Analysis of Code (and Design) Defect Injection and Removal in PSP

Diego Vallespir

Grupo de Ingeniería de Software
Universidad de la República
Uruguay

William Nichols

Software Engineering Institute
Carnegie Mellon University
United States

Agenda

The Role of Defects

Our Research

The Data Set

Where the Defects Are Injected

Analysis of Code Defects

- Defect types injected during Code
- When are the defects removed
- Cost to remove the defects injected in Code

Using data for planning

Conclusions and Future Work

The Role of Defects

- A primary goal in SPI: more efficient software development
- Software defects work against this goal
- To prevent or remove defects efficiently, we have to understand them:
  - Where and when are defects injected and removed?
  - Which defect type is most frequently injected?
  - Which type is most expensive to remove?
  - How many and which types of defects escape into unit test?
  - Other considerations

Our Research

Research goal:

analyze PSP data to learn about the characteristics of defects injected during design (presented at TSP Symposium 2011) and code

The Data Set

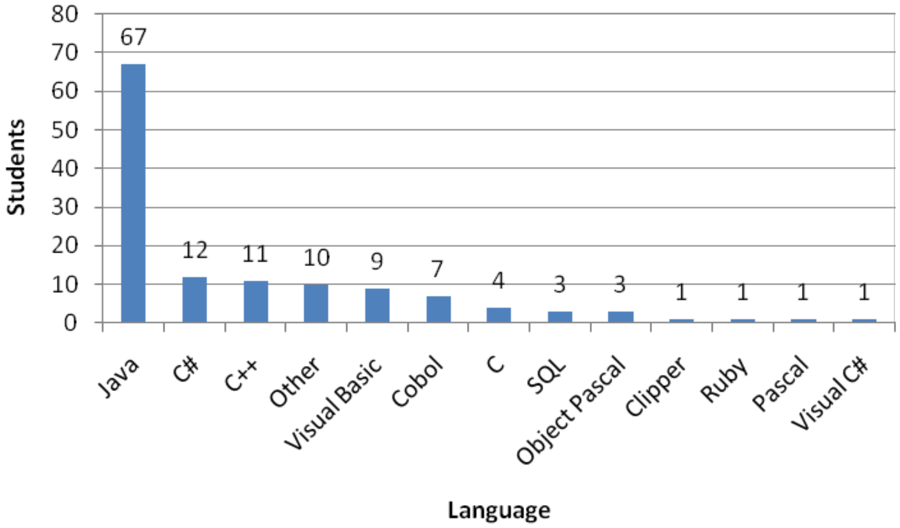
PSP 8 program course

- From October 2005 to January 2010

Only PSP2.1 was considered

- Programs 6, 7 and 8
- Threat to validity: the students who generated the data were in a learning process, so the PSP techniques may not have been well applied

The Data Set (2)



94 engineers used the Java, C++, C# and C programming languages

Reason: these languages used similar syntax, subprogram and data constructs

Threat to validity: Java, C++, and C# are OO languages but C is not and we are analyzing code defects. (Thanks, reviewers, for pointing this out.)

However, the C language was used only by 4 engineers.

The Data Set (3)

94 engineers

2 did not record any defects in the last 3 programs.

4 whose records of injected defects (injected during Code) were uncertain regarding their correctness and therefore were dismissed for this analysis.

8 did not record defects in the Code phase, so they were dismissed, as well.

