UNIVERSITÀ DEGLI STUDI DI CATANIA
DIPARTIMENTO DI MATEMATICA E INFORMATICA
CORSO DI LAUREA IN INFORMATICA, PRIMO LIVELLO

Francesco Borzì

Strumenti per valutare la modularità del software

RELAZIONE PROGETTO FINALE

Relatore:
Chiar.mo Prof. Emiliano Tramontana

Anno Accademico 2012 - 2013

Indice

Capitolo 1. Introduzione

L'ingegneria del software è la disciplina che si occupa delle metodologie di sviluppo e dei processi produttivi finalizzati alla realizzazione di sistemi software. Essa si propone quindi una serie di obiettivi riguardanti l'evoluzione dello sviluppo del software dal punto di vista tecnologico e, soprattutto, dal punto di vista metodologico, ad esempio per migliorare il ciclo di vita del software.

Ci sono diversi aspetti da tenere in considerazione ai fini di migliorare il ciclo di vita di un software e gli sviluppatori devono tenere conto di questi aspetti durante la sua realizzazione. Un software ben progettato è più facile da comprendere, modificare, aggiornare e riutilizzare. Una buona progettazione del software permette quindi una durata di vita più lunga dell'applicazione, in quanto tutte le applicazioni necessitano di essere regolarmente aggiornate.

Il programmatore deve quindi “tenere sotto controllo” costantemente la progettazione del software e assicurarsi che essa sia adeguata.
Per venire incontro al programmatore esistono le **metriche software**, cioè degli standard per la misura di alcune proprietà del software. Tramite il calcolo delle metriche il programmatore può ottenere un immediato riscontro sulla qualità della progettazione del software.

Uno degli aspetti più importanti da tenere in considerazione per una buona progettazione del software è quello della **modularità**. Dotare un software di modularità significa suddividerlo in più moduli, ognuno dei quali svolge un preciso compito. Ogni modulo dev'essere tendenzialmente indipendente dagli altri, e deve avere il minor numero possibile di interazioni con gli altri moduli. L'obiettivo di questo paradigma di programmazione è in particolare semplificare lo sviluppo, i test e la manutenzione soprattutto di programmi di grosse dimensioni.

Raggiungere un alto livello di modularità in sistemi molto grandi spesso può essere un compito difficile, persino per sviluppatori dotati di una certa esperienza. Per semplificare tale procedimento si dovrebbero innanzitutto individuare le diverse categorie a cui i vari componenti del software appartengono, ovviamente in base alla funzione che essi svolgono.

Possiamo considerare i componenti principali di un'applicazione scritta in linguaggio Java, l'insieme di metodi chiamati dalle classi che compongono tale applicazione. Viene dunque naturale eseguire il processo di classificazione dei componenti di un applicazione Java raggruppando i metodi, chiamati dalle classi che compongono

l'applicazione, in diverse categorie. Ci si dovrebbe aspettare, in un'applicazione che gode di un buon livello di modularità, che ogni classe dell'applicazione chiami principalmente metodi di una determinata categoria, ovvero la categoria corrispondente all'ambito per cui quella classe è stata ideata. Altrimenti, se una classe dovesse chiamare tanti metodi provenienti da categorie differenti, la sua modularità potrebbe essere compromessa.

L'obiettivo del progetto software realizzato in collaborazione con il collega *Gabriele Gelardi,* è proprio quello di fornire uno strumento adeguato che sia in grado di esplorare un'applicazione Java, classificare i metodi chiamati da ogni classe di tale applicazione e, per ogni classe, calcolare una metrica che esprima il livello di modularità.

Tale metrica è **fragm** [1] e quantifica l'eterogeneità di una classe nel seguente modo: se una classe chiama dei metodi appartenenti a diverse categorie, allora ha un valore di frammentazione che cresce all'aumentare del numero delle categorie.

Una volta constatato un alto livello di fragm per una classe, è possibile effettuare un operazione di refactoring per migliorare la progettazione del software. Per esempio spostando i gruppi di metodi appartenenti a categorie diverse in altre classi.

Riassumendo, più una classe contiene metodi che svolgono la stessa funzione (quindi appartenenti alla stessa categoria) e minore sarà il livello di fragm.

Per calcolare tale metrica, ci serviremo sia della programmazione sequenziale che di quella parallela, confrontando i tempi impiegati da ognuna di esse.

Capitolo 2. Obiettivi

Attualmente le metriche più note tengono conto solo delle interazioni tra classi/oggetti senza tenere conto delle diverse categorie a cui le classi dovrebbero appartenere.

Il nostro scopo è quello di separare "il più possibile" parti di codice che operano in contesti totalmente differenti, per garantire un alto tasso di modularità al sistema software.

