Software Reliability Modeling: a Comparative Discussion

Jorge Luis Romeu
IIT Research Institute
http://web.cortland.edu/romeu
Invited Paper Presented to the INFORMS FALL 1999 Meeting
Outline

• Model Origins (hardware and software)
• The Dichotomy Problem
• Dichotomy Contradictions
• Software Model Assumptions
• Validity and Applicability
• Proposed Working Strategy
• Summary and Conclusions
Model Origins

- What is Reliability? HW/SW? For whom?
- HW Models: developed/understood (50s)
- Developers: Systems/Electrical Engineers
- SW Modeling Needs: increasing in 70s
- Similar Development Personnel
- Similar Approaches and Techniques
- Natural to Adapt/Extend HW Models
Types of Reliability Models

• I) Structural (theoretical)
• III) Black Box (empirical): time domain (time between failures and failure count); time series; fault seeding/tagging;
• IV) other (simulation, FMEA, etc.)
HW/SW Models

- HARDWARE:
  - Empirical (Duane)
  - NHPP Models (Crow, AMSAA)
  - Parts-based models (MIL-HDBK-217)
  - Other (Simulation, time series, etc.)

- SOFTWARE:
  - Empirical (runs)
  - Piecewise PP/NHPP (JM, SW, Musa, Goel)
  - SW Science/CycComp (Halstead, McCabe)
  - Other (Prop. Hazards, time series, etc.)
Dichotomy Problem Categories:

- **PEOPLE**: personnel involved in model development or in model use activities
- **PRODUCT**: outcomes for which the software people are formally evaluated on
- **PROCESS**: methods and materials used to develop their intended products
- **RESULTS**: perceived or real assessment stemming from their activity products
The Dichotomy Contradictions:

- **MODELERS:**
  - Academicians/Resear.
  - Dissertations and publications
  - Markov, Distribution Theories
  - Satisfied: Ph.D.s, tenure, promotions

- **USERS:**
  - Practitioners/Devel.
  - Releases, schedules, cost/staff estimates
  - Programmers, small and deficient info.
  - Unsatisfied: Small Improvements.
Practitioner’s Needs

- Approaches that are feasible and practical
- FEASIBLE: implemented without incurring exorbitant costs, excessive work burden or unrealistic data collection activities.
- PRACTICAL: provide tools for scheduling, costing, staffing, testing, demonstrating and establishing product release time
Some SW R Model Assumptions

• Definition and criticality of a failure
• Time Units (execution, calendar time, etc.)
• Fixed No. faults (N); no new/same faults
• All faults are equally likely (same order?)
• All SW faults are always exposed
• Faults are immediately removed
• Only one failure at a time (Poisson model)
More SW R Assumptions (2)

- Software testing is homogeneous
- Failure rate proportional to error content
- Time between failures is independent
- No. failure in disjoint intervals, independent
- Testing proceeds only after fault is removed
- All the code is tested, all the time
- Run time versus thinking (desk) time
Assumptions (3) & Problems

• Bayesians assume a prior distribution
• Development phases and fault exposure
• Seeded faults; same place/impact as real
• Input dom/user profile difficult/dependent
• Failure data is weak (collection secondary)
• Reliability increases with time (growth)
• Software doesn’t wear out (time indep.) but maintenance increases complexity...
Some Additional Problems

• Experim. probls/subjects not representative
• Problem requirements, SW environment, documentation, user profile and experience not accounted for factors in the models.
• Multiple models; choice is another problem
• Initial Estim. = f(P_speed, lang., expos_rate)
• Fault exposure ratio obtained (when, where)
• L_Expansion ratios obtained (when, where)
• Environment/methodology Learning Curve
Proposed Modeling Approach

• Models assess the end result (SW product)
• Results = f (people, project, environment)
  1) Improve People (training, organization)
  2) Understand Environment (shop): forensic analysis for strong/weak pts, parameters
  3) Strengthen Projects: rqmts/specs definitions, error prevention versus error correction, staff/time/stress management considerations
  4) Then measure using SW_R models
Improve People/Process:

- Fault Avoidance/Prevention: CMM practices, requirements/specifications definitions
- Fault Tolerance/Redundancy: recovery blocks; n-version programming, voters
- Fault Detection/Correction: clean-room, walkthroughs, configuration management, module/integration and system testing, etc.
- Reused, adaptive, variant code precautionary practices, FMECA/TQM procedures
Example: CMM

- Capability Maturity Model (CMM/SEI)
- Focuses on SW devel. process/practices
- L1 - Basic practices performed informally
- L2 - Practices are Planned and Trackted
- L3 - Standardized, Well Defined Practices
- L4 - Quantitatively Controlled Practices
- L5 - Continuous Improving/Quant. Process
Summary

- SW RELIABILITY MODELS IMPROVED:
  - software estimation, predictions, assessment
  - understanding of the development process
  - many/new software development techniques

- BUT MODELS ARE BASED UPON:
  - poor (few, weak measurement scale) data
  - weak (invalid, unrealistic) assumptions
  - incomplete (do not consider all) factors
Conclusions

• Don’t ask unrealistic results from models
• Work with software model developers
• Provide incentive for adapting models
• Assess your organization (strength/weak)
• Improve it (programming/processes)
• Error prevention rather than correction
• THEN, use one model, judiciously
Final Conclusion: