

Reflective Software Engineering

Module 09:

Defects

(1)



Objectives

To understand:

- what defects are
- how to classify defects
- strategies for defect collection
- defect analyses (given a collection strategy)
- how to use defect data to improve your capabilities.

(2)



What are defects?

Pragmatically, we will think of "defects" as:

- Any event or activity generating REWORK.

Rework has been shown in one study to account for 44% of overall development cost!

Rework also introduces volatility into time and schedule estimates.

(3)



Basic forms of defect collection

Individual defect collection

- No external developer involvement
- Typically "in-process" defect collection
- Intermixed with other development effort or part of a separate "personal review" phase.
- Minimal overhead critical to accuracy

Group Review

- Involves a group of people
- Typically a separate phase or project
- Intrinsically high overhead
- Can obtain wider diversity of defect data than possible with individual collection

(4)



Example generic defects

A missing semi-colon in source code

- Causes rework through re-editing the file, and recompiling the system.

A missing class in the system design

- Causes rework through redesign of other classes to accomodate this class, re-editing the files, recompiling the system, and re-testing the behavior of the system to make sure no other errors were introduced.

A customer-reported missing requirement:

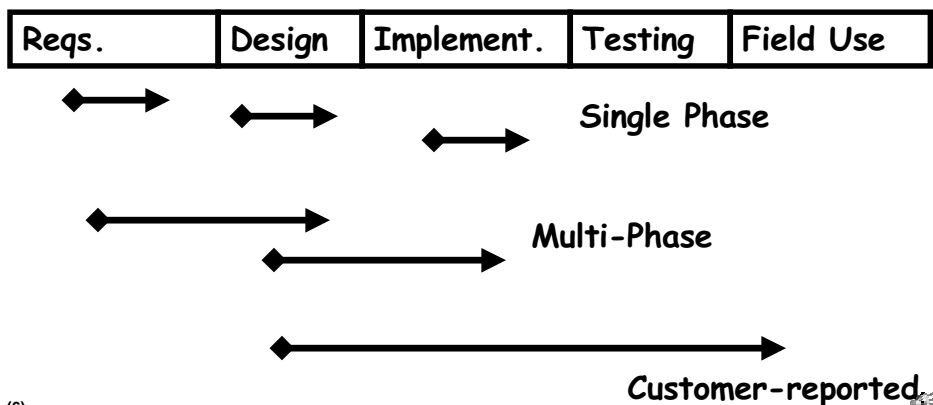
- Causes rework through redesign, re-editing, recompiling, retesting, and redistribution.

(5)



Cost of defect removal

The cost and impact of a defect depends upon the "distance" between where it was injected and where it is removed:



(6)



IBM's "rule of thumb"

Relative cost to correct defects:

• During design	1.5
• Prior to coding	1.0
• During coding	1.5
• Prior to test	10.0
• During test	60.0
• In field use	100.0

(7)



Strategies for defect management

Find and fix defects before they "escape" into the next phase of development.

Find and fix defects before they reach testing and field use.

Collect data on defect injection, removal, and fix times.

Analyze defect data to:

- Determine most important types of defects
- Prevent their occurrence or minimize cost.

(8)



Leap defect support

Tools:

- Pueo: specifies defect types.
- Honu: records/displays defect data.
- Mano: single defect entry tool

Data:

- Project, DocType, Defect Type, Occurrences, Injected, Removed, Fix Time, Description, etc.

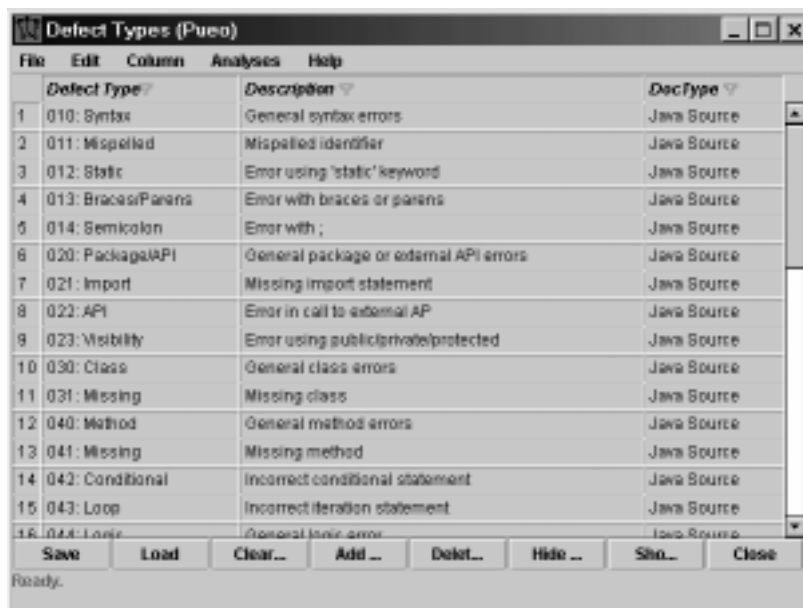
Analyses:

- Occurrence and fix time trends
- Individual and multi-project defect statistics

(9)



Pueo



(10)



Honu

Defects (Honu)

File Edit Column Analyses Help

	Date ▾	Project ▾	# ▾	DefectType ▾	Description ▾
1	05/12/1999	ISERN'99	1	30: Content	Should explain cost performance index on slide
2	05/12/1999	ISERN'99	1	10: Speech	Talking too fast
3	05/12/1999	ISERN'99	1	20: Format	Need an intro slide
4	05/12/1999	FSE'99	1	30: Content	Objectives of the talk not clearly stated up front.
5	05/12/1999	FSE'99	1	20: Format	Would be good to provide "road map" of where talk is going on slides.
6	05/12/1999	PCF'99	1	30: Content	Transition from SE research to entrepreneurial activities is rough.
7	05/12/1999	ISERN'99	5	30: Content	Prepare for pedagogy vs. methodology discussion
8	05/12/1999	FSE'99	7	30: Content	Need to provide total numbers of defects.
9	05/12/1999	ISERN'99	5	30: Content	Estimation should show both linear and averages

Save Load Clear... Add ... Delet... Hide ... Sho... Close Mail

Ready.

(11)

Mano

Defect (Mano)

Project: psae-1 # 1 Doc Type: Java Source Defect Type: 044: Logic Fix Time: 2:10

Injected: Design Removed: Testing Checklist: Record Clear Save

Desc: Did not design a check for answer sets that don't add to 100.



Defects (Honu)

File Edit Column Analyses Help

	Date ▾	Project ▾	FixTime ▾	# ▾	DefectType ▾	Description ▾	By ▾	Rev. ▾
12	10/11/1999	psae-1	41	1	044: Logic	Did not design a check for answer sets that don't add to 100.	Design	Testing
12								
12								

Save Load Clear To... Add Ro... Delete R... Hide Ro... Show R... Close Mail

Ready.

(12)

Example analyses

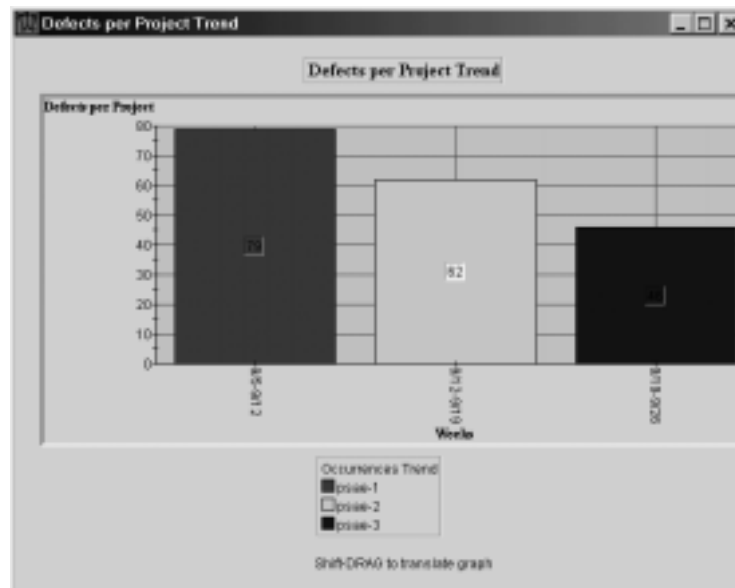
Here are some example analyses on some sample project data:

- Total defects
- Defect types
- Defect density
- Fix times
- Defect rates

(13)