Una volta individuati dunque i diversi contesti in cui una o più classi possono operare, serve uno strumento per misurare l'effettiva indipendenza delle porzioni di codice operanti in contesti differenti. A questo scopo definiamo come **categoria** un insieme di metodi che condividono gli stessi scopi e le stesse caratteristiche.
Data una classe *cl* e una categoria *k* definiamo inoltre come $\mathbf{ITC_k(}\boldsymbol{cl}\mathbf{)}$ il numero di metodi chiamati dalla classe *cl* appartenenti alla categoria *k*.

Supponendo di avere in tutto *K* categorie, calcoliamo il valore di *fragm* per la classe *cl* come segue:

$$fragm(cl) = \frac{1}{\sqrt{K}-1}\left(\frac{\sum_{k=1}^{K} ITC_k(cl)}{\sqrt{\sum_{k=1}^{K} ITC_k(cl)^2}} - 1\right)$$

Si pone intuitivamente il problema di determinare i diversi contesti in cui un sistema software può operare, ovvero classificare i metodi invocabili dall'applicazione nelle diverse categorie.

Una possibile organizzazione di metodi in categorie è stata suggerita [2] classificando innanzitutto i metodi forniti dalla piattaforma sottostante, ovvero, nel caso di applicazioni Java, i metodi della **Java standard library**. Per esempio, il package *java.net* contiene delle funzionalità orientate alla rete, il package *java.io* invece offre una serie di funzionalità legate all'input/output, filesystem, etc...

In questo modo siamo in grado di capire che, se una data classe chiama un certo numero di metodi del package *java.net*, probabilmente il compito di quella classe è interfacciare l'applicazione alla rete e dunque, per mantenere un buon livello di modularità, non dovrebbe svolgere altre funzioni bensì quest'ultime verranno svolte da un'altra classe. In particolare i metodi della standard library di Java sono stati organizzati in 9 categorie, nella seguente maniera:

CONCURRENT

java.concurrent.Executors
java.concurrent.ExecutorService
java.concurrent.Future
java.concurrent.Thread
java.concurrent.Runnable

IO

java.io.File
java.io.FileReader
java.io.FileWriter
java.io.FileInputStream
java.io.FileOutputStream
java.io.BufferedReader
java.io.BufferedWriter
java.io.StringWriter
java.io.PrintStream
java.io.PrintWriter
java.io.OutputStreamWriter

STRING

java.lang.Character
java.lang.String
java.lang.StringBuilder
java.lang.StringBuffer

COMPUTE

java.lang.Byte
java.lang.Short
java.lang.Integer
java.lang.Long
java.lang.Float
java.lang.Double
java.lang.Enum

EXCEPTION

java.lang.Exception
java.lang.RuntimeException
java.lang.IllegalArgumentException
java.lang.ClassNotFoundException
java.lang.IllegalAccessException
java.lang.InvocationTargetException

REFLECT

java.reflect.Class
java.reflect.Constructor
java.reflect.Method
java.reflect.Field
java.reflect.Modifier
java.reflect.Array
java.reflect.ParameteraizedType

ANNOTATION

java.annotation.Annotation
java.annotation.RetentionPolicy
java.annotation.Target

CONTAIN

java.util.List
java.util.LinkedList
java.util.Arrays
java.util.ArrayList
java.util.Map
java.util.HashMap
java.util.Collection
java.util.Collections
java.util.Set
java.util.Vector
java.util.HashTable
java.util.HashSet
java.util.WeakHashMap
java.util.Enumeration

NET

java.net.DatagramPacket
java.net.DatagramSocket
java.net.Socket
java.net.ServerSocket

Il nostro compito è quindi quello di realizzare un sistema software che sia in grado di:

- esplorare uno o più **Java class file**, cioè file con estensione *.class* che contengono il Java bytecode (istruzioni eseguite dalla Java Virtual Machine) che costituisce una classe;

- analizzarne il contenuto verificando, per ogni classe, quali metodi di altre classi vengono chiamati;

- identificare, per ogni metodo chiamato da ciascuna classe analizzata, la categoria di appartenenza;

- conteggiare, per ogni classe analizzata, il numero di metodi chiamati appartenenti ad ogni categoria;

- calcolare il valore di ***fragm*** per ogni classe analizzata e restituirlo in output;

Bisogna considerare inoltre che il calcolo del fragm avviene mediante l'elaborazione di un array di dati di dimensione pari al numero *K* di categorie prestabilite, nel nostro caso 9. Considerando inoltre che il sistema software può prendere in input e analizzare più di un Java class file, quindi più di una classe, il calcolo del fragm avviene elaborando una matrice di dati ***K*** x ***N*** dove N è il numero di classi analizzate.

Dal momento in cui le applicazioni di grandi dimensioni sono formate da un elevato numero di classi, il nostro sistema software si potrebbe trovare ad elaborare matrici **K x N** di grandi dimensioni (a causa dell'elevato valore di N), questo ci spinge a provare ad eseguire tali elaborazioni di dati tramite il **calcolo parallelo**.

In generale, il calcolo parallelo è l'esecuzione simultanea del codice sorgente di uno o più programmi (diviso e specificamente adattato) su più microprocessori o più core dello stesso processore allo scopo di aumentare le prestazioni di calcolo del sistema di elaborazione.

Per i nostri scopi, ci serviremo del calcolo parallelo per calcolare il valore di fragm per ogni classe analizzata. Confronteremo il tempo impiegato per eseguire i calcoli in maniera sequenziale con quella parallela, sia per sistemi di piccole dimensioni che per sistemi di grandi dimensioni (dotati di un numero molto elevato di classi).

Capitolo 3. Realizzazione e dettagli del sistema software

Il nostro sistema software dev'essere innanzitutto in grado di aprire i class file ed esplorarli. Per far ciò ci siamo serviti di un'applicazione open source che esegue già questo lavoro: ***ckjm***.

L'applicazione Java ***ckjm - Chidamber and Kemerer Java Metrics*** (sito web: http://www.spinellis.gr/sw/ckjm/) è stata realizzata da *Diomidis Spinellis*. Essa prende in input uno o più Java class files e, tramite le librerie ***BCEL*** (*Byte Code Engineering Library*), esegue un' esplorazione approfondita del codice ai fini di calcolare un set di metriche note (WMC, DIT, NOC, CBO, RFC, LCOM, etc...). Il codice di ckjm è reperibile presso **GitHub** al seguente indirizzo:

https://github.com/dspinellis/ckjm

In particolare **GitHub** è un servizio web di hosting per lo sviluppo di progetti software che usa **Git**, il sistema software di controllo di versione distribuito creato da *Linus Torvalds*. GitHub può essere considerato come un vero e proprio social network per sviluppatori, contenente diverse funzionalità che agevolano la realizzazione di software in gruppo.

github
SOCIAL CODING

GitHub permette di creare con molta semplicità dei **fork**. Un fork, nell'ambito dell'ingegneria del software, indica lo sviluppo di un nuovo progetto software che parte dal codice sorgente di un altro già esistente, ad opera di uno o più programmatori.

Noi abbiamo creato un fork dell'applicazione *ckjm* su cui abbiamo basato il nostro lavoro, il repository del nostro fork di ckjm è reperibile presso questo indirizzo:

https://github.com/ShinDarth/ckjm

Il package ***gr.spinellis.ckjm*** è composto dalle seguenti classi principali:

- ***MetricsFilter***: contiene il *main* dell'applicazione;

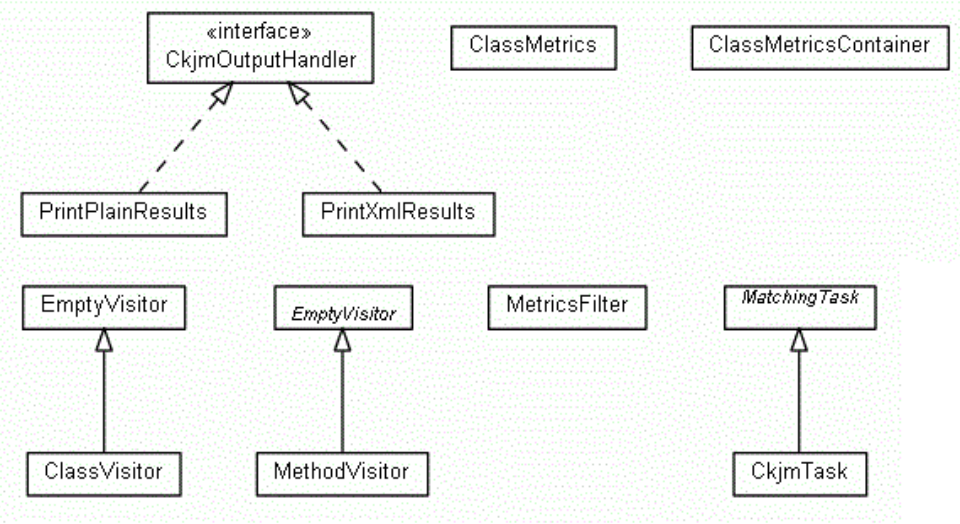
- ***MethodVisitor***: