



A Product Family Approach to Graceful Degradation

Bill Nace

Philip Koopman

Carnegie Mellon University

Pittsburgh, PA USA

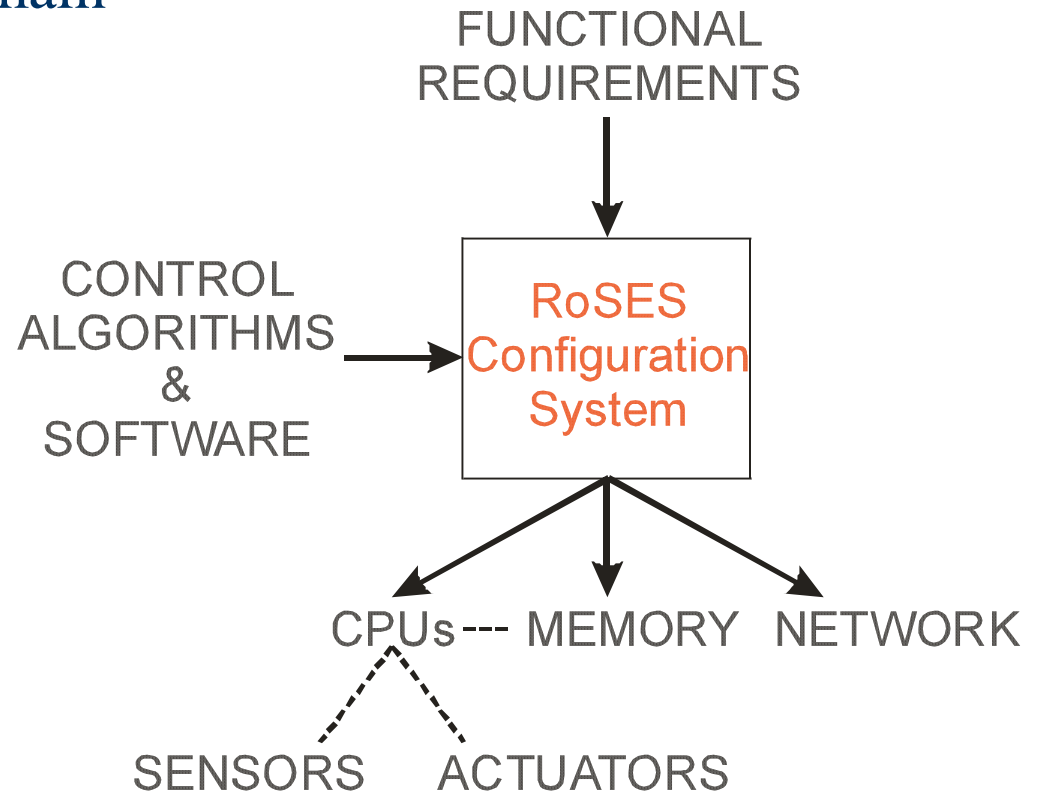


Agenda

- ◆ RoSES = Robust Self-configuring Embedded Systems
 - Examines automatic graceful degradation
- ◆ Graceful Degradation
 - Desirable, but difficult to achieve
- ◆ Product Family Architecture
 - Common, but not used for robustness
- ◆ Main point: achieve Graceful Degradation through Product Family Architectures
 - Mechanism: automatic reconfiguration

