

Lecture Topics

Review of Process Improvement
Characterize and analyze current performance
Understanding Root Causes
Improving Quality Performance



Review of Process Improvement

From the Planning presentation, recall that the steps are to measure, analyze, and improve.

We will apply the following performance improvement steps

- Characterize and analyze current performance
- Generate Process Improvement Proposals (PIPs)
- Set performance goals
- Evaluate and select PIPs for implementation
- Implement PIPs
- Gather more data and evaluate the results

Characterizing the current performance means to measure your performance and set baseline against which improvements will be compared.

The analysis will attempt to gain insight into the strengths and weaknesses of your current performance. Will look at how to view data, segment or stratify the data, and identify high leverage areas for improvement.



Characterize and analyze current performance

In understanding the current quality performance, at least the following questions must be answered.

- Are there enough data points to draw conclusions?
- What kinds of defects are injected?
- How effective are my defect removal activities?
- How does quality affect my productivity?
- Am I consistently delivering a zero to low defect product?

To address these issues and begin to understand the quality data available to us, we are going to take a journey through a single student's data set.



The data journey

As we go through a student's quality data, begin to ask yourself the following questions:

- Where can they improve?
- What issues are effecting them?
- How could they address these issues?
- How could they improve?
- What can they do different?
- Why should they change?
- What are the root causes?



© 2011 Carnegie Mellon University 5

As the instructor goes through the following slides, they should note any ideas that come from the students on a white board. This will aid in the PIP discussion and subsequent quality planning and tracking presentations. It will also help them in starting to think about what kind of things they could do differently with their own work.



PSP Advanced: Understanding and Improving Quality Performance	
Assignment - Process	Process Addition
Program 1 - PSP 0.0	
Program 2 - PSP 0.1	<ul style="list-style-type: none"> estimating and reporting software size distributing development time over planned project phases using a size counting and coding standard recording process problems and improvement ideas
Program 3 - PSP 1.0	<ul style="list-style-type: none"> PROBE size estimating method and size estimating template test report template The project plan summary was expanded. <ul style="list-style-type: none"> Summary section was added with plan, actual, and to-date productivity Program Size summary includes planned size for all size accounting types All values except the Total Size under Actual in the Program Size Summary are calculated.
Program 4 - PSP 1.1	<ul style="list-style-type: none"> task planning template schedule planning template <ul style="list-style-type: none"> These are typically used for projects that take several days or weeks, which are not required for the PSP exercises. The project plan summary was expanded to include basic process statistics.
Program 5 - PSP 2.0	<ul style="list-style-type: none"> design review checklist code review checklist <ul style="list-style-type: none"> Design and code review checklists are described separately. PSP2 adds two key capabilities to the PSP <ul style="list-style-type: none"> design and code reviews quality planning The PSP2 project plan summary supports these two new capabilities.
Program 6 - PSP 2.1	<ul style="list-style-type: none"> PSP2.1 design review script PSP2.1 design review checklist Design specification template
Program 7 & 8 - PSP 2.1	None

© 2011 Carnegie Mellon University 6

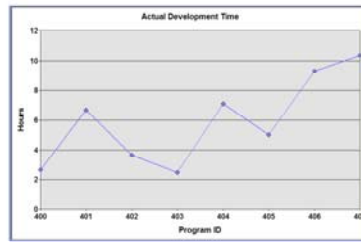
Process used in generating the following Student #4 data. This student took the traditional PSP 1 and 2 courses, which may vary slightly from the PSP Fundamental and Advanced programming assignments, thus this slide is included in helping the instructor and students interpret the data.



Actual Development Time

How much time did the student spend on each program?

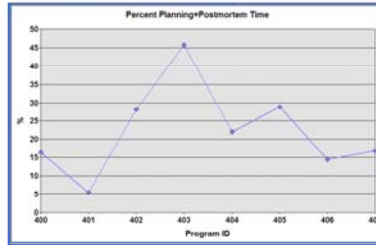
Where did the student spend most of their time?
How did the distribution of time change as the process changed?



Percent Planning and Postmortem Time

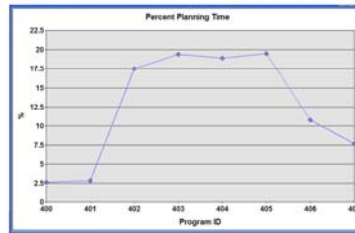
Why is the percentage so high for assignment 4?

