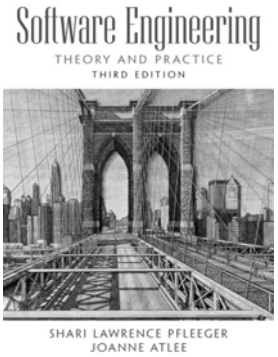
Chapter 13

Improving Predictions, Products, Processes, and Resources



ISBN 0-13-146913-4
Prentice-Hall, 2006

Contents

Chapter 13 Objectives

- Improving predictions
- Improving products by using reuse and inspections
- Improving processes by using cleanroom and maturity models
- Improving resources by investigating trade-offs

13.1 Improving Prediction

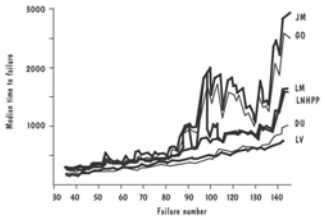
- Need to have the predicted value be close to the actual value
- Need to understand ways to improve the prediction process
  - Reliability models and techniques

13.1 Improving Prediction
Reliability Models

- The Jelinski-Moranda model (JM)
- The Goel-Okumoto model (GO)
- The Littlewood model (LM)
- Littlewood's nonhomogenous Poisson process model (LNHPP)
- The Duane model (DU)
- The Littlewood-Verrall model (LV)

13.1 Improving Prediction
Reliability Models Comparison

- Each model applied on the same dataset (the Musa dataset)
- Each model was used to generate 100 successive reliability estimates



13.1 Improving Prediction
Predictive Accuracy

- Predictions are biased when they are consistently different from the actual value
- Predictions are noisy when successive predictions fluctuate more wildly than the actual value

13.1 Improving Prediction
Dealing with Bias

- Compare how often the observed times of failure are less than the predicted ones
- When a given model predicts that the next failure will occur at a particular time
  - Record interfailure times; t_1 to t_n
  - Compare the observed time with predicted time (T_1 trough T_n)
  - Count the number of times that t_i is less than T_i
  - If the number is less than $n/2$, we have bias in our prediction
- *U*–plots can help us understand and reduce bias

13.1 Improving Prediction
The *U*–Plot: Steps

- Formally expressing bias by forming a sequence of numbers $\{u_i\}$
  - u_i is an estimate of the probability that t_i is less than T_i
- Calculating a distribution function for this data sequence, from which we calculate the u values
- Constructing a graph called a u–plot

13.1 Improving Prediction
The *U*–Plot: Generating U_i Values

- Based on the Musa data

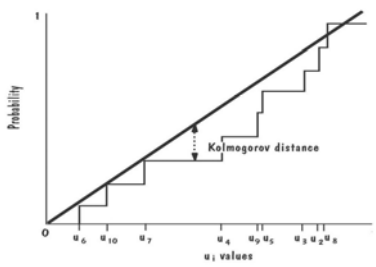
i	t_i	Predicted Mean Time to ith failure	u_i
1	3		
2	30	16.5	0.84
3	113	71.5	0.79
4	81	97	0.57
5	115	98	0.69
6	9	62	0.14
7	2	5.5	0.30
8	91	46.5	0.86
9	112	101.5	0.67
10	15	63.5	0.21

13.1 Improving Prediction
The *U*–Plot: Constructing The Graph

- Placing the u_i values along the horizontal axis
- Drawing a step function, where each step has height $1/(n+1)$
- Drawing the line with slope 1
- Comparing the line with the u–plot
  - The difference represents the deviation between prediction and actual
  - The degree of deviation: Kolmogorov distance

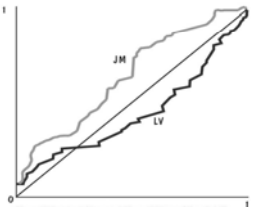
13.1 Improving Prediction
The *U*–Plot

- Based on u_i values from Musa data



13.1 Improving Prediction
The *U*-Plot Example

- Jelinski-Moranda and Littlewood-Verrall Models, the Kolmogorov distance
  - JM = 0.190, significant at 1% level
  - LV = 0.144, significant at 5% level



13.1 Improving Prediction
Dealing with Noise

- The estimate values are very far from the actual values, and fluctuate wildly
  - A lot of noise in the prediction
- **Unwarranted noise**: actual reliability is not fluctuating, but the estimates are
- Prequential likelihood helps reduce noise