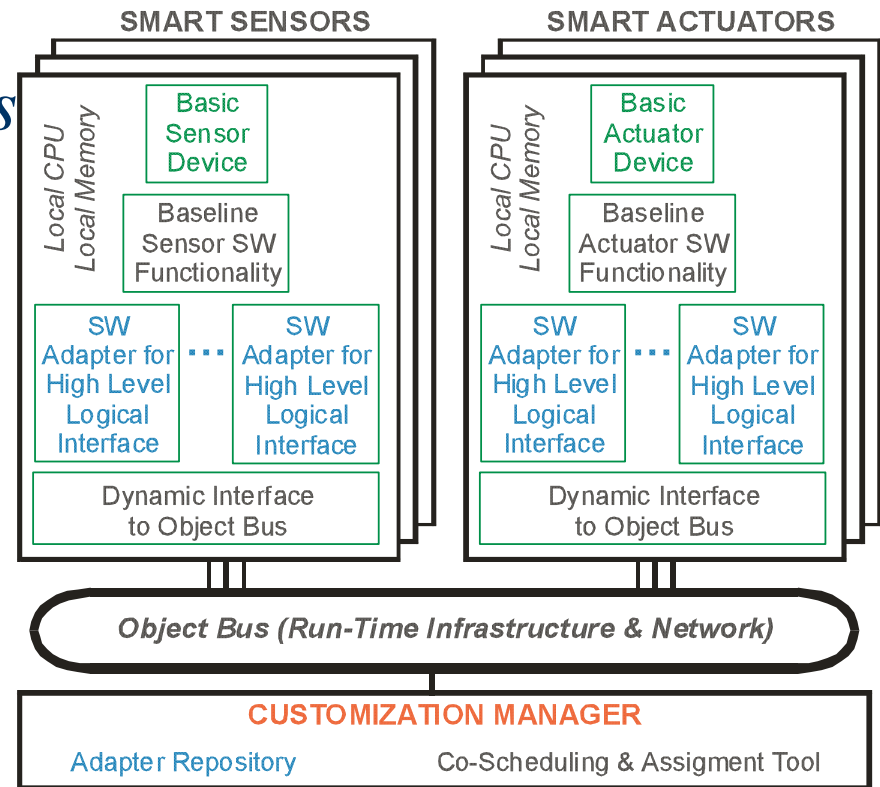
Intro to RoSES

- ◆ Robust Self-configuring Embedded Systems
- ◆ Goal: investigate automatic graceful degradation
 - Initially automotive domain
 - Elevators for teaching
- ◆ Static reconfiguration
 - Dynamic in future??



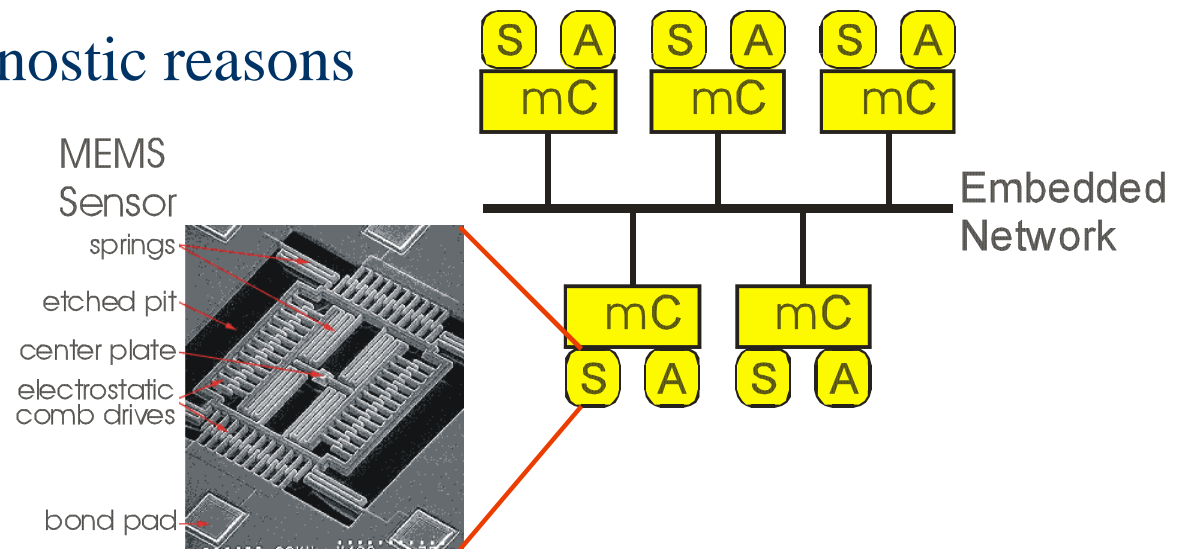
Details on RoSES

- ◆ Components modeled as abstract functionality blocks
- ◆ Using *mobile object adapters* to match configuration requirements to sensor functionality
 - Loaded by *Reconfiguration Manager* from *Adapter Repository*



Embedded Systems

- ◆ Sensors/Actuators built on integrated microcontrollers
 - MEMS accelerates trend
- ◆ Dozens/Hundreds of “smart” Sensors/Actuators
 - Attached to real-time network(s)
 - Often only for diagnostic reasons



- ◆ Bottom Line: centralized → federated

Embedded *Systems*

- ◆ Electronics growth in system mainly used for optimization
- ◆ Base system (critical) functionality 10-30%
- ◆ Remainder optimization
 - For fuel economy, remote diagnostics, marketing features, etc



Graceful Degradation

- ◆ System loses functionality incrementally when faced with failure
- ◆ Hard to design -- must anticipate failure modes
 - Fault-Tolerant systems traditionally rely on brute force replication and failover algorithms
- ◆ Ideally non-critical functionality is shed first



