Platform-Based Embedded Software Design for Multi-Vehicle Multi-Modal Systems

T. John Koo, Ph.D.

Dept. of Electrical Engineering and Computer Sciences
University of California at Berkeley
Berkeley, CA94720

Embedded Software Conference
Grenoble, France
October 7-9, 2002

koo@eecs.berkeley.edu

Research Collaborators

- Judith Liebman
- Cedric Ma
- Benjamin Horowitz,
- Alberto Sangiovanni-Vincentelli
- Shankar Sastry

Berkeley
University of California

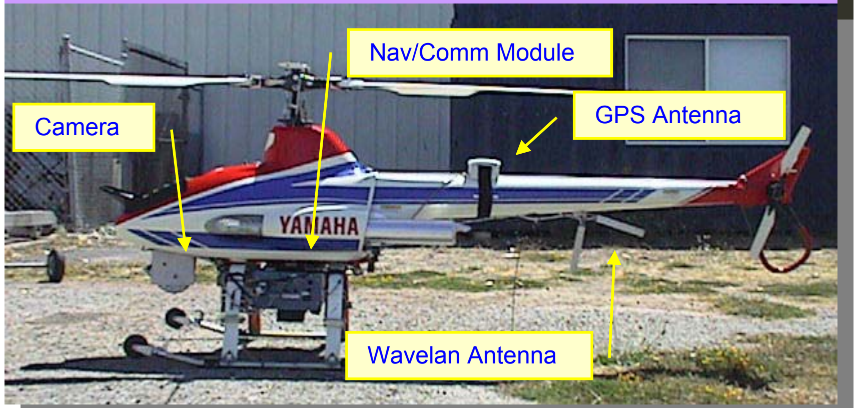
Outline

- Motivation
- Platform-Based Design
- Autonomous Vehicle Design
- Hardware-In-The-Loop Simulation
- Conclusion

Autonomous Flight of R-50

Autonomous Landing of R-50

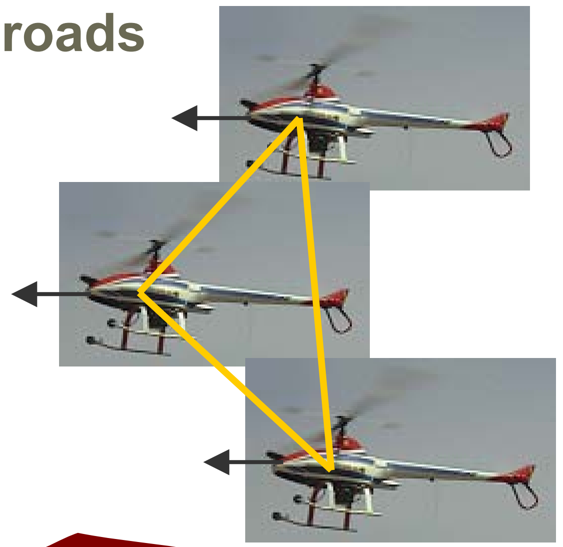
Berkeley Aerial Robot: R-50



Motivation

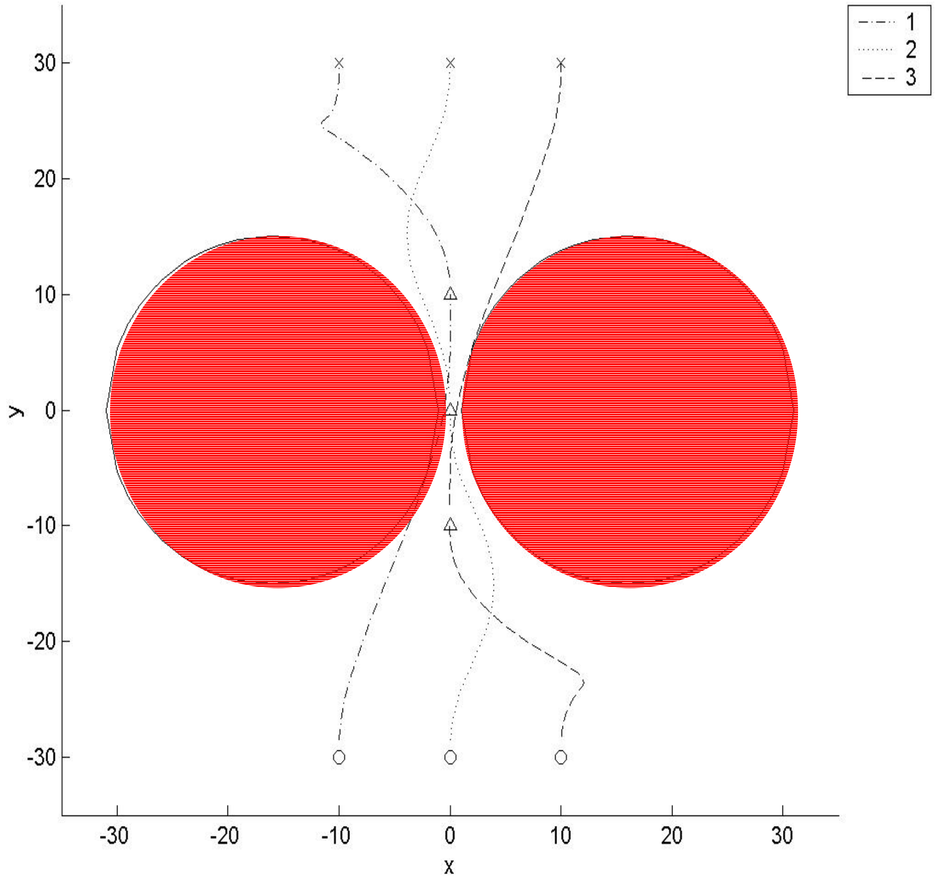
- Multiple Autonomous Vehicle Applications
  - Unmanned aerial vehicles perform mission collectively
  - Satellites for distributed sensing
  - Autonomous underwater vehicles performing exploration
  - Autonomous cars forming platoons on roads

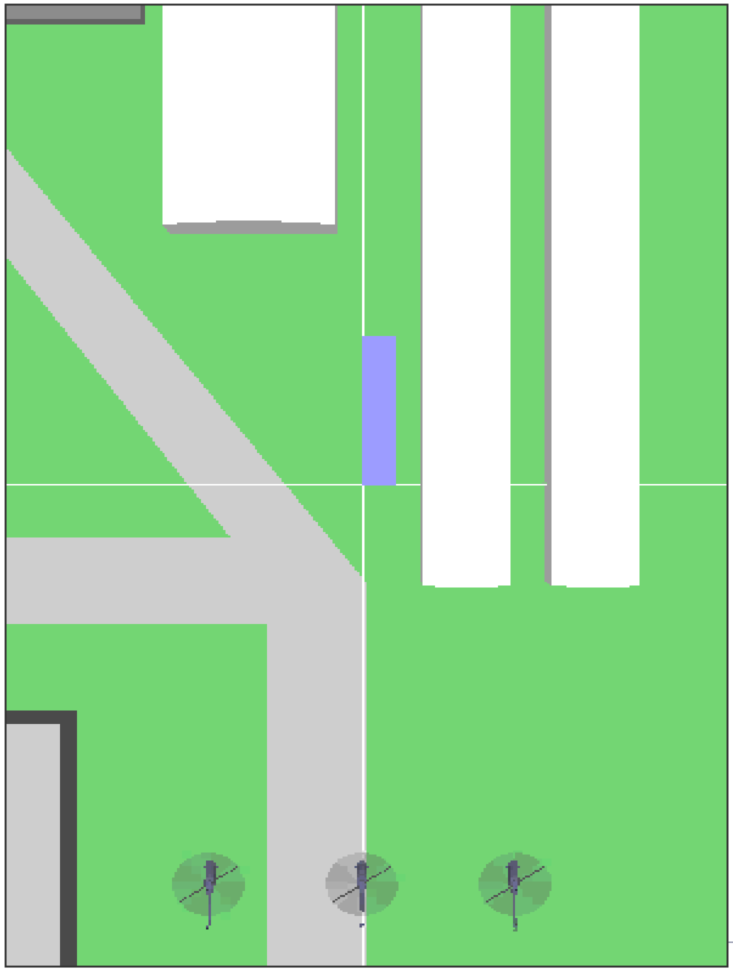
- Enabling Technologies
  - Hierarchical control of multi-agents
  - Distributed Sensing and Actuation
  - Computation
  - Communication
  - Embedded Software



Motivation

- Formation of Autonomous Vehicles

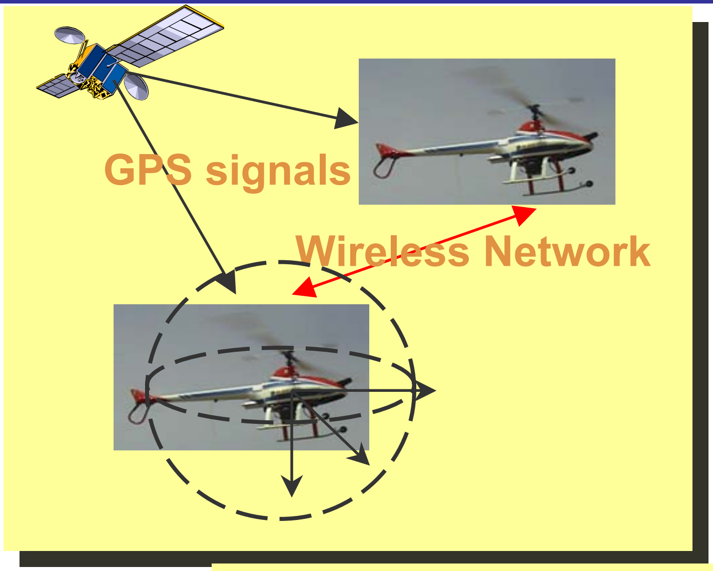


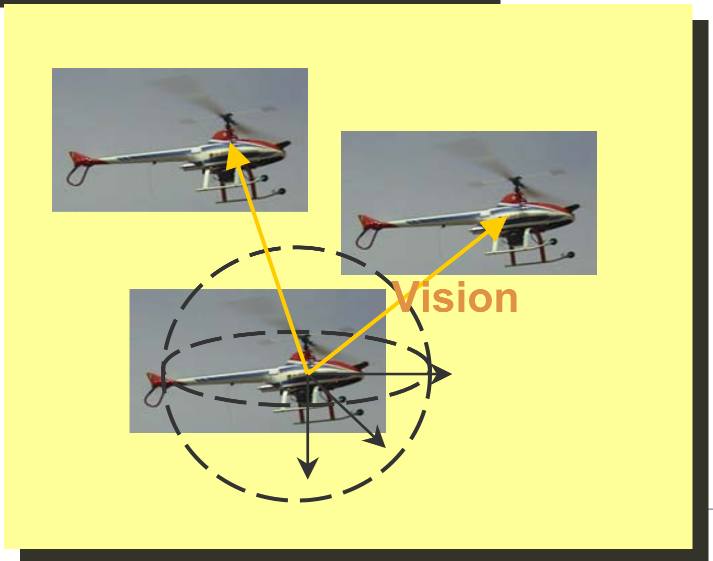


Shannon Zelinski, T. John Koo, Shankar Sastry. “Optimization-based Formation Reconfiguration Planning,” submitted to Int. Conf. on Robotics and Automation, 2003.

