Progress in Robust Embedded System Architectures

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

Prof. Philip Koopman & Prof. Priya Narasimhan
Bill Nace – Charles Shelton – Chris Martin – Beth Latronico – Tridib Chakravarty – Yang Wang
Outline

- RoSES Strategic Vision
  - Feasibility assessment
  - Key technical research areas
  - Technology transition to GM

- Demo – Chris Martin
  - Including “workarounds” as a form of dependability
Generic RoSES System Architecture

SMART SENSORS

Basic S/A Device
Baseline Sensor SW Functionality
SW Adapter for High Level Logical Interface
Dynamic Interface to Object Bus

Local CPU & Memory

Smart Sensors

Dynamic Interface to Object Bus

State Variables on Real-Time Embedded Network

RECONFIGURATION MANAGER

Adapter Repository

SMART ACTUATORS

Basic S/A Device
Baseline Sensor SW Functionality
SW Adapter for High Level Logical Interface
Dynamic Interface to Object Bus

Local CPU & Memory

Smart Actuators

Dynamic Interface to Object Bus

State Variables on Real-Time Embedded Network

RECONFIGURATION MANAGER

Adapter Repository

Co-Scheduling & Assignment Tool

3
RoSES Strategic Vision:

◆ Goal:

Develop theory, techniques, & key tools for robust distributed embedded systems

◆ Grand Hypothesis:

Graceful degradation will provide cost-effective dependability

◆ Approach:

• Understand problem & demonstrate feasibility
  – Prototypes for key points to explore issues

• Resolve key research issues
  – Structure approach to spin off capabilities over time

• Transition knowledge to industry
  – Work with GM Software Architecture team for mutual benefit
Overview: Problem Understanding

- Run-time infrastructure
  - Why can’t we just buy this stuff?

- Configuration management
  - Is this really just a known software partitioning problem?

- Architectural definition & patterns
  - Getting past having to ignore the man behind the curtain
Run Time Infrastructure

- **Why can’t we just buy one?** (Meredith Beveridge)
  - Many are just paper – look at real tools
  - Corba is too “fat”
  - Jini looked attractive …
    - and sort of worked …
    - but had significant shortcomings

- **Getting something that will really work** (Yang Wang)
  - Key requirements based on Jini and other experiences
  - What can we learn from other research middleware?
  - How compatible can we be with desktop middleware?
    - Differ where it is important to do so
    - Remain compatible wherever possible
  - Support key needs for graceful degradation
    
(work starting Spring 2002)
How do we track fine-grain distributed components?

(Bill Nace)

- Which software component goes where in the system?
- Given a fixed set of hardware, optimize system functionality
  - In general, not all possible software will fit on hardware
  - Various feature classes contain overlapping functionality
- Progress
  - Good heuristics for quick solution
  - Representation & method successful on pilot problem
  - Working on a larger problem
Architectural Definition & Patterns

- **Robust architectural patterns** (Charles Shelton)
  - Are there generic approaches to attain robustness?
  - Can we evaluate “robustness”? 

  • Progress:
    - Using realistic elevator example to demonstrate methodology
    - First results for quantifying robustness

  • Plan: work with GM architecture team
Overview: Resolve Key Research Issues

- **Project focus areas:**
  - Can we use UML or do we have to invent something?
  - Embedded to people interface
  - Embedded to enterprise interface

- **Long-term items:**
  - Formal representation & quantification
  - Appropriate robustness approaches
  - NP-hard issues in specification & evaluation
Can UML handle real embedded systems?

- Spring 2001: class to build realistic systems
- Uncovered several problems; several solutions invented

- Compiler theory helps with stitching scenarios  
  (Beth Latronico)
- Statechart clustering helps with global modes  
  (Elissa Newman)
- SW architecture different than for desktops  
  (Owen Cheng)

State transition diagrams:

1. **Radio 1**
   - User → Button → Radio
   - \( U_{press} \rightarrow B_{press} \) \([\text{Time of } B_{release} - \text{Time of } B_{press} < 2 \text{ seconds}]\)
   - \( U_{release} \rightarrow B_{release} \) \(\text{change station}\)

2. **Radio 2**
   - User → Button → Radio
   - \( U_{press} \rightarrow B_{press} \) \([\text{Current Time} - \text{Time of } B_{press} >= 2 \text{ seconds}]\)
   - \( B_{release} \rightarrow \text{station set} \)

**SD → message duration response SD | \( \varepsilon \)**

**message duration response \( \alpha \) B_{release} change_station | \( \beta \) station_set**
Embedded To People Interface

- **People can help with robustness(!) (Chris Martin)**
  - Concept of “workaround” is important, but neglected
  - Minor user flexibility can improve system-level robustness

- Most real systems have several ways to accomplish goals
- They can be represented as paths through UML scenarios
- Min-cut graph algorithm can expose robustness bottlenecks
- Elevator system results demonstrate feasibility

Proposed Workaround:

- **Hallway**
  - Push button in desired direction
- **Wait**
  - Push button in opposite direction
What happens when Embedded meets Enterprise?

