Identifying and Addressing Uncertainty in Architecture-Level Software Reliability Modeling

Leslie Cheung
Leana Golubchik
Nenad Medvidovic
Gaurav Sukhatme

University of Southern California

{lcheung, leana, neno, gaurav}@usc.edu
Software Architecture-level Reliability Modeling

• Assessing reliability of software early is desirable
  – Fixing major problems discovered late in development is too costly

• Doing so at the level of software architecture would be preferable
  – Architecture is a linchpin of software system development
  – A set of abstractions, notations, techniques, and tools for developing large, complex software-intensive systems

• Challenge: Uncertainty
  – e.g., How do you know the runtime behavior of the system before it is implemented?
  – Needed information includes
    ➢ Operational profile
    ➢ Failure characteristics
Source of Uncertainty

- Components’ reliabilities
  - Existing approaches assume these are known
- Development scenario
  - Develop a system from scratch vs. from existing components
- Needed information about a system
  - Domain expertise
  - Software system requirements
  - Simulated architectural model
  - Functionally similar system
- Granularity of architectural models
  - Coarse-gained vs. detailed models of components
- Reliability modeling techniques
  - Different techniques are effective in different situations
Architecture-Level Reliability Prediction Framework

- Specifically targeted at the level of individual components
Architecture-Level Reliability Prediction Framework
Framework in Action

- Goal: Leverage dynamic behavior of a component
- Often available in an architectural model

Diagram:
- Initializing
- Estimating Sensor Data
- Updating
- Turning
- Going Straight
- Reset / Initialize
- Update DB
- Get Sensor Data
- Go Straight
- Update DB
- Mission Parameters
- Get Sensor Data
- Obstacle Ahead / Turn Left
- Too Close / Turn Left
- Too Far / Turn Right

Activities:
- B1
- B2
- B3
- B4
- B5
Framework in Action

- And build a stochastic reliability model
Architecture-Level Reliability Prediction Framework

1. **Phase 1**
   - Determining States
   - States of Reliability Model

2. **Phase 2**
   - Determining Transitions
   - Reliability Model

3. **Phase 3**
   - Computing Reliability

**Granularity of Architectural Models**

- Architectural Models
- Information Sources
- Reliability Modeling Techniques

**Comp Reliability**
Leveraging the Information Sources

• Little or no information
  – Explore the design space

• Domain knowledge
  – Use operational profiles suggested by expert(s)
  – Beware of expert inaccuracies (or worse)!

• Requirements documents
  – Contain typical use cases of a component

• Simulation of architectural models
  – Capable of handling complex state spaces

• Functionally similar component
  – Runtime behavior *might* be similar to the component of interest
  – Similarities may be misleading
Architecture-Level Reliability Prediction Framework

- **Phase 1**: Determining States
  - Architectural Models
  - Granularity of Architectural Models

- **Phase 2**: Determining Transitions
  - States of Reliability Model
  - Information Sources

- **Phase 3**: Computing Reliability
  - Reliability Modeling Techniques
  - Comp Reliability
Techniques for Modeling Reliability

- **Discrete-Time Markov Chains**
- **Hidden Markov Models**
  - **Input:** Operational and failure profile from above sources
  - **Process:** Standard approaches to solve the model
  - **Output:** Transition probabilities
- **Bayesian Networks**
  - We have been exploring them for system-level reliability prediction
- ...

...
Evaluation Strategy

• Study the framework in many representative scenarios
• Establish the framework’s predictive power vs. implementation-level reliability estimation techniques
• Evaluate the framework’s sensitivity to changes in different parameters
  – Architectural model and its granularity
  – Source of operational profile and failure characteristics
  – Reliability modeling technique
Example Software Component at Multiple Granularities
Example Software Component at Multiple Granularities
Sensitivity to Architectural Model Granularity – DeSi
• Only one defect is present (Turn defect)
• Each curve corresponds to different failure probabilities
• Vary recovery probabilities from 0.2 to 1 (at 0.2 intervals)
• Expert can be wrong!
Sensitivity to Failure Probabilities – DeSi

- Considering *Missing Model Validation Rules* defect
- Again, each curve corresponds to different failure probabilities
- Vary recovery probabilities from 0.2 to 1 (at 0.2 intervals)
- Expert was relatively close, but functionally similar component was not (close or similar)!
- Simulated model was imprecise
Sensitivity to Operational Profile Estimation – DeSi
Conclusions and Current Directions

• One focus to date has been component reliability prediction at the architectural level
  – Uncertainty is a major challenge

• We explored different information sources available at the architectural level

• Current Directions
  – Predicting system reliability at the architectural level
    • Scalability – how to model a system with many components in a scalable way?
  – Firmware modeling
    • Operating system, device drivers, middleware, etc. also impact a software system’s reliability