Is this spike in time found in Planning, Postmortem, or both?



Percent Planning Time

What percentage of time should the student plan for in the future for planning?

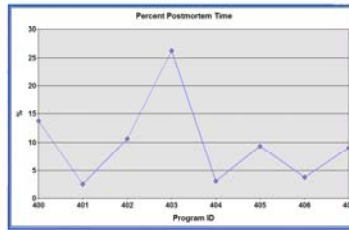


Percent Postmortem Time - 1

Why is the percentage so high for program assignment four?

Is it caused by the introduction of PSP 1.1?

Does the time spent in PM correlate to program size?

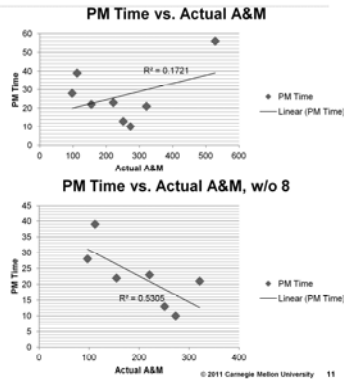


Percent Postmortem Time - 2

Assignment 8 was identified as an outlier because it took twice as long as planned to complete due to the fact it was completed weeks after the program was initially coded and unit tested. In all other cases PM was conducted right after UT. This can be seen by reviewing the student's time log.

The time spent performing the PM in Assignment 4 seems to be consistent with other data points when compared to size.

So why was the percentage of time spent in PM so much higher than other programming assignments?



The instructor should take this time to discuss correlation between any two measures as it applies to quality.

The instructor should also talk about outlier in quality data. As you change your process, previous data points can become obsolete and skew the data.

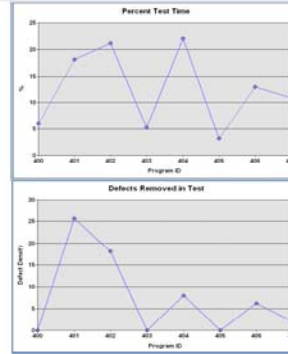


Percent Postmortem Time - 3

Looking at *Percent Test Time* you will notice that percent of time spent in Test for Assignment 4 dropped significantly.

This is partially due to the fact that no defects were found in Test, which can be seen by looking at the Defects Removed in Test.

Thus percent distribution for other phases, such as PM increased proportionately.

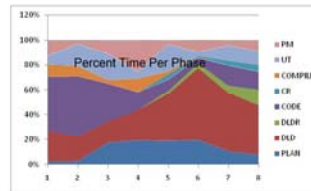


© 2011 Carnegie Mellon University 12



How did the distribution of time change as the process changed?

Phase	% Time Spent
PLAN	~ same
DLD	Increased
DLDL	Increased
CODE	Decreased
CR	Increased
COMPILE	Decreased
UT	Decreased
PM	Decreased



How did these additional phases and tasks effect productivity?

© 2011 Carnegie Mellon University 13

As the process changed and more phases were added such as personal reviews and design templates

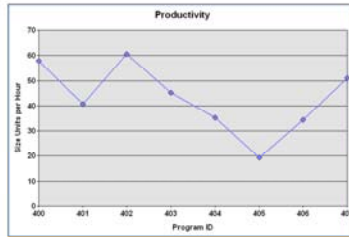


How did additional phases and tasks affect productivity?

Productivity declined with each change in process.

Once the process stabilized, productivity increased to initial levels.

How is the student able to do more work in the same amount of time?



© 2011 Carnegie Mellon University 14

The decline in productivity can be associated with the student's learning curve



How is the student able to do more work in the same amount of time?

Where are most Defects Injected? Removed?

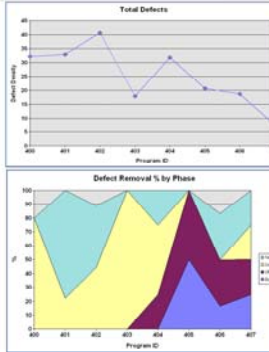
How effective are the student's reviews?

How effective are the different process steps at removing defects?

How does review effectiveness affect the time spent in test?

What is the cost of quality?

What does the student know about the defects he/she injects/removes?



