartition
}

tempSensor: INTERFACE {
    read_temperature: integer
 }
```

Robot ADL: Mappings

```
controllerPartition: PARTITION {
  OS Ubuntu 12.0.4
  PACKAGES
    ...
  NODES
   thermometer
   thermostat
   heater
  DEVICES
   temp1: tempSensor
   heat1: heatingUnit
}
```

Buses

```
CertiKOS_IPC: BUS {
  ENDPOINTS
    thermostatPartition
    thermometerPartition
    heaterPartition
}

Ethernet: BUS {
  ENDPOINTS
    Controller
    ResilientStateEstimator
    Firewall
}

Radio: BUS {
  ENDPOINTS
    ocu
    firewall
```

Using the Robot ADL

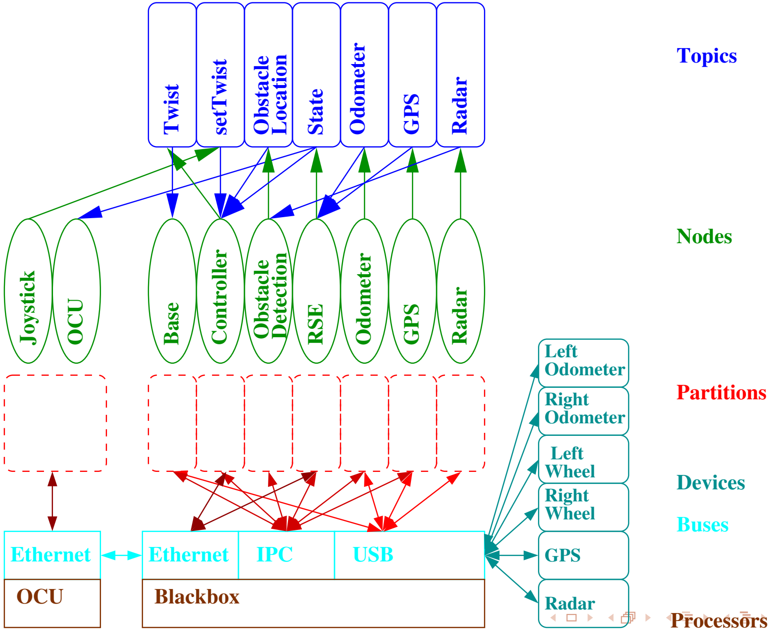
- The architecture description is used to
    1. Check the architectural integrity to ensure that the message types, rates, and device assignments match
    2. Relate the model of computation at the engineering layer to the ROS configuration and platform in the computation layer
    3. Generate glue code to handle initialization, and access to devices and communication
    4. Build and certify the running ROS configuration to conform to the architecture
- The model(s) of computation in the engineering layer is correctly realized
    1. The system is properly initialized
    2. The nodes are scheduled at their specified periodic rate
    3. The node computations are executed correctly within the hypervisor
    4. The messages are delivered with integrity and authenticity within the latency bounds
    5. The fault assumptions in the model are met by the architecture

Minimal Landshark Architecture

- Nodes:
    - Sensors: Odometer, GPS, Radar
    - Actuators: Left and Right wheel motors
    - Controllers: Resilient State Estimator (RSE), Obstacle Detector (OD), Vehicle Controller (VC), Base
    - Operator: Joystick, OCU

  Topics: Radar, Odometer, GPS, setTwist, Twist, State, Obstacle Location
- Devices: Left/Right odometers, IMU, GPS, Left/Right motors, Joystick
- Processors: OCU, Blackbox
- Buses: Radio, OCU Ethernet, Blackbox Ethernet, CertiKOS IPC, USB 3.0

Minimal Landshark Architecture



Architecture Consistency

- Node consistency
  1. Nodes can only read subscribed topics and write to published topics
  2. Types/rates of variables should match those of the corresponding topics
  3. The rates of topics (published or subscribed) correspond to those of the topic
  4. Device access is performed through the given API
- Topic consistency check is trivial since it is subsumed by node consistency.

Platform Consistency

- Each node is assigned to a partition where it is scheduled at the specified rate and given virtualized or exclusive access to its devices.
- Each partition is assigned to a processor with virtualization provided by a hypervisor to isolate multiple partitions.
- Each publisher/subscriber pair of nodes has a channel through the available buses.
- For each such channel, any received message must have been sent by a valid publisher no earlier than the latency bound (see next slide).
- Each device is assigned to a single ROS node or virtualized by the hypervisor.

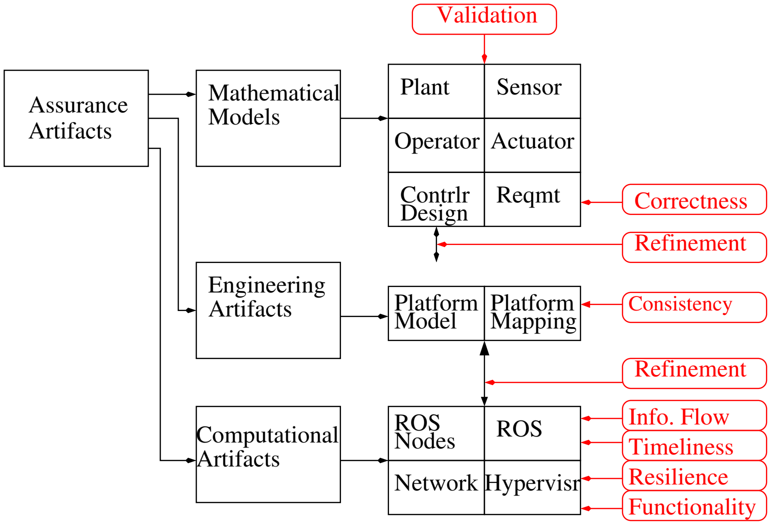
Channel Consistency

- All device access is through the USB with authenticity, integrity, and latency guarantees.
- All ROS communication between nodes on the Blackbox is through CertiKOS IPC with authenticity, integrity, and latency guarantees ensured by CertiKOS, the ROS Master, and the RVMaster.
    1. The ROS installation on the Blackbox ensures that messages are exchanged through CertiKOS IPC
    2. All the Landshark ROS nodes are constructed from the architecture and conform to their publish/subscribe authorizations