Motivation

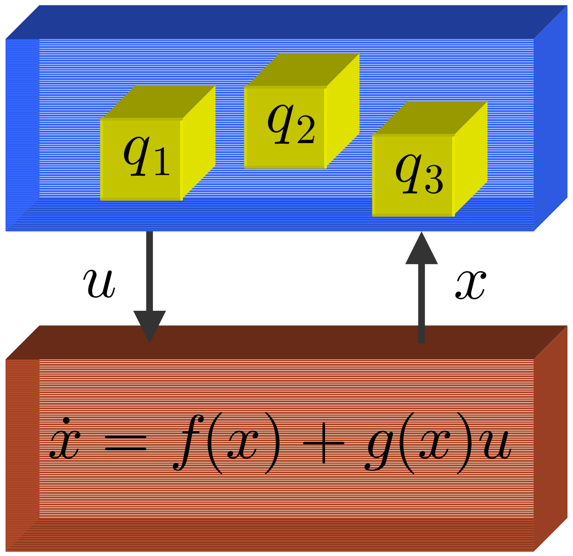
- Loose Formation Flight
  - GPS provides global positioning information to vehicles
  - Wireless network is used to distribute information between vehicles
  - Navigation computer on each vehicle calculates relative orientation, distance and velocities
- Tight Formation Flight
  - Vision system equipped with omni-directional camera can track neighboring vehicles
  - Structure from motion algorithms running on vision system provides estimates of relative orientation, distance and velocities to navigation computer





Motivation

- **Multi-Modal Systems**
  - Given a continuous control system, a collection of *control modes* are designed.
  - Each high-level task is specified as a sequence of control modes.



For control mode q_i,
Given

$$\dot{x} = f(x) + g(x)u$$
$$y_i = h_i(x), \quad X_i$$
$$u = k_i(x, r_i)$$

Assume that

$$r_i \in \mathcal{R}_i$$
$$x(t_0) \in S_i(r_i) \subseteq X_i$$

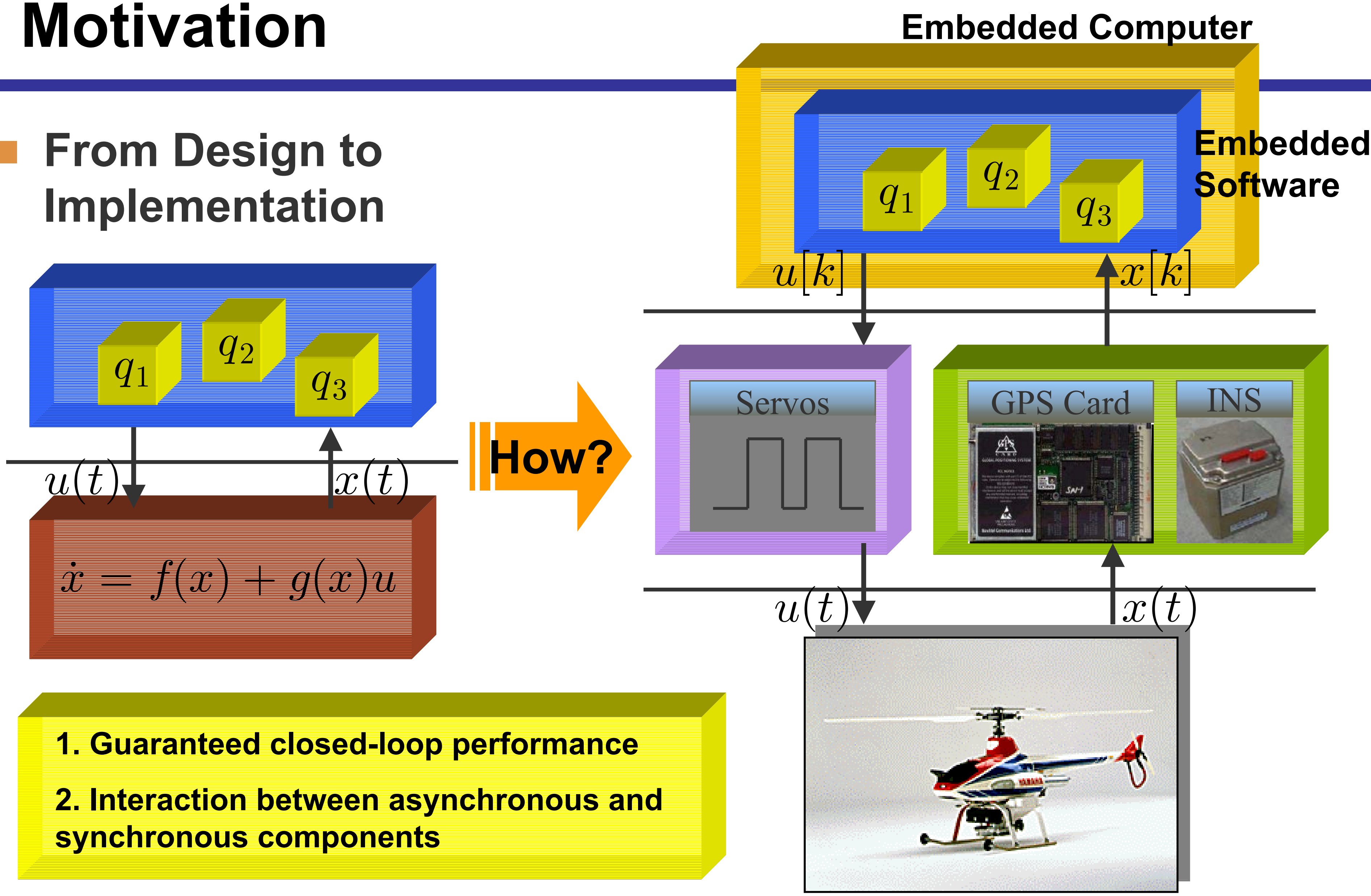
Guarantee that

$$y_i \longrightarrow r_i$$
$$x(t) \in X_i, \quad t \geq t_0$$

T. J. Koo, G. J. Pappas, and S. Sastry, “Mode Switching Synthesis for Reachability Specifications,” Hybrid Systems: Computation and Control, Lecture Notes in Computer Science, Springer, 2001.