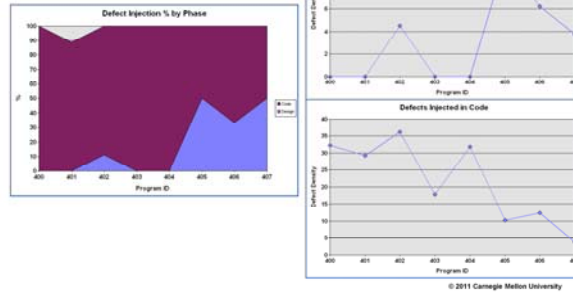
© 2011 Carnegie Mellon University 15

In general the student had less defects to fix and was able to remove more defects earlier in the process where it is less costly to fix. Thus, was able to spend more time on design and quality.



Where are most Defects Injected?

Prior to the introduction of PSP2.1, most of the defects were injected in the Code phase.

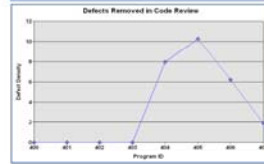
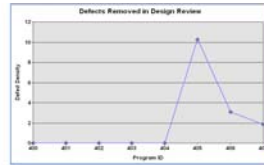


It could be possible that some defects recorded as injected in CODE prior to PSP 2.1 were actually design issues.

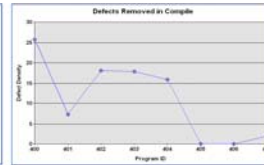
Defects seem to be evenly injected in both code and design phases after the introduction of formal Design specification and verification concepts in PSP 2.1.

Where are most Defects Removed?

After PSP2.1



Prior to PSP 2.1

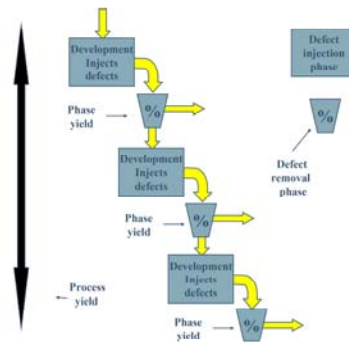


© 2011 Carnegie Mellon University 17



How effective are the student's reviews? - Yield

Yield measures the efficiency of each filtering phase at removing defects.



© 2011 Carnegie Mellon University 18



How effective are the student's Reviews? – Phase Yield

Phase Yield represents the defects removed during a phase as a % of those present at the start of the phase, plus those injected during that phase.

Looking at the student's Phase yield for programs 5 – 8 (PSP2 and PSP2.1), which review is least effective?

How could the student possibly improve their phase yield?

Phase	Program 5 Injected	Program 5 Removed	Program 6 Injected	Program 6 Removed	Program 7 Injected	Program 7 Removed	Program 8 Injected	Program 8 Removed	To-Date Injected	To-Date Removed	Phase Yield
Planning	0	0	0	0	0	0	0	0	0	0	0.00%
Detailed Design	0	0	1	0	2	0	2	0	5	0	0.00%
OLD Review	0	0	0	1	0	1	0	1	0	3	60.00%
Code	8	0	1	0	4	1	2	0	15	1	5.88%
Code Review	0	2	0	1	0	2	0	1	0	6	37.50%
Compile	0	4	0	0	0	0	0	1	0	5	50.00%
Unit Test	0	2	0	0	0	2	0	1	0	5	100.00%
PM	0	0	0	0	0	0	0	0	0	0	0.00%

© 2011 Carnegie Mellon University 19

Instructor should ask the students why we only care about program's 5-8?

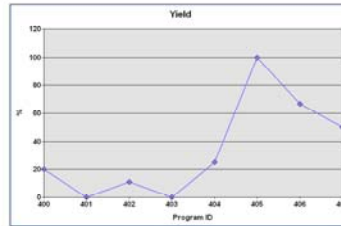
Main reason: Programs 1-4 did not have reviews, thus would make the Phase Yield smaller than they actually are using PSP 2 and 2.1.



How effective are the student's Reviews? – Process Yield

Process Yield is the percentage of defects removed before the first compile or test.

The percentage of defects removed prior to compile and test significantly increased with the introduction of reviews.



$$\text{Process Yield} = \frac{100 * (\text{defects found before the first compile or test})}{(\text{defects injected before compile or test})}$$



How efficient are the different process steps at removing defects? - 1

Defect Removal Leverage (DRL) measures the relative efficiency of a process step at removing defects.

