RoSES
Robust
Self-configuring
Embedded
Systems

William Nace
Philip Koopman

Building a Graceful Degradation Algorithm

A Framework for Distributed Embedded Systems

Graceful Degradation: The next step for dependable systems

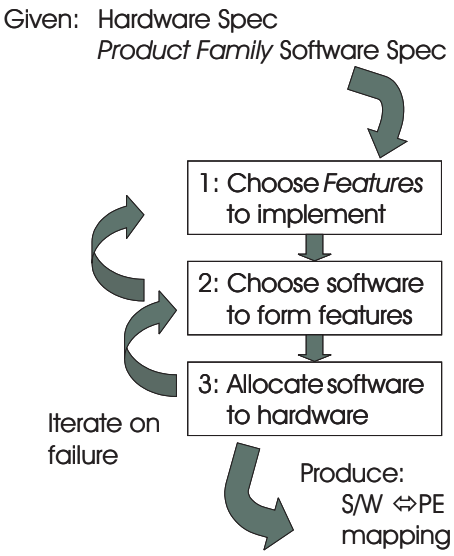
- Economically sensitive systems cannot use standard fault-tolerance techniques (e.g. modular redundancy)
- Luckily, such systems have many non-mission critical resources that can be diverted to critical uses -- and thus the potential for graceful degradation. For example, use a car stereo processor to execute brake control S/W
- System complexity requires automated methods to determine fallback configurations, not pre-specification by design engineers. An automobile with 100 network nodes has an unmanageable 2^{100} possible configurations

Mission Statement: What does the algorithm do?

- Maximize the functionality of available hardware. For instance, after a hardware fault event
- Assign software components from a large library to particular processing elements

Input: Hardware and Software Specification

- Hardware: List of Processing Elements (PE), Network and their resource sizes
- Software: Data Flow Graph (DFG) for individual configurations, merged into Product Family Architecture (PFA) graph to express all available configurations
  - PFA graph provides flexibility for configuration tradeoffs



Phase 1: Feature Selection

- Choose which features will, if implemented, maximize functionality
- Combinatorial algorithm: attempt collections of features from disparate classes

Phase 2: Component Selection

- Choose software components that make up the features selected in Phase 1
- Dependencies among s/w components are communicated via PFA graph
- Find paths through PFA graph that include feature nodes

Phase 3: Component Allocation

- A Bin-Packing Algorithm -- well trodden research area. Can the software components be made to fit within the PE/Network resource constraints?
- Novel approach for embedded systems: Transducer-sensitive algorithm implemented and tested. 2.7x speedup realized

Output: Component ⇔ PE Mapping

Where will this research go from here?

CHALLENGES:

- Determine proper algorithm parameters for each phase
- Must exploit information from failure of one phase to better choose inputs for re-try
  - Phase 3 is NP-complete -- must not simply wrap loop constructs around it

GOALS:

- Proof of concept -- use robust elevator simulators developed for 18-849
- Determine tuning parameters for:
  - Design-time vs. Run-time use
  - Pre-positioned fallback configuration

GM

ISRI

BOSCH

Institute for Complex Engineered Systems

Electrical & Computer ENGINEERING

Carnegie Mellon