Motivation

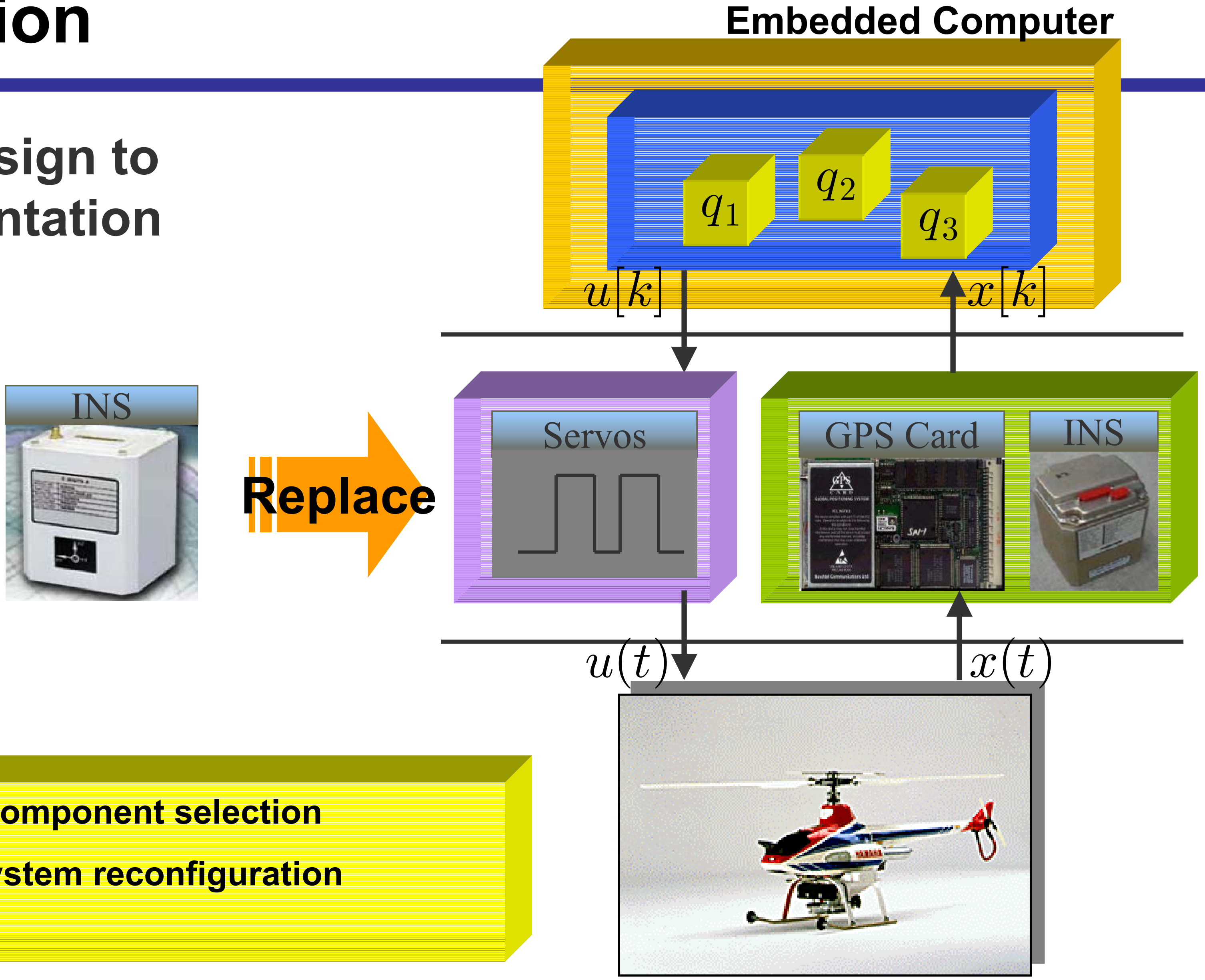
- From Design to Implementation



1. Guaranteed closed-loop performance
2. Interaction between asynchronous and synchronous components

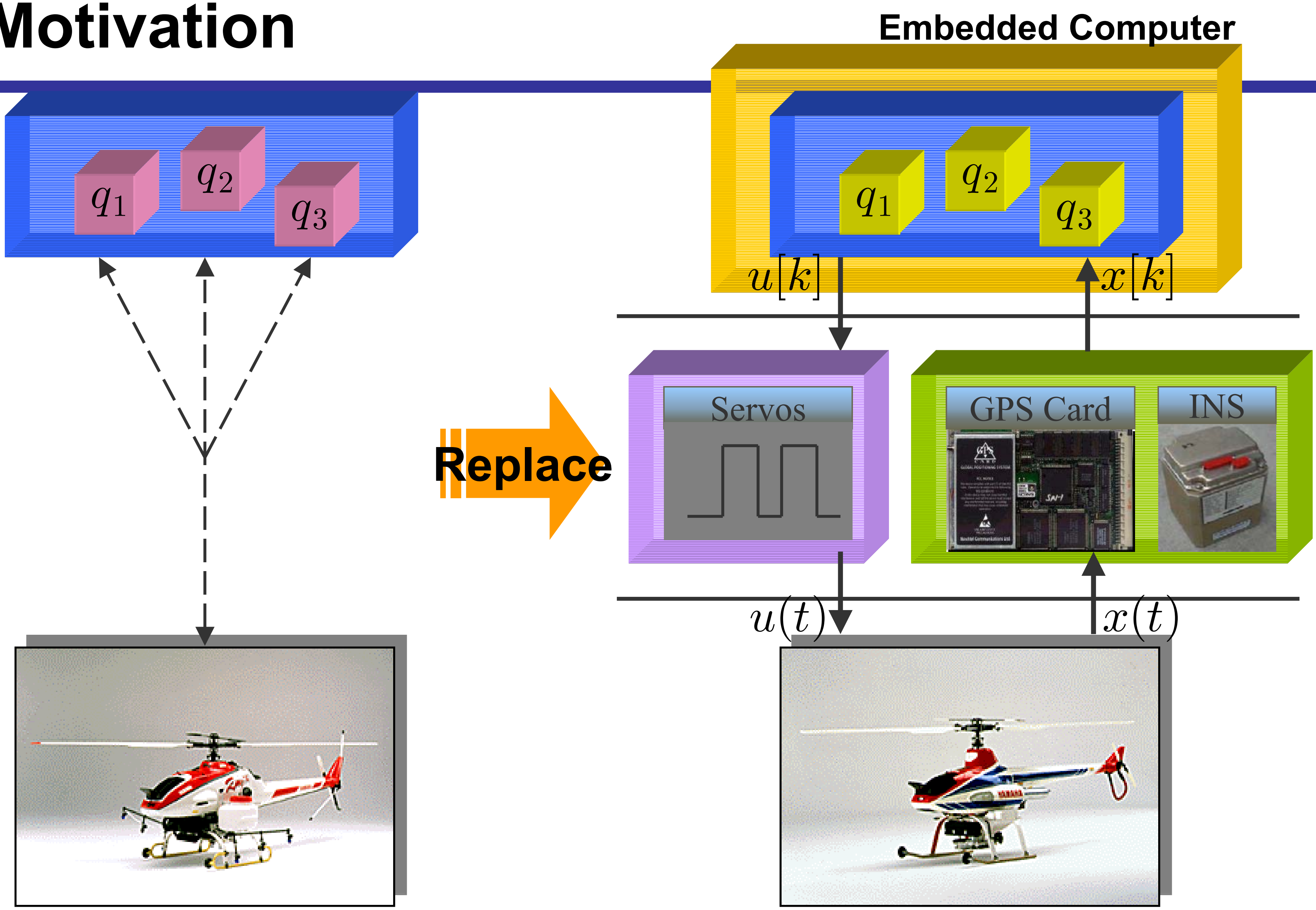
Motivation

- From Design to Implementation



1. Versatile component selection
2. Flexible system reconfiguration

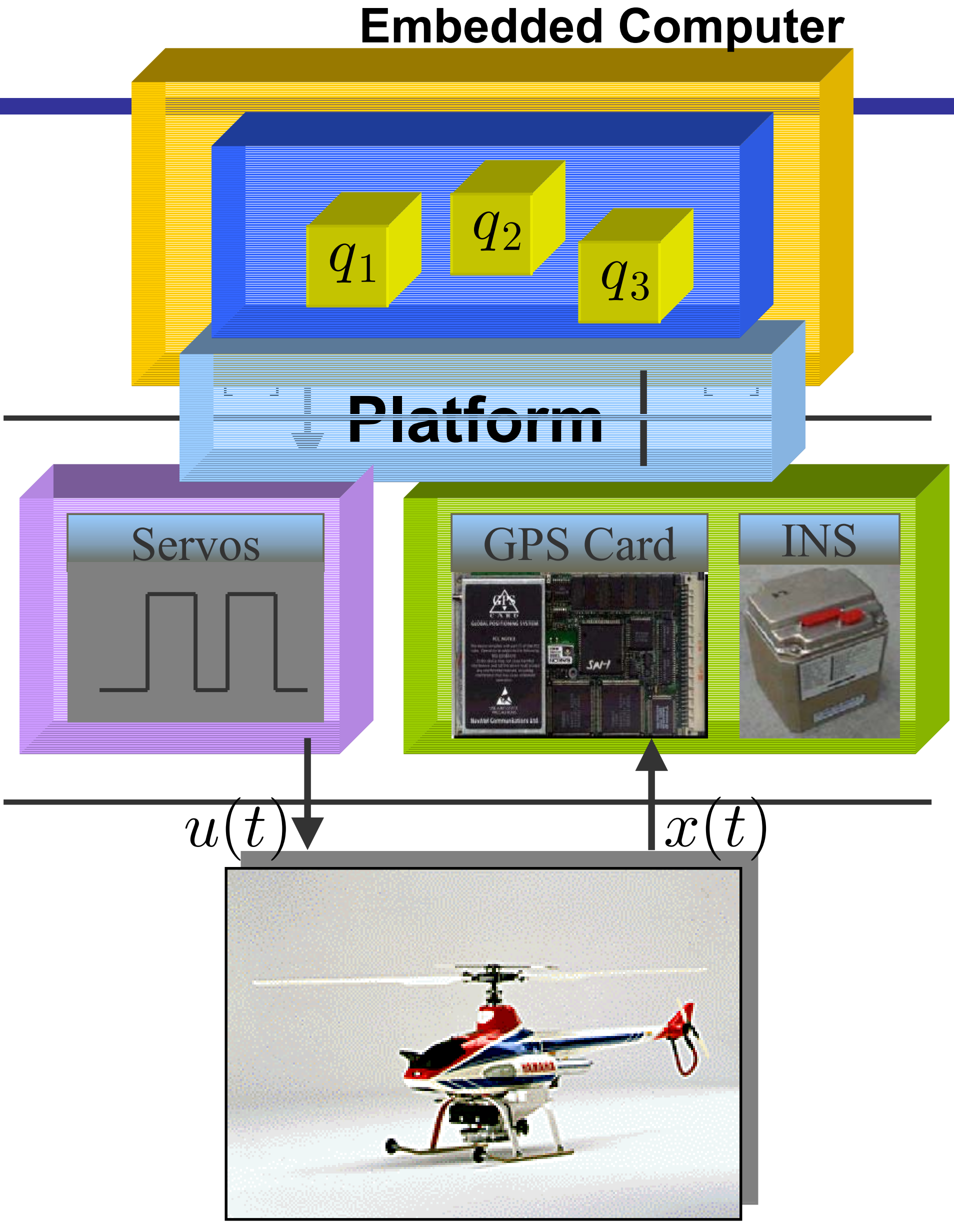
Motivation



Embedded Computer
q_1
q_2
q_3
q_1
q_2
q_3
$u[k]$
$x[k]$
Replace
Servos
GPS Card
INS
$u(t)$
$x(t)$

Motivation

- Time-based design
  - Predictable closed-loop performance
  - Boundary between asynchronous and synchronous components
- Modular design
  - Versatile components selection
  - Flexible system reconfiguration
- ***Platform-Based Design***



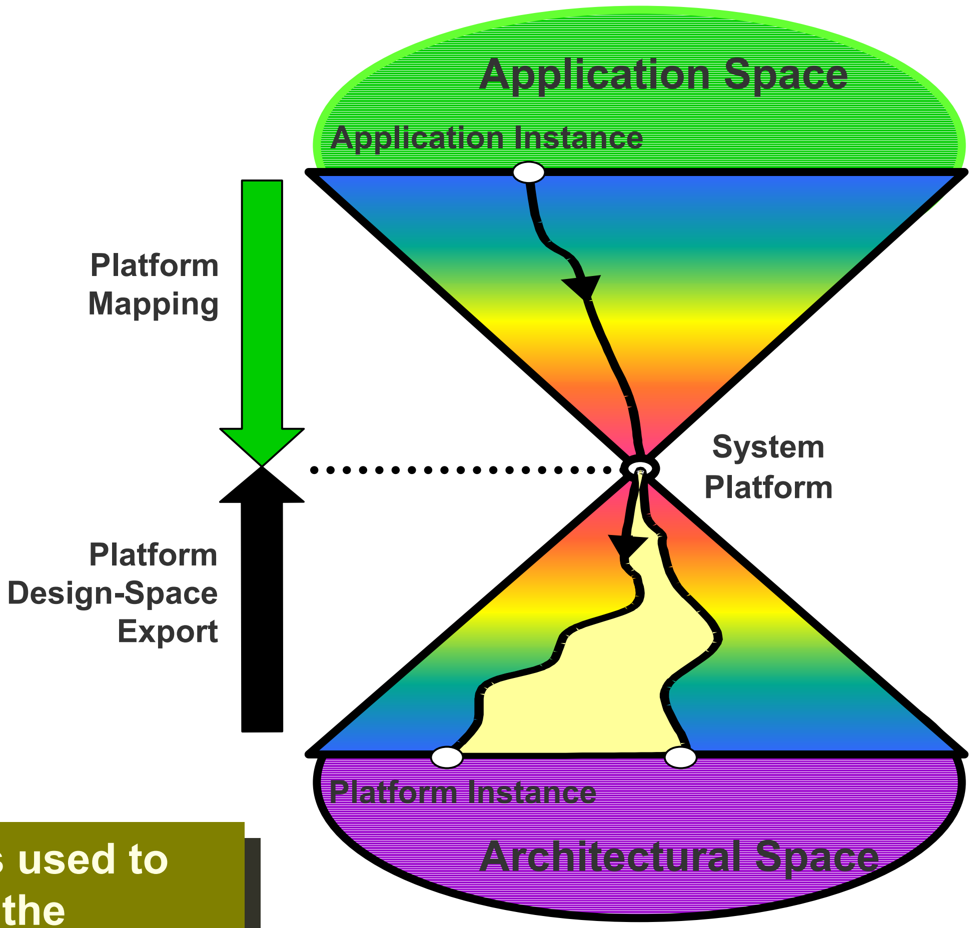
Platform-Based Design

- In general, a platform is an abstraction layer that covers *a number of possible refinements* into a lower level.