DRL is the number of defects removed per hour by a process step relative to a base process.

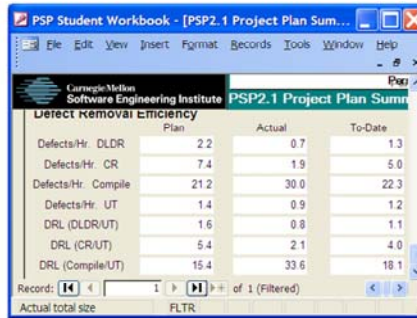
- The usual base is unit test (UT).
- DRL (X/UT) is the DRL for phase X with respect to unit test.

$$\text{DRL}(X/\text{UT}) = \frac{(\text{defects/hour phase X})}{(\text{defects/hour unit test})}$$



How efficient are the different process steps at removing defects? - 2

For Assignment 8



The screenshot shows a software window titled "PSP Student Workbook - [PSP2.1 Project Plan Sum...]" with a menu bar (File, Edit, View, Insert, Format, Records, Tools, Window, Help) and a toolbar. The main content area displays a table titled "Defect Removal Efficiency" with the following data:

	Plan	Actual	To-Date
Defects/Hr. DLDR	2.2	0.7	1.3
Defects/Hr. CR	7.4	1.9	5.0
Defects/Hr. Compile	21.2	30.0	22.3
Defects/Hr. UT	1.4	0.9	1.2
DRL (DLDR/UT)	1.6	0.8	1.1
DRL (CR/UT)	5.4	2.1	4.0
DRL (Compile/UT)	15.4	33.6	18.1

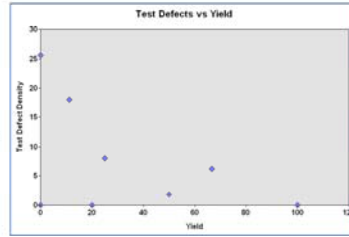
Below the table, there is a status bar showing "Records: 1 of 1 (Filtered)" and "Actual total size FLTR".



How does review effectiveness affect the time spent in test?

The higher the Process Yield the less defects found in test.

Less defects in test results in less time in test.



$$\text{Test Defect Density} = \frac{\text{Test Defects}}{\text{KLOC}} = \frac{1000 * (\text{test defects})}{(\text{actual A \& M})}$$



What is the cost of quality (COQ)?

COQ measures process quality in a way that is meaningful to management.

The COQ elements are failure, appraisal, and prevention costs.

Failure cost is the time spent in repair and rework plus the cost of any scrap. In PSP, it is compile and test time.

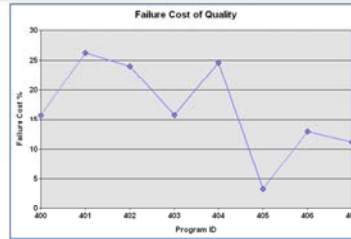
Appraisal costs are the costs of inspecting for defects. In PSP, appraisal cost is design and code review time.

In the TSP, inspections are included in appraisal costs.



Failure Cost of Quality

In general, the % Failure COQ decreased with the introduction of PSP2.1

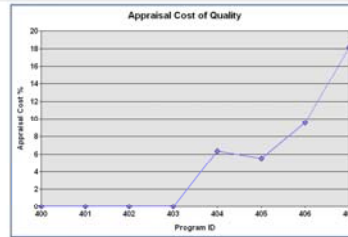


$$\% \text{ Failure COQ} = \frac{100 * (\text{compile time} + \text{test time})}{\text{Total development time}}$$



Appraisal Cost of Quality

In general, the % Appraisal COQ began increasing with the introduction of PSP2.0

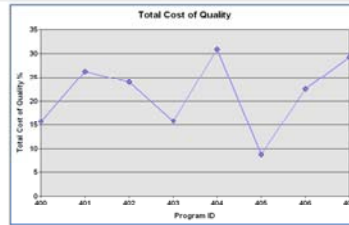


$$\% \text{ Appraisal COQ} = \frac{100 * (\text{design review time} + \text{code review time})}{\text{Total development time}}$$



Total Cost of Quality

Quality is Free when proper attention and discipline is applied to reviews.