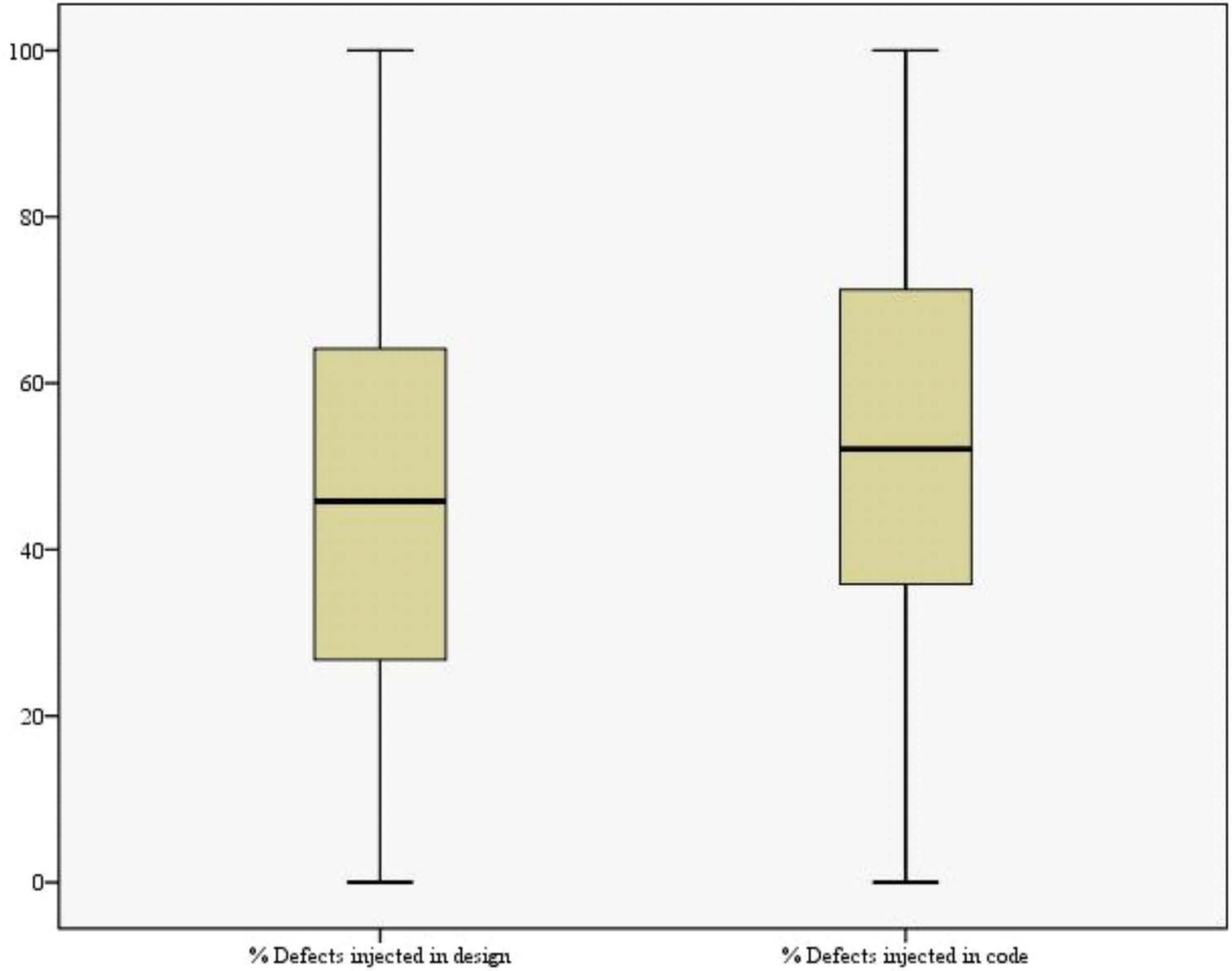
Finally, we use data from 80 engineers.

Where the Defects Are Injected

	DLD	DLDR	Code	CR	Comp	UT
Mean	46.4	0.4	52.4	0.3	0.03	0.5
Lower	40.8	0.2	46.7	0.0	0.0	0.2
Upper	52.0	0.7	58.1	0.7	0.09	0.9
Std. Dev.	27.2	1.7	27.4	1.8	0.3	1.8

As we expected, almost 99% of the defects are injected in the DLD and Code phases.

Where the Defects Are Injected (2)



Where the Defects Are Injected (3)

The variability between individuals is substantial. For example, some engineers don’t inject defects during design and some of them don’t inject defects during code.

Future work: try to understand the characteristics of individuals exhibiting different defect injection patterns

Analysis of Code Defects

In this work, we focus on code defects.

Based on our analysis, we will discuss:

- What types of defects are injected during code
- When those defects are removed
- The effort required to find and fix defects

Defects Types Injected During Code

	Docs.	Syn.	Buil	Assign	Inter.	Chec	Data	Func	Syst	Env
Mean	3.8	40.3	0.6	14.0	5.5	2.7	5.8	26.4	0	0.9
Mean D.	6.9	6.0	0.1	12.6	10.0	4.6	9.8	46.6	0.2	3.1
Lower	1.5	33.7	0.0	9.9	3.1	1.0	3.1	19.9	0	0.0
Upper	6.0	46.9	1.1	18.1	8.0	4.4	8.6	32.8	0	1.7
Std. dev.	10.1	29.5	2.5	18.4	11.1	7.4	12.4	29.1	0	3.9

To improve the detection of code defects, we first want to know which types of defects were injected during the Code phase.

Defects Types Injected During Code (2)

	Docs.	Syn.	Buil	Assign	Inter.	Chec	Data	Func	Syst	Env
Mean	3.8	40.3	0.6	14.0	5.5	2.7	5.8	26.4	0	0.9
Mean D.	6.9	6.0	0.1	12.6	10.0	4.6	9.8	46.6	0.2	3.1
Lower	1.5	33.7	0.0	9.9	3.1	1.0	3.1	19.9	0	0.0
Upper	6.0	46.9	1.1	18.1	8.0	4.4	8.6	32.8	0	1.7
Std. dev.	10.1	29.5	2.5	18.4	11.1	7.4	12.4	29.1	0	3.9

Build, System and Environment: almost no defects of this type were found.

This may be due to the PSP course exercises:

- Small programs where the build/package is simple and the systems problems (configuration, timing, etc.) are unlikely to be present

Threat to validity: the programs of the PSP course are small.

Future work: try to find more of these defect types in TSP projects.

Defects Types Injected During Code (3)

	Docs.	Syn.	Buil	Assign	Inter.	Chec	Data	Func	Syst	Env
Mean	3.8	40.3	0.6	14.0	5.5	2.7	5.8	26.4	0	0.9
Mean D.	6.9	6.0	0.1	12.6	10.0	4.6	9.8	46.6	0.2	3.1
Lower	1.5	33.7	0.0	9.9	3.1	1.0	3.1	19.9	0	0.0
Upper	6.0	46.9	1.1	18.1	8.0	4.4	8.6	32.8	0	1.7
Std. dev.	10.1	29.5	2.5	18.4	11.1	7.4	12.4	29.1	0	3.9

Documentation, Interface, Checking and Data few defects of these types were found.

Defects of these types were injected (from 2.7% to 5.8%).

Defects Types Injected During Code (4)