13.1 Improving Prediction
Prequential Likelihood

- Allows us to compare the predictions from two models
  - Help to choose the most accurate model

13.1 Improving Prediction
Prequential Likelihood Calculation

i	t_i	T_i	Prequential Likelihood
3	113	16.5	6.43E-05
4	81	71.5	2.9E-07
5	115	97	9.13E-10
6	9	98	8.5E-12
7	2	62	1.33E-13
8	91	5.5	1.57E-21
9	112	46.5	3.04E-24
10	15	101.5	2.59E-26
11	138	63.5	4.64E-29
12	50	76.5	3.15E-31
13	77	94	1.48E-33

13.1 Improving Prediction
Prequential Likelihood Comparing Two Models

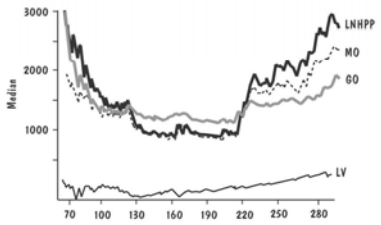
n	Prequential Likelihood LNHPP:JM
10	1.28
20	2.21
30	2.54
40	4.55
50	2.14
60	4.15
70	66.0
80	1516
90	8647
100	6727

13.1 Improving Prediction
Recalibrating Prediction

- Models behave differently on different datasets
- Results are different even on the same dataset
- Recalibrating is the way to deal with overall inaccuracy

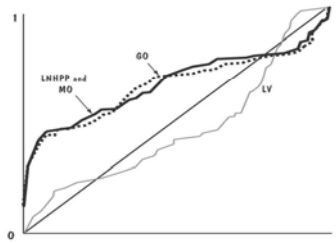
13.1 Improving Prediction
Recalibrating Prediction Example

- Reliability prediction of several models, using data from Musa SS3 data



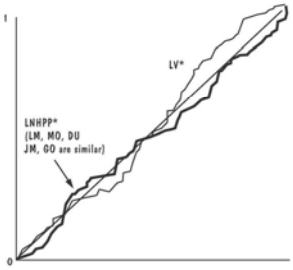
13.1 Improving Prediction
Recalibrating Prediction Example (continued)

- *U*–plots of models using data from Musa SS3 data



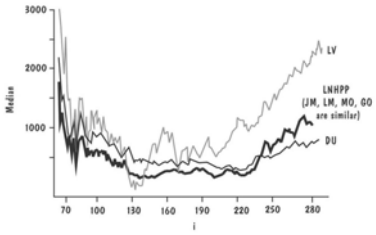
13.1 Improving Prediction
Recalibrating Prediction Example (continued)

- *U*–plots for recalibrated models of Musa SS3 data



13.1 Improving Prediction
Recalibrating Prediction Example (continued)

- Prediction of recalibrated models using data from Musa SS3 data



13.1 Improving Prediction
Benefits of Recalibrating

- Models in closer agreement than before
- New models with less bias than original ones

13.2 Improving Products

- Two product improvement strategies
  - Inspections
  - Reuse

13.2 Improving Products
Inspections Metrics

- A set of nine measurements
  - generated by business needs
  - aimed at planning, monitoring, controlling, and improving inspections
- Tell
  - whether the code quality is increasing as a result of inspections
  - how effective that staff is at preparing and inspecting code

13.2 Improving Products
Code Inspections Statistic from AT&T

Measurements	First Sample Project	Second Sample Project
Number of inspections in sample	27	55
Total thousands of lines of code inspected	9.3	22.5
Average lines of code inspected (module size)	343	409
Average preparation rate (lines of code per hour)	194	121.9
Average inspection rate (lines of code per hour)	172	154.8
Total faults detected (observed and nonobserved) per thousands of lines of code	106	89.7
Percentage of reinspections	11	0.5

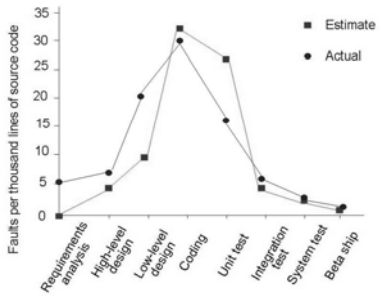
13.2 Improving Products
Sidebar 13.1 Monitoring Fault Injection and Detection

- Techniques for monitoring faults and measuring inspection effectiveness
  - Creating a fault database
  - Track activities when the fault was injected into product
  - Calculate the yield of several review activities