COQ = % Appraisal COQ + % Failure COQ

© 2011 Carnegie Mellon University 27

Properly conducted reviews significantly reduce testing time and produce high-quality results. Unless the engineer is committed to producing high-quality products, the review process is likely to be ineffective. Engineers whose objective is to begin testing as soon as possible rarely perform code reviews or perform them so poorly that they are a waste of time.

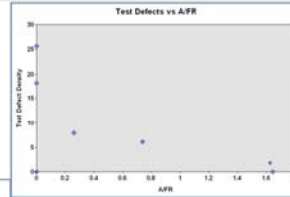
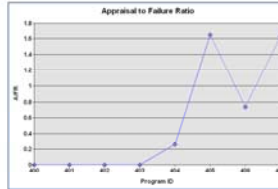


Appraisal to Failure Ratio (A/FR)

When A/FR is greater than 1, the student is spending more time in reviews than in compile and test.

The number of test defects is typically much lower for higher values of A/FR (≥ 2 is a good goal). However, too high a value ($>> 10$) could mean excessive time is being spent in reviews.

In this case the student is approaching the goal of 2 or higher, but there is still room for improvement.



$$A/FR = \frac{\% \text{ Appraisal COQ}}{\% \text{ Failure COQ}}$$



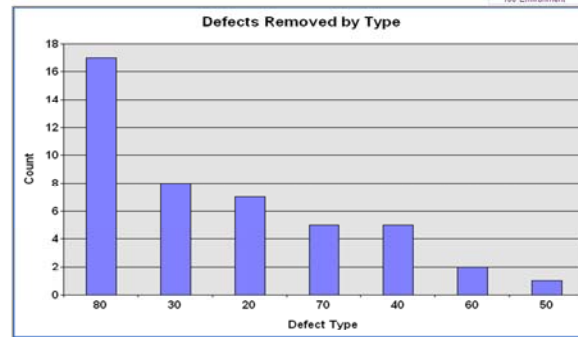
What does the student know about the defects they inject/remove?

For the student to understand their defects, they need to be able to answer questions such as:

- What are the most common types of defects?
- Which defects cost the most?
- What types of defects are injected most in the design and code phases?
- What types of defects are primarily removed in compile and test phases?
- What phase had the largest average fix time?
- What phase had the largest total fix time?



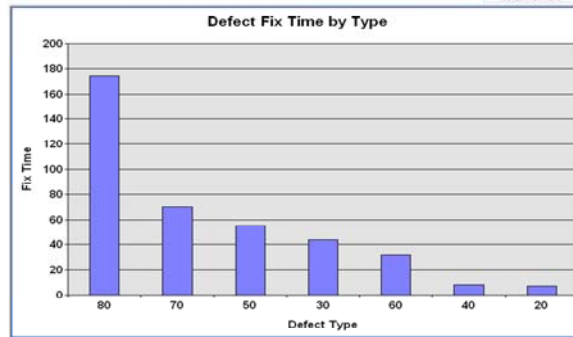
What are the most common types of defects?



© 2011 Carnegie Mellon University 30



Which defects cost the most?



© 2011 Carnegie Mellon University 31



PSP Advanced: Understanding and Improving Quality Performance

What types of defects are injected most in the design and code phases?

10 Documentation
20 Syntax
30 Build Package
40 Assignment
50 Interface
60 Checking
70 Data
80 Function
90 System
100 Environment

PSP Student Workbook - [PSP Percent Injected and Removed by Type Report]

File Edit View Insert Format Records Tools Window Help

Carnegie Mellon Software Engineering Institute Personal Schedule Process™

PSP Percent Injected/Removed by Type Report

Type	Number Injected		Percentage Injected		Number Removed		Percentage Removed	
	Design	Code	Design	Code	Compile	UT	Compile	UT
20		6		15.8%	6	1	35.3%	6.3%
30		8		21.1%	8		47.1%	
40		5		13.2%	2	1	11.8%	6.3%
50		1		2.6%		1		6.3%
60		2		5.3%		1		6.3%
70	2	3	33.3%	7.9%	1	1	5.9%	6.3%
80	4	13	66.7%	34.2%		11		68.8%
90								
100								
Totals	6	36			17	16		

Typenumber of defect type

© 2011 Carnegie Mellon University 32

Design, Types 70 and 80

Code, 20, 30, 80 (If you look at the defect log, you will notice that the type 80 could have been miss categorized. They were most likely injected in Design, however it was prior to PSP2 and PSP2.1)



PSP Advanced: Understanding and Improving Quality Performance

What types of defects are injected most in the design and code phases?

