RoSES

Robust Self-configuring Embedded Systems

http://www.ece.cmu.edu/roses

Prof. Philip Koopman

Bill Nace – Charles Shelton – Meredith Beveridge – Tridib Chakravarty – Chris Martin – Mike Bigrigg

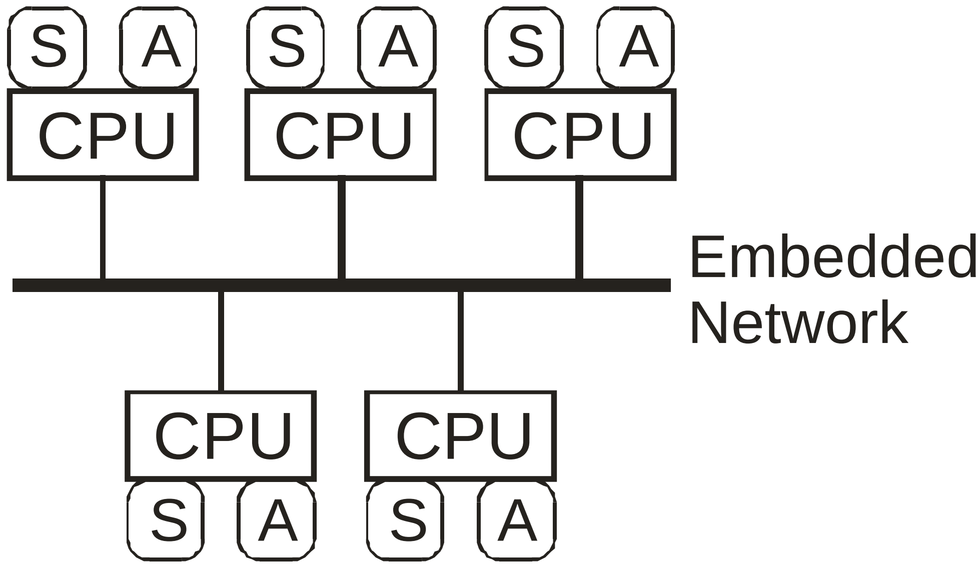
Institute for Complex Engineered Systems

Carnegie Mellon

Electrical & Computer ENGINEERING

RoSES Objectives

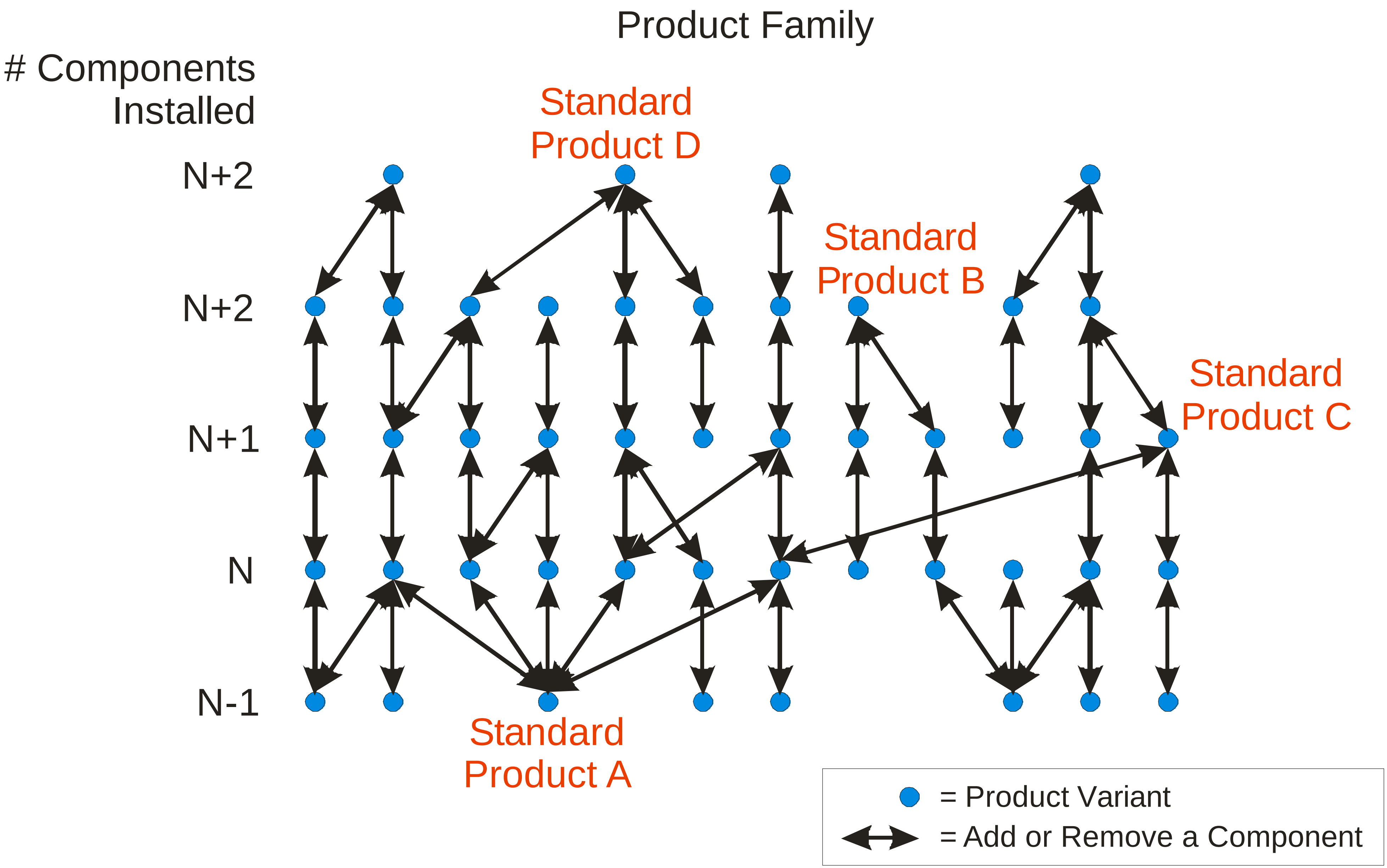
- **Context: distributed embedded systems**
  - Multiple "smart" sensors/actuators connected to embedded real-time network



- **Research goal – automatic reconfiguration for:**
  - Graceful degradation
  - Graceful reintegration after repair
  - Graceful acceptance of non-exact spares
  - Graceful upgrade with new capabilities

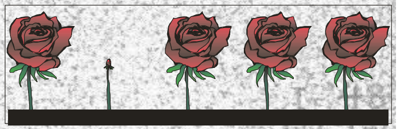
Product Family Architectures

- **Fine grain distributed systems yield dense product lattices**



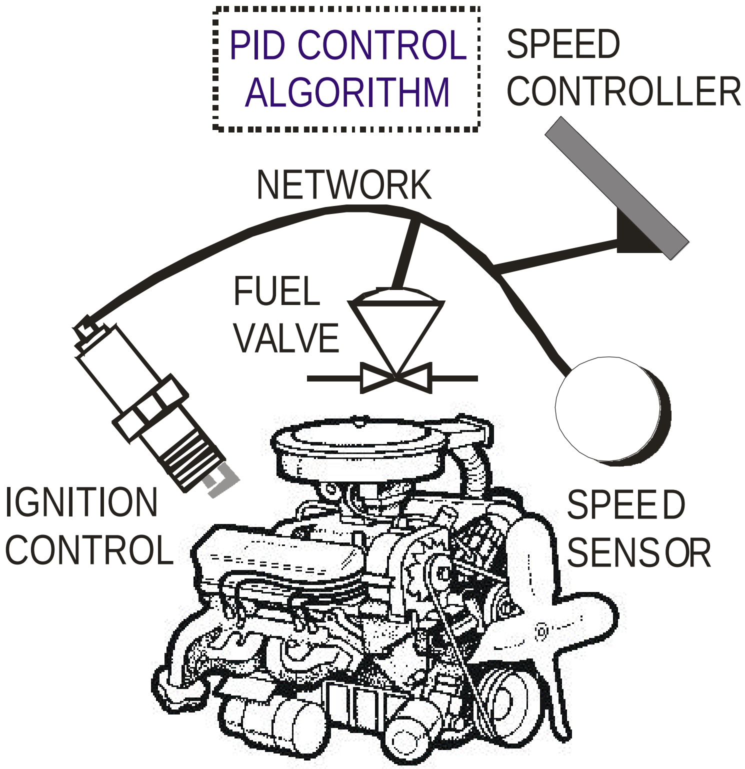
Why Self-Configuration?