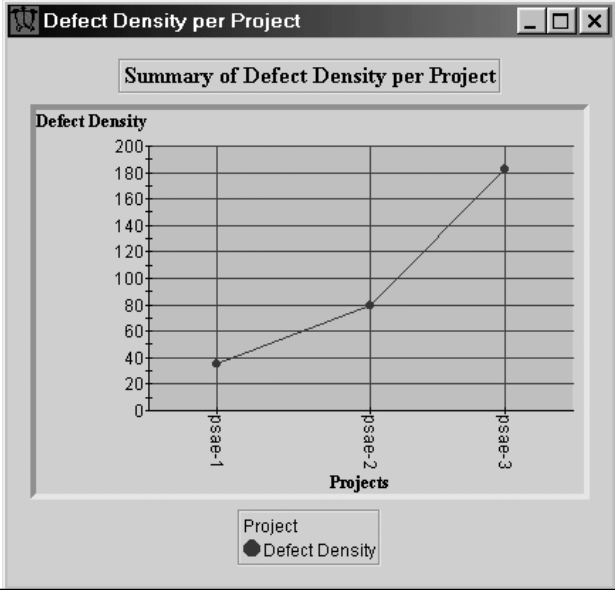
Total Defects



(14)



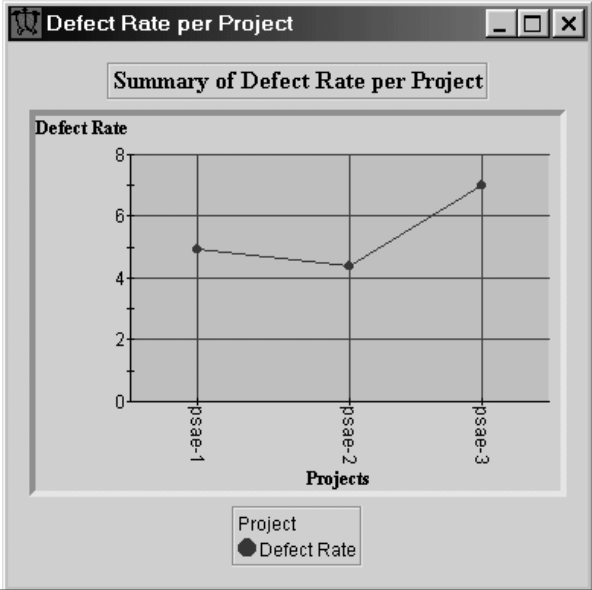
Defect Density



(15)



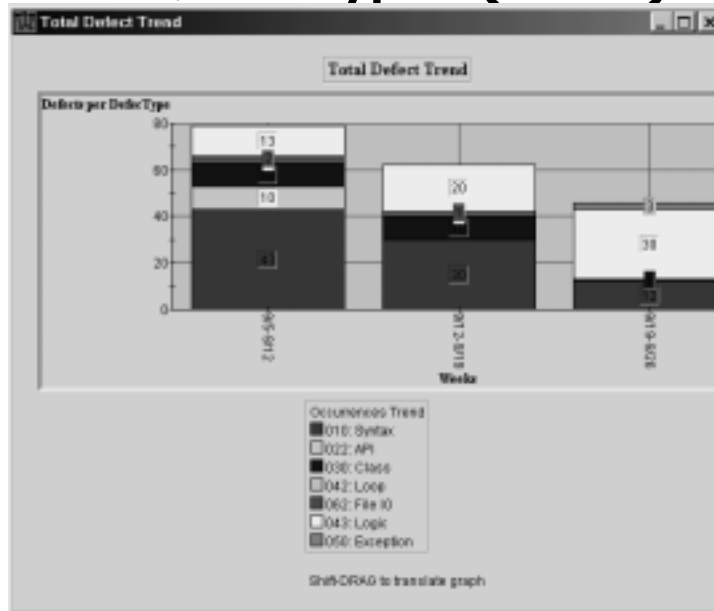
Defect rates



(16)



Defect Types (Chart)



(17)