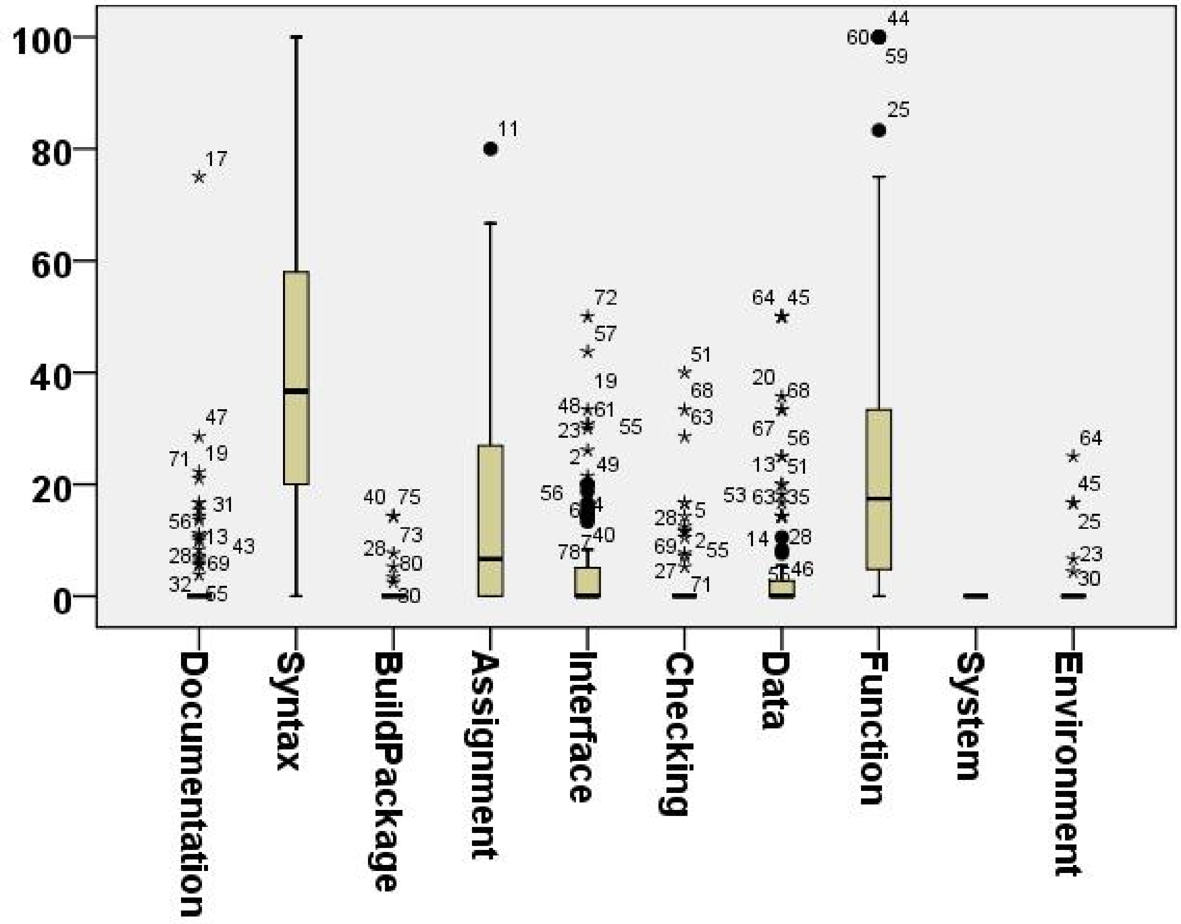
	Docs.	Syn.	Buil	Assign	Inter.	Chec	Data	Func	Syst	Env
Mean	3.8	40.3	0.6	14.0	5.5	2.7	5.8	26.4	0	0.9
Mean D.	6.9	6.0	0.1	12.6	10.0	4.6	9.8	46.6	0.2	3.1
Lower	1.5	33.7	0.0	9.9	3.1	1.0	3.1	19.9	0	0.0
Upper	6.0	46.9	1.1	18.1	8.0	4.4	8.6	32.8	0	1.7
Std. dev.	10.1	29.5	2.5	18.4	11.1	7.4	12.4	29.1	0	3.9

Syntax, Assignment and Function: many defects of this type were found.

These type of defects account for 80% of the Code defects.

Defects Types Injected During Code (5)

Variability between individuals and assignments



This suggests that individuals have different behaviors.

When Are the Defects Removed

For each engineer who injected Code defects, we identified the phases in which the engineers found these defects.

This work used a limited sample size that did not allow further analysis of removal phases.

Future work: when we get more data, examine the removal phases based on the defect types.

When Are the Defects Removed (2)

	CODE DEFECTS		
	CR	**Comp**	**UT**
Mean	62.0	16.6	21.4
Lower	55.0	11.7	15.4
Upper	69.0	21.6	27.3
Std. dev.	31.3	22.4	26.9

62% of the defects are found early in the CR phase.

However, 21% of the defects injected during Design escapes all phases prior to UT.

- How can we improve this? We first need to know the types of defects that escape to UT.

Cost to Remove the Defects Injected in Code

We analyze the differences in cost segmented by:

- Removal phase
- Defect type

It would also be interesting to segment and analyze both the removal phase and the defect type jointly.