- **Product Families + Automatic configuration management creates a unifying capability**
  - Product families can include degradation as well as intentional price/performance tradeoff points
- **Consider component failure as an example:**
  - Component fails –
    triggers reconfiguration for degraded operation
  - Component replaced –
    reconfiguration to integrate repair part
  - New component added –
    reconfiguration to upgrade system
- **That’s a lot to attempt all at once…**
  - Static configuration at first
  - On-the-fly configuration as an eventual goal



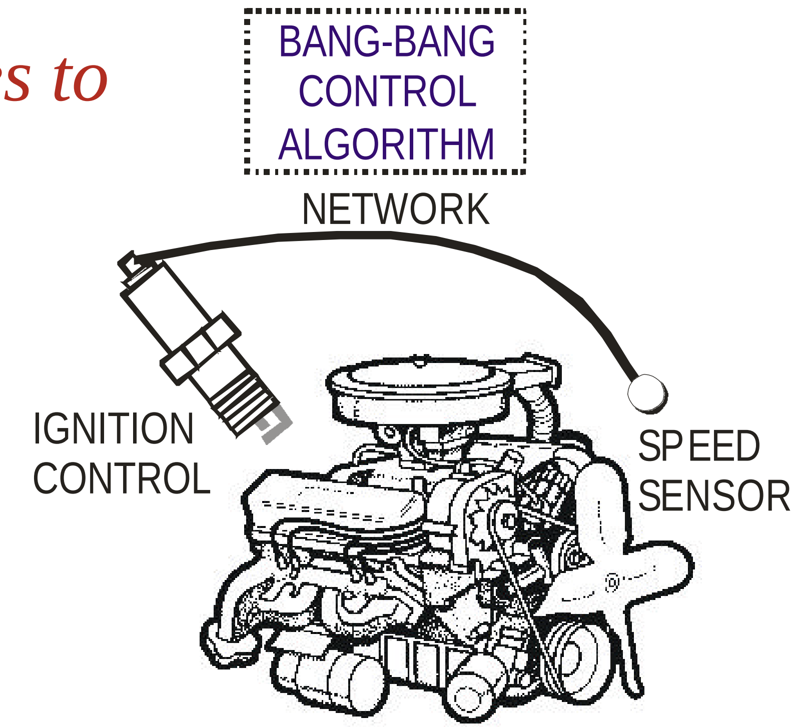
A Simplistic Example…

- **Control of gasoline engine speed**
  - Complicated system controls fuel if valve is installed/operational
  - But, baseline capability is retained in case of failure