13.2 Improving Products
Yield Calculation

Activity	Faults Injected						
	Fault Found	Design Inspection	Code	Code inspection	Compile	Test	Post-development
Planning	0	2	2	2	2	2	2
Detailed design	0	2	4	5	5	6	6
Design inspection	4						
Code	2			2	7	10	12
Code inspection	3						
Compile	5						
Test	4						
Post development	2						
TOTAL	20						
Design inspection yield		4/4=100%	4/6= 67%	4/7= 57.1%	4/7= 57.1%	4/8= 50%	4/8=50%
Code inspection yield				3/5=60%	3/10= 30%	3/14= 25.5%	3/16=18.8%
Total yield		4/4=100%	6/6= 100%	9/9= 100%	9/14= 64.3%	9/16= 56.3%	9/20=45%

13.2 Improving Products
Projected vs. Actual Faults Found During Inspection and Testing



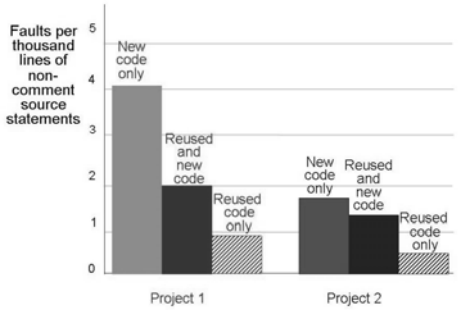
13.2 Improving Products
Fault Density

- When fault density is lower than expected
  - The inspections are not detecting all the faults they should
  - The design lacks sufficient content
  - The project is smaller than planned
  - Quality is better than expected
- If the fault density is higher than expected
  - The product is larger than planned
  - The inspections are doing a good job of detecting fault
  - The product quality is low

13.2 Improving Products
Reuse

- At HP, Lim (1994) shows how reuse improves quality
  - Two case studies to determine whether reuse actually reduces fault density
- Moller and Paulish (1993) investigated the relationship involving fault density and reuse at Siemens
  - Be careful how much code we modify

13.2 Improving Products
Fault Density of New Code vs. Reused Code



13.3 Improving Processes

- Process and capability maturity
- Prototyping and Cleanroom
  - Reduce maintenance time

13.3 Improving Processes
Process and Capability Maturity

- CMM
- ISO 9000
- SPICE

13.3 Improving Processes
Drawbacks of Process and Capability Maturity

- Process maturity questionnaires only capture a small number of the characteristics of good software practice
- Process maturity model assumes a manufacturing paradigm for software
- Process maturity approach does not dig deep enough into how software development practices are implemented

13.3 Improving Processes
Benefits of Process and Capability Maturity

- Aggregate results from the SEI benefit study

Category	Range	Median
Total yearly cost of software process improvement activities	$49,000 to $1,202,000	$245,000
Years engaged in software process improvement	1 to 9	3.5
Cost of software process improvement per engineer	$490–$2,004	$1,375
Productivity gain per year	9–67%	35%
Early detection gain per year (faults discovered pretest)	6–25%	22%
Yearly reduction in time to market	15–23%	19%
Yearly reduction in postrelease fault reports	10–94%	39%
Business value of investment in software process improvement (value returned on each dollar invested	4.0 to 8.8	5.0

13.3 Improving Processes
Sidebar 13.2 Process Maturity and Increased Visibility

- The lowest level of visibility (akin to CMM Level 1): the requirements are ill-defined
- The next higher level (similar to CMM level 2): the requirements are well-defined, but process activities are not
- Higher level still (much like CMM level 3), the process activities are clearly differentiated

13.3 Improving Processes
Maintenance

- Key questions in selecting maintenance estimation techniques
  - How can we quantitatively assess the maintenance process?
  - How can we use that assessment to improve the maintenance process?
  - How do we quantitatively evaluate the effectiveness of any process improvements?