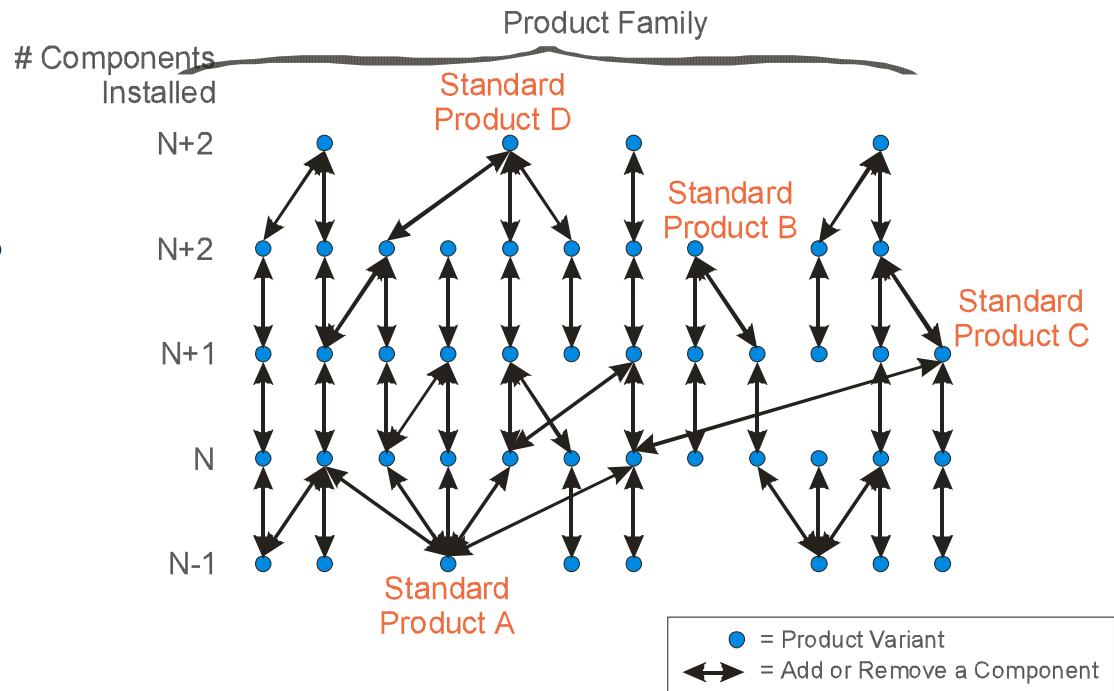
Graceful Degradation Approach

- ◆ On failure
 - Shed optimizations
 - Allocate remaining resources to mission fulfillment
- ◆ BUT, what if optimizations can't be discarded?
 - Ensure system built with such separation in mind
 - Architecture is important!



