ESEG Empirical Software Engineering Group

Centro de Informática U·F·P·E

Universidade Federal de Pernambuco

HARMFULNESS OF CODE DUPLICATION

This briefing reports evidence on circunstances under which code duplication harms system quality based on scientific evidence from a systematic review.

FINDINGS

Figure 1 contains a causal model of the effects of duplication on system quality. The model has been constructed after analysis of the reviewed literature. The names in the figure are variables and the arrows are causal effects. Each arrow represents a separate hypothesis such as, “duplication increases cochange”. This should be read thus: in general, all else being equal, a system with more duplication will exhibit more cochange than an equivalent system with less duplication.

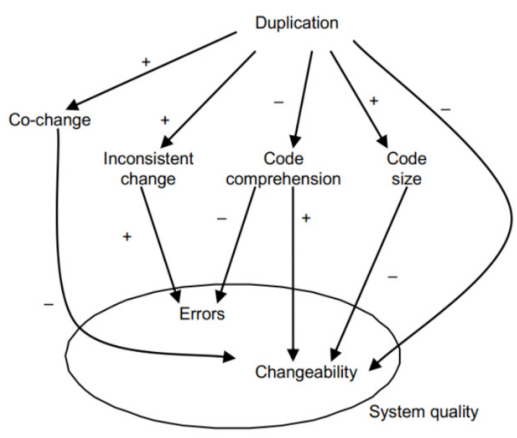

Figure 1: Causal model for code duplication

Co-Change

- A significant portion of clones are co-changed. For cochanging clones that cannot be refactored, consider using change propagating tools, such as Linked Editing.
- Of the clonesthat are removed in later versions, a minority is removed because of different changes, while the rest is presumably removed through refactoring.
- Clones that are not refactorable tend to live longer and are very likely to exhibit cochange. 45% to 55% of clone sets are cochanged during a long part of the life cycle of systems.
- Clones with classlevel granularity have more cochange than clones with smaller granularities. These also cause more increase in code size. These should be the first to refactor.
- Files between which there is a very high ratio of cloning are very often changed in the same release.

INCONSISTENT CHANGE

- A small portion of clones are changed inconsistently which is repaired in later versions. However only a part of the cases in which a change is applied to other occurrences later on (late propagation) stem from inconsistent change.
- In a small number of cases it was observed that the act of copying code led to errors. Consider using the approaches mentioned as errors in the code, especially when a lot of copypaste has happened.

CODE COMPREHENSION

- Programmers experience that duplication hinders code comprehension. Consider adding clone detection results to system documentation to aid comprehension.
- Issues with code ownership may introduce duplication.
- Changes that involve aspects (in the AOP sense) are especially hard to analyze. When aspectrelated code is cloned and changes to those aspects are expected in the future, use AOP techniques to avoid cochange.
- Duplicated code is not necessarily more complex or harder to understand than the equivalent nonduplicated code. Be careful when refactoring clones to not increase the complexity.

Code Size

- It is simple to measure the total size of code involved in a clone set. However it is difficult to estimate the size of code needed in an alternative solution, for example after refactoring. Removal of clones does not always result in reducing the code size. Be careful when refactoring clones to not increase the code size.
- Researchers should be aware of types of duplication that go beyond syntactical clones.
- Clones created by different programmers are harder to understand and harder to refactor than clones created by one programmer.

Keywords:

Code duplication
Code clone
Software Maintainance

Who is this briefing for?

Software engineers practitioners who want to make decisions about code duplication based on scientific evidence.

Where the findings come from?

All findings of this briefing were extracted from the systematic review conducted by Hordjik et al.

What is a systematic review?

cin.ufpe.br/eseg/slrs

What is included in this briefing?

The main findings of the original systematic review.

What is not included in this briefing?

Additional information not presented in the original systematic review.

Detailed descriptions about the studies analised in the original systematic review.

To access other evidence briefings on software engineering:

cin.ufpe.br/eseg/briefings

For additional information about ESEG:

cin.ufpe.br/eseg

ORIGINAL SYSTEMATIC REVIEW REFERENCE

Wiebe Hordijk, María Laura Ponisio, and Roel Wieringa. 2009. Harmfulness of code duplication: a structured review of the evidence. In Proceedings of the 13th international conference on Evaluation nd Assessment in Software Engineering (EASE’09), David Budgen, Mark Turner, and Mahmood Niazi (Eds.). British Computer Society, Swinton, UK, UK, 8897.