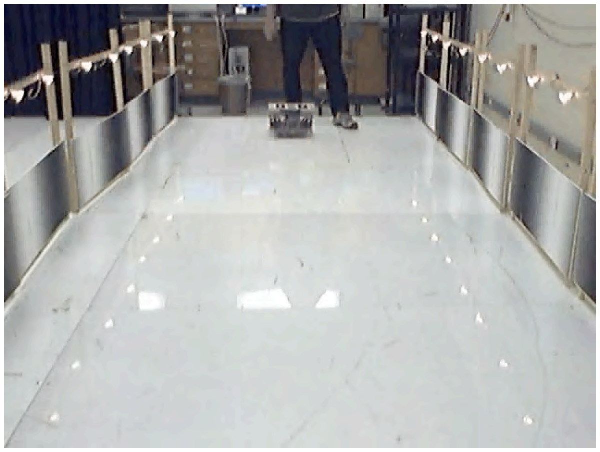
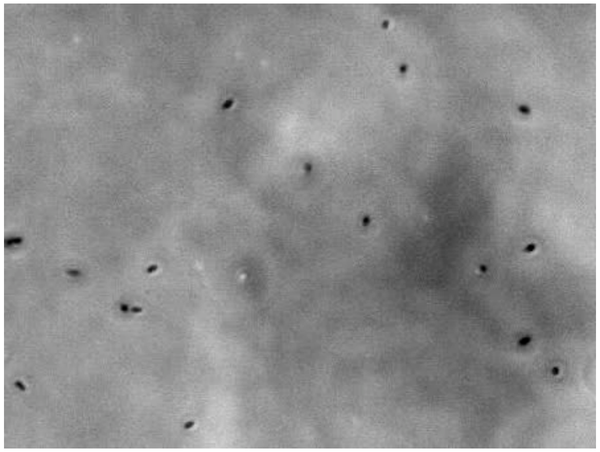
# Systems Engineering and Architecture

**Richard M. Murray**
Control and Dynamical Systems
California Institute of Technology

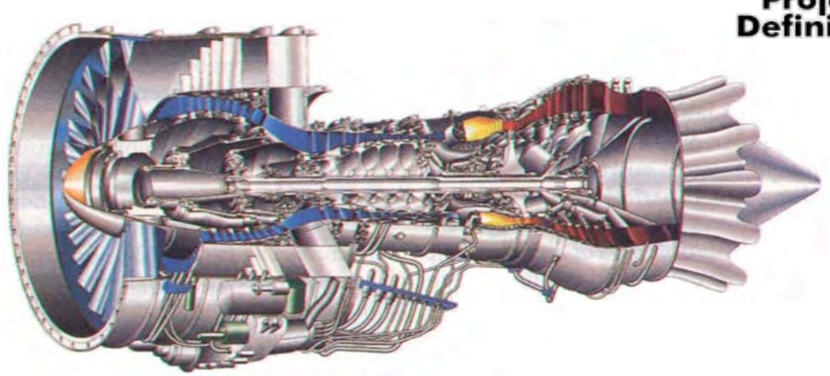
Design Principles in Biological Systems
21 April 2008

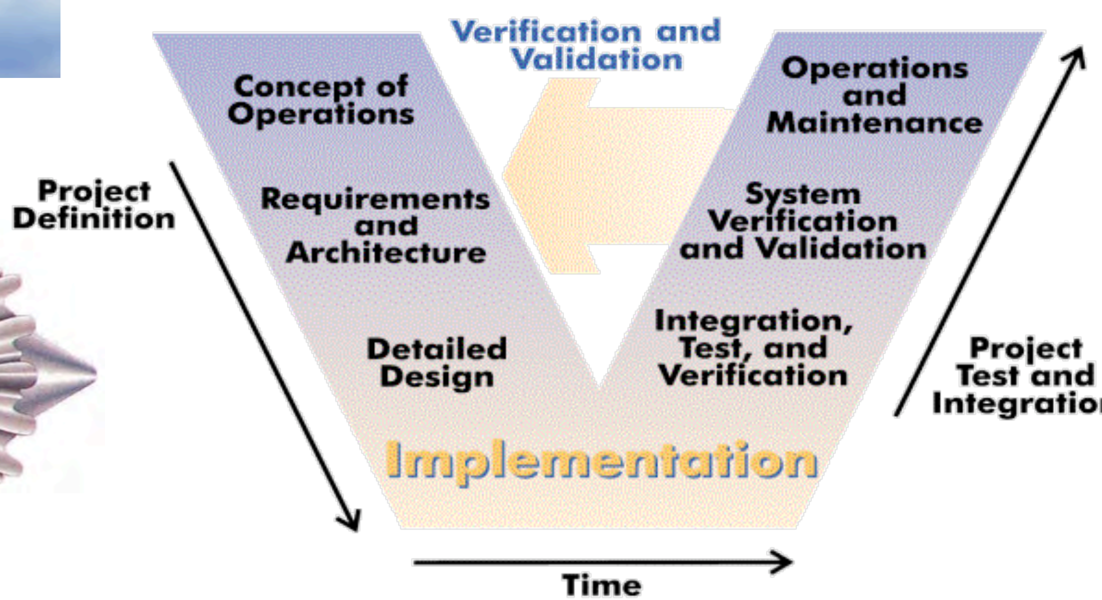




# Product Systems Engineering


**Systems engineering methodology**

- requirements capture and analysis
- systems architecture and design
- functional analysis
- interface design and specification
- communications protocol design & specs
- simulation and modeling
- verification and validation
- fault modeling





