The Plastic Surgery Hypothesis

E. Barr, Y. Brun, P. Devanbu, M. Harman, F. Sarro

TR available at http://www.cs.ucl.ac.uk/research/research_notes/

Federica Sarro
Research Associate, CREST centre
Department of Computer Science
University College London
f.sarro@ucl.ac.uk

London, 19/05/2014
2009

GenProg  Clearview
FixWizard  SemDiff
LibSync  PAR

...
Why Do They Work?

The content of new code can often be assembled out of code fragments that already exists elsewhere in the system under evolution (*plastic surgery*)

Contributions of Our Work

- Formal statement and validation of the Plastic Surgery Hypothesis (PSH)

- Large-scale, empirical study of the extent to which changes can be reconstructed from code already available during the development

- Analysis of the distribution of grafts in codebase to which a change applies
The Plastic Surgery Hypothesis

Changes to a code base contain snippets that already exist in the code base at the time of the change,

(1) Changes are repetitive related to the program to which they are applied (parent)
The Plastic Surgery Hypothesis

Changes to a code base contain snippets that *already exist* in the code base *at the time of the change*, and these snippets can be *efficiently found* and exploited.

1. Changes are repetitive related to the program to which they are applied (parent).
2. This repetitiveness is exploitable.
The Plastic Surgery Hypothesis

Changes to a code base contain snippets that already exist in the code base at the time of the change, and these snippets can be efficiently found and exploited.

(1) Changes are repetitive related to the program to which they are applied (parent)
(2) This repetitiveness is exploitable

• Previous works have found repetitiveness of changes across the project history
  – neglecting the primordial code that remained unchanged from the first version to the last

<table>
<thead>
<tr>
<th>Project</th>
<th>Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camel</td>
<td>26%</td>
</tr>
<tr>
<td>CXF</td>
<td>85%</td>
</tr>
<tr>
<td>HIVE</td>
<td>97%</td>
</tr>
<tr>
<td>....</td>
<td></td>
</tr>
<tr>
<td>Wicked</td>
<td>&lt;0.5%</td>
</tr>
</tbody>
</table>


Federica Sarro - f.sarro@ucl.ac.uk
The Plastic Surgery Hypothesis

Changes to a code base contain snippets that *already exist* in the code base *at the time of the change*, and these snippets can be *efficiently found* and exploited.

1. Changes are repetitive related to the program to which they are applied (parent)
2. This repetitiveness is exploitable

“How much of each change to a codebase can be constructed from the existing code?”

“What is the cost of finding these snippets?”

- Previous works have found repetitiveness of changes across the project history
  - neglecting the primordial code that remained unchanged from the first version to the last
  - ignoring the cost of finding redundancies

---


Federica Sarro - f.sarro@ucl.ac.uk
The Plastic Surgery Hypothesis

How much of each change to a codebase can be constructed from the existing code?

Graftability of a change
percentage of snippets in a commit that match a snippet in the codebase
- line-granularity
- exact matching, ignoring whitespace
The Plastic Surgery Hypothesis

How much of each change to a codebase can be constructed from the existing code?

Graftability of a change
percentage of snippets in a commit that match a snippet in the codebase

- line-granularity
- exact matching, ignoring whitespace
The Plastic Surgery Hypothesis

*What is the cost of finding these snippets?*

Density of a search-space

number of grafts found in a given search-space over its size
Corpus

- 12 open-source software projects
  - 15,723 commits from 2004 to 2012
  - 1,038,761 LOC

- 5 types of changes
  - Bug, Improvement, New Feature, Task, Custom Issue
RQ1: PSH Validation

Changes to a code base contain *snippets that already exist* in the code base *at the time of the change*, and these snippets can be *efficiently found and exploited*

- Changes are 43% graftable on average
- What percentage of the changes are x% graftable?
  - 16% of the changes are novel
  - 42% of the changes are more than 50% graftable
  - 10% of the changes are fully graftable
RQ1: PSH Validation

Changes to a code base contain snippets that *already exist* in the code base *at the time of the change*, and these snippets can be *efficiently found and exploited*

- Changes are 43% graftable on average
- What percentage of the changes are x% graftable?
  - 16% of the changes are novel
  - 42% of the changes are more than 50% graftable
  - 10% of the changes are fully graftable
RQ1: PSH Validation

Changes to a code base contain snippets that already exist in the code base at the time of the change, and these snippets can be efficiently found and exploited

- How do parents fare as possible source of grafts, when compared to nonparental ancestors and other projects?
  - non-parental ancestors contribute only 5% more grafts than the parents, while other projects only provide 9% on average
  - there is statistical significant difference in favour of the parent codebase with high effect size
RQ1: PSH Validation

Changes to a code base contain snippets that already exist in the code base at the time of the change, and these snippets can be efficiently found and exploited.

