

An Adaptive Software Reliability Prediction Approach

Meng-Lai Yin * Lawrence E. James Samuel Keene
Rafael R. Arellano Jon Peterson

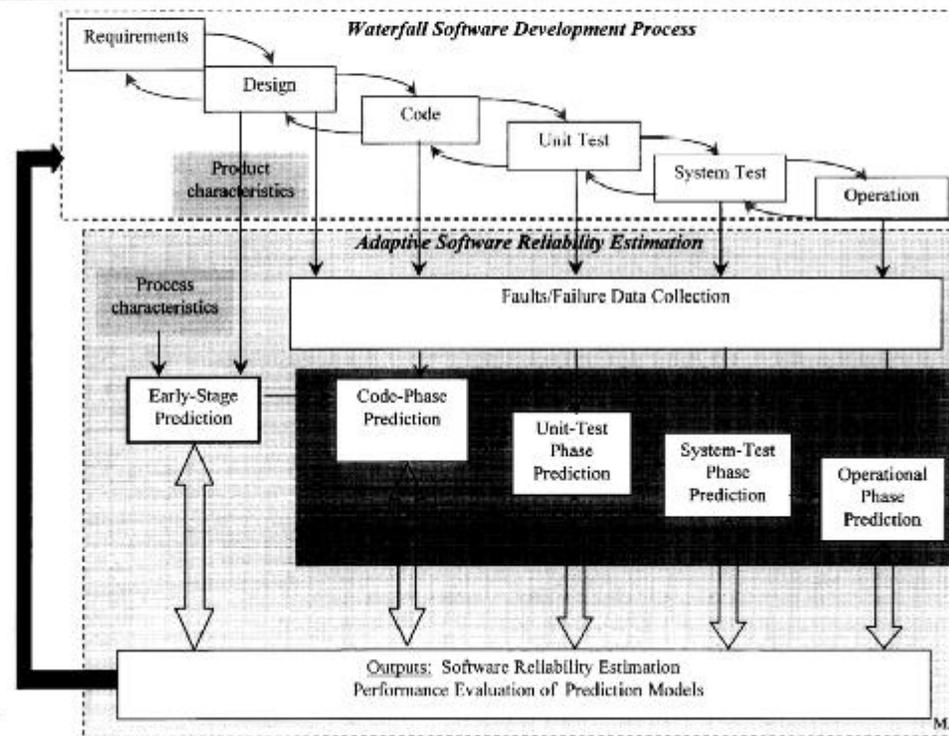
Software Reliability Prediction Approaches

- No “standard” model to be used
- Blind approach
- Autopsy approach
- Adaptive approach
 - analyze software reliability adaptively and progressively

Adaptive Approach for SW Rel. Prediction

- Software reliability is modeled as the software development proceeds
- The prediction can be refined and justified progressively
- Continuously provide estimation based on the most current failure information
- Prediction at the beginning of software development, when no failure data are available, is also necessary (early-stage prediction)
- This process has been implemented in a software development program in progress

The Process



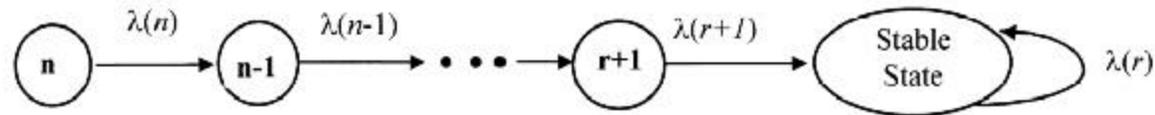
Tools Consideration

- **SWEEP** (SoftWare Error Estimation Program)
 - phase-based model
- **SMERF** (Statistical Modeling and Estimation of Reliability Functions)
 - various growth models
- **CASRE** (Computer-Aided Software Reliability Estimation)
 - various growth models

Early-Stage Prediction

- Provide rough estimation on various software reliability measurements, based on **limited information** (Accuracy is the goal of the overall adaptive process, not the main concern of the early-stage prediction)
- Information needed are
 - the size of the code
 - the software process maturity level
 - the schedule of software development

The Early-Stage Prediction Model



n : the number of inherent faults

r : the number of remaining faults

Stable State: the state where the software failure rate remain the same (asymptotic property)

$\lambda(i)$: failure rate function

Calculation Process

1. Determine n , r , and T

n = size of code * inherent fault density

r = n * remaining faults percentage

T is from the beginning of operational phase to
the time the software reaches stable state

2. Specify failure rate function

3. Solve λ from $T = 1/\lambda(n) + 1/\lambda(n-1) + \dots + 1/\lambda(r+1)$

4. Various measurements can be obtained from the result

Example 1. Fault Densities for different Software life-cycle Phases

Phase	Faults/ KSLOC
Coding	99.5
Unit Test	19.7
System Test	6.01
Operation	1.48

(copied from Musa[8], Table 5.4)

Example 2. Fault Densities for different SEI CMM Level

SEI CMM Level	Faults/ KSLOC (at the beginning of operation phase)
5	0.5
4	1.0
3	2.0
2	3.0
1	5.0
Un-rated	6.0

(copied from keene[5])

Example

- 360 KSLOC
- SEI CMM Level 4 Process
- 4 years to reach the stable state after software deployment
- 1.48 faults/KSLOC (533 faults) for phase-based inherent fault density prediction
- 1.0 faults/KSLOC (360 faults) for process-maturity-level-based prediction

Summary

- The adaptive approach and the early-stage prediction method have been implemented in an on-going software development program and provided adequate prediction
- As more experience is gained, the process and the method will be improved, and will benefit other software development products.