10 Documentation
20 Syntax
30 Build Package
40 Assignment
50 Interface
60 Checking
70 Data
80 Function
90 System
100 Environment

PSP Student Workbook - [PSP Percent Injected and Removed by Type Report]

File Edit View Insert Format Records Tools Window Help

Carnegie Mellon Software Engineering Institute Personal Software Process®

PSP Percent Injected/Removed by Type Report

Type	Number Injected		Percentage Injected		Number Removed		Percentage Removed	
	Design	Code	Design	Code	Compile	UT	Compile	UT
20		6		15.8%	6	1	35.3%	6.3%
30		8		21.1%	8		47.1%	
40		5		13.2%	2	1	11.8%	6.3%
50		1		2.6%		1		6.3%
60		2		5.3%		1		6.3%
70	2	3	33.3%	7.9%	1	1	5.9%	6.3%
80	4	13	66.7%	34.2%		11		68.8%
90								
100								
Totals	6	36			17	16		

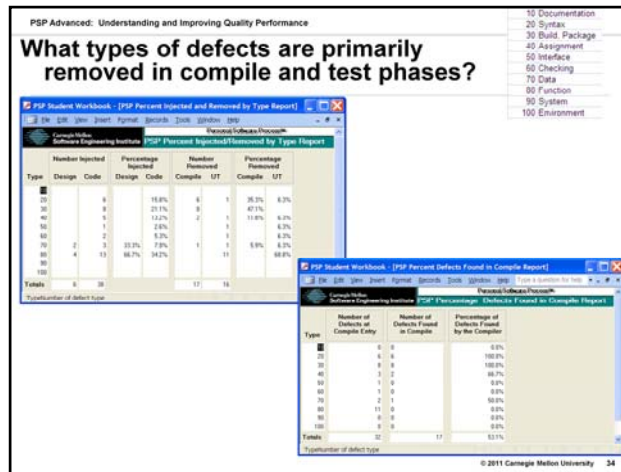
Type number of defect type

© 2011 Carnegie Mellon University 33

Design, Types 70 and 80

Code, 20, 30, 80 (If you look at the defect log, you will notice that the type 80 could have been miss categorized. They were most likely injected in Design, however it was prior to PSP2 and PSP2.1)



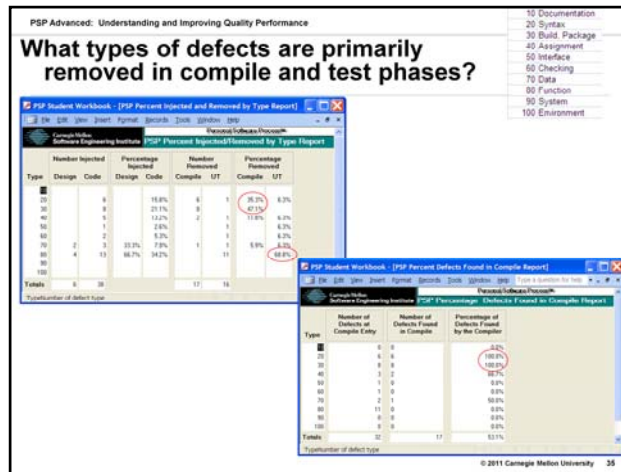


For compile, both tables show Types 20 and 30

For Test, Types 80

Instructor should note that Type 80 defects are usually injected in the design phase, thus they have escaped both the Design Review and Code Review.





For compile, both tables show Types 20 and 30

For Test, Types 80

Instructor should note that Type 80 defects are usually injected in the design phase, thus they have escaped both the Design Review and Code Review.



What had the largest average fix time? What had the largest total fix time?