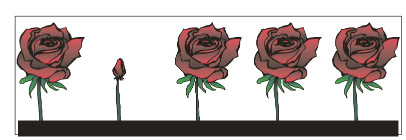


Throttled Engine Controller

Degrades to

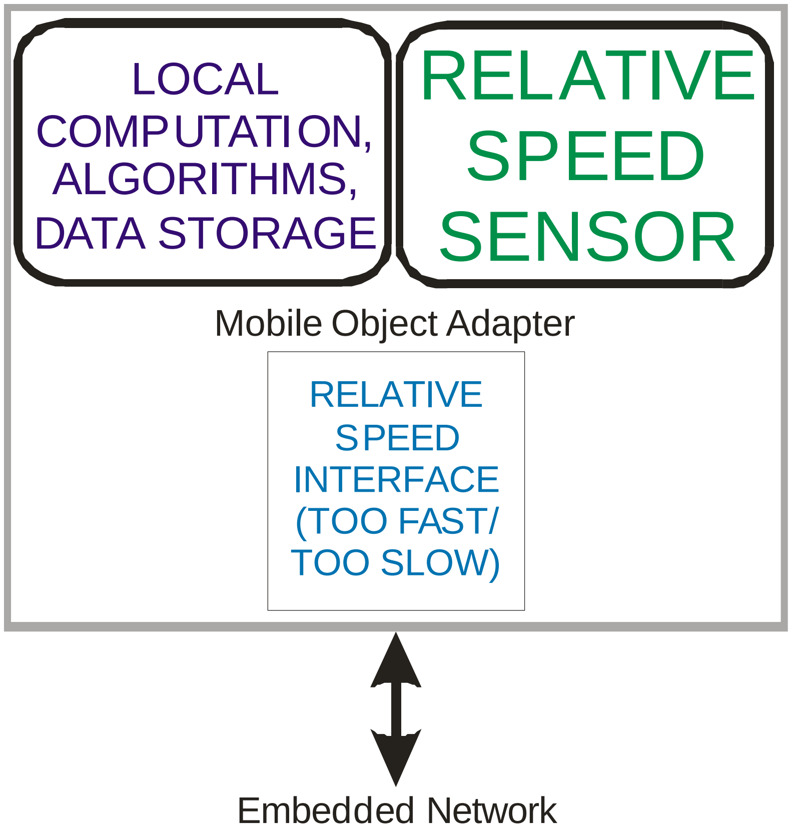


Hit-or-Miss Constant Speed Engine Controller

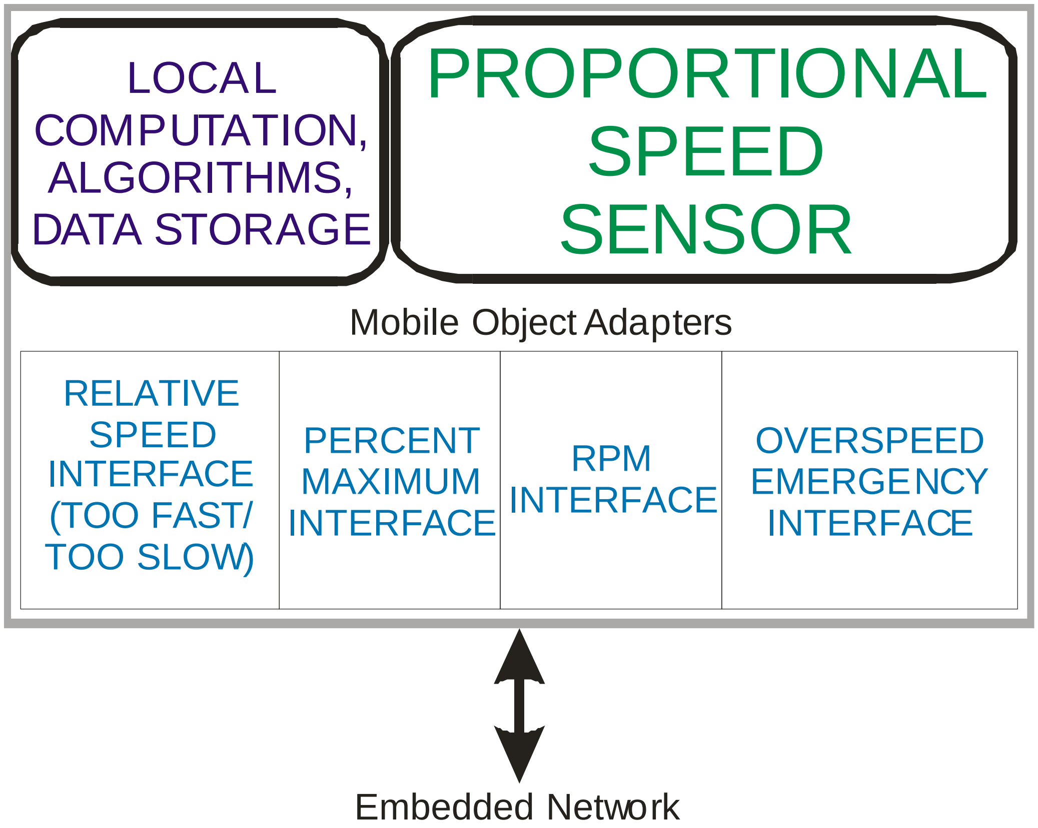


Different Sensors / Different Capabilities

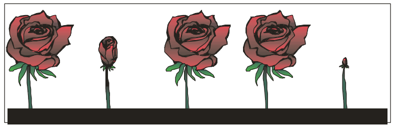
- **Similarly, different actuators have different capabilities**
  - *Mobile Object Adapters* translate raw capability into desired interface