13.3 Improving Processes
Maintenance (continued)

- Lesson learned from maintenance process when evaluating improvement
  - Use statistical techniques with care
  - In some cases, process improvement must be very dramatic if the quantitative effects are to show up in the statistical results
  - Process improvement affects linear regression results in different ways

13.3 Improving Processes
Sidebar 13.3 Is Capability Maturity Holding NASA Back?

- NASA's space shuttle was built and is maintained by a CMM level 5 organization
- Software is driven primarily by tables
  - Before each launch, tables must be updated; which is costly and time consuming
- Major change in the development process, in part to overhaul the table-based approach and make the system more flexible, may result in a process that receives a lower CMM rating

13.3 Improving Processes
Sidebar 13.4 Comparing Several Maintenance Estimation Techniques

- Inductive logic programming models were more accurate than
  - top-down induction trees
  - top-down induction attribute value rules
  - covering algorithms

13.3 Improving Processes
Organization of Cleanroom Studies

- Controlled experiment comparing reading with testing
- Controlled experiment comparing cleanroom with cleanroom-plus-testing
- Case study of cleanroom on 3-person development team and 2-person test team
- Case study on 4-person development team and 2-person test team
- Case study on 14-person development team and 4-person test team

13.3 Improving Processes
Results of Reading vs. Testing Experiment 1

	Reading	Functional Testing	Structural Testing
Mean number of faults detected	5.1	4.5	3.3
Number of faults detected per hour of use of technique	3.3	1.8	1.8

13.3 Improving Processes
Second Experiment Findings

- Cleanroom developers were more effective at doing offline reading
- Cleanroom-plus-testing focused more on functional testing than on reading
- Cleanroom teams spent less time online and were more likely to meet their deadlines
- Cleanroom products were less complex, had more global data, and had more comments
- Cleanroom products met the system requirements more completely, and they had a higher percentage of successful independent test cases
- Cleanroom developers did not apply the formal methods very rigorously
- Almost all Cleanroom participants were willing to use cleanroom again on another development project

13.3 Improving Processes
Results of SEL Case Studies

	Baseline Value	Cleanroom Development	Traditional Development
Lines of code per day	26	26	20
Changes per thousand lines of code	20.1	5.4	13.7
Faults per thousand lines of code	7.0	3.3	6.0

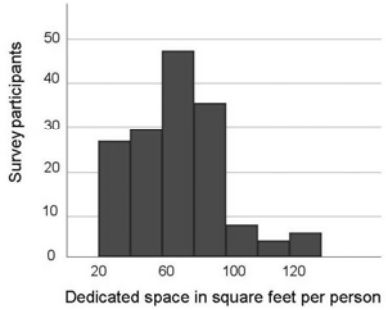
13.4 Improving Resources

- Some resources are fixed, leaving no room for improvement
- Other resources are highly variable
  - Human resources

13.4 Improving Resources
Work Environment

- Giving people the environment they need to do a good job
  - acceptable work space
  - tolerable noisy and quiet office
- Considering the team size and communication path
- Emphasizing the importance of team "jell," where team members work smoothly, coordinating their work and respecting each other's abilities

13.4 Improving Resources
Work Space for Developers Survey

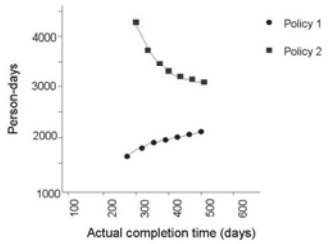


13.4 Improving Resources
Sidebar 13.5 Viewing Users as A Resource

- Reasons for the success of SSNS (Sale Service Negotiation System) at Bell Atlantic
  - its developers' use of users as a resources
  - performance issues were addressed by having the user work side by side with the software engineers

13.4 Improving Resources
Cost and Schedule Trade-offs

- Trade-off between person-days and schedule for two management policies



13.5 General Improvement Guidelines

- Are the goals the same?
- Are the priorities of the goals the same?
- Are the questions the same?
- Are the measurements the same?
- Is the maturity the same?
- Is the process the same?
- Is the audience the same?

13.6 Information System Example
Piccadilly System

- Improvement strategies that Piccadilly maintainers should follow
  - Perform perfective maintenance
  - Examine other similar software systems at Piccadilly

13.7 Real-Time Example
Ariane-5

- Several improvements have been suggested
  - The team should perform a thorough requirements review
  - The team should do ground testing
  - The guidance system's precision should be demonstrated by analysis and computer simulation
  - Reviews should become a part of the design and qualification process

13.8 What This Chapter Means for You

- Prediction can be improved by
  - using *u*-plot
  - prequential likelihood
  - recalibration
- Products can be improved as part of a reuse program or by instituting an inspection process
- Process can be improved by evaluating their effects and determining relationships that lead to increased quality and productivity
- There is promise of improvement in resource allocation as we learn more about human variability and examine the trade-offs between effort and schedule