PSP Student Workbook - [PSP Defect Fix Time Report]

File Edit View Insert Format Records Tools Window Help

Carnegie Mellon Software Engineering Institute Personal Software Process PSP Defect Fix Time Report

Defects Injected In	Removed in Compile	Removed in UT	Removed in Compile and UT
BUILD			
Fix Time		60	60
Total		1	1
Avg. Fix Time		60	60
CODE			
Fix Time	55	190	245
Total	17	14	31
Avg. Fix Time	3.23529412	13.57142857	7.903225806
TOTALS			
Fix Time	55	250	305
Total	17	15	32
Avg. Fix Time	3.23529412	16.66666667	9.53125
Symbol of phase			



What had the largest average fix time? What had the largest total fix time?

PSP Student Workbook - [PSP Defect Fix Time Report]

Carnegie Mellon Software Engineering Institute

PSP Defect Fix Time Report

Defects Injected In		Removed in Compile	Removed in UT	Removed in Compile and UT
DLD	Fix Time		60	60
	Total		1	1
	Avg. Fix Time		60	60
CODE	Fix Time	55	190	245
	Total	17	14	31
	Avg. Fix Time	3.23529412	13.57142857	7.903225806
TOTALS	Fix Time	55	250	305
	Total	17	15	32
	Avg. Fix Time	3.23529412	16.66666667	9.53125

Symbol of phase

DLD/UT had largest average fix time

CODE/UT had largest total fix time



Generate Process Improvement Proposals (PIPs)

Now that we have taken a journey through the data:

- Where can they improve?
- What issues are affecting them?
- How could they address these issues?
- How could they improve?
- What can they do different?
- Why should they change?
- What are the root causes?



HOW DOES THE DATA SUPPORT YOUR ANSWERS?

The instructor should be having this discussion with the class as they are going through the data, or while they are on the journey.



Example PIPs

1. Process Yield has been declining over the last 2 programs, which will lead to more defects being found in Test. To improve Process Yield, focus on improving the code review phase yield as it is only 37.5%. Try reviewing at a slower pace and take a break between the code and code review phase.
2. Review type 70 & 80 defects in the defect log and update review checklists to find and fix defects prior to test. They are the most costly and they are currently making it through to the failure phases.
3. Implement Design Verification Techniques to improve the effectiveness of design reviews and reduce the number of defects making it into test.



Review of Process Improvement

From the Planning presentation, recall that the steps are to measure, analyze, and improve.

We will apply the following performance improvement steps

- Characterize and analyze current performance
- Generate Process Improvement Proposals (PIPs)
- ➔ Set performance goals
- Evaluate and select PIPs for implementation
- Implement PIPs
- Gather more data and evaluate the results

Characterizing the current performance means to measure your performance and set baseline against which improvements will be compared.

The analysis will attempt to gain insight into the strengths and weaknesses of your current performance. Will look at how to view data, segment or stratify the data, and identify high leverage areas for improvement.



Set performance goals

What quality goals should the student consider for next time?

How can the student change their process to meet the new quality goals?



© 2011 Carnegie Mellon University 41



A Quality Goal Consideration

Quality Goal: Increase Process Yield by 20%.

- Current Yield = 50%
- Desired Yield = 60%

How can the process change to meet the new quality goals?

- Improve Code Review effectiveness



Phase	Current Phase Yield	Desired Phase Yield
Planning	0.00%	0.00%
Detailed Design	0.00%	0.00%
DLD Review	60.00%	60.00%
Code	5.88%	5.88%
Code Review	37.50%	50.00%
Compile	50.00%	50.00%
Unit Test	100.00%	100.00%
PM	0.00%	0.00%

© 2011 Carnegie Mellon University 42



Review of Process Improvement

From the Planning presentation, recall that the steps are to measure, analyze, and improve.

We will apply the following performance improvement steps

- Characterize and analyze current performance
- Generate Process Improvement Proposals (PIPs)
- Set performance goals
- • Evaluate and select PIPs for implementation
- Implement PIPs
- Gather more data and evaluate the results

Characterizing the current performance means to measure your performance and set baseline against which improvements will be compared.

The analysis will attempt to gain insight into the strengths and weaknesses of your current performance. Will look at how to view data, segment or stratify the data, and identify high leverage areas for improvement.



Which PIPs could improve Code Review effectiveness?

1. Process Yield has been declining over the last 2 programs, which will lead to more defects being found in Test. To improve Process Yield, focus on improving the code review phase yield as it is only 37.5%. Try reviewing at a slower pace and take a break between the code and code review phase.
2. Review type 70 & 80 defects in the defect log and update review checklists to find and fix defects prior to test. They are the most costly and they are currently making it through to the failure phases.
3. Implement Design Verification Techniques to improve the effectiveness of design reviews and reduce the number of defects making it into test



Which PIPs could improve Code Review effectiveness?