Platform stack
Platform
Mapping Tools
Platform

Platform-Based Design

- The design process is meet-in-the-middle:
  - Top-down: map an instance of the top platform into an instance of the lower platform and propagate constraints
  - Bottom-up: build a platform by defining the "library" that characterizes it and a performance abstraction



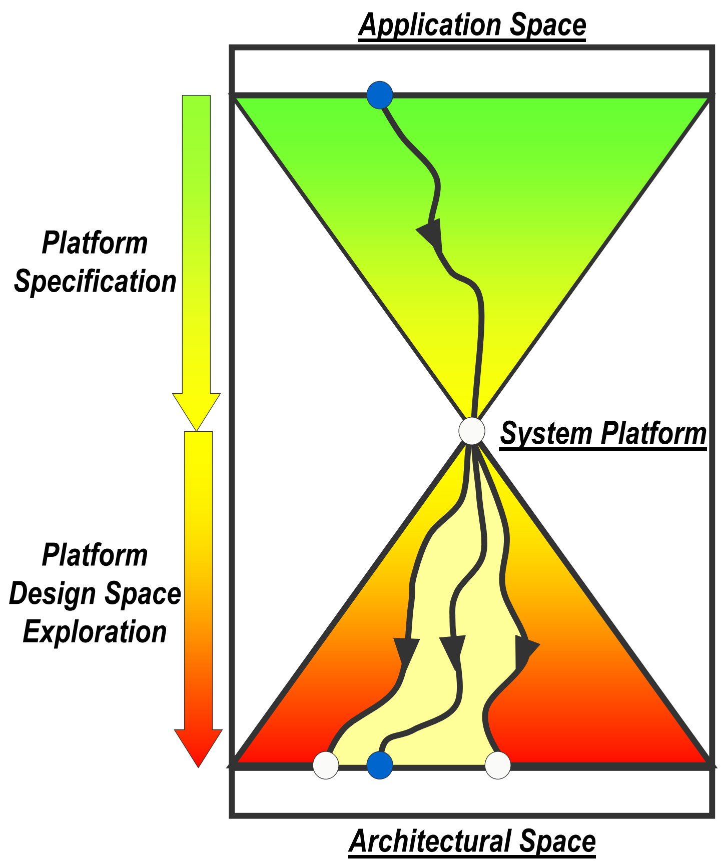
For every platform, there is a view that is used to map the upper layers of abstraction into the platform and a view that is used to define the class of lower level abstractions implied by the platform.

Berkeley University of California

Platform-Based Design

- Proposed by Alberto Sangiovanni-Vincentelli and adopted by Cadence for SOC design

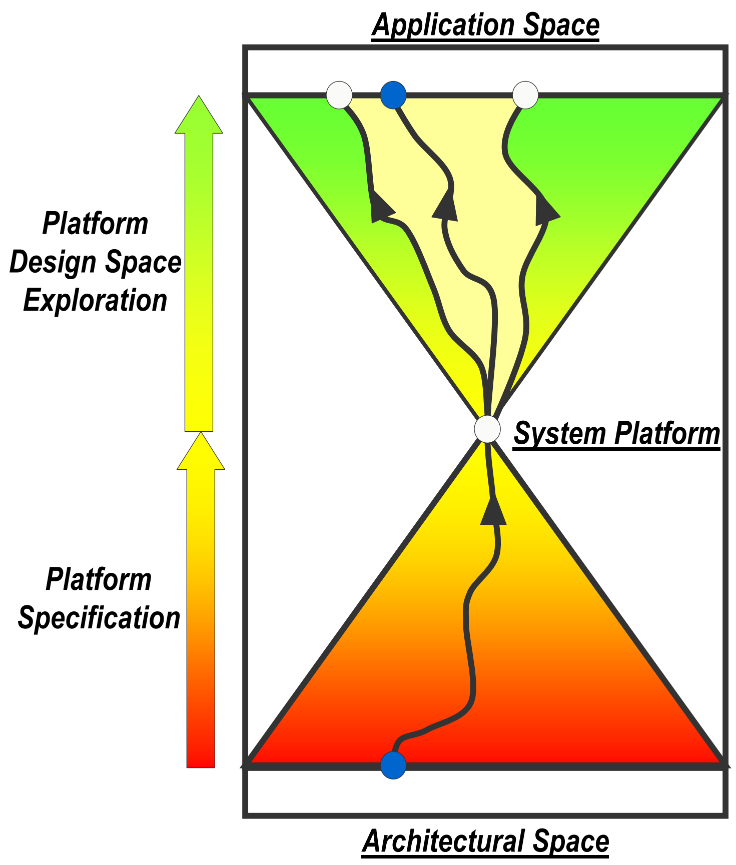
- Define the application instance to be implemented to satisfy system design requirements defined by application
- Specify the system platform according to possible instances of implementations
- Evaluate top down different instances of system platforms



Berkeley
University of California

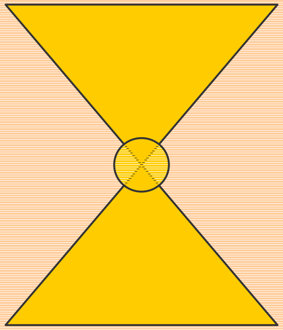
Platform-Based Design

- Define the system platform instance so that multiple instances of applications can be mapped to the same system platform
- Present this to system designers as system Design-Kit and optimally leverage economy of scale for system platform instance
- Provide bottom up instances of system platform for evaluation without disclosing the details of the implementation details



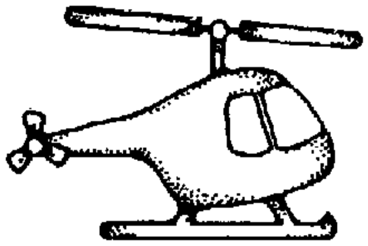
Platform-Based Design of Autonomous Vehicles

I



Platform-Based Design

II



UAV System

III

Time-Based Embedded Control

Platform-Based Autonomous Vehicle Design

II. UAV System

R-50 Hovering

GPS Card

GPS Antenna

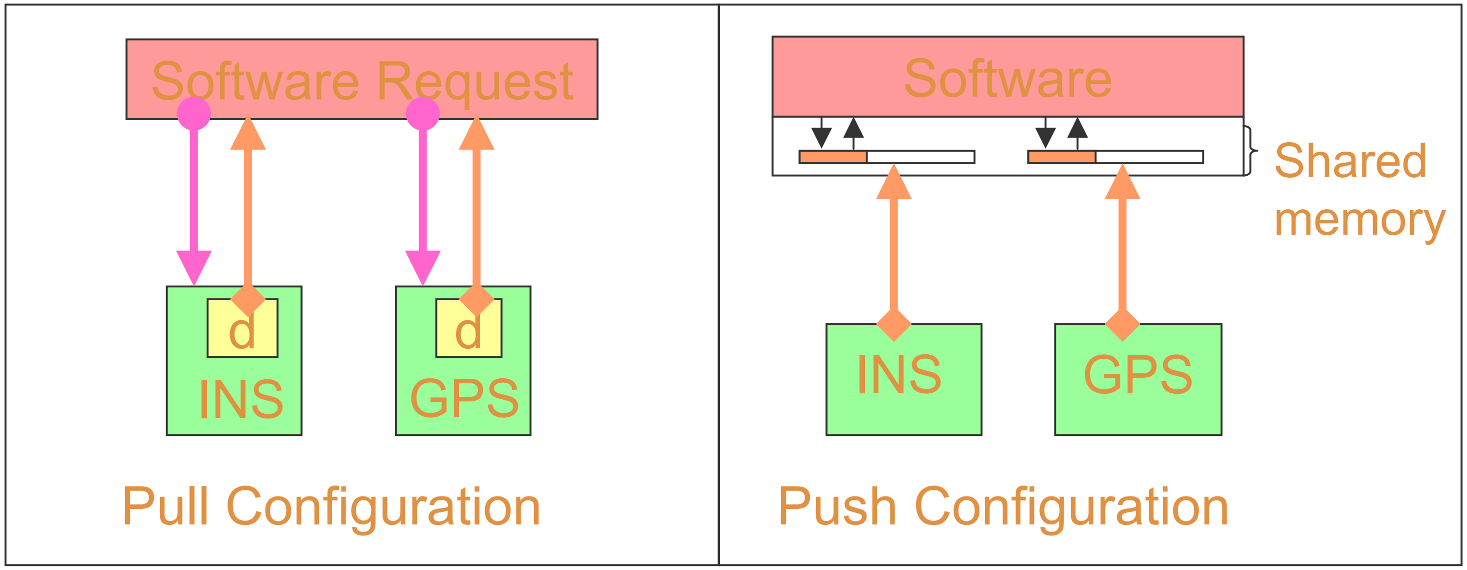
- Inertial navigation system (INS).
  - Accelerometers, rotational rate sensors.
  - Frequent (100 Hz) measure of position, velocity, orientation, and rate of rotation.
  - Low position accuracy. Error can grow unbounded over time.
- Global positioning system (GPS).
  - 4 Hz measure of position – too slow for stabilization.
    2 cm accuracy.
- Sensor Fusion - Kalman filter.
  - Prediction (100 Hz) – use INS to estimate location.
  - Correction (4 Hz) – use GPS to correct estimate.

INS

Berkeley
University of California

II. UAV System

- Sensors may differ in:
  - Data formats, initialization schemes (usually requiring some bit level coding), rates, accuracies, data communication schemes, and even data types
- Differing Communication schemes requires the most custom written code per sensor
- Sensors asynchronous w.r.t. control computer.



III. Time-Based Embedded Control

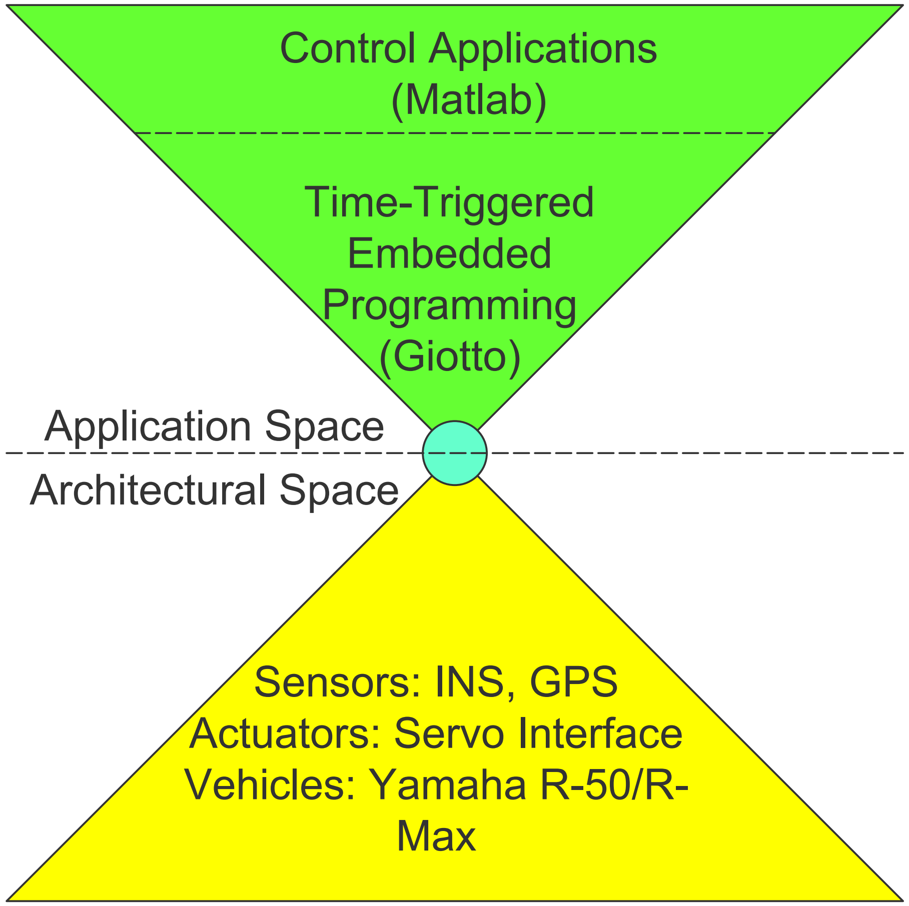
- Advantages of time-triggered framework:
  - Allows for composability and validation
    - These are important properties for safety critical systems
    - Timing guarantees ensure no jitter
- Disadvantages:
  - Bounded delay is introduced
  - Implementation and system integration become more difficult
- Platform design allows for time-triggered framework for the time-based embedded controller
  - Use Giotto as a software platform to ease implementation:
    - provides real-time guarantees for control blocks
    - handles all processing resources
    - Handles all I/O procedures

T. John Koo, Judith Liebman, Cedric Ma, and Shankar Sastry. "**Hierarchical Approach for Design of Multi-Vehicle Multi-Modal Embedded Software,**" Embedded Software Conference, 2001.

Berkeley University of California

Platform-Based Design for Autonomous Vehicles

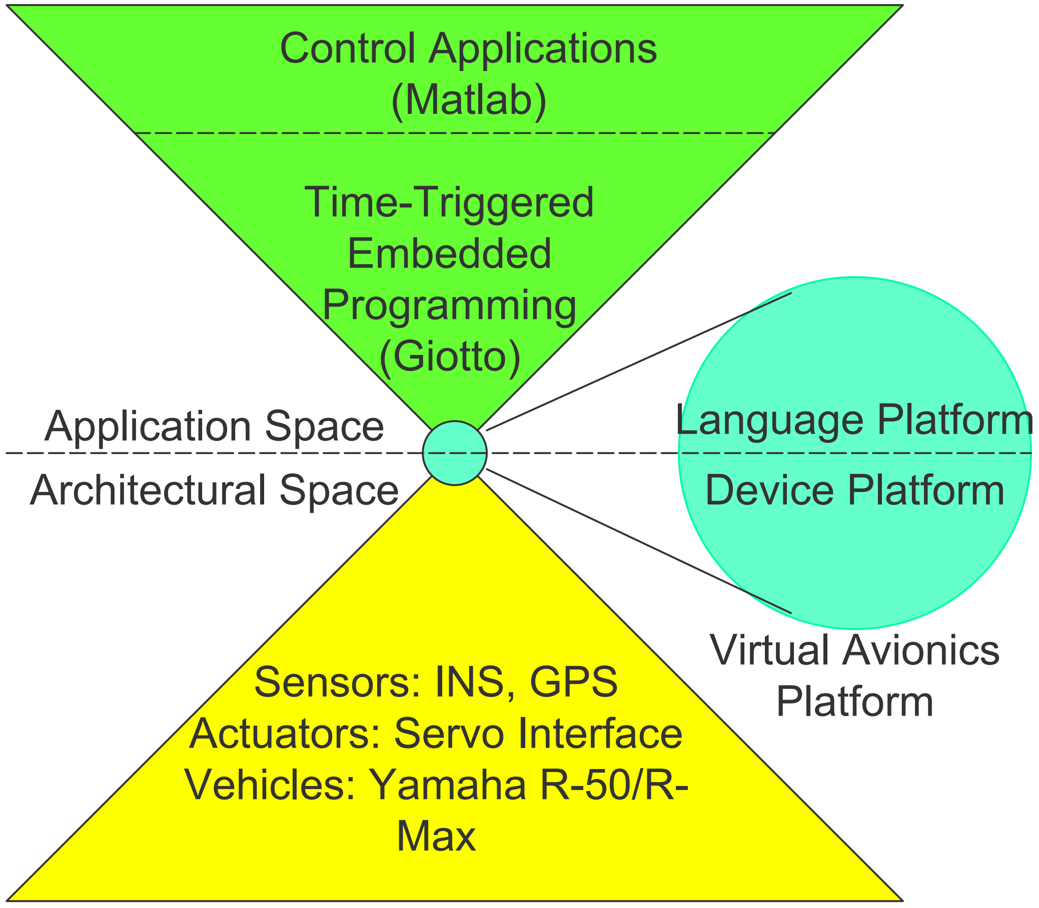
- Objective
  - Abstract details of sensors, actuators, and vehicle hardware from control applications
- How?
  - Time-triggered Embedded Programming Language (i.e. Giotto)
  - Platform


Control Applications (Matlab)
Time-Triggered Embedded Programming (Giotto)
Application Space
Architectural Space
Sensors: INS, GPS
Actuators: Servo Interface
Vehicles: Yamaha R-50/R-Max

Berkeley University of California

Platform-Based Design for Autonomous Vehicles

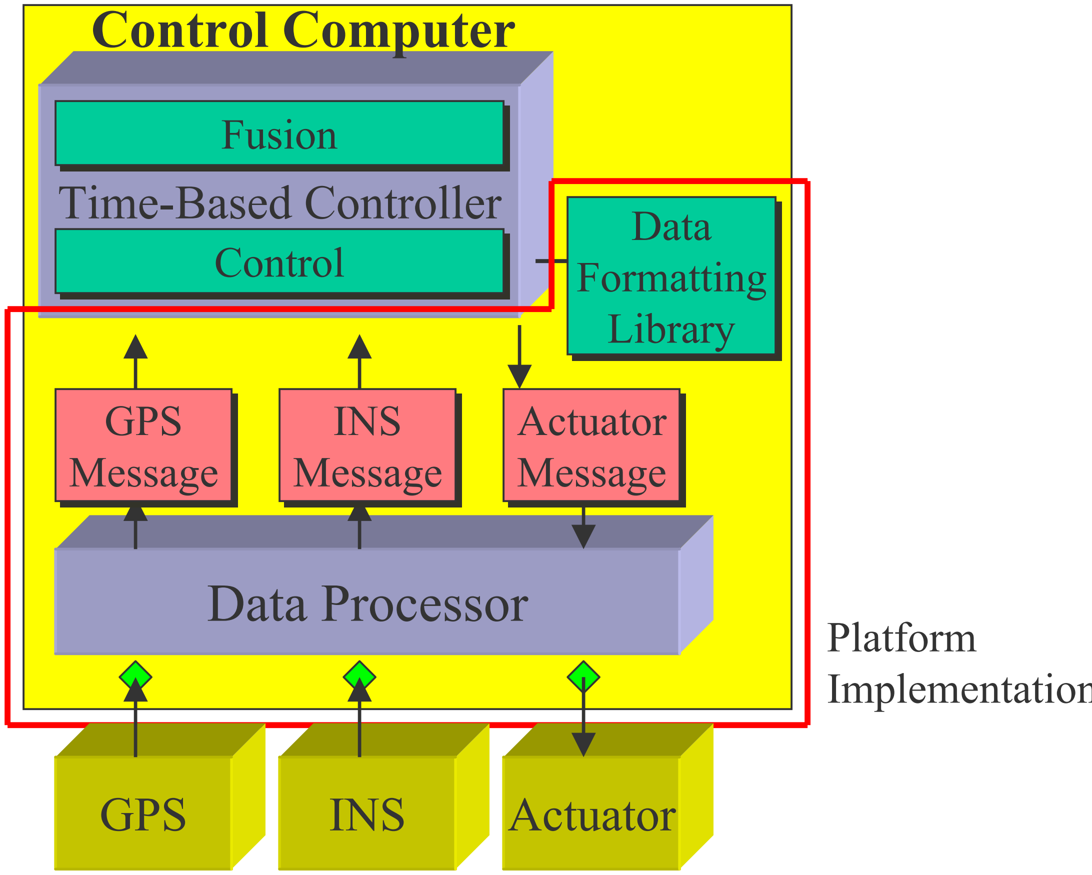
- **Language Platform**
  - **Provides** an environment in which time-based control programs can be scheduled and run
  - **Assumes** the use of generic data formats for sensors/actuators made possible by the Device Platform
- **Device Platform**
  - **Isolates** details of sensor/actuators from embedded control programs
  - **Communicates** with each sensor/actuator according to its own data format, context, and timing requirements
  - **Presents** an API to embedded control programs for accessing sensors/actuators


Control Applications (Matlab)
Time-Triggered Embedded Programming (Giotto)
Application Space
Architectural Space
Language Platform
Device Platform
Virtual Avionics Platform
Sensors: INS, GPS
Actuators: Servo Interface
Vehicles: Yamaha R-50/R-Max