From Jini experience we know to expect incompatibilities

- Event-driven vs. periodic
- Transactional vs. continuous control
- Rollback/retry vs. maintaining control stability
Embedded To Enterprise Interface

What happens when Embedded meets Enterprise?

(Priya Narasimhan & Phil Koopman)

From Jini experience we know to expect incompatibilities

- Event-driven vs. periodic
- Transactional vs. continuous control
- Rollback/retry vs. maintaining control stability

Class in Spring 2002 to build one and see what happens
Formal Representation & Quantification

◆ What is system architecture?  (Shelton)
  • Multiple viewpoints onto a single system
    – Hardware + software + communications + control
    – Human interface + upgrades + safety/security + validation + run-time infrastructure + fault management + …
  • Patterns for different architectural styles
    – General tradeoffs inherent to each style

◆ Can there really be a “safety architecture”?  (Latronico)

◆ What is graceful degradation?  (everyone)
  • For that matter, in a partially disabled system, what does “working” mean?
  • Perhaps it is related to vulnerability to mission failure (Martin)
Appropriate Robustness Approaches

- Can we characterize the robustness tradeoff space?
  - Brute force replication
    - Expensive – many more components in system
    - Not entirely effective for software
  - Failover modes
    - Design intensive, but known to work
    - Can we create more systematic ways to do this?
  - Reconfiguration (current emphasis)
    - Can work together with product family configuration management (Nace)
    - Whether it is even feasible is a research topic (yes, so far)
  - Heterogeneous redundancy
    - If two sensors/actuators are almost the same, can they be interchanged?
    - Few existing techniques, although analytic redundancy fits here
    - People can use systems differently (people are “system components” too) (Martin)
NP-Hard Issues In Specification & Evaluation

Many hard problems encountered as we go

- Allocating software to components (Nace)

- System specification
  - Product family architecture specification (Shelton)
  - Specification of utility for different features & feature sets

- Evaluation
  - When is a system really “working” when it is partially disabled? (Martin)
  - Safety/certification of component-based systems (Latronico)

- Implementation
  - Software runtime infrastructure (Wang)
  - Real time scheduling for distributed networked system
  - Security of embedded+enterprise combined system
  - What baseline set of components gives most reconfiguration flexibility?

...
Overview: Transition Knowledge To Industry

◆ Work with GM architecture team
  • Trips both ways
  • Students create representative vehicle subsets for research
  • GM benefits from experience gained in RoSES implementation

◆ Teaching
  • Stream of CMU grads. trained in robust embedded system design
    – Soon to include robust enterprise systems as well
  • Opportunity for GM-based course projects
    – 6-12 months advanced planning required
    – Topic area must be carefully selected
Related Work – Embedded Protocols

- **CRC error detection effectiveness** (Chakravarty)
  - Train Communication Protocol design review
  - Found that error codes could be much more effective
    - Error codes optimized for long messages
    - But embedded networks have short messages – different design tradeoff point

- **FlexRay & TTP protocols** (Koopman)
  - Were already being evaluated for another customer
  - Expertise available when GM joined FlexRay consortium
RoSES Publications In 2001

2001 Workshop on Reliability in Embedded Systems
- Nace – Component allocation framework
- Shelton – Architectural principles
- Martin & Latronico – User workarounds

2001 UML Conference
- Latronico – sequence diagrams as a formal language

IBM Ubiquitous Computing Workshop
- Nace – Internet meets embedded systems (invited)

Theses:
- Beveridge – Jini meets CAN (also invited paper at WORDS 2002)
- Martin – User workarounds + graph analysis
- Chakravarty – Optimal embedded network error detection
Conclusions

- **Results coming in on understanding the problem area**
  - Run-time infrastructure
  - Configuration management (PhD thesis next year)
  - Architectural definition & patterns (PhD thesis in about 2 years)

- **Progress on key technology areas**
  - UML isn’t dead (yet) – but will require augmentation
  - Embedded to people interface is an emergent opportunity
  - Embedded to enterprise interface (Spring 2002 course)

- **Pieces of long-term issues being solved as we go**
  - Formal representation & quantification
  - Appropriate robustness approaches
  - NP-hard issues in specification & evaluation

- **Participation with GM SW architecture team**