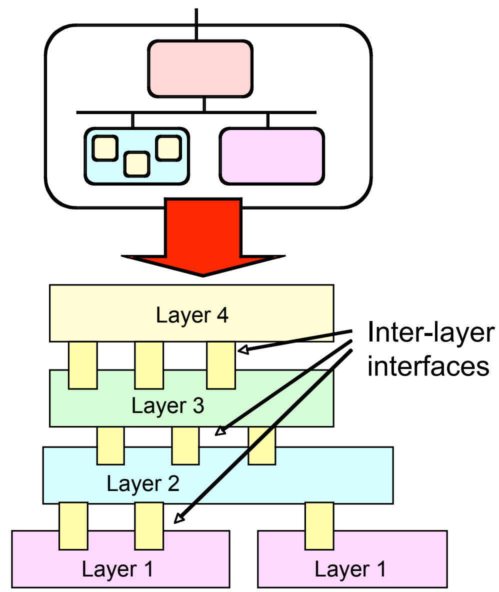
# Systems of Systems Engineering

**Little centralized control over the design**

- Individual systems build for specific purpose
- No global requirements document + *evolution*

**Example: air operations center (think ATC)**

- Multiple aircraft, designed over the last 50 years (with lots of variations in capabilities)
- Ground control stations + imagery analysis design to run independent of AOC
- All running on COTS computers, networks



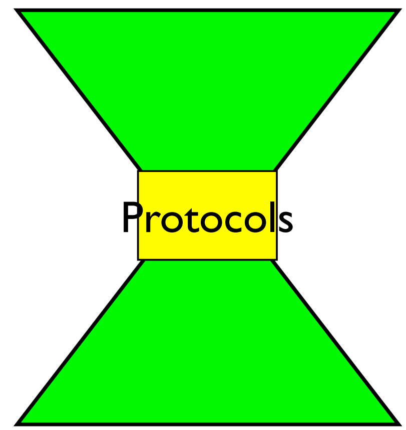
# The Role of Architecture

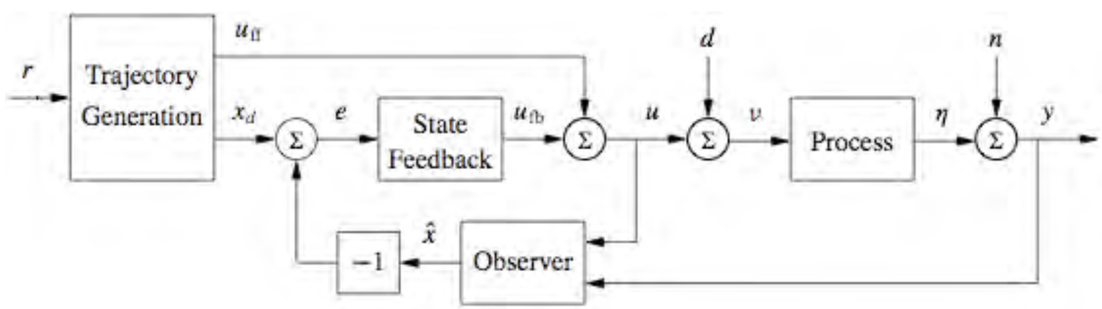
**How do we define architecture?**

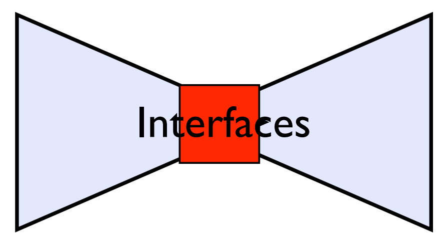
- IEEE: "The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution."
- Doyle (following Gerhart and Kirschner): "The constraints that deconstrain"
- Partha (following from building architecture): Integration of structure and function

**Some useful concepts**

- *Functional decomposition*: how do we break down a system into functionally independent modules
- *Interfaces and standards*: how to we specify consistent interfaces that let us integrate functional modules
- *Protocols*: how do we build layered abstractions that allow designers to ignore the details above and below



# Design Example: "Alice"

**DARPA Grand Challenge**

- 150 miles of autonomous desert driving
- Key challenge: uncertainty route/env
- Diversity: 198 teams → 120 → 43 → 23

**Alice**

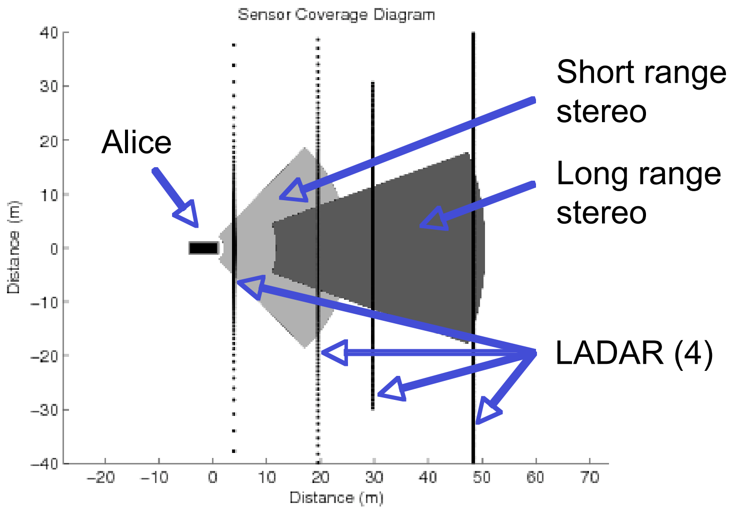
- 50 Caltech undergraduates, 1 year
- 5 cameras: 2 stereo pairs, roadfinding
- 5 LADARs: long, med*2, short, bumper
- 2 GPS units + 1 IMU (LN 200)

