

# Extreme Programming And Software Clones

Eric Nickell and Ian Smith

*Palo Alto Research Center, Incorporated (PARC)*

*3333 Coyote Hill Rd, Palo Alto, CA, 94304*

*{nickell, iansmith}@parc.com*

## Abstract

*In this study, we ran a near-clone detector over software that has been developed while making heavy, light, or no use of extreme programming (XP) practices. XP is one of several agile methodologies gaining adherents, and may become a significant factor in industrial software development over the next decade.*

*We examined Java packages developed within our organization, using a tree edit distance algorithm on all pairs of methods within a single project. Initial results indicate that near-clones found in XP-developed software differ from other software in their frequency, kind, and location. Our study may signal to clone detection practitioners a coming shift in the types of clone detection that will be needed in the future, and to XP practitioners that there are continued opportunities for refactoring that many XP developers are missing.*

## 1. Introduction

The authors have adopted extreme programming practices [1] to develop most software over the last few years [2, 3]. When we started developing near-clone detection algorithms as part of a larger system, we naturally employed test-driven development and refactoring, two of the key practices of XP. Once smaller tests were passing, we needed a larger body of working code over which we had control as input, and we turned the system on itself. Fewer clones were reported than we expected in a code base of this size. We found similar rates when applying the algorithm to other code we had written, until we came to software which had been written during the period when we were just becoming XP-infected. This serendipitous find prompted us to broaden and harden our findings to include XP, non-XP, and semi-XP projects of comparable size.

In this study, we examine eight software projects of similar size (tens of thousands of lines of code), all developed in Java by teams of two to four PARC researchers over the last 4 years. For each project, we

did a pairwise comparison of methods within that project, using ANTLR [4] to produce the syntax trees, and a performance-enhanced variant of the tree edit distance algorithm of Zhang and Shasha [5]. From the edit distance and the tree sizes of the two method trees we produce a score, which is essentially the size of the largest alignment between the two methods, less an approximation of the amount of alignment that occurs when two unrelated trees are compared. For each project, we recorded the highest 100 scores for more detailed analysis.

## 2. Results

The XP projects produce significantly lower scores on our cloning index than either the non-XP or semi-XP projects (Table 1), and the project with the lowest score was written by the team with the longest experience and highest commitment to XP. The semi-XP and non-XP projects produced a wide range of scores; two of the non-XP projects produced lower average scores than one of the semi-XP projects. There is evidence that the XP projects gain some of their advantage in lower scores in part because XP developers tend to produce smaller methods.

Given the results shown in Table 1, we conducted a more careful audit of ardor and donquixote, the projects producing the lowest and highest scores, and less extensive audits of the other projects. The detections in ardor are mostly in smaller methods, and would produce 2-8 lines savings per occurrence if refactored, but for minimal gains in clarity. Three method pairs in different compilation units involve walking the same data structure, and were apparently written independent of each other, possibly indicating a need for a new abstraction.

Most of the near-clones detected in the semi-XP code were related to the test code rather than the production code, and the remainder were false positives. Nearly all (95%) of the detections for donquixote are typical near-clones: someone has done cut-and-paste with minimal changes, for methods up to 150 lines.

### 3. Conclusion

As expected, XP-developed software is more clone-resistant than software developed in a more traditional manner. Further research will be needed to understand cause and effect relationships.

XP practitioners should note the tendency to build a refactoring debt in the test code, while they are keeping the production code simple. This tendency should be resisted: Brittle tests can eventually slow a project as well as brittle production code.

Those working in the area of clone detection may note a shift in the number, locus, and nature of clones in new code as agile methodologies such as XP gain adherents. The clones will be fewer and they most likely will be found in the test code. When future clone detectors find refactorable areas of XP-developed software, it may indicate a need for a new abstraction, rather than stemming from cut-and-paste.

### 4. References

[1] K. Beck, *Extreme Programming: Embrace Change*, Addison-Wesley, Boston, 1999.

[2] S. Johnson, E. Nickell, J. Mao, and I. Smith, "Extreme Makeover: Bending the Rules to Reduce Risk Rewriting Complex Systems", *Proceedings Of Extreme Programming 2003*, Genoa, Italy, May 2003, pp. 307-314.

[3] V. Bellotti, N. Ducheneaut, M. Howard, C. Neuwirth, and I. Smith, "XP in Research Lab: The Hunt for Strategic Value", *Proceedings Of Extreme Programming 2002*, Alghero, Italy, May 2002 pp 55-61.

[4] <http://www.antlr.org/>

[5] K. Zhang, D. Shasha, "Simple, Fast Algorithm For the Editing Distance Between Trees and Related Problems", *SIAM Journal on Computing* (18), 1989, pp. 1245-1262.

Project	XP Use	Avg. Score	Avg. # Nodes	In Test Code	Exact clones	Exact Clones In Test Code
Alcatraz	None	110	154		5	
Donquixote	None	381	471		19	
Spandex	None	143	232		2	
Iago	Semi	165	249	95%	0	0
Pdq	Semi	81	167	76%	0	0
Ontology	Full	50	94	33%	5	3
Tantalus	Full	62	110	24%	1	0
Ardor	Full	33	56	91%	0	0
Ardor_Test	Full	65	102	N/A	4	4

Table 1

- "In test code" refers to the percent of methods that were found to be in test code for projects that we studied more closely.
- "100% clones in test code" refers to the number of identical methods found in test code (rather than production code). Note that the Non-XP projects did not have explicit test code so we do not measure this value for those projects.
- Ardor\_Test is the test code for the Ardor project. The testing was a separate project and so we were able to measure it separately from Ardor itself. The "In Test Code" value given for Ardor is the fusion of the results of measuring Ardor and Ardor\_Test.