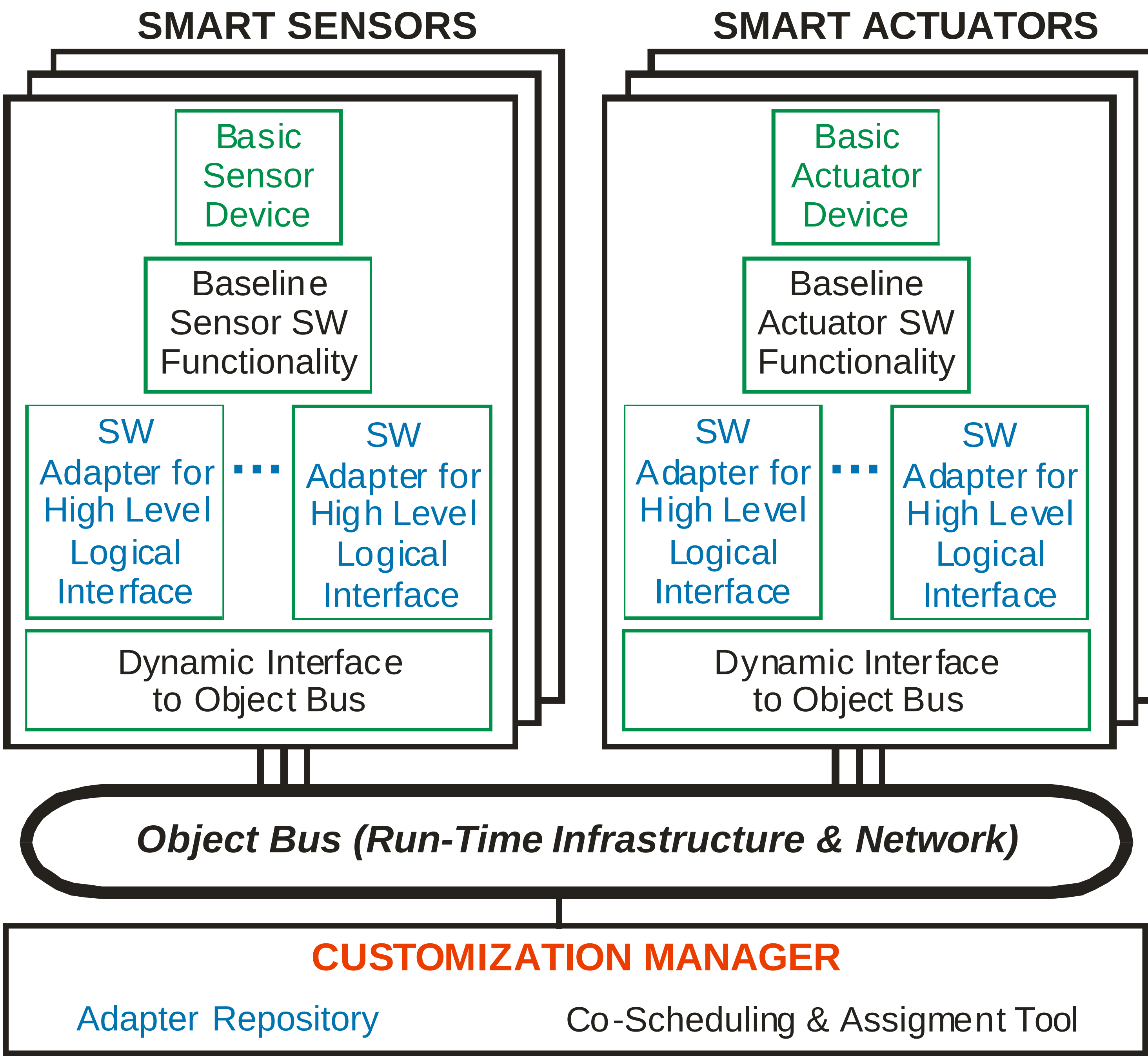
EXAMPLE SIMPLE SPEED SENSOR



EXAMPLE HIGH-END SPEED SENSOR with MULTIPLE INTERFACES

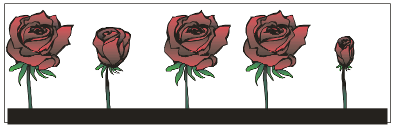


Generic RoSES System Architecture



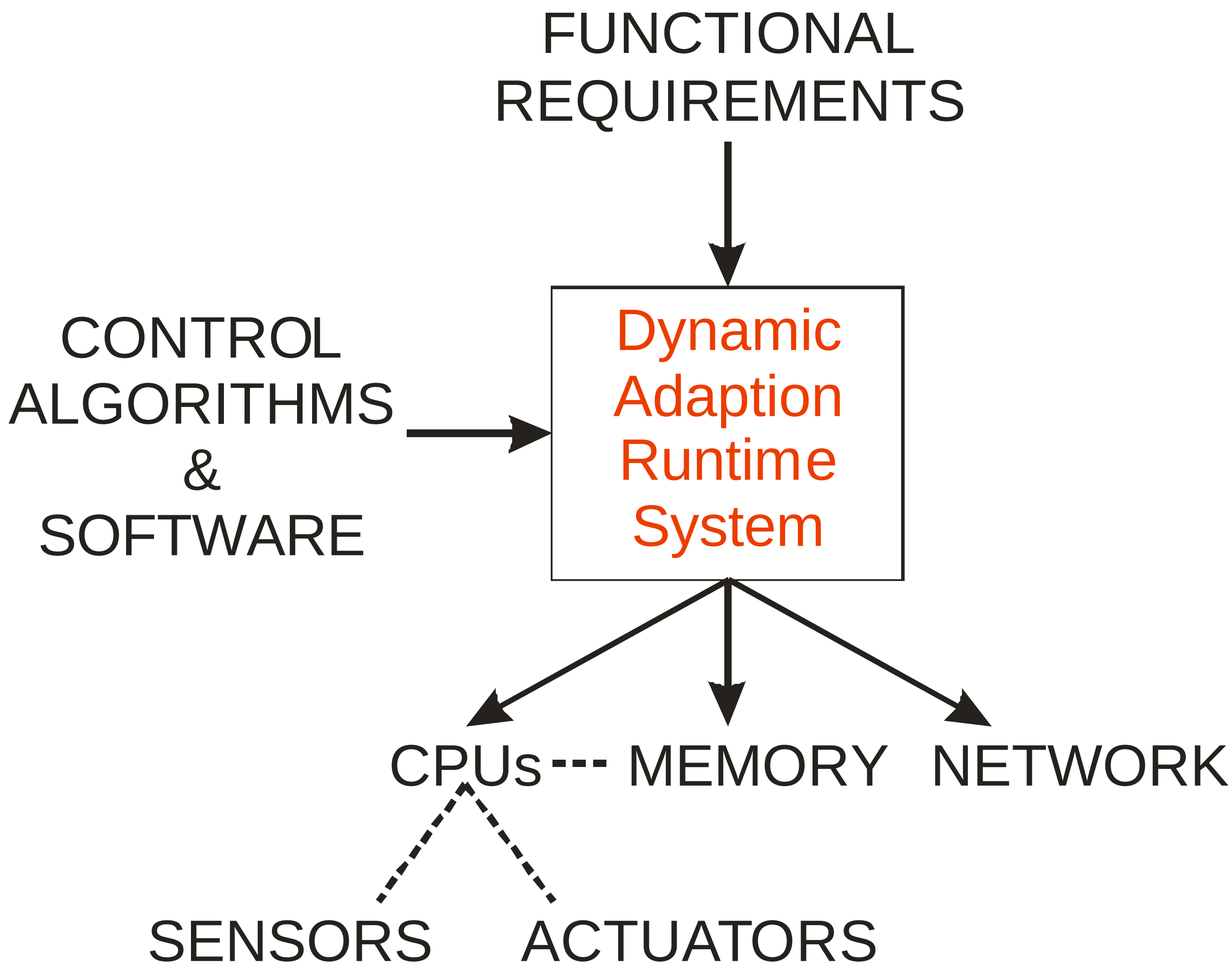
Near-Term Research Challenges

- **Mapping functionality onto hardware**
  - Maximize utility of result given constrained resources
- **Achieving real-time operation**