**Computing**

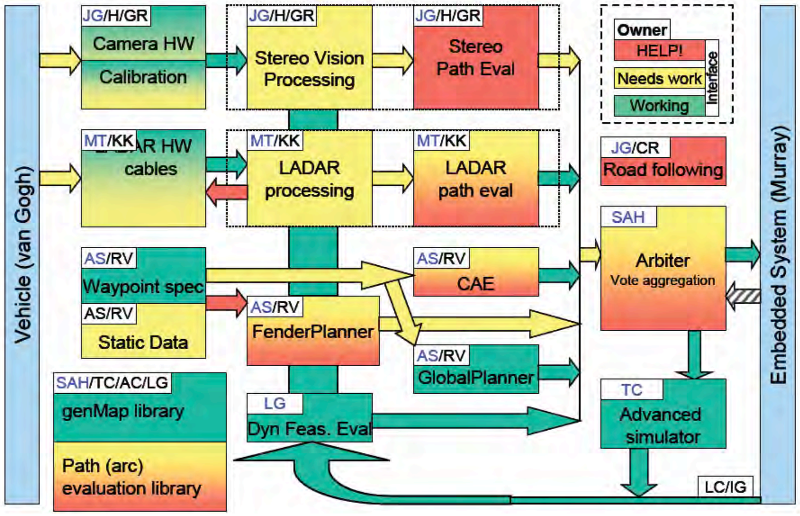
- 6 Dell PowerEdge Servers (P4, 3GHz)
- 1 IBM Quad Core AMD64 (fast!)
- 1 Gb/s switched ethernet

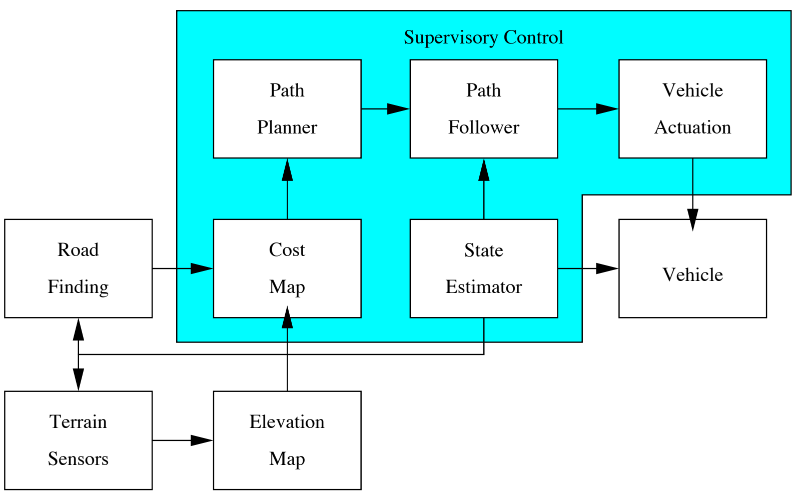
**Software**

- 15 programs with ~100 exec threads
- 100,000+ lines of executable code



# Evolution of Alice's Architecture




**Bob's architecture: arbiter based**

- Key idea: independent sensors "vote" for direction that vehicle should drive
- Key feature: once interface protocol for a "voter" is established, can work on many sensor processing approaches in parallel
- Limitation: very limited ability to "reason" about environment; no contingency plans
- Complexity: 20k (est) lines of C++ code

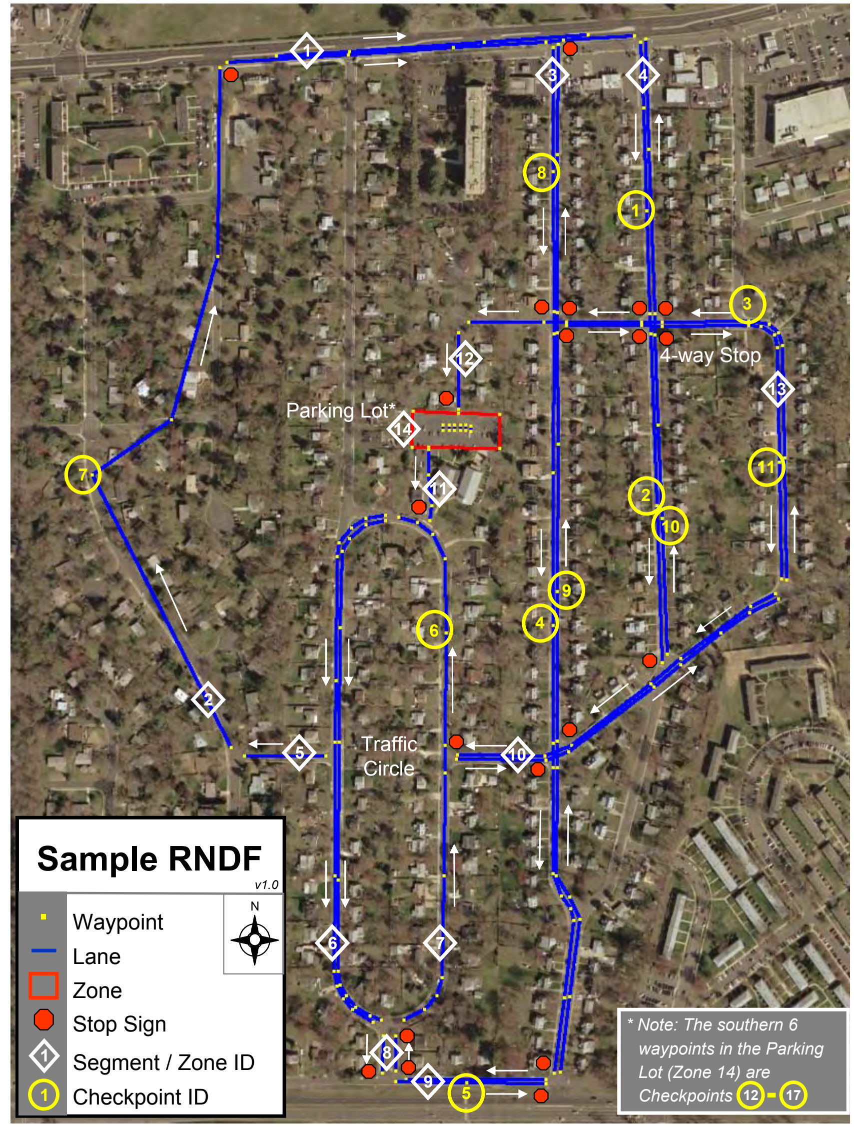
**Alice's architecture: cost map + planning**