    3. Messages from any intruder nodes are monitored to check authorizations
    4. CertiKOS IPC ensures that the latency bounds are not violated
    5. ROS+CertiKOS ensures that messages are not corrupted
- All ROS communication between OCU and Blackbox nodes is through the firewall:
    1. Only messages with valid source and destination IP addresses are allowed through the firewall

Construction of Assurance Case

- Definition of model layers: PVS
- Model properties: HybridSAL, KeyMaera
- Architecture/Platform definition: Simulink, SAL, ROS
- Code correctness: Bedrock, Spiral, K, VST
- CertiKOS: Coq
- Authentication: Coq
- Network stack: Specware
- Semantic integration of claims: PVS
- Syntactic integration of assurance artifacts and claims: ETB

Landshark Assurance in ETB



Landshark Assurance in ETB: Mathematical Layer

The OCU generates **setTwist** messages that guide the vehicle through the specified waypoints based on the **State** information.
Reliable State Estimator (RSE) computes **State** estimate with bounded error (see Ashish's SAL verification of U Penn. claim) with one faulty sensor.
Obstacle detection (OD) procedure computes accurate **Obstacle Location** information relative to the vehicle. [CMU using KeyMaera and SPIRAL]
The Landshark Controller (LC) node ensures that the vehicle navigates to the waypoints but stops to *avoid* any detected obstacles — the vehicle must maintain a minimum separation from any obstacle.
The Landshark Base (LB) node navigates the vehicle based on the given **Twist** input from LC.

Landshark Assurance in ETB: Engineering Layer

The ROS nodes and topic channels are specified in the architecture description.
The sensor and actuator devices satisfy the models used in the mathematical layer.
The sensor nodes record the sensor readings accurately, and the actuator nodes operate the actuators as commanded.
The ROS node operate at rates that satisfy the assumptions used in the mathematical layer.
The channel latencies satisfy the assumptions used in the mathematical layer.
The ROS controller nodes implement the algorithms used in the mathematical layer.

Landshark Assurance in ETB: Computation Layer

The ROS build is generated from the architecture description.
The Platform Consistency claim is valid for the platform consisting of the OCU and Blackbox+CertiKOS.
The start-up procedure (CertiKOS + ROS + Master) initializes the ROS nodes (and associated devices) and the topic channels between these nodes.
The Channel Consistency claim holds for all of the ROS message channels and is ensured by CertiKOS + ROS Authentication + Firewall + RVMaster.
Together, these claims ensure that the platform faithfully simulates the rate synchronous model of computation on the configuration specified by the architecture.

Conclusions

- Cyber-physical systems range from engine controllers, cars, and robots to factories, buildings, and power grids.
- The incorporation of software and networking makes the safety and security of these systems a critical challenge.
- The construction of the assurance case should drive the design of cyber-physical systems.
- The assurance-driven design (ADD) starts with an eight-variables model of the system developing three layers of design and assurance: the mathematical, engineering, and computation layers.
- We have outlined a minimalist Landshark configuration, an Architecture Definition Language, and an assurance case structure.
- We have show how the assurance argument and artifacts can be assembled using the Evidential Tool Bus (ETB).
- The Landshark assurance argument will be adapted to the American Build Automobile.