

Chicago Software Process Improvement Network

Building and Using a Defect Prediction Model

Laura Zawadzki, IT Quality Manager,

Motorola Mobility Inc.,

Tatiana Orlova, Quality Analyst, PMP,

Motorola Mobility Inc.



Common IT Program Issues:

- ➔ Programs are massive and complex
- ➔ Resources are constrained
- ➔ Schedules are extremely aggressive
- ➔ Testing was planned based on schedule constraints
- ➔ Actual testing activity was often reduced to meet schedule commitments
- ➔ Teams 'knew' it was probably not enough testing but 'felt' that meeting schedule commitment was worth sacrificing some project quality



➔ Incorporate Defect Prediction Model

- ➔ Assess testing effectiveness and predict the quantity of post release defects
- ➔ Enables quantitative decision about production go-live readiness based on:
 - ➔ current state of testing effectiveness and
 - ➔ prediction of the number of remaining undiscovered defects which will escape to production

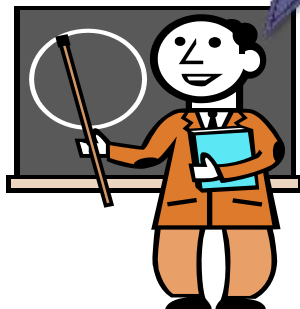


Defect Prediction Objectives

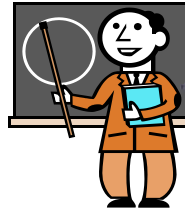
What number of defects would be considered reasonable versus signs of low quality?



To estimate the defect level for a project we need to consider:

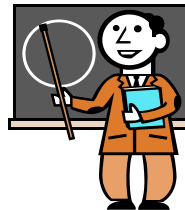


1) Project size and defect potential



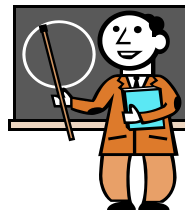
- ➔ Project size
- ➔ Number of defects
- ➔ Defect potential for each phase

2) How we prevent defects from occurring?



- ➔ Roles & Responsibilities
- ➔ Formalized Procedures
- ➔ Processes and Tools
- ➔ Controls & Measures

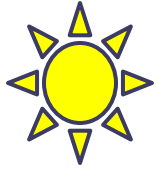
3) How we detect defects during the project?



- ➔ Static Techniques
- ➔ Dynamic Techniques



Defect Potentials



The term “defect potentials” refers to the total quantity of defects that will be found in five software artifacts: requirements, design, code, documents, and “bad fixes” or secondary defects.

U.S. average for defect potentials is about 5 defects per function points (defect potentials range is from 1.00 defect per function point (FP) to about 10.00 defects per function point): [1]

Defect Origin	Defects / Function Point
Requirements	1
Design	1.25
Coding	1.75
Documentation	0.6
Bad Fixes	0.4

Defect potentials correlate with size of Software, maturity level of the organization, level of expertise, system knowledge, etc..



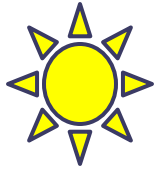
Estimating Project Size and Defect Levels

There was a strong relationship between number of Development requirements and number of total defects for the project observed. Analysis of data indicated that Development requirements times 1.84 estimated the number of total defects for the project.

- ➔ ***We will define Project Size in terms of Detailed requirements (DRQ) in our Model***
- ➔ ***We will calculate Number of defects based on the Number of Development Requirements:
Maximum Possible defects= DRQ * 1.84***



Historical Data Analysis



Development requirements are detailed-level requirements or technical specifications, written for the technical practitioner and business technical subject matter expert. Development requirements include all functional, performance and setup requirements.

- ➔ Historical information was collected from IT projects in our organization for a period of four years.
- ➔ All projects were using standard defined processes for gathering, analyzing and documenting requirements.
- ➔ All projects were using standard templates for producing development requirements, i.e. documentation was consistent across projects .
- ➔ All projects were using standard defects prevention activities, such as requirement reviews, design reviews, code review and unit testing.
- ➔ All projects were using standard processes for detecting and removing defects, such as system and integration testing, regression and performance testing (when required) and acceptance testing.



Defect Removal Efficiency



Defect Removal Efficiency (DRE) measures the defects reported by phase as a percentage of the overall defects recorded across a project. Can be used to measure how effective a particular phase is at detecting and removing defects.

➔ **DRE Calculation:**

$$\text{DRE} = \frac{\text{No. of In process Defects}}{\text{Total No. of Defects (In Process + Post-release)}}$$

U.S. averages for defect removal efficiency against each of the five defect categories : [1]

Defect Origin	Defect Removal
Requirements defects	77%
Design defects	85%
Coding defects	95%
Documentation defects	80%
Bad Fixes	70%
Total	85%



Defect Removal Efficiency (cont.)

Quality control procedures such as testing and reviews (inspections) vary in their effectiveness at removing defects:

Quality Activity	Average Defect Removal Rate	Peak Defect Removal Rate
Requirements review	30%	50%
Design review	40%	65%
Personal review (design or code)	35%	60%
Code reviews or pair programming	50%	70%
Unit testing (automated or manual)	25%	50%
Functional testing	30%	45%
Regression testing	20%	30%
Performance testing	15%	25%
System testing	35%	50%
Acceptance testing	30%	45%

Achieving top removal efficiency requires a combination of formal inspections and formal testing. Testing alone is insufficient for optimal defect removal efficiency.



Defect Removal Efficiency (cont.)

If an organization has no defect prevention methods in place then they are totally reliant on defect removal efficiency. These numbers [4] were used as an “Industry standard” for Defect Removal Efficiency in our model:

Defect Removal activity by phase	Defect Removal Efficiency (DRE)
Requirements Review	15%
Design Review	30%
Code Review & UT	20%
Formal Testing	25%

Analysis of historical data showed that our organization has different DRE per phase. Most of the defects were found in Formal testing (70%). The model was adjusted to make DRE consistent with historical data DRE.

Defect Removal activity by phase	Defect Removal Efficiency (DRE)
Requirements Review	13%
Design Review	3%
Code Review & UT	4%
Formal Testing	70%
Escaped Defects (PRD)	10%
Total	100%

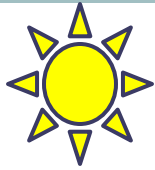


Defect Removal Efficiency (cont.)

- ➔ Best in Class organisations have a DRE of approximately 95% with peak of 99.5%. [1]
- ➔ Average organizations have a DRE of 85% [1]
- ➔ Historical DRE for past projects in our organization was calculated **as 90%**
- ➔ ***Several pilots of the model showed DRE of 94% for the program it was intended for***
- ➔ ***We will use the DRE of 94% for our further calculations***



Defect Prevention



Defect Prevention Effectiveness (DPE) is a measure of how effective an organisation's processes, procedures & controls are at preventing defects occurring in the first place.

The following processes allow to reduce number of defects created:

- **Roles and Responsibilities Clearly Defined -- up to 15% reduction**
- **Formalized Procedures -- up to 25% reduction**
- **Repeatable Processes -- up to 35% reduction**
- **Controls and Measures in place -- up to 30% reduction [4]**

➔ **DPE Calculation:**

$$\text{DPE} = \frac{\text{No. of defects removed by org. process}}{\text{No. of possible defects}}$$



Defect Prevention (cont.)

- ➔ Best in Class organizations have a DPE of between 92.5% and 99%. [1]
- ➔ DPE for Average US Company is between 75% and 85%. [1]
- ➔ We do not have historical DRE for past projects in our organization as we do not measure how our processes reduce/prevent defects.
- ➔ ***Assumption for Model:
DPE assumed to be in the range of 75% to 85% for defect estimation model***



Goal:

Create a mechanism for estimating the potential defects for a project based upon the requirements which can be used for:

- ➔ Decision making on testing effort, scope and schedule for a project
- ➔ Monitoring of project & deliverable quality
- ➔ Early warning of quality and testing issues
- ➔ Understand the balance between testing effort and quality

Design Criteria:

- ➔ Simple to use with graphical output
- ➔ Based upon an agreed measurable factor available for every project
- ➔ Updateable throughout a project to track progress against estimate
- ➔ Can be applied to any new project in our organization

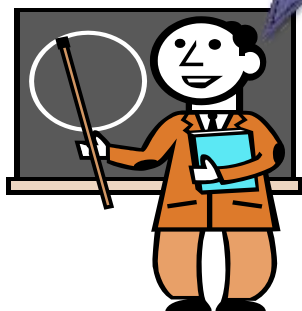


Defect Prediction Components

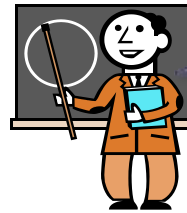
How many defects could I expect in acceptance test or production?



To build the model we need:

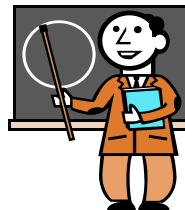


1) Project size and defect potential



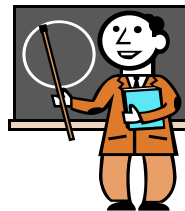
- ➔ Project size defined in terms of Detailed requirements (DRQ)
- ➔ Number of defects calculated from DRQ: $\text{No. of defects} = \text{DRQ} * 1.84$
- ➔ Defect potentials derived for each phase based on historical data

2) How we prevent defects from occurring?



- ➔ Processes, Procedures, Controls...
- ➔ DPE is assumed to be in the range of 75% to 85%

3) How we detect defects during the project?



- ➔ Formal reviews, Testing activities...
- ➔ DRE is 94%



Building a Model for Estimating the Number of Defects

Step 1: Estimate Total Potential Defects

- Project size defined in terms of Detailed requirements (DRQ)
 - Number of defects calculated from DRQ:
Maximum Possible defects = $DRQ * 1.84$
- Note: 1.84 is the number of possible defects for a single DRQ spread across all project phases (RQ, Design, Test, Production)

Example:

New Project has 1000 Detailed Requirements
Maximum Possible Defects = 1840
Test Case Executions Estimate = 1840

Step 2: Apply Defect Prevention Effectiveness

- DPE assumed to be in the range 75% to 85% for defect estimation model
- Number of Maximum Possible defects should be adjusted by applying the upper and lower boundaries for defect prevention effectiveness

Example : Maximum Possible Defects = 1840

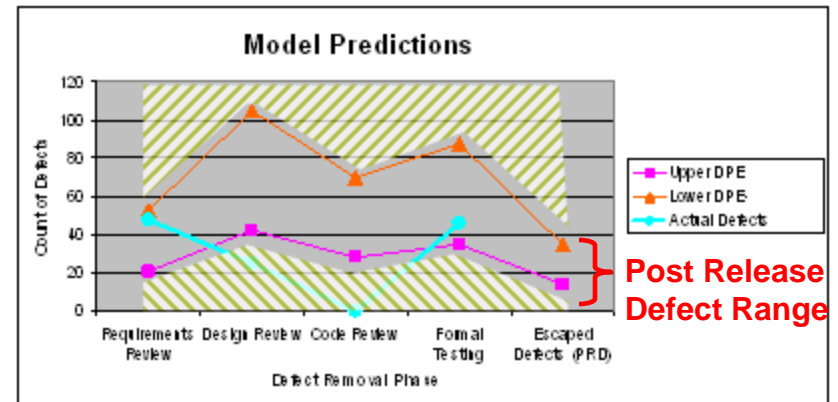
Estimate Defects that **could** escape to production (PRDs):
Calculate lower control limit (85% defects prevented) = 276
Calculate upper control limit (75% defects prevented) = 460
Number of predicted defects would be [276..460]

Step 3: Apply Defect Removal Efficiency

Defect Removal activity by phase	Expected DRE	Defects @ DPE 85%	Defects @ DPE 75%
Requirements Review	13%		
Design Review	3%		
Code Review & UT	4%		
Formal Testing	74%		
Escaped Defects (PRD)	6%		
Total Defects:	100%		

Example: 1000 DRs, Maximum Possible Defects = 1840
Estimate Defects that **could** escape to production (PRDs):
DRE = 94% => predicted number of PRDs would be
 $PRD = 0.06 \times [276..460] = [17..28]$
Estimate min & max defects per phase:
RR = [36..60], DR = [8..14], CR = [11..18], FT = [17..28]

Step 4: Estimate Defects per Phase & PRDs



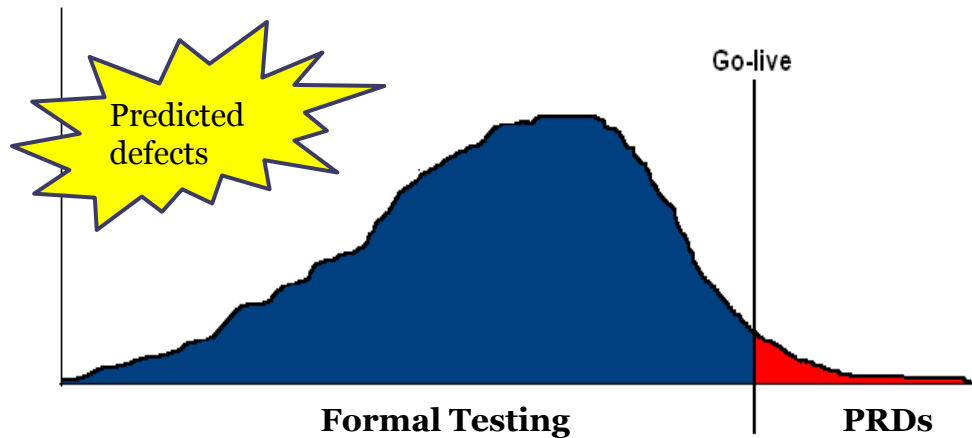
Plot actual defects against estimate to track progress and quality, and estimate Post Release Defects.