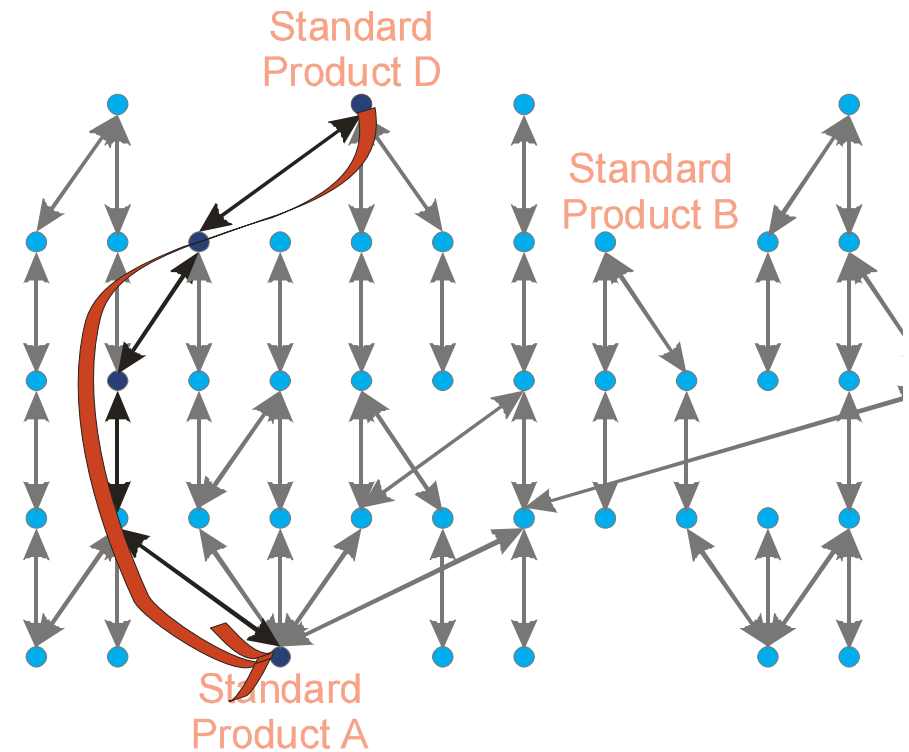
Product Family Architecture

- ◆ System design space populated by configuration lattice
- ◆ Each configuration represents a potential product model
 - Familiar to consumers of electronics, autos...
- ◆ Generally optimized assuming no failures



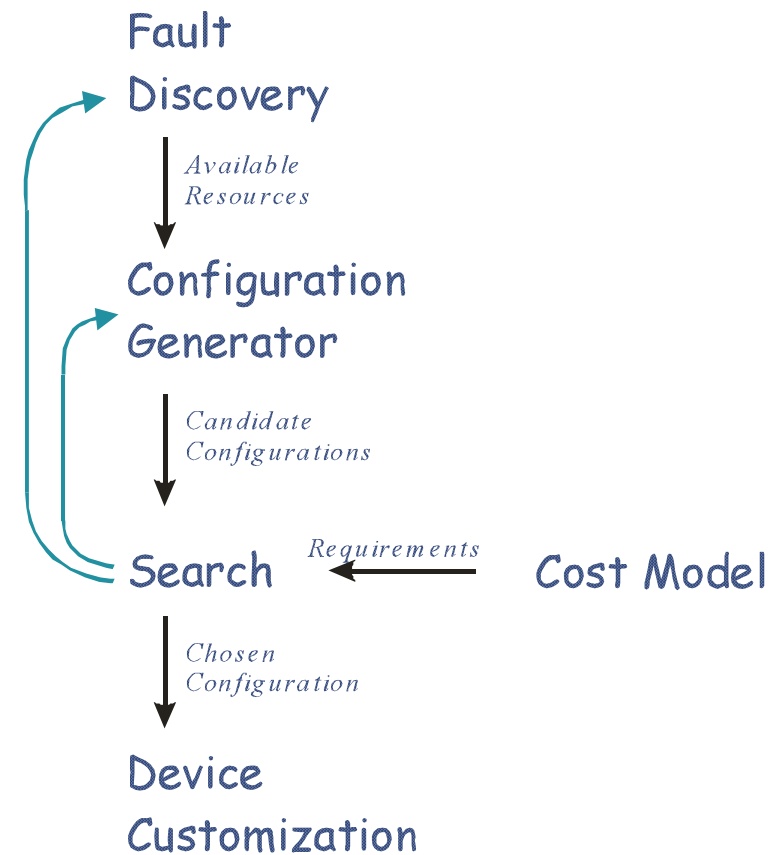
Reconfiguration

- ◆ Unifying mechanism for graceful degradation
- ◆ Optimize system functionality based on available resources
- ◆ Graceful degradation is now merely movement down the PFA lattice
 - Path determined by reconfiguration manager



Reconfiguration Manager

- ◆ Configuration changes not restricted to lattice
 - Re-synthesize functional architecture on the fly
- ◆ Fault Discovery/System Model
- ◆ Configuration Generator
 - Dependency Model
 - Validity Checker
- ◆ Cost Model
- ◆ Device Customization



Other Uses

- ◆ Graceful Repair Reintegration
 - Upgrade the system when a component is fixed
- ◆ Reconfiguration as Logistical Support
 - Replacement with non-exact spares
 - Legacy support requirements minimized



Challenges

- ◆ Debugging/Technical
- ◆ Certification
- ◆ Error Detection
- ◆ Multi-Vendor Challenges
 - Design for cross-vendor reconfiguration
 - Liability
- ◆ Dynamic reconfiguration



Related Work

- ◆ Herlihy91 “Specifying Graceful Degradation”
 - Lattice formulation of system behavior constraints
- ◆ Altenbernd97 “The Slack Method”
 - Allocation of tasks to distributed system
- ◆ Beck98 “Automatic Configuration of Embedded Multicomputer Systems”
 - Specification of hardware resources to support
 - Allocation of tasks to distributed system
- ◆ Shimomura95 “Development of Self-Maintenance Photocopiers”
 - Physical model based approach





More info

www.ece.cmu.edu/roes

wnace@cmu.edu

koopman@cmu.edu