Unfortunately, because of limited sample size after a two dimensional segmentation, we could not perform that analysis with statistical significance.

Future work: when we get more data, examine the segmentation in two dimensions.

Cost to Remove Defects Segmented by Phase

For each engineer, we calculated the average task time to removing a code defect in each of the different phases.

Because some engineers did not remove code defects in one or more phases, our sample size varied by phase.

Cost to Remove Defects Segmented by Phase (2)

	CODE DEFECTS		
	CR	**Com**	**UT**
Mean	1.9	1.5	14.4
Lower	1.5	1.1	9.8
Upper	2.3	1.9	19.0
Std. dev.	1.9	1.3	16.4

The cost of removing code defects in Unit Test are 7 times higher that the ones removed in Code Review.

Cost (in minutes) of "find and fix" defects injected during code segmented by removal phase

Cost to Remove Defects Segmented by Type

	Docs.	Syn.	Assign.	Inter.	Data	Func.
Mean	3.4	1.9	2.7	2.3	12.2	9.4
Lower	1.3	1.4	1.8	1.4	0.0	6.8
Upper	5.4	2.3	3.7	3.3	27.2	12.1
Std. dev.	4.2	2.0	3.1	2.2	32.9	10.7

Cost (in minutes) of find and fix defects injected during code segmented by type

Not enough data to present Build/Package, System or Environment defects.

Three clearly different groups

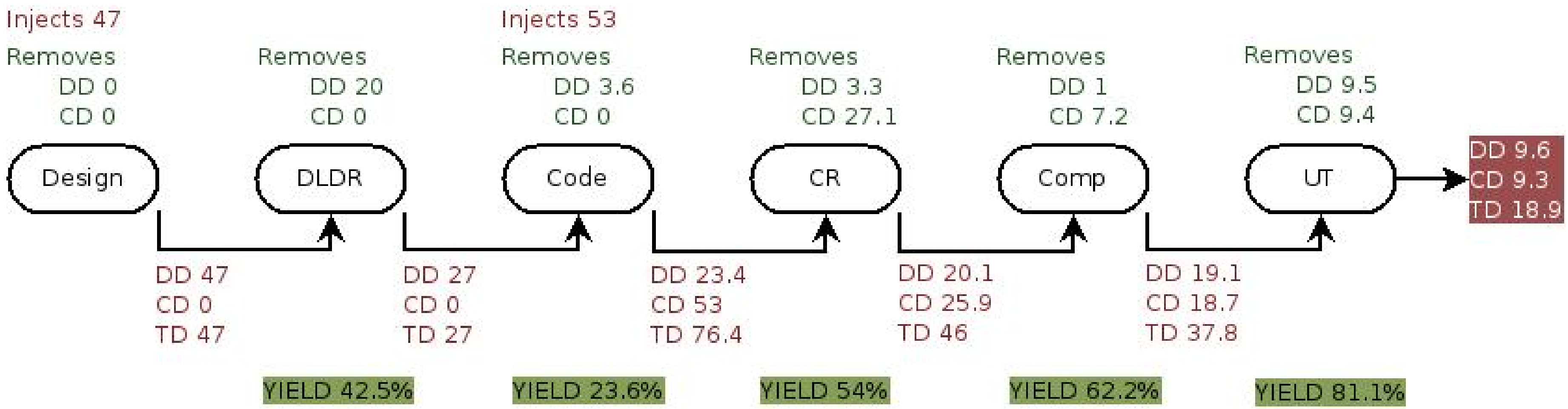
Using Data for Planning

We will assume that UT yield is 50%

We will assume that all the defects are injected in design or code (this is almost true)

Suppose we developed a program in which we injected 100 defects

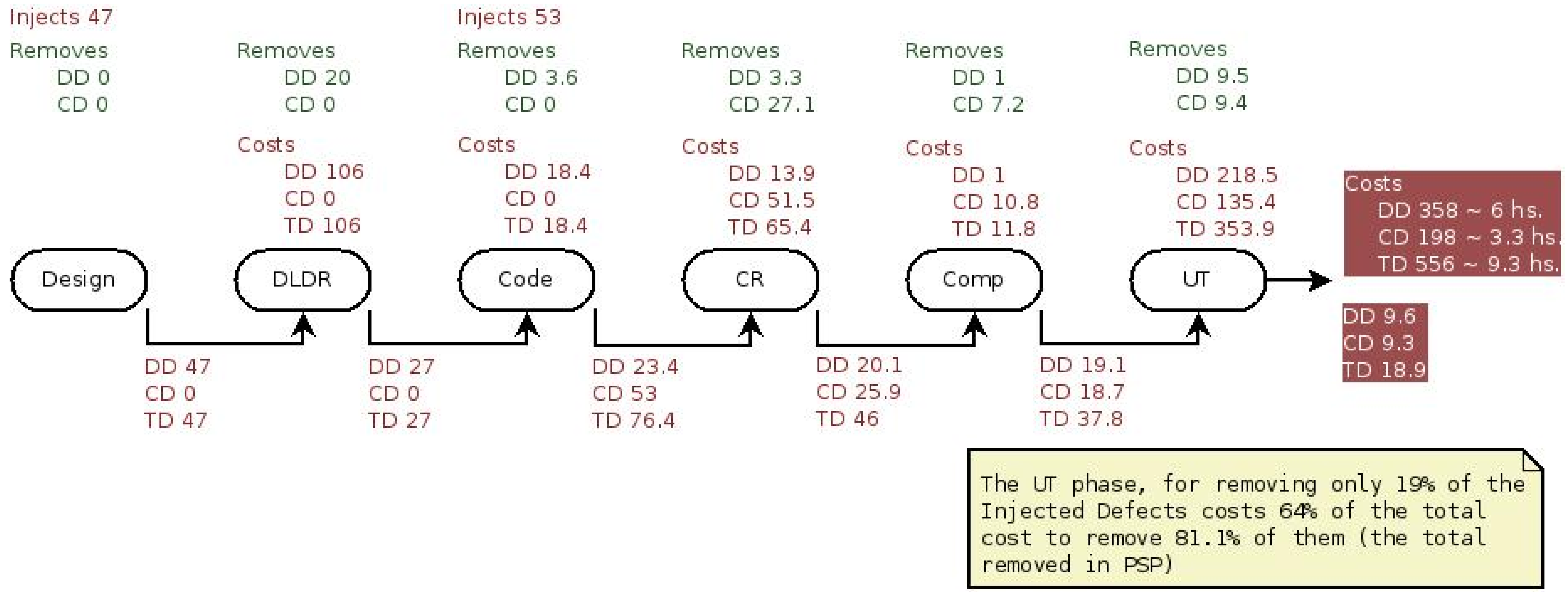
Using Data for Planning (2)



Excellent yield!

Consider that: Inspections are missing and that the data come from the courses (learning process)

Using Data for Planning (3)



Unit testing is really expensive

It is important to remove more defects before arriving to UT

Conclusions

(We observe a high variability between individuals and assignments).

Around 38% of the injected defects arrives to UT.

Phases prior to UT have similar defect find and fix costs

Defects are 5 times more expensive to find and fix in UT than in the earlier PSP phases for design defects and 7 times more expensive for code defects.

The estimated yield of the PSP (during the course) is 81%

Future work

Future work was mentioned during the presentation

The most important things we are planning to do is:

- Repeat the analysis with more data
- Characterize things that are pending
- Moving our research to TSP

We hope that this new analysis will enable us to analyze improvement opportunities to achieve better process yields

Questions

Diego Vallespir

dvallesp@fing.edu.uy

Grupo de Ingeniería de Software
Universidad de la República
Uruguay

William Nichols

wrn@sei.cmu.edu

Software Engineering Institute
Carnegie Mellon University
United States

Analysis of Code (and Design) Defect Injection and Removal in PSP

TSP Symposium 2012: Delivering agility with discipline
September 17-20, 2012 in St. Petersburg, Florida, USA