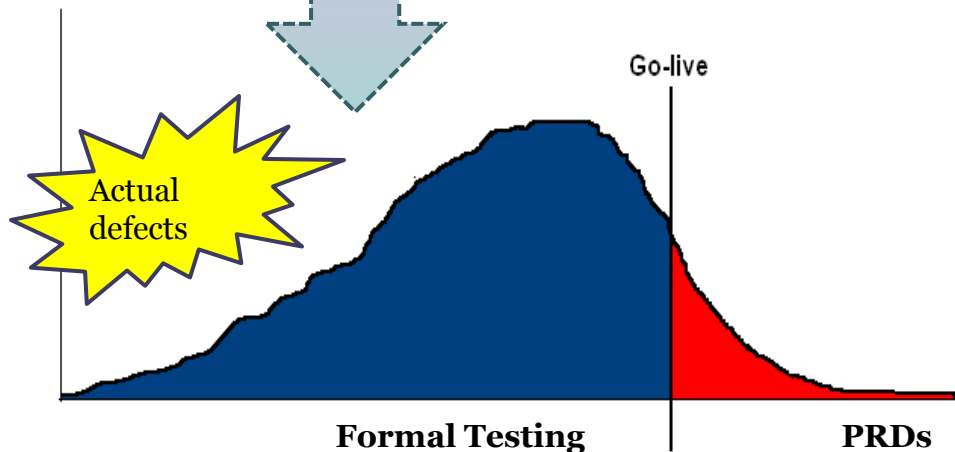


Defect Prediction Model

Case Study 1



Distribution of defects shifted towards post-release defects



Description: Formal testing was complete, the defects were below expected number of defects, however due to schedule and budget constraints the decision was made to stop testing and release product.

Outcome: Defects that were not found in Formal testing, were discovered in Post-production



Case Study 2

Number of requirements = 3386

Defect Removal activity by phase	Expected DRE	Defects @ DPE 85%	Defects @ DPE- 75%	Actual Defects	Actual DRE per phase
Requirements Review	13%	121	202	212	16%
Design Review	3%	28	47	51	4%
Code Review & UT	4%	37	62	59	5%
Formal Testing	74%	692	1153	976	75%
Escaped Defects (PRD)	6%	56	93	N/A	not released
Total Defects:	100%	935	1558	1298	100%

Description:

- Project was in UAT
- Metrics were trending towards the higher side of the defect range in each phase
- Defect discovery in Formal Testing phase was significantly lower than expected

Outcome:

Based on Quantitative analysis, the decision was made to add another cycle of UAT and postpone the Go-live date for one week. Additional 7% UAT defects were found during this week.



Benefits of the model

The model was implemented and currently is being used in two major Business Application Development Programs

The model has demonstrated its effectiveness in:

- ➔ quantitative decision-making about production go-live readiness;
- ➔ predicting the number of remaining undiscovered defects which may escape to production;
- ➔ providing visibility into defect removal effectiveness and deliverable quality

Benefit calculations show that a 1% improvement in DRE equals to approximately \$20,000 of savings in terms of reduced PRDs found and fixed in production.

Using the defect estimation model on large projects could potentially improve the project DRE by at least 5%, through:

- ➔ Better estimation of project defects and testing required to capture them before production
- ➔ Monitoring defects during the project and making course corrections if required.



1. Jones, Capers; **Estimating Software Costs: Bringing Realism to Estimating**; McGraw-Hill Osborne Media; 2007; (2nd edition.)
2. Jones, Capers; **Software Assessments, Benchmarks, and Best Practices**; Addison Wesley Longman, Boston, MA, 2000
3. Jones, Capers; **Applied Software Measurement**; 3rd edition; McGraw-Hill, New York, NY, 2008
4. Longstreet, David; **Test Cases & Defects**, Longstreet Consulting Inc.
<http://www.softwaremetrics.com/Articles/defects.htm>
5. Blackford, Craig, [Improve Testing Effectiveness](#), 11 June 2010





Thank You