1. ✓ Process Yield has been declining over the last 2 programs, which will lead to more defects being found in Test. To improve Process Yield, focus on improving the code review phase yield as it is only 37.5%. Try reviewing at a slower pace and take a break between the code and code review phase.
2. ✓ Review type 70 & 80 defects in the defect log and update review checklists to find and fix defects prior to test. They are the most costly and they are currently making it through to the failure phases.
3. Implement Design Verification Techniques to improve the effectiveness of design reviews and reduce the number of defects making it into test



Review of Process Improvement

From the Planning presentation, recall that the steps are to measure, analyze, and improve.

We will apply the following performance improvement steps

- Characterize and analyze current performance
- Generate Process Improvement Proposals (PIPs)
- Set performance goals
- Evaluate and select PIPs for implementation
- • Implement PIPs
- Gather more data and evaluate the results

Characterizing the current performance means to measure your performance and set baseline against which improvements will be compared.

The analysis will attempt to gain insight into the strengths and weaknesses of your current performance. Will look at how to view data, segment or stratify the data, and identify high leverage areas for improvement.



Implement PIPs - 1

1. Process Yield has been declining over the last 2 programs, which will lead to more defects being found in Test. To improve Process Yield, focus on improving the code review phase yield as it is only 37.5%. Try reviewing at a slower pace and take a break between the code and code review phase.
 - Update process script to take short break between code and code review phase
 - Increase plan code review time



Implement PIPs - 2

2. Review type 70 & 80 defects in the defect log and update review checklists to find and fix defects prior to test. They are the most costly and they are currently making it through to the failure phases.
 - Review defect log
 - Update Code Review Checklist to look for the defects that are escaping to compile and test



Review of Process Improvement

From the Planning presentation, recall that the steps are to measure, analyze, and improve.

We will apply the following performance improvement steps

- Characterize and analyze current performance
- Generate Process Improvement Proposals (PIPs)
- Set performance goals
- Evaluate and select PIPs for implementation
- Implement PIPs

→ • Gather more data and evaluate the results

Characterizing the current performance means to measure your performance and set baseline against which improvements will be compared.

The analysis will attempt to gain insight into the strengths and weaknesses of your current performance. Will look at how to view data, segment or stratify the data, and identify high leverage areas for improvement.



Gather more data and evaluate the results

1. Follow the updated process on a few additional assignments
2. Evaluate the results. Realize that some changes to your process will invalidate historic performance.
3. If the data shows improvement, keep the change. If no improvement is found, then re-evaluate the process changes and revise.
4. Continue the Quality Journey
 - Characterize and analyze current performance
 - Generate Process Improvement Proposals (PIPs)
 - Set performance goals
 - Evaluate and select PIPs for implementation
 - Implement PIPs
 - Gather more data and evaluate the results



© 2011 Carnegie Mellon University 50



Evaluate the results

To determine if implemented process improvements have been effective:

- you should periodically repeat the steps for baselining your work processes and comparing the baseline performance to previously established improvement goals.
- be careful to avoid the complications of bolstering and clutching.

Bolstering is the selective recall of only those results that reinforce an opinion or belief, usually manifest by forgetting failures and remembering only successes. Use of all PSP data from all projects should preclude bolstering.

Clutching is the tendency to perform badly when under pressure or when a good outcome is especially critical, thereby negating successful performance on past projects when using the same processes. By following established processes and using data (rather than instinct) as a basis for instantiating process changes, clutching can be minimized or avoided.

© 2011 Carnegie Mellon University 51



Continue the Quality Journey

Set realistic and challenging goals by using data analysis to

- identify problem areas
- determine the root cause of problems
- determine the potential effects of process change
- set quantifiable and challenging improvement goals



© 2011 Carnegie Mellon University 52



Messages to Remember



You cannot know your quality performance without detailed tracking of actual data.

Performance consistency depends on the early removal of defects.

Understanding the causes behind defect injections and escapes can lead to many ideas for process changes.

You can improve quality performance by learning from your data and trying new process ideas.

Focus on regularly making small improvements and major changes will take care of themselves.