- Higher level reasoning about environment based on cost map
- Key features:
    - Fuse elevation maps to allow parallel development of sensor pathways
    - Supervisor controller for contingencies
- Limitation: much more complex software
- Complexity: 100k lines of code; some reuse
- Built on top of lots of existing code + COTS

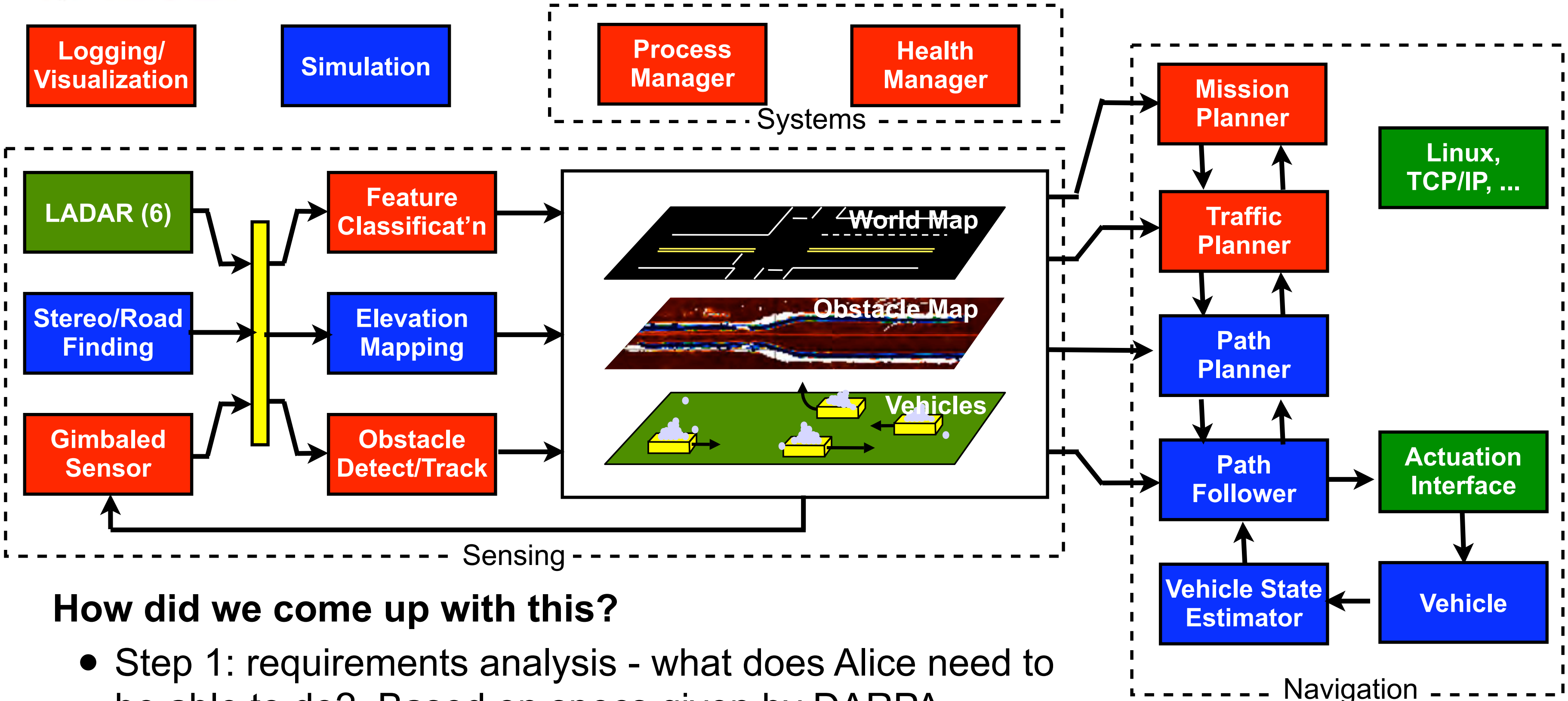
# 2007 DARPA Grand Challenge (Urban Challenge)

**Autonomous Urban Driving**

- 60 mile course, less than 6 hours
- City streets, obeying traffic rules
- Follow cars, maintain safe distance
- Pull around stopped, moving vehicles
- Stop and go through intersections
- Navigate in parking lots (w/ other cars)
- U turns, traffic merges, replanning
- Prizes: $2M, $1M, $500K



# DGC07 System Architecture (Gen 3)



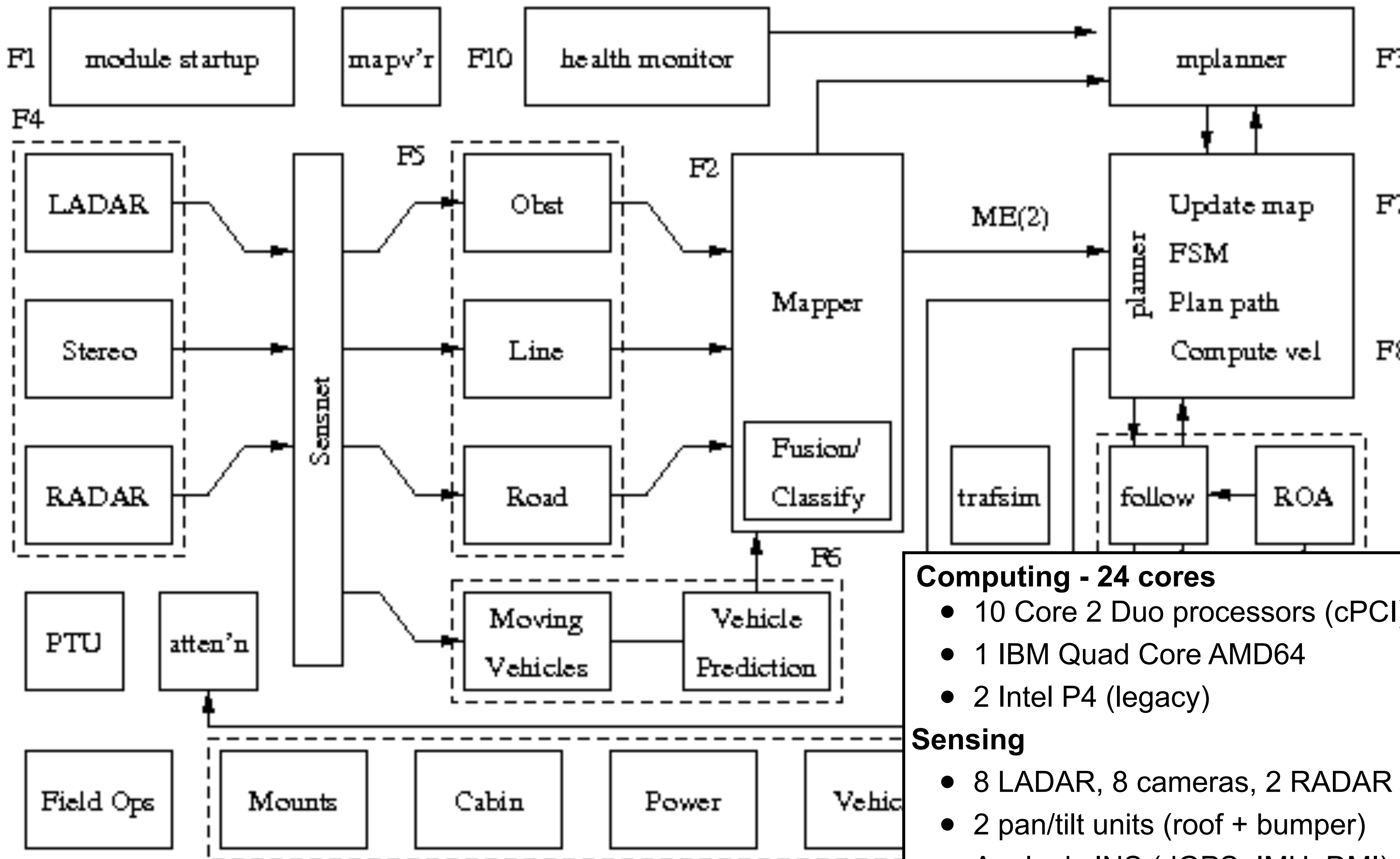

**How did we come up with this?**

- Step 1: requirements analysis - what does Alice need to be able to do? Based on specs given by DARPA
- Step 2: functional decomposition - what are the basic elements required to function? Designer choice
- Step 3: scenario generation and iteration - can it do what we want? Some simulation; mainly paper-based
- Step 4: interface specs (50% inherited ⇒ software reuse)

**Properties**

- Highly modular
- Rapidly adaptable
- Constantly viable
- Robust ???