Berkeley
University of California

Platform Implementation

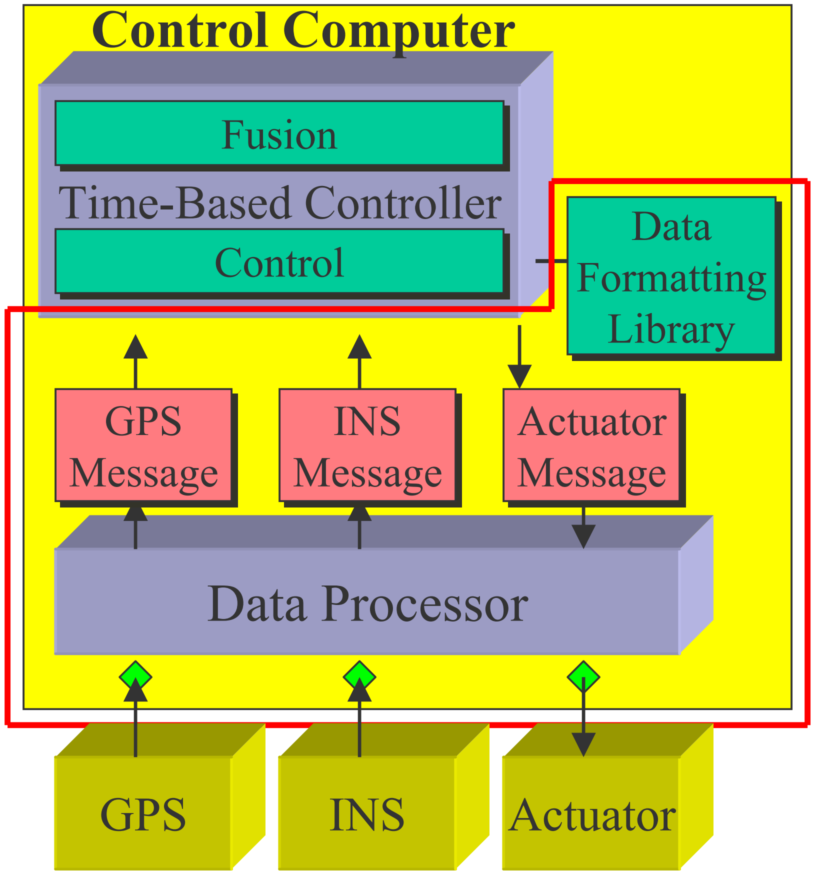
- Data Processor
  - Isolates the timing of sensors and actuators
  - Moves sensor/actuator data to shared memory
    - No format conversion
    - Uses 'negligible' computation time: saves processor time for Giotto tasks
- Shared Memory
  - Serves as bridge between synchronous and asynchronous parts of system
  - Circular buffer: allows simultaneous read/write



Berkeley
University of California

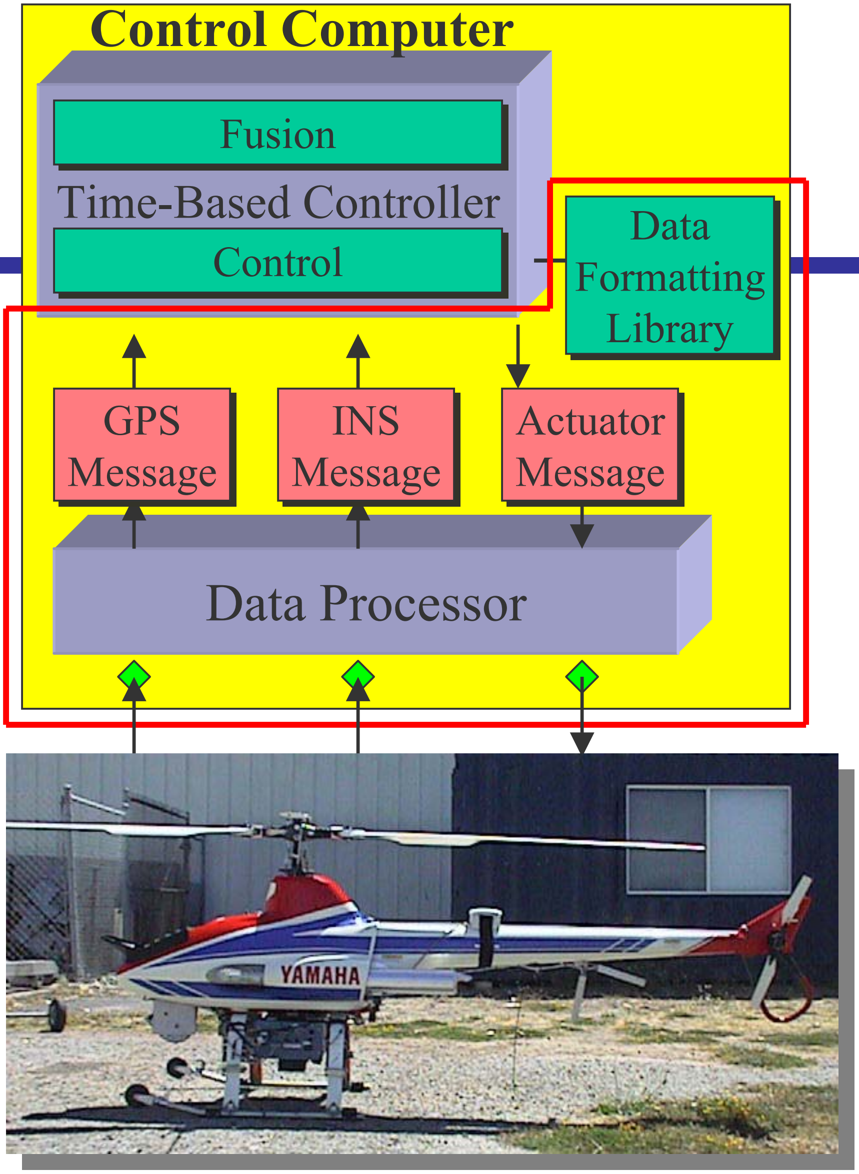
Platform Implementation

- Time-based controller
  - Where control algorithms (Control) and Kalman filter (Fusion) reside as Giotto *tasks*
- Data formatting library
  - Allows control programs to interpret sensor data and send data to actuator as generic, device independent format
  - Implemented as C routines



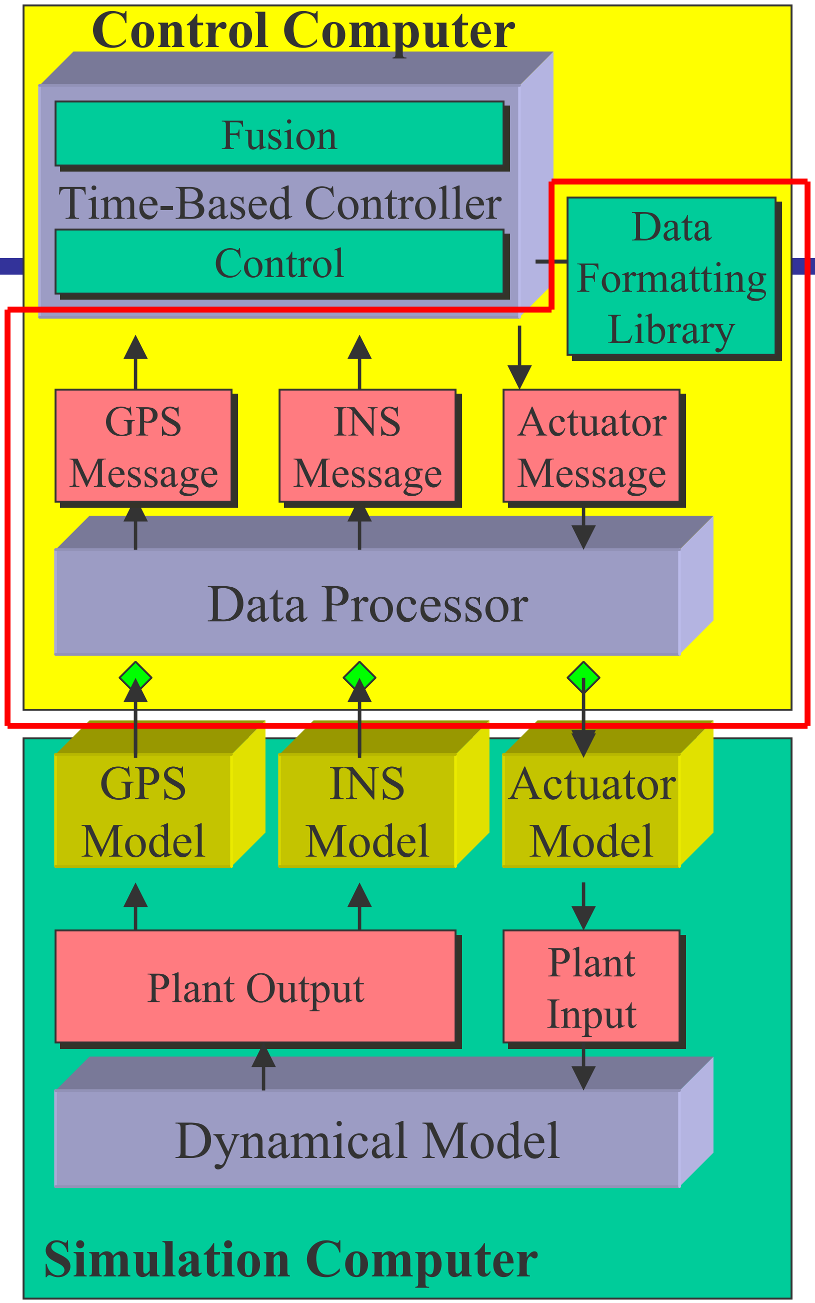
Hardware-in-the-Loop Framework

- Hardware-in-the-loop enables:
  - Safe & inexpensive testing
  - Rapid design iterations
  - Partial simulations of newly developed technologies
  - Repeatable tests
  - Testing of non-deterministic components

