# Scalable Graceful Degradation in Distributed Embedded Systems



**Charles P. Shelton** Philip Koopman

### Specifying Graceful Degradation is Exponentially Complex:

Must rank 2<sup>N</sup> system configurations of N software components, sensors, and actuators

### Our Model Achieves Scalable Specification for Data Flow-Centric Embedded Systems:

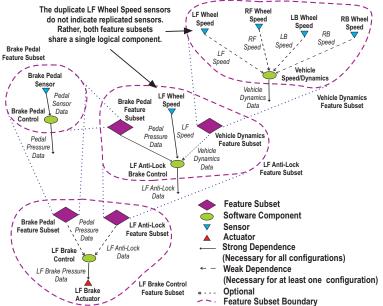
Analysis reduced from  $O(2^{N})$  to  $O(N^{*}2^{k})$ ; k = number of components per subsystem

- 1. Separate system into orthogonal software and hardware views; focus on software configs
- 2. Partition system into feature subsets based on component input and output interfaces
- 3. Allow feature subsets to overlap and share components
- 4. Rank relative utility of configurations of subsystems, not all configurations of entire system
- 5. Determine system configuration utility as a composition of working subsystem utilities

### **Proof of Concept:**

#### Scalable Specification of an example Brake-**By-Wire system**

- Real-world electromechanical anti-lock braking system transformed to hypothetical embedded network control system
- 10 Software Components, 5 Sensors, 4 Actuators = **19 System Components**
- 2<sup>19</sup> = 524.288 possible system configurations; 89,600 valid with positive utility
- In our model: 10 defined subsystems, max 5 components per subsystem
- Specify utility of only 52 subsystem configurations for complete utility function for all system configurations



## **Application to Real Distributed Embedded Systems:**

#### Autonomous Robot Navigation:

2.50 of accuracy and Combine line following Config 1 Config 4 sensor system with location (Nothing broken) (Dead reckoner software. tracking and navigation 2.00 to complete course) Algorithms path planner Performance Metric software broken) Config 2 Run robot with combination Config 5 1.50 (Line follower sensors of failed sensors and (Dead reckoner software, broken) software to observe path planner software, combined measure graceful degradation front wheel encoder sensor Config 3 1.00 broken) (Line follower sensors, front wheel encoder sensor Config 6 broken); backup rear wheel 0.50 time 1 (Everything except sensors did not provide line sensors and expected utility in the motor actuators broken) implementation 0.00 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 Utility Value from Model Institute **Carnegie Mellon** 

## **Graceful Degradation Validation:**

Relative utility differences of system configurations in the model match relative performance differences of system configuration implementations







Mobot Utility Vs. Performance