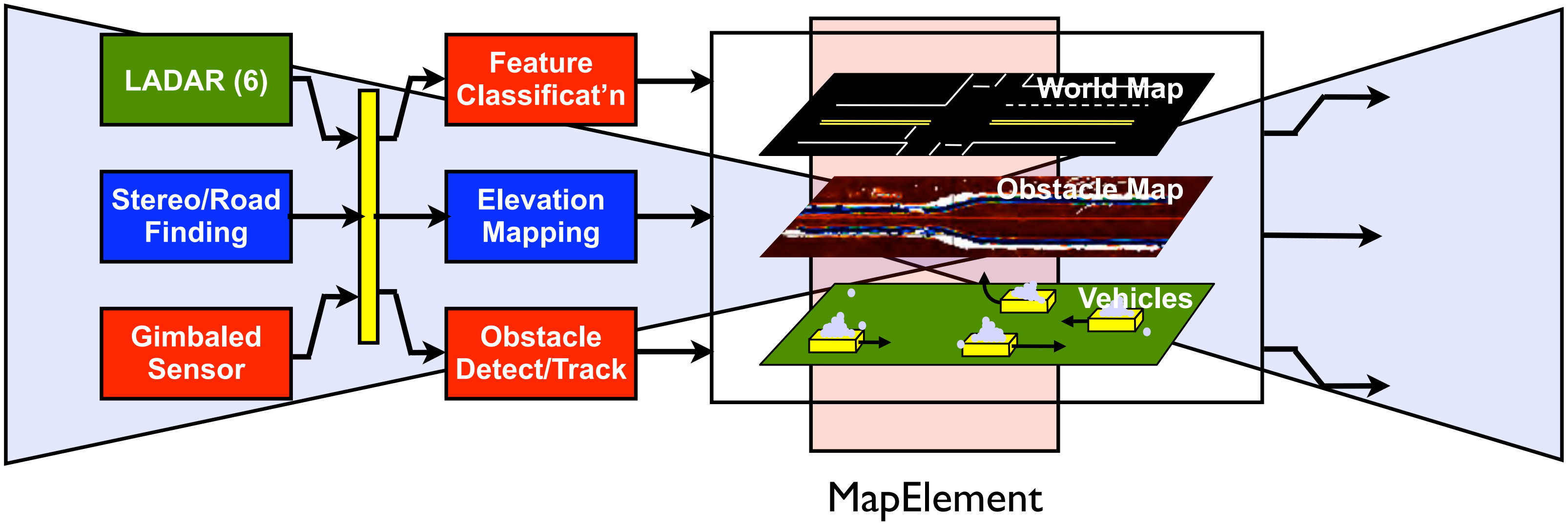
# Architecture, July 2007




**Computing - 24 cores**
- 10 Core 2 Duo processors (cPCI)
- 1 IBM Quad Core AMD64
- 2 Intel P4 (legacy)

**Sensing**
- 8 LADAR, 8 cameras, 2 RADAR
- 2 pan/tilt units (roof + bumper)
- Applanix INS (dGPS, IMU, DMI)

# Sensing Bowtie



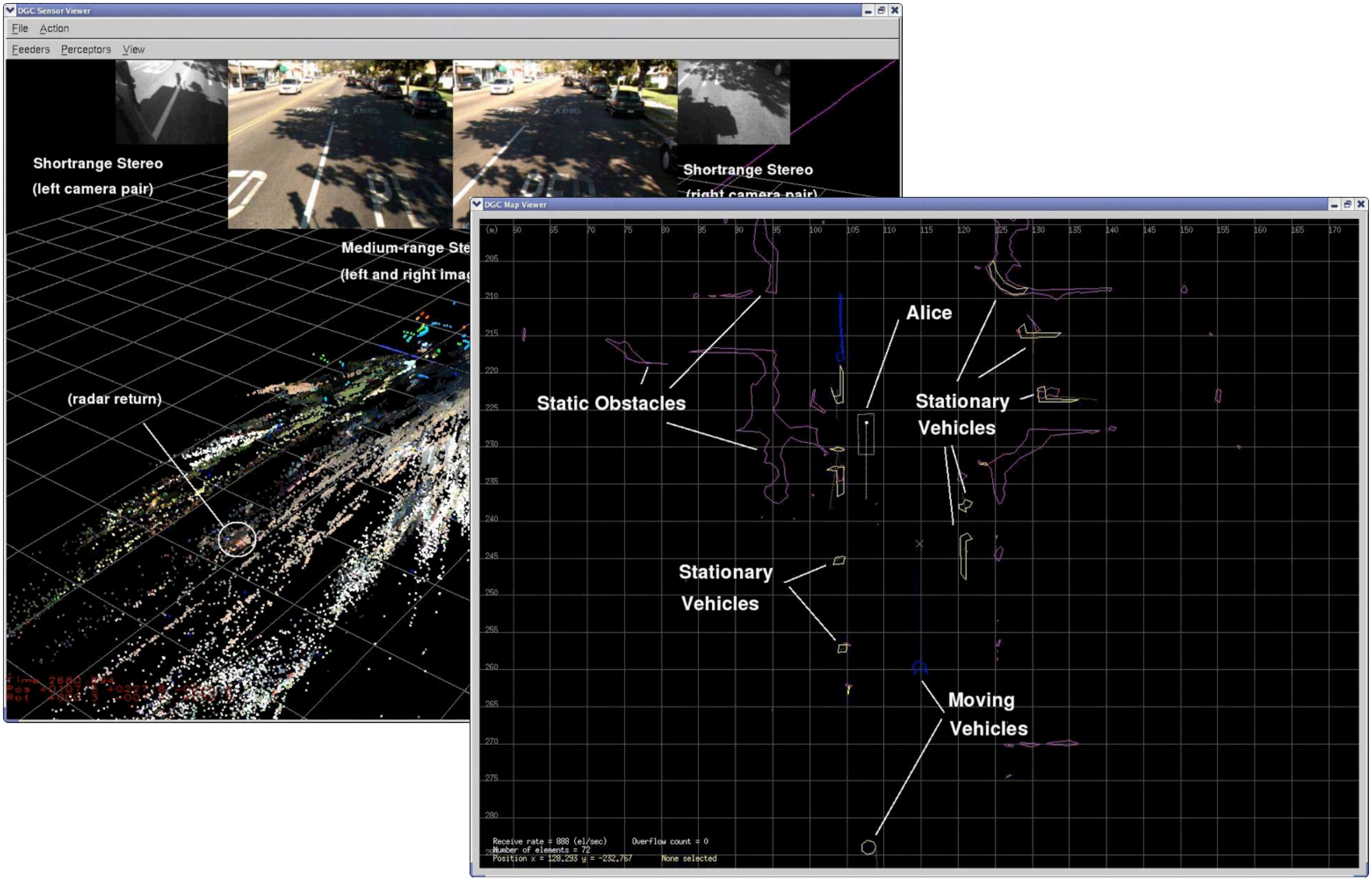

**MapElement serves as constraint that deconstrains**

- Fix the structure of the elements in the world map
- Left end: sensors → perceptors → MapElements
- Right end: MapElements → environment descriptions → planners

**Engineering principle: allow parallel development (people and time) + flexibility**

- Fixing the map element structure allows 15 people to work simultaneously
- We can evolve/adapt our design over time, as we get closer to the race

# Feeder → Perceptors → Mapper



# Planning Hourglass

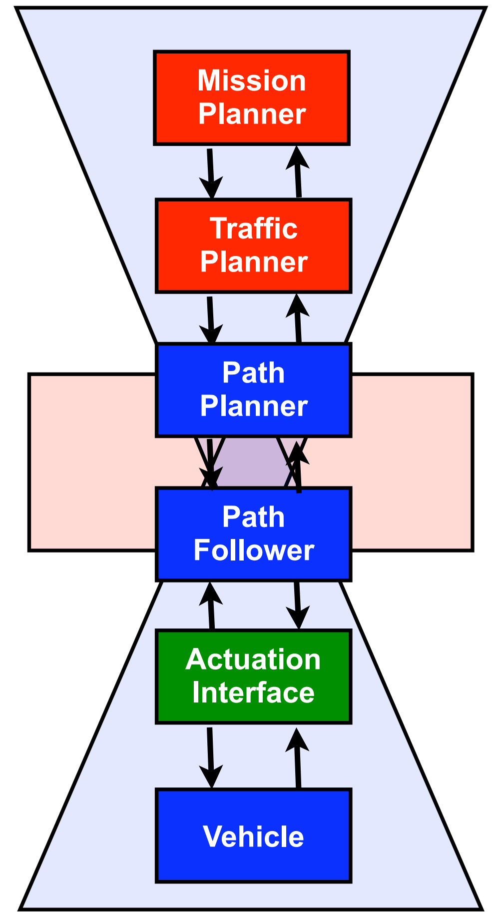
[redacted]

