Pattern Insight™ Clone Detection

pattern insight™

The fastest, most effective way to discover all similar code segments

What is Clone Detection?

Pattern Insight Clone Detection is a powerful pattern discovery technology using unique data mining and program analysis techniques. This allows Clone Detection to quickly and accurately discover all similar code segments that appear multiple times in a given code base. What's unique about Clone Detection is its pattern discovery capability — using unique data mining and program analysis techniques, Clone Detection quickly and accurately discovers all similar code segments that appear multiple times in a given code base.

These similar code segments may be the result of versioning, persistent branches, a copy-paste-modify approach to development, or the use of similar code in multiple products. The pattern discovery capability can find "fuzzy" matches, not just identical matches, since it tolerates modifications to code segments such as renaming of functions and variables, and insertions or deletions of statements.

Clone Detection Helps You Manage Large Code Bases

You can apply Clone Detection in a number of ways to help simplify the management of large code bases. Here are some applications that Clone Detection supports out of the box:

Finding copy-paste bugs

A common development practice is "copy-paste programming." Developers copy-paste code and then make updates to fit its new context, rather than writing the code from scratch. If a developer doesn't apply the updates consistently within the replicated block, he can introduce bugs. Bugs like these can't be easily discovered, either visually or by using other tools. Clone Detection helps you find such copy-paste bugs that can be very difficult to track down by any other means.

Refactoring support

Large, replicated code bases often get to the point where you have to refactor to manage explosion in code size, or to enable reuse. Clone Detection's pattern discovery capability can help you with your refactoring efforts.

Managing license compliance

Clone Detection can help you identify use of third-party code or open source code within your code base to avoid potential license violations.

Common to addressing all these challenges is that given a segment of code, you must be able to scan the code base, and discover all possible similar code segments or patterns within the code base—regardless of whether the similar code segments are identical or just fuzzy matches; whether the similarity exists at the granularity of a statement or a code segment; and whether it is semantic or syntactic. And you must be able to discover all similar segments fast enough to support analysis, regardless of the size of the code base.

pattern insight™

Why Isn't Search Adequate for Discovering Patterns?

Most existing solutions cannot tackle the problem since they rely on search. Search is not adequate for addressing the problem of discovering all code patterns within a large code base, especially if you're trying to discover both identical matches and fuzzy matches. This requires a scalable data mining capability with advanced program analysis, like Clone Detection.

Here's why. If you're trying to find all similar code segments, and looking for identical as well as fuzzy matches, the size of the problem space explodes even for a small code base. Search-based solutions are simply not adequate for addressing such a huge problem space—for a code base with millions of lines of code, it can take days or weeks just to find all identical matches. In contrast, Clone Detection can find all similarities—both identical and fuzzy matches—in a matter of minutes.

How Clone Detection is Unique

Clone Detection is a breakthrough solution that combines unique data mining technology with an understanding of code syntax and semantics to deliver deep pattern discovery and analysis capabilities. It tolerates function and variable name changes as well as insertion and deletion of statements. Compared to search-based solutions, Clone Detection is very fast—for example, on a commodity machine, it took only minutes to find all similar code segments in the Linux code base with around four million lines of code. It is also highly scalable, due to unique optimizations.

Clone Detection Features

Discover Code Patterns

The Code Patterns feature helps you analyze code bases and identify similarities with a high degree of accuracy.

Using the Code Patterns feature, you can:

- Find all similar code patterns within a single version or a branch of a code base
- Compare two versions or branches of a single code base, and identify all code segments that are similar or identical between them
- Compare two different code bases and identify all code segments that are similar or identical between them

When reviewing the results, you can sort the code patterns either by the number of copies or by the size of the pattern.

The Code Patterns feature helps you address a number of the challenges of managing large code bases:

Refactoring a large, replicated code base

If a code base has been continuously developed over time, it is likely that there is a great deal of replicated code due to branching, versioning, or developers using copy-paste techniques to simplify development. One of the goals of a refactoring effort is to identify similar code segments, and extract them into reusable components, modules or libraries to reduce the size of the code base, as well as ensure reuse.

If you've ever tried to refactor a large, replicated code base, you probably realize that a manual approach to finding what to refactor isn't fast or effective; but you've probably found as well that there aren't many tools that can help with the task. The Code Patterns feature can help jumpstart your refactoring effort in a way that wasn't possible before. You can quickly and accurately identify all similar code patterns, and then sort the results by number of patterns to identify functions that are candidates for refactoring into a reusable module.

Identifying use of third-party or open source code

If you want to identify open source code or third-party source code that is being used in your application in potential violation of license agreements, you can use the Code Patterns feature to identify such usage. You start by creating a tree of only the open source or third-party code. Then, you can compare this source code against your code base using the Code Patterns feature, and identify the similar code across the two source trees.