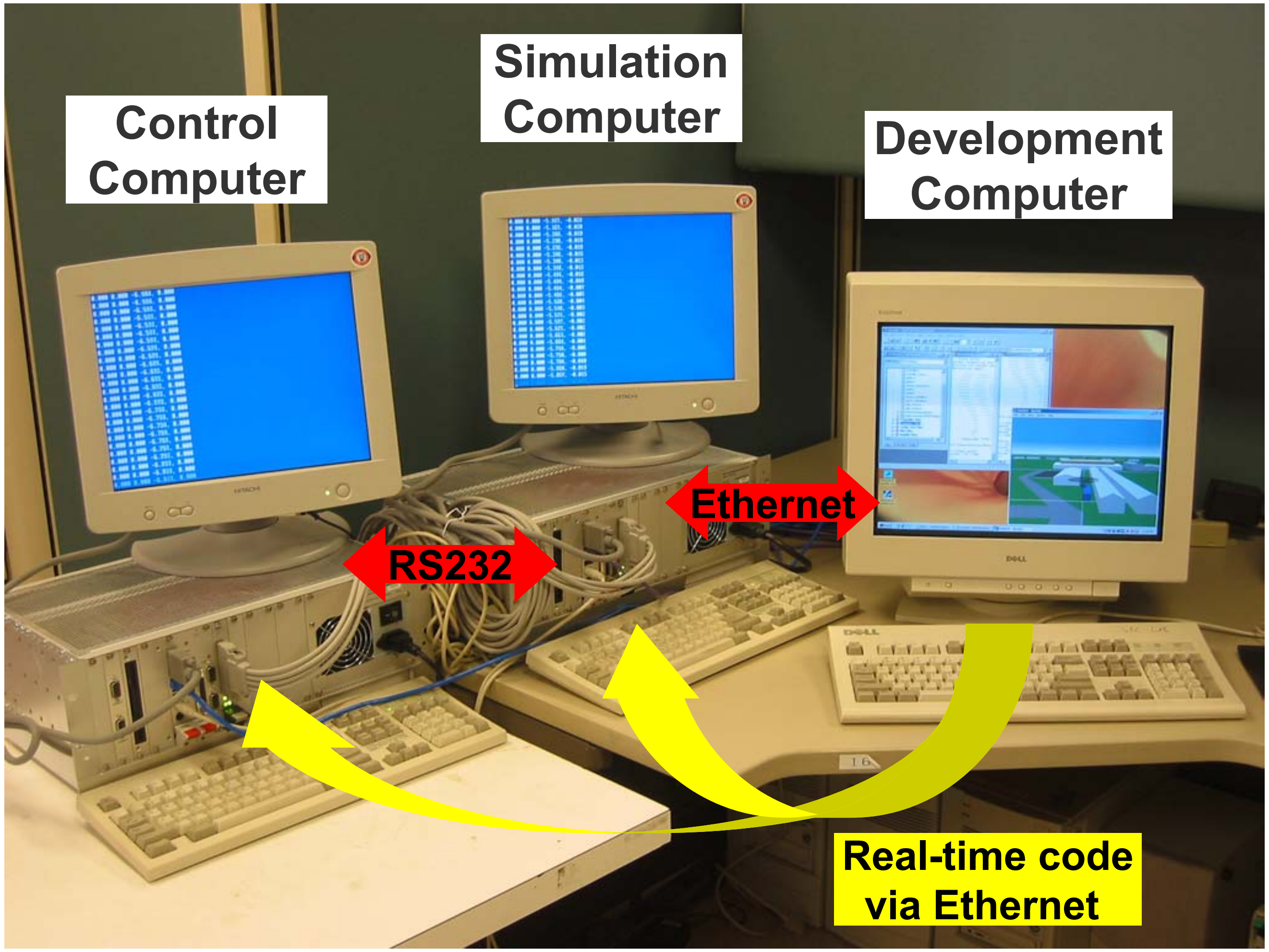


Hardware-in-the-Loop Framework

- Hardware-in-the-loop enables:
  - Safe & inexpensive testing
  - Rapid design iterations
  - Partial simulations of newly developed technologies
  - Repeatable tests
  - Testing of non-deterministic components

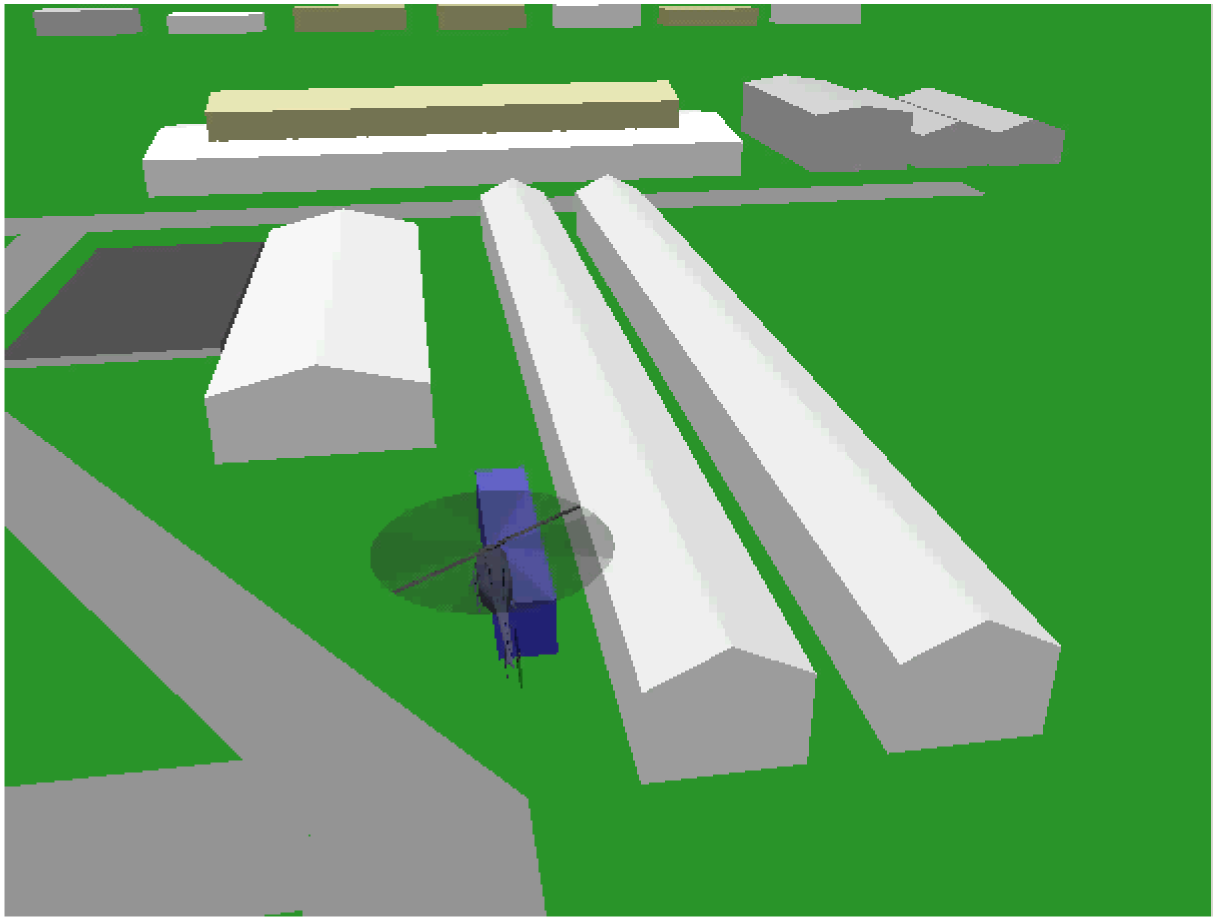


Testing High Performance Maneuvers using Hardware-in-the-Loop Simulation



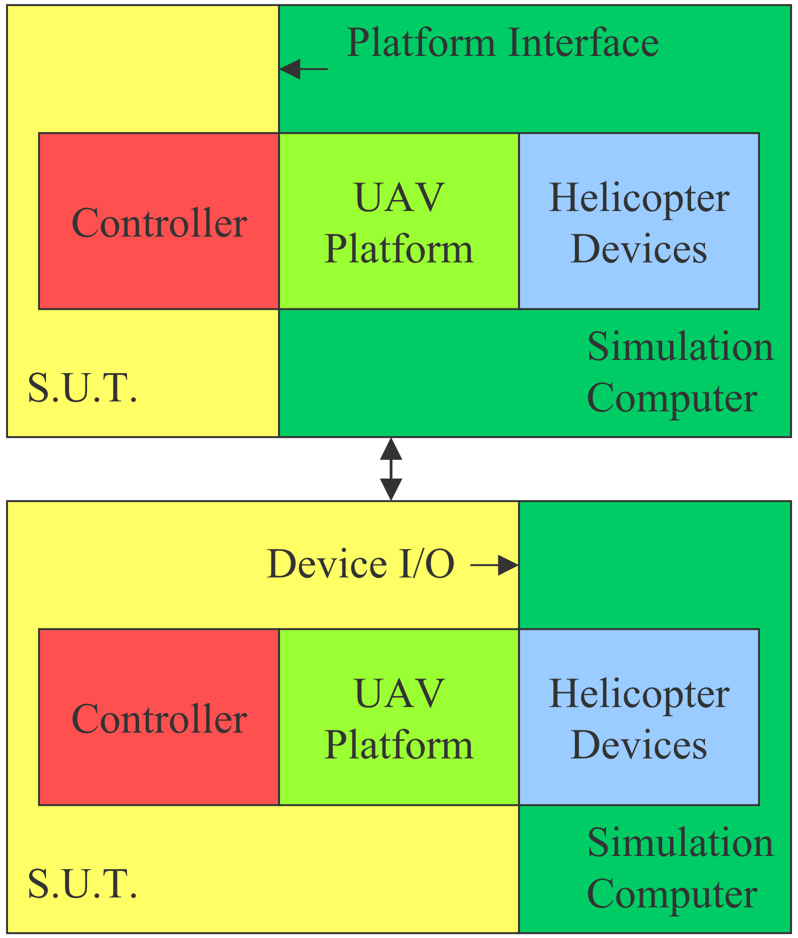
Testing High Performance Maneuvers using Hardware-in-the-Loop Simulation

Nose-in circle maneuver

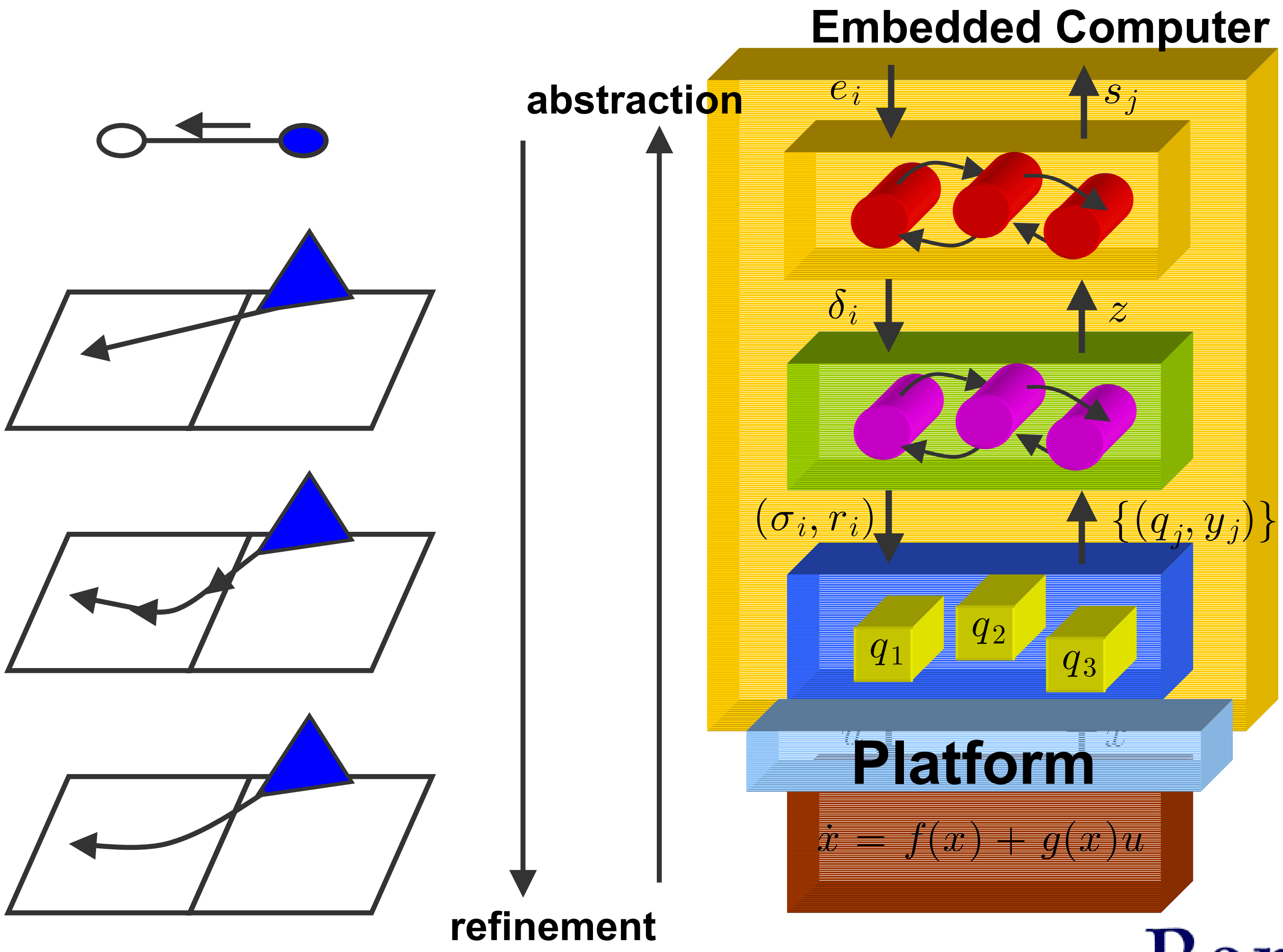


Flexible Testing

- Testing with different focus
  - Adapts to testing either controller or platform

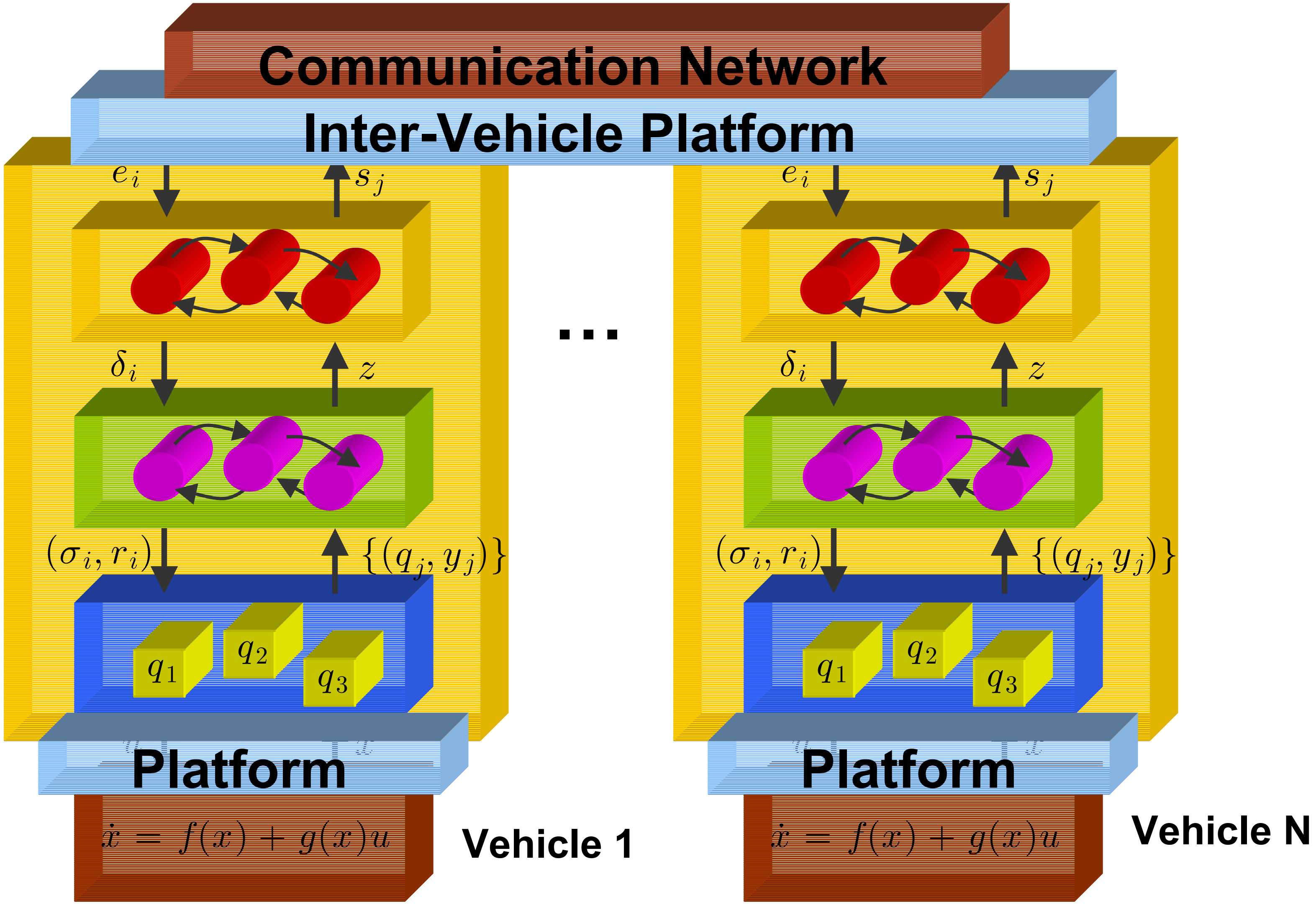


Platform-Based Design: Multiple Levels of Abstraction



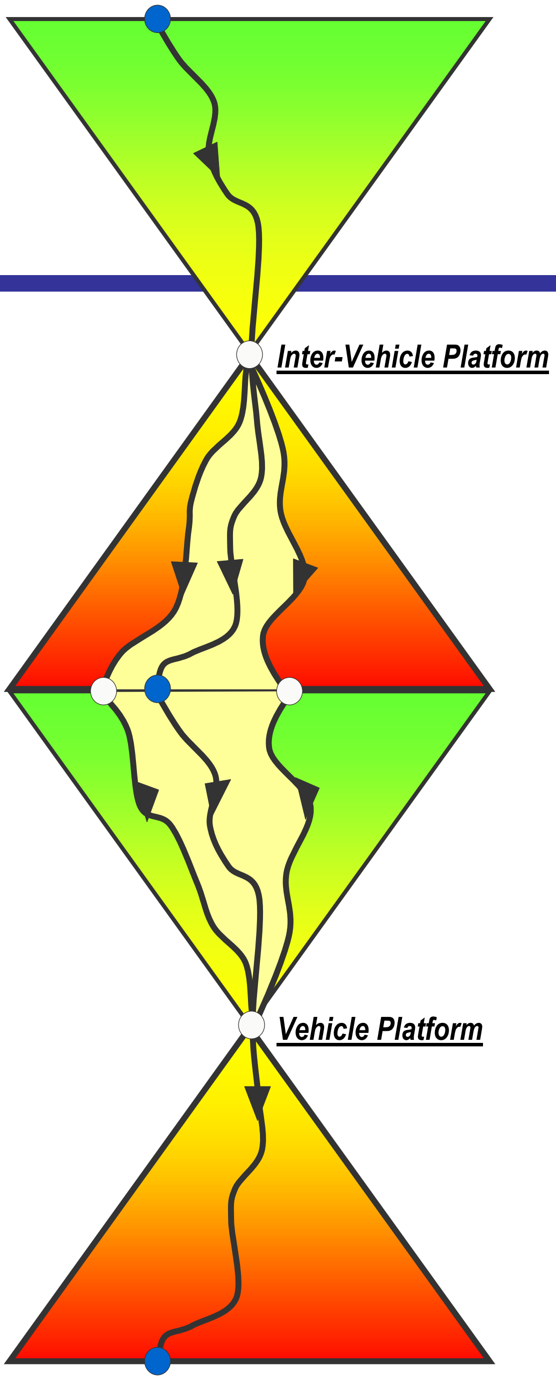
T. J. Koo and S. Sastry, "Bisimulation based Hierarchical System Architecture for Single-Agent Multi-Modal Systems," Hybrid Systems: Computation and Control, 2002.

Platform-Based Design: Platforms of Platforms



Platform-Based Design: Platforms of Platforms

- Platforms eliminate *large loop iterations* for affordable design
- Restrict design space via new forms of regularity and structure that surrender *some* design potential for lower cost and first-pass success
- The number and location of intermediate platforms is the essence of platform-based design
- Critical step is defining intermediate platforms to support:
  - Predictability: abstraction to facilitate higher-level design
  - Verifiability: ability to ensure correctness



Conclusion

- **Modular autonomous vehicle platform deisgn.**
  - Bridges between asynchronous sensors and synchronous controller.
  - Allow flexible interchange sensors, actuators, vehicles.
- **Time-based controller.**
  - Facilitates analysis of closed-loop system.
  - Employs the time-triggered programming language - Giotto.
- **Hardware-in-the-loop simulation.**
  - Rapid, inexpensive, repeatible testing.

Berkeley
University of California

End