  - Co-schedule CPU, Memory, Network usage to meet real-time deadlines
- **Achieving "plug & play" capabilities**
  - *e.g.*, What would it take to put CORBA on a CAN network?
  - Avoid re-inventing CORBA if possible…
- **Tractable demonstration**
  - Generic automotive testbed



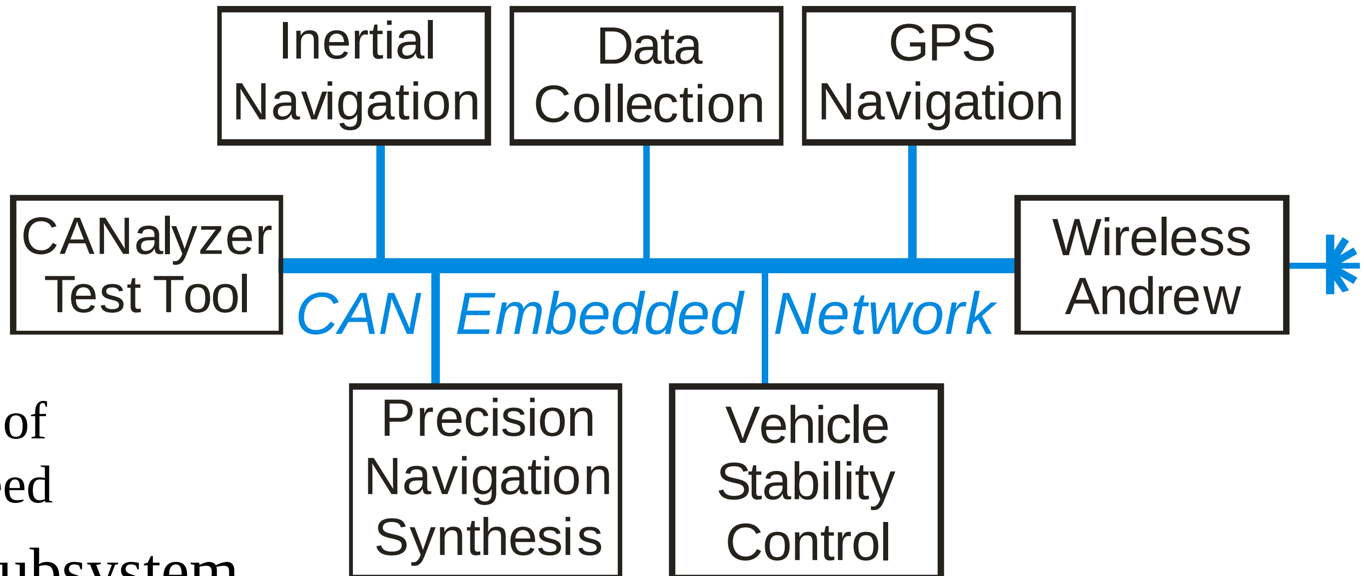
Functionality To Hardware Mapping

- **Automatic allocation of HW & SW components**
  - Maximize utility of functions within hardware constraints



Proposed Testbed

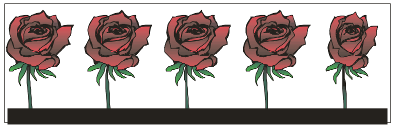
- **Navigation + active vehicle stability control**
  - Inertial sensors / dead reckoning subsystem
    - 3d inertial platform with acceleration and speed readouts
    - Steering angle
    - Wheel speed
  - GPS-based navigation subsystem
    - External source of position and speed