- How do parents fare as possible source of grafts, when compared to nonparental ancestors and other projects?
  - non-parental ancestors contribute only 5% more grafts than the parents, while other projects only provide 9% on average
  - there is statistical significant difference in favour of the parent codebase with high effect size
RQ1: PSH Validation

Changes to a code base contain snippets that already exist in the code base at the time of the change, and these snippets can be efficiently found and exploited.

- How do parents fare as efficient source of grafts, when compared to other projects?
  - The density of parent is significantly higher than those of the other search-spaces with a high effect size.
RQ1: PSH Validation

Changes to a code base contain snippets that already exist in the code base at the time of the change, and these snippets can be efficiently found and exploited.

• How do parents fare as efficient source of grafts, when compared to other projects?
  – The density of parent is significantly higher than those of the other search-spaces with a high effect size.
Further Insights

- Graftability by Commit Size
- Graftability by Commit Type
- Graft Contiguity
- Graft Clustering
Graftability by Commit Size

How does graftability vary with commit size?

A. Large commits are not graftable
B. Size does not matter
C. Small commits are fully graftable
D. Small commits are more graftable than the large ones
Graftability by Commit Size

How does graftability vary with commit size?

A. Large commits are not graftable
B. Size does not matter
C. Small commits are fully graftable
D. Small commits are more graftable than the large ones
Graftability by Commit Size

RQ2: How does graftability vary with commit size?

Regression model, graftability as response.

<table>
<thead>
<tr>
<th></th>
<th>Model 1</th>
<th>Model 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>0.29*** (0.00)</td>
<td>0.30*** (0.00)</td>
</tr>
<tr>
<td>Commit Size (Log scaled)</td>
<td>0.07*** (0.00)</td>
<td>0.08*** (0.00)</td>
</tr>
<tr>
<td>Improvement vs. Bug</td>
<td></td>
<td></td>
</tr>
<tr>
<td>New Feature vs. Bug</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task vs. Bug</td>
<td></td>
<td>-0.04*** (0.01)</td>
</tr>
<tr>
<td>Custom Issue vs. Bug</td>
<td></td>
<td>0.00 (0.01)</td>
</tr>
<tr>
<td>R²</td>
<td>0.04</td>
<td>0.04</td>
</tr>
<tr>
<td>Adj. R²</td>
<td>0.04</td>
<td>0.04</td>
</tr>
<tr>
<td>Num. obs.</td>
<td>15723</td>
<td>15723</td>
</tr>
</tbody>
</table>

*** p < 0.001, ** p < 0.01, * p < 0.05
Graftability by Commit Type

RQ3: Do different kinds of commits exhibit same graftability?

**Bug**
A problem which impairs/prevents the functions of the product

**New Feature**
A new feature of the product

**Task**
A project task that needs to be done

**Improvement**
An enhancement to an existing feature

**Custom Issue**
A custom issue type, as defined by the organization
Graftability by Commit Type

RQ3: Do different kinds of commits exhibit same graftability?

![Box plots showing graftability and commit size for different types of commits.](image)
Graft Contiguity

RQ4: To what extent are grafts contiguous?
Graft Contiguity

RQ4: To what extend are grafts contiguous?

How big are contiguous grafts? The figure reports the size (log scale) of both host and donor snippets.

How many host and donor snippets have the same size? The figure shows the number (sqrt scale) of those host and donor snippets having the same size.
Graft Clustering

RQ5: Are the donor snippets needed to graft a host snippet in the same file?

30% 9%
Changes to a code base contain snippets that already exist in the code base at the time of the change, and these snippets can be efficiently found and exploited.

1. Changes are 43% graftable on average. 16% are novel, 42% are more than 50% graftable, 10% are fully graftable.
2. The parent is a rich and effective search space.
3. The size and type of a commit have no significant practical impact on its graftability.
4. 53% of the snippets can be fully grafted from a single donor; in the remaining cases two donors are needed on average.
5. Donor snippets are often found in the same file, not requiring more extensive search.
Future Work

The complement of graftability measures the novelty of changes

- explore whether the feature set of novel changes is more predictable than we have found grafts to be
The Plastic Surgery Hypothesis

Changes to a code base contain snippets that already exist in the code base at the time of the change, and these snippets can be efficiently found and exploited.

RQ2: How does graftability vary with commit size?

RQ3: Do different kinds of commits exhibit same graftability?

RQ4: To what extend are grafts contiguous?

RQ5: Are the donor snippets needed to graft a host snippet in the same file?

A project task that needs to be done

A new feature of the product

An enhancement to an existing feature

A custom issue type, as defined by the organization

Federica Sarro - f.sarro@ucl.ac.uk