Changed, Added and Deleted SLOC: A Simple Example

An important consideration for Software Project Managers is whether the requirements are being altered in a given project without their knowledge or consent. The impact of unauthorised or unscheduled departure from requirements can lead to considerable problems in maintaining and managing large source projects.

Having received requests from several major C/C++ and Java projects to measure ‘changes' in source code projects, Power Software was commissioned to manufacture an automated tool to measure changed lines of source code.

The first question we asked was 'are there any existing tools that can already accomplish this task?' The obvious starting place was 'Change Management' tools like CMVC (Configuration Management Version Control) systems. These tools indicate which files have changed between projects but do not gather and present the information in the format our customers’ required. One problem with CMVC systems was that a file shown to be ‘changed’ might have changed only by time and date. This is misleading if we are looking for actual changes in source code.

The next place we looked was at existing Software Metrics tools. These tools measure source code parameters including SLOC (source lines of code). However it is possible that a Software Metrics tool could report that the compared files or projects have the same total number of source lines but these could have changed intrinsically. The actual lines of code could have changed but the total number of lines will be unchanged so the 'changed line' is not detected.

For our customers the particular information required is 'changed source lines of code', 'added source lines of code' and 'deleted source lines of code'. This needs to be gathered across two releases of the same project to show exactly what has 'changed'. The information needs to be presented as 'changed lines' at the file and project level to assist management in their job of maintaining highest quality through proper control mechanisms.

The following example shows how the ‘Krakatau Project Manager’ tool meets these requirements.

This document will show how the values for changed, added and deleted source lines of code are derived by Krakatau Project Manager for C/C++ or Java code.

Consider the following example:

Figure 1 – Old and New File

Old File

```
void functionA(int k) {
        cout << "param: " << k << endl ;
}

void functionB() {
        //Unused function
        int j=0 ;
}

int main(int argc, char **argv) {

        int i ;
        int j=100 ;
        int count=0 ;

        for (i=0 ; i<10 ; i++) {

                // Calling functionA()
                functionA(i) ;

                while (j>0) {
                        j-= 5 ;
                }
        }
}
```

New File

```
void functionA(int k) {
        cout << "param: " << k << endl ;
}

int main(int argc, char **argv) {

        int i ;
        int j=100 ;
        int count=0 ;

        for (i=0 ; i<10 ; i++) {

                // Calling functionA()
                functionA(i) ;

                while (j>0) {
                        j-= 10 ;
                }

                while(count<10) {
                        count++ ;
                }
        }

}
```

Color Key

Added Source Line
Changed Source Line
Deleted Source Line

In the above example, since the absolute or total value for Lines of Code and Source Lines of Code has not changed many tools would not show changes for this file. However, Krakatau PM compares the two files (using a standard[1] comparison algorithm), and gives values for changes/additions and deletions with respect to the actual Source Lines of Code. *This indicates the reality of the activity within the source file or project.*

File Level

At the file level the metrics are produced from the file comparison as follows:

ADD_SLOC

From the example, in the new file these lines have been added:

```
while(count<10) {
        count++ ;
}
```

for which Krakatau PM would give ADD_SLOC = 3.

DEL_SLOC

From the example, the following lines existed in the old file but were no longer present in the new file:

```
void functionB() {
        //Unused function
        int j=0 ;
}
```

This would give DEL_SLOC = 3. Since comments are not treated as Source lines of code they are not included in the values.

CHG_SLOC

From the example only one line changed:

Figure 2 – Changed Line

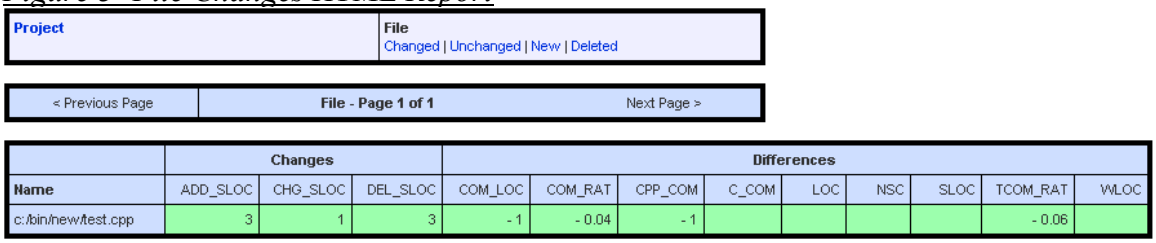
Old File	New File

`while (j>0) {`
`        j-= 5 ;`
`}`
	`while (j>0) {`
	`        j-= 10 ;`
	`}`

This would be counted as CHG_SLOC = 1.

The following is an HTML report showing the changes between the files produced from the source code used in the example above:

Figure 3 -File Changes HTML Report



Project	File Changed | Unchanged | New | Deleted

< Previous Page	File - Page 1 of 1	Next Page >

	Changes			Differences								
Name	ADD_SLOC	CHG_SLOC	DEL_SLOC	COM_LOC	COM_RAT	CPP_COM	C_COM	LOC	NSC	SLOC	TCOM_RAT	WLOC
c:/bin/new/test.cpp	3	1	3	- 1	- 0.04	- 1					- 0.06	

From this report (fig. 3) the values 3, 1 and 3 for ADD_SLOC, CHG_SLOC and DEL_SLOC respectively as described above can be seen.

The second section of this report (fig. 3) shows the differences between the metrics between the two files. In this example it can be seen that the value for CPP_COM (C++ style comments) has decreased by 1. This is explained because (as previously shown) the comment was removed in the function:

```
void functionB() {
        //Unused function
        int j=0 ;
}
```

As explained above the comment is not included in the DEL_SLOC value as it is not a source line of code, however, the Difference section of the report will highlight that the number of comments has gone down.

Project Level

Krakatau Project Manager also gives numbers for these metrics at the project level as well the file level.

Figure 4 -Project Changes HTML Report

Project	File Changed | Unchanged | New | Deleted

Changes

ADD_FILE	0
ADD_METH	0
ADD_SLOC	3
CHG_FILE	1
CHG_METH	1
CHG_SLOC	1
DEL_FILE	0
DEL_METH	1
DEL_SLOC	3

Differences

COM_LOC	- 1
COM_RAT	- 0.04
CPP_COM	- 1
C_COM	
LOC	
NCLASS	
NFILE	
NMETH	- 1
NSC	
SLOC	
TCOM_RAT	- 0.06
WLOC	+ 1

It can be seen (fig. 4) that at the project level again both "Changes" and "Difference" metrics are produced.

If we firstly consider the "Changes" metrics, added/deleted and changed SLOC are present – these values are obtained from the sum of their corresponding metrics at the file level. As well as these, the project level "Changes" metrics shows how many files and methods have been added/changed/and deleted. *This gives a quick high-level view of what has changed between two versions of a project.*

The "Difference" metrics are calculated from the taking the difference of the metrics for the old and the new projects. This again gives an indication at the project level of what is changing.

Further information can be obtained from support@powersoftware.com.

References

[1] "An O(ND) Difference Algorithm and its Variations", Eugene W. Myers, Algorithmica Vol. 1 No. 2, 1986, p 251.