  - Data collection subsystem
    - Stores and forwards data for failure diagnosis
  - Gateway to wireless internet connection
    - Simulated using Wireless Andrew
  - CANalyzer system to simulate rest of vehicle network
    - Provides messages for realistic operating environment

Experiments

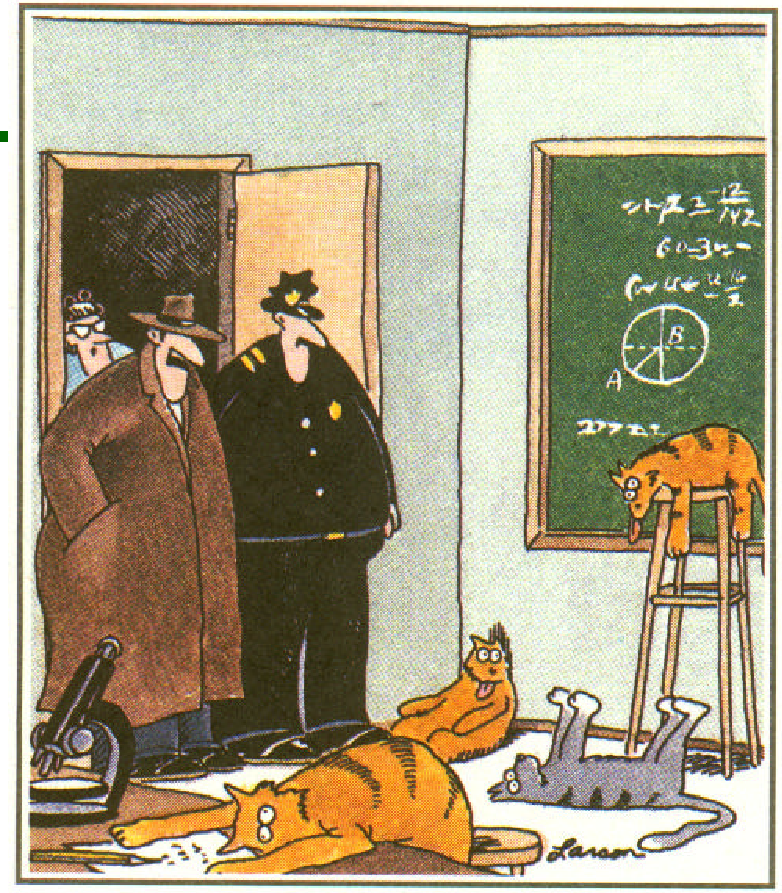
- **Applications**
  - Phase 1: Precision navigation
    - Gracefully degrading navigation based on sensor information
    - Combine inertial & GPS for result better than either alone
  - Phase 2: Active vehicle stability control
    - Vehicle stability algorithms vary degree of control based on quality of sensor information
    - Graceful degradation rather than brute force redundancy
- **Year 1 goal – lab demo:**
  - Demonstrate automatic reconfiguration when sensors/actuators/ computers fail for navigation application
- **Later goals:**
  - Demonstrate automatic reconfiguration in a test vehicle
  - Demonstrate upgrade of baseline component with advanced/proprietary component
  - Demonstrate graceful failure and restoration after repair in a test vehicle



Possible Extensions

- **Embedded system point of view for system security**
  - Vehicle / external firewall
  - Protection against malicious or faulty components on networks

- **Safe upgrade strategies**
  - Even if new components have software defects (?!)



"Notice all the computations, theoretical scribblings, and lab equipment, Norm. ... Yes, curiosity killed these cats."

People

- **Prof. Phil Koopman**
  - 10+ years industry experience embedded hardware & software
- **Bill Nace**
  - Ph.D. student: functionality-to-resource mapping
- **Charles Shelton**
  - Ph.D. student: software architecture
- **Meredith Beveridge**
  - M.S. student: testbed/baseline applications + Jini-on-CAN
- **Chris Martin**
  - M.S. student: simulation infrastructure (details TBD)
- **Tridib Chakravarty**
  - M.S. student: embedded protocol infrastructure
- **Mike Bigrigg –** staff member

R o S E S