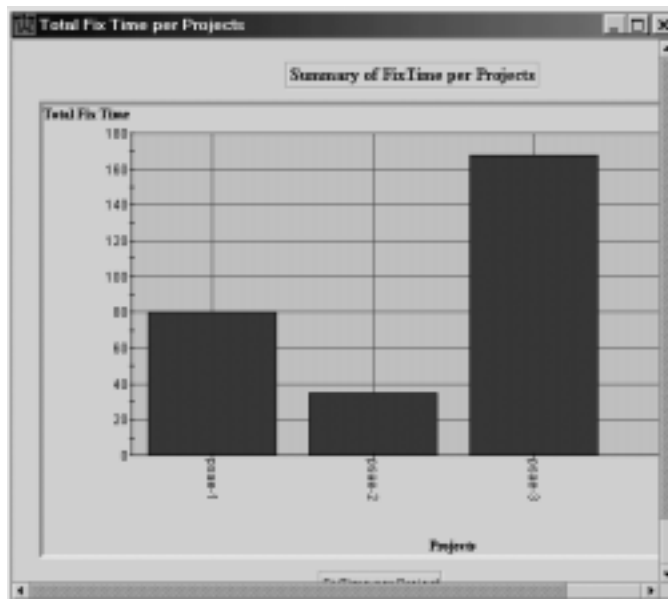
Defect Types (tabular)

Size	Time	Defects	Combined
Defects per DefectTypes			
010: Syntax	43	54.430	
022: API	10	12.659	
030: Class	10	12.659	
042: Loop	1	1.366	
043: Logic	13	16.458	
062: File IO	2	2.532	
Total	79		
Defects per DefectTypes			
010: Syntax	30	48.387	
030: Class	10	16.129	
042: Loop	1	1.813	
043: Logic	20	32.258	
062: File IO	1	1.813	
Total	62		
Defects per DefectTypes			
010: Syntax	12	28.887	
030: Class	1	2.174	
043: Logic	30	85.217	
050: Exception	3	6.532	
Total	46		
Testing	92	48.158	
Design	0	0	
Total	197		
Defects by Type			
030: Class	21	11.23	
050: Exception	3	1.604	
010: Syntax	05	45.455	
022: API	10	5.348	
<Undefined>	0		
043: Logic	63	33.69	
062: File IO	3	1.604	
042: Loop	2	1.07	
Total	197		

OK

(18)

Fix Times



(19)

Reflections upon this data

Total defects dropped from psae-1 to psae-3
•But density of defects increased!

Defect rate between 4-7 defects/hour

Syntax errors decreasing, logic errors on the rise

Exception errors just starting

Total rework (fix time) very high on last project.

(20)

What defects to record?

1. Record every single defect made.
 - High overhead---does backspace count?
2. Record every defect crossing a phase.
 - Whenever injected does not equal removed.
 - Practical, but some overhead.
 - Provides useful trend and density analyses.
3. Record only "major" defects per phase.
 - Both cross-phase and intra-phase collected.
 - Very practical, little overhead
 - Trend/density analyses not typically useful.

(21)



What defects to record? (cont.)

4. Record every defect of a specific type (or small set of types)
 - Very practical, low overhead.
 - Supports trends/densities for chosen types.
 - Good way to target problem areas.
 - Can miss "new" types of defects.
5. Record every defect injected in a specific phase.
 - Very practical, low overhead.
 - Supports trends/densities for chosen phase.
 - Good for targetting problem phases (design)
 - Doesn't address costs of other phases, etc.

(22)



What to record about each defect?

Project

- Required for almost all analyses

Fix Time

- Required for cost analyses

Occurrences

- Simplifies entry of repeated defects

Defect type

- Supports analyses on categories of defects

Description

- Provides useful textual info on defects

Injected

- Required for phase-based analyses

Removed

- Required for phased-based analyses

Location, DocID, Severity, Valid, Checklist

- To be used in review

(23)



What to do with defect data?

Target particular defect types for

- increased design effort
- review (I.e. logic, loops)

Target phases for increased QA effort.

Monitor trends upward or downward

Identify projects with "suspicious" data (very high defect rates/densities/etc.)

- Could indicate low quality
- Could also be artifact of collection strategy!

(24)



More implications of defect data

Defect data can also:

- Drive changes to defect type list.
- Generate new checklist items and patterns.
- Drive changes to the way defects are collected. (From general to targetted).
- Lead to use of reviews for help with certain defect types.
- Change code/design style to prevent problematic defects from occurring.

(25)



Initial defect data collection

For next project:

- download file with defect type definitions.
- You should record:
 - Every defect that crosses a phase.
 - In addition, at least one major defect from each phase.
- The following fields should be filled out:
 - date, project, fix time, occurrences, defect type, description, injected, removed.

(26)



Initial defect analysis

What defect types occurred most frequently?

What defect types were most expensive?

In which phase did the most rework (I.e. fix time) occur?

What can you do to decrease the occurrence of these defects in your next project?

(27)