**Protocol stack based architecture**

- Planners uses directives/responses to communicate
- Each layer is isolated from the ones above and below
- Have 4 different path planners under development, two different traffic planners.  Rewriting the controllers as we speak (literally)

**Engineering principle: protocols isolate interactions**

- Define each layer to have a specific purpose; don't rely on knowledge of lower level details
- Important to pass information back and forth through the layers; a fairly in an actuator just generate a change in the path (and perhaps the mission)
- Higher layers (not shown) monitor health and can act as "hormones" (affecting multiple subsystems)



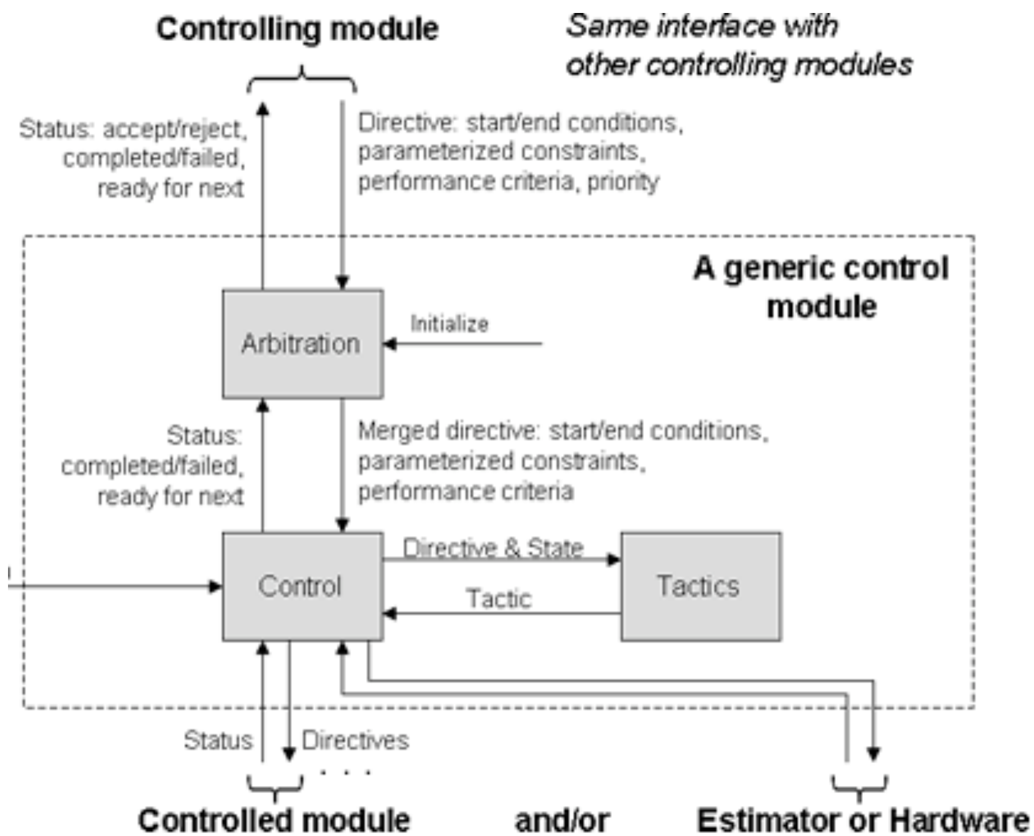
# Canonical Software Architecture

**Directive/response framework**

- Each component communicates with its neighbors through directives and status
- Separate taking directives from other components (in their terms) from a given component's core function and directives (in its own terms)
- Build on JPL "State Analysis" (Rasmussen et al)

**Modularity**

- Interfaces are defined independently from the module structure, such that when one module gets rebuilt,the modules that it talks to can remain the same
- Each component is divided into three parts
  - Arbitration: accept/reject
  - Control: execute
  - Tactics: success/fail



# Testing at El Toro

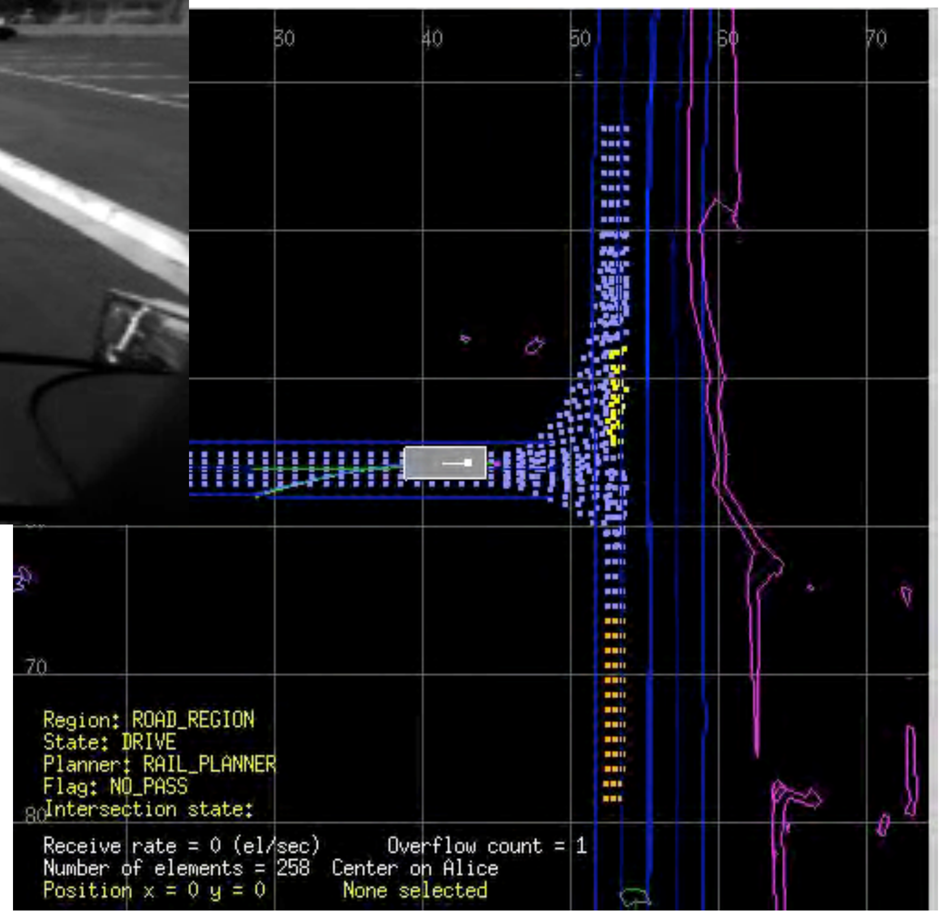
**Approximate 300 miles of testing over 2 months**

