Analysis of 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 Design Defects
  – Defect types injected during design
  – When are the defects removed
  – Cost to remove the defects injected in design
“Playing” with the Data
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
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 design 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.
11 did not record any defects during design phase.
For our analysis, we sometimes use data from 92 engineers,
while other times using data from 83 engineers,
depending on the analysis needs.
Where the Defects Are Injected

<table>
<thead>
<tr>
<th></th>
<th>DLD</th>
<th>DLDR</th>
<th>Code</th>
<th>CR</th>
<th>Comp</th>
<th>UT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>46.4</td>
<td>0.4</td>
<td>52.4</td>
<td>0.3</td>
<td>0.03</td>
<td>0.5</td>
</tr>
<tr>
<td>Lower</td>
<td>40.8</td>
<td>0.2</td>
<td>46.7</td>
<td>0.0</td>
<td>0.0</td>
<td>0.2</td>
</tr>
<tr>
<td>Upper</td>
<td>52.0</td>
<td>0.7</td>
<td>58.1</td>
<td>0.7</td>
<td>0.09</td>
<td>0.9</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>27.2</td>
<td>1.7</td>
<td>27.4</td>
<td>1.8</td>
<td>0.3</td>
<td>1.8</td>
</tr>
</tbody>
</table>

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 Design Defects

In this work, we focus on design defects. We reduced our data set to the 83 engineers who injected design defects.

Based on our analysis, we will discuss:

– What types of defects are injected during design
– When those defects are removed
– The effort required to find and fix defects
Defects Types Injected During Design

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>6.9</td>
<td>6.0</td>
<td>0.1</td>
<td>12.6</td>
<td>10.0</td>
<td>4.6</td>
<td>9.8</td>
<td>46.6</td>
<td>0.2</td>
<td>3.1</td>
</tr>
<tr>
<td>Lower</td>
<td>3.3</td>
<td>2.5</td>
<td>0.0</td>
<td>8.2</td>
<td>5.1</td>
<td>1.6</td>
<td>6.3</td>
<td>39.7</td>
<td>0.0</td>
<td>0.9</td>
</tr>
<tr>
<td>Upper</td>
<td>10.5</td>
<td>9.5</td>
<td>0.3</td>
<td>17.0</td>
<td>15.0</td>
<td>7.6</td>
<td>13.3</td>
<td>53.5</td>
<td>0.6</td>
<td>5.3</td>
</tr>
<tr>
<td>Std. dev.</td>
<td>16.6</td>
<td>16.0</td>
<td>0.8</td>
<td>20.2</td>
<td>22.5</td>
<td>13.8</td>
<td>16.0</td>
<td>31.5</td>
<td>1.7</td>
<td>10.1</td>
</tr>
</tbody>
</table>

To improve the detection of design defects, we first want to know which types of defects were injected during the Design phase.
Defects Types Injected During Design

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>6.9</td>
<td>6.0</td>
<td>0.1</td>
<td>12.6</td>
<td>10.0</td>
<td>4.6</td>
<td>9.8</td>
<td>46.6</td>
<td>0.2</td>
<td>3.1</td>
</tr>
<tr>
<td>Lower</td>
<td>3.3</td>
<td>2.5</td>
<td>0.0</td>
<td>8.2</td>
<td>5.1</td>
<td>1.6</td>
<td>6.3</td>
<td>39.7</td>
<td>0.0</td>
<td>0.9</td>
</tr>
<tr>
<td>Upper</td>
<td>10.5</td>
<td>9.5</td>
<td>0.3</td>
<td>17.0</td>
<td>15.0</td>
<td>7.6</td>
<td>13.3</td>
<td>53.5</td>
<td>0.6</td>
<td>5.3</td>
</tr>
<tr>
<td>Std. dev.</td>
<td>16.6</td>
<td>16.0</td>
<td>0.8</td>
<td>20.2</td>
<td>22.5</td>
<td>13.8</td>
<td>16.0</td>
<td>31.5</td>
<td>1.7</td>
<td>10.1</td>
</tr>
</tbody>
</table>