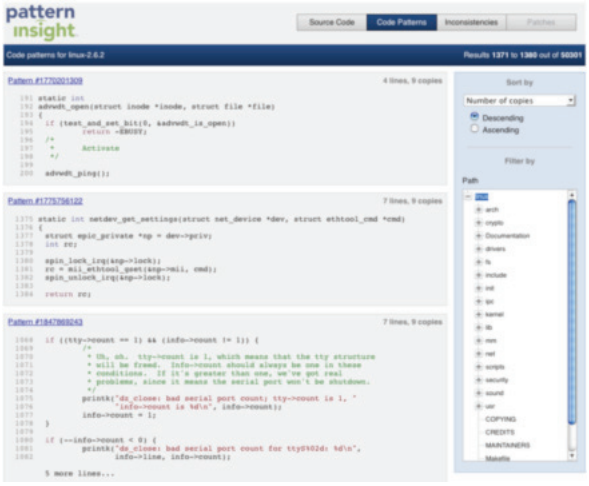

Figure 1: Code Patterns

Find Severe Copy-Paste Bugs

Clone Detection can help you identify copy-paste bugs in your code base. Developers often use copy-paste to save time when they need to quickly implement code that is largely similar to existing code, but want to make minor modifications to fit into its new context. It is easy to introduce bugs during this process—for example, when a developer copy-pastes a code segment from one place to another, he often needs to modify one variable name (e.g., variable i), to another (e.g., variable j). If the developer modifies the segment in most instances, but forgets to modify one or more places, a bug is introduced that is hard to recognize visually.

Figure 2 shows an example from the Linux kernel—the loop in lines 111-118 was copied from lines 92-99. In the new copy-pasted segment (lines 111-118), the variable prom_phys_total is replaced with prom_prom_taken in all cases except in line 117 (shown in bold font). As a result, the pointer prom_prom_taken[iter].theres_more incorrectly points to the element of prom_phys_total instead of prom_prom_taken. It's very difficult to detect copy-paste errors—since they are semantic errors, they cannot be detected visually or by any other static or dynamic analysis tool. Further, in many cases, copy-paste code is slightly modified by the addition or deletion of statements.

```
( linux-2.6.6/arch/sparc64/prom/memory.c )
68 void __init prom_meminit(void)
69 {
     .....
92   for(iter=0; iter<num_regs; iter++) {
93     prom_phys_total[iter].start_adr =
94       prom_reg_memlist[iter].phys_addr;
95     prom_phys_total[iter].num_bytes =
96       prom_reg_memlist[iter].reg_size;
97     prom_phys_total[iter].theres_more =
98       &prom_phys_total[iter+1];
99   }
     .....
111  for(iter=0; iter<num_regs; iter++) {
112    prom_prom_taken[iter].start_adr =
113      prom_reg_memlist[iter].phys_addr;
114    prom_prom_taken[iter].num_bytes =
115      prom_reg_memlist[iter].reg_size;
116    prom_prom_taken[iter].theres_more =
117      &prom_phys_total[iter+1];      // bug
118  }
     .....
143 }
```

Figure 2: Copy-paste error in Linux kernel

Clone Detection efficiently detects similar code patterns that appear multiple times. It then flags code segments with multiple copies that have inconsistent changes between the copies, indicating potential bugs. When the results are displayed, they are prioritized and ranked based on an algorithm that takes into account several parameters—the number of total modifications between replicated segments; the type of inconsistency found relative to the location it was found; and the number of inconsistencies within the segments.

Figure 3: Copy-paste inconsistencies

You can also view the inconsistencies in a split-screen view of both locations where the copy-paste inconsistency has been discovered, with replicated code segments clearly highlighted to show which sections have been replicated.

You can then review each inconsistency, and once you verify whether it's a real bug or a false positive, you can annotate it accordingly. Each copy-paste inconsistency can be annotated as Unverified, Real Bug, False Positive, or Ignore and stored with notes. Each bug is associated with a specific bug ID, so the bug will retain all of its annotation and comments the next time you use this feature on your code base.

How it Works

Clone Detection first parses the source code and then performs data mining with program analysis on the parsed code. All similar code segments are identified, followed by inconsistency detection. The entire procedure takes a few minutes to hours, depending on the code base size. Results are stored in a database for future use.

A web-based interface supports effective viewing and analysis of all patterns. Command line and API interfaces are available for scripting and custom integrations.

Server System Requirements	
Architecture	32-bit X86
Operating Systems	Linux (RHEL 4, RHEL 5, Ubuntu 7.10, 8.04)
Memory	4G minimum, 8G recommended
Disk Space	200 MB disk space for installation and 4-5X code base size for operational use and temporary storage
User Interface	Web browser (Chrome 8.0, IE 7.0, Firefox 3.0, Safari 3.0 and above)

pattern insight™

About Pattern Insight

Pattern Insight improves its customers' software quality and security. Its products are the first in the industry to make it significantly faster, easier and more accurate to find and address defects and security issues in source code. Designed for engineering teams, Pattern Insight's Code Assurance is a sophisticated software solution that identifies and removes all "known" (previously fixed) defects in code before it is released.

Pattern Insight was founded by a team of University of Illinois researchers with cutting-edge expertise in systems mining. Key investors include Venture Investors, John Lovitt and Dr. Kai Ki. Pattern Insight is headquartered in Mountain View, CA.

PatternInsight.com

Pattern Insight Headquarters
465 Fairchild Drive, Suite 209 Mountain View, CA 94043 P: 866 582 2655 F: 408 573 7855 E: info@patterninsight.com