- Longest run without intervention: 11 miles
- Top average speed: ~10 mph

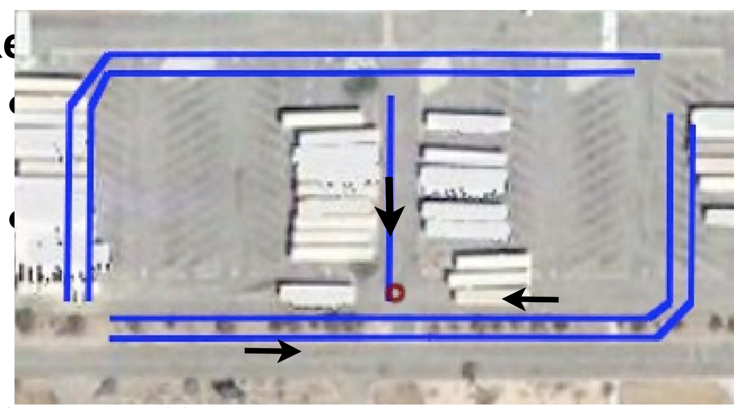
# 2007 National Qualifying Event

**Merging test**

- 10-12 cars circling past inters’n
- Count “perfect runs” in 30 min



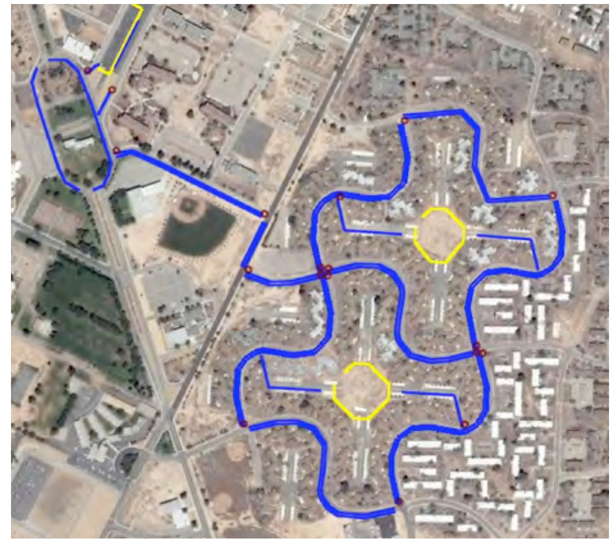

Re



# 2007 National Qualifying Event

**Driving test**

- 2 mile run - roads, parking lots, obstacles on road



**Results**

- First run - safety buffers too large => slow progress
- Second run - completed course in 22 minutes; minor errors
- 1 of ~8 vehicles completed

# Richard's Observations (JPL, Feb 08)

**Things that worked**

- Basic architecture was sound; capable of completing the race
- Sensing subsystem performed well (some spurious/misclassified obstacles)
- Rail planner was very versatile: could handle on-road/off-road, forward-reverse, etc
- Canonical software architecture worked better than previous supervisory control
- Alice was good at not getting stuck and not running into things

**Things that didn't work**

- Original planning approach (OTG) was too slow; late switch to rail planner
- Canonical software architecture implementation slowed us down
- Traffic logic was complex; difficult to debug and modify; didn't get stressed until late in the cycle
- Planned project schedule was too aggressive (but probably required to qualify)

# Credits (2007)

**Technical Contributers (79):** Daniel Alvarez, Mohamed Aly, Jessica Austin, Brandt Belson, Philipp Boettcher, Julia Braman, Joel Burdick, William David Carrillo, Vanessa Carson, Arthur Chang, Edward Chen, Steve Chien, Jay Conrod, Iain Cranston, Lars Cremean, Josh Doubleday, Tom Duong, Stefano di Cairano, Noel duToit, Luke Durrant, Josh Feingold, Matthew Feldman, Tony and Sandie Fender, Nicholas Fette, Ken Fisher, Melvin Flores, Brent Goldman, Scott Goodfriend, Sven Gowal, Steven Gray, Rob Grogan, Jerry He, Phillip Ho, Andrew Howard , Mitch Ingham, Nikhil Jain, Michael Kaye, Aditya Khosla, Ryan Lim, Magnus Linderoth, Laura Lindzey, Christian Looman, Ghyrn Loveness, Jeremy Ma, Justin McAllister, Joe McDonnell, Richard Murray, Russell Newman, Noele Norris, Josh Oreman, Kenny Oslund, Robbie Paolini, Jimmy Paulos, Celia Peina, Humberto Pereira, Rich Petras, Sam Pfister, Christopher Rasmussen, Bob Rasumussen, Dominic Rizzo, Miles Robinson, Henrik Sandberg, Chris Schantz, Kristian Soltesz, Chess Stetson, Sashko Stubailo, Tamas Szalay, Klimka Szwaykowska, Daniel Talancon, Daniele Tamino, Pete Trautman, David Trotz, Glenn Wagner, Yi Wang, David Waylonis, Nok Wongpiromsarn, Albert Wu, Francisco Zabala, Johnny Zhang

**Major Sponsors:** DARPA, Caltech, IST, Big Dog Ventures, Mohr-Davidow