Build and System: 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 Design (2)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mean</strong></td>
<td>6.9</td>
<td>6.0</td>
<td>0.1</td>
<td>12.6</td>
<td>10.0</td>
<td>4.6</td>
<td>9.8</td>
<td>46.6</td>
<td>0.2</td>
<td>3.1</td>
</tr>
<tr>
<td><strong>Lower</strong></td>
<td>3.3</td>
<td>2.5</td>
<td>0.0</td>
<td>8.2</td>
<td>5.1</td>
<td>1.6</td>
<td>6.3</td>
<td>39.7</td>
<td>0.0</td>
<td>0.9</td>
</tr>
<tr>
<td><strong>Upper</strong></td>
<td>10.5</td>
<td>9.5</td>
<td>0.3</td>
<td>17.0</td>
<td>15.0</td>
<td>7.6</td>
<td>13.3</td>
<td>53.5</td>
<td>0.6</td>
<td>5.3</td>
</tr>
<tr>
<td><strong>Std. dev.</strong></td>
<td>16.6</td>
<td>16.0</td>
<td>0.8</td>
<td>20.2</td>
<td>22.5</td>
<td>13.8</td>
<td>16.0</td>
<td>31.5</td>
<td>1.7</td>
<td>10.1</td>
</tr>
</tbody>
</table>

Documentation, Syntax, Assignment, Interface, Checking, Data, and Environment: few defects of these types were found.

Defects of these types were injected (from 3.1% to 12.6%).

Most of the remaining defect types (except Function) are in this category.
**Defects Types Injected During Design (3)**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>6.9</td>
<td>6.0</td>
<td>0.1</td>
<td>12.6</td>
<td>10.0</td>
<td>4.6</td>
<td>9.8</td>
<td>46.6</td>
<td>0.2</td>
<td>3.1</td>
</tr>
<tr>
<td>Lower</td>
<td>3.3</td>
<td>2.5</td>
<td>0.0</td>
<td>8.2</td>
<td>5.1</td>
<td>1.6</td>
<td>6.3</td>
<td>39.7</td>
<td>0.0</td>
<td>0.9</td>
</tr>
<tr>
<td>Upper</td>
<td>10.5</td>
<td>9.5</td>
<td>0.3</td>
<td>17.0</td>
<td>15.0</td>
<td>7.6</td>
<td>13.3</td>
<td>53.5</td>
<td>0.6</td>
<td>5.3</td>
</tr>
<tr>
<td>Std. dev.</td>
<td>16.6</td>
<td>16.0</td>
<td>0.8</td>
<td>20.2</td>
<td>22.5</td>
<td>13.8</td>
<td>16.0</td>
<td>31.5</td>
<td>1.7</td>
<td>10.1</td>
</tr>
</tbody>
</table>

Function: many defects of this type were found. Only defects of type Function were in this category. 46.6% of all the defects injected during design were Function defects.
Defects Types Injected During Design (4)
Variability between individuals and assignments

This suggests that individuals have different behaviors.
When Are the Defects Removed

For each engineer who injected Design 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)

<table>
<thead>
<tr>
<th></th>
<th>DLDR</th>
<th>Code</th>
<th>CR</th>
<th>Comp</th>
<th>UT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>53.4</td>
<td>9.6</td>
<td>8.9</td>
<td>2.5</td>
<td>25.7</td>
</tr>
<tr>
<td>Lower</td>
<td>45.8</td>
<td>5.7</td>
<td>5.2</td>
<td>0.0</td>
<td>19.3</td>
</tr>
<tr>
<td>Upper</td>
<td>61.0</td>
<td>13.4</td>
<td>12.5</td>
<td>5.2</td>
<td>32.0</td>
</tr>
<tr>
<td>Std. dev.</td>
<td>34.8</td>
<td>17.5</td>
<td>16.7</td>
<td>12.3</td>
<td>29.2</td>
</tr>
</tbody>
</table>

Half of the defects are found early in the DLDR phase.

However, one of every four 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.
When Are the Defects Removed (3)

Again we find a high variability between individuals.
Cost to Remove the Defects Injected in Design

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 design defect in each of the different phases. Because some engineers did not remove design defects in one or more phases, our sample size varied by phase. We excluded the cost of finding design defects in the Compile phase because we had insufficient data for that phase.
### Cost to Remove Defects Segmented by Phase (2)

<table>
<thead>
<tr>
<th></th>
<th>DLDR</th>
<th>CODE</th>
<th>CR</th>
<th>UT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>5.3</td>
<td>5.1</td>
<td>4.2</td>
<td>23.0</td>
</tr>
<tr>
<td>Lower</td>
<td>3.7</td>
<td>2.5</td>
<td>2.6</td>
<td>11.6</td>
</tr>
<tr>
<td>Upper</td>
<td>6.9</td>
<td>7.6</td>
<td>5.7</td>
<td>34.3</td>
</tr>
<tr>
<td>Std. dev.</td>
<td>6.6</td>
<td>6.7</td>
<td>4.1</td>
<td>42.0</td>
</tr>
</tbody>
</table>

Cost (in minutes) of “find and fix” defects injected during design segmented by removal phase.
Cost to Remove Defects Segmented by Phase (3)

The cost remained almost constant during DLDR, Code, and CR.

- We expected an increased defect find and fix cost in each phase.
- The design defects that are removed during DLDR cost approximately the same as removing the ones that escape from Design into Code and those that escape from Design into CR.
- However, we are calculating phase removal costs with different defects.

Unit Test cost are almost five times higher that DLDR costs.
Cost (in minutes) of find and fix defects injected during design segmented by type

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

Three clearly different groups:
- Group 1: approximately 5 minutes cost (Documentation, Syntax, Interface, Checking)
## Cost to Remove Defects Segmented by Type (2)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>5.6</td>
<td>4.3</td>
<td>7.3</td>
<td>5.4</td>
<td>4.9</td>
<td>11.0</td>
<td>9.3</td>
<td>10.5</td>
</tr>
<tr>
<td>Lower</td>
<td>3.6</td>
<td>1.8</td>
<td>1.9</td>
<td>2.5</td>
<td>2.2</td>
<td>2.2</td>
<td>6.9</td>
<td>3.0</td>
</tr>
<tr>
<td>Upper</td>
<td>7.6</td>
<td>6.7</td>
<td>12.7</td>
<td>8.2</td>
<td>7.5</td>
<td>19.8</td>
<td>11.7</td>
<td>17.9</td>
</tr>
<tr>
<td>Std. dev.</td>
<td>4.1</td>
<td>3.7</td>
<td>16.3</td>
<td>7.3</td>
<td>5.2</td>
<td>25.6</td>
<td>10.1</td>
<td>11.7</td>
</tr>
</tbody>
</table>

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

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

Three clearly different groups:

- Group 1: approximately 5 minutes cost (Documentation, Syntax, Interface, Checking)
- Group 2: a group only with Assignment defects
## Cost to Remove Defects Segmented by Type (3)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>5.6</td>
<td>4.3</td>
<td>7.3</td>
<td>5.4</td>
<td>4.9</td>
<td>11.0</td>
<td>9.3</td>
<td>10.5</td>
</tr>
<tr>
<td>Lower</td>
<td>3.6</td>
<td>1.8</td>
<td>1.9</td>
<td>2.5</td>
<td>2.2</td>
<td>2.2</td>
<td>6.9</td>
<td>3.0</td>
</tr>
<tr>
<td>Upper</td>
<td>7.6</td>
<td>6.7</td>
<td>12.7</td>
<td>8.2</td>
<td>7.5</td>
<td>19.8</td>
<td>11.7</td>
<td>17.9</td>
</tr>
<tr>
<td>Std. dev.</td>
<td>4.1</td>
<td>3.7</td>
<td>16.3</td>
<td>7.3</td>
<td>5.2</td>
<td>25.6</td>
<td>10.1</td>
<td>11.7</td>
</tr>
</tbody>
</table>

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

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

Three clearly different groups:

- Group 1: approximately 5 minutes cost (Documentation, Syntax, Interface, Checking)
- Group 2: a group only with Assignment defects
- Group 3: approximately 10 minutes cost (Data, Function, Environment)
Using Data for Planning

Suppose we developed a program in which we injected 100 defects during design
Using Data for Planning (2)

100 defects injected during design

- We will find 53 in DLDR, 10 in Code, 9 in CR, 2 in Compile and 26 in UT
- Using the average cost of find and fix by phase (we assume 1 minute in Compile) we have:

We will use 598 minutes to find and fix the 26 defects that escapes all the other phases

This represents the **61% of the total time** of removing the defects injected during design
Conclusions

(We observe a high variability between individuals and assignments)

Function type are the most common design defects (46%)

Function type defects are in the most costly find and fix group. (Data and Environment are also in this group)

Half of the defects injected in design are found early, in the process DLDR phase

25% of the design defects escape to UT.

Phases prior to UT have similar defect find and fix costs

Defects are 5 time more expensive to find and fix in UT than in the earlier PSP phases.
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
- Include an equal analysis to defects injected during coding

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 Design Defect Injection and Removal in PSP

TSP Symposium 2011: A dedication to excellence  
September 19-22, 2011 in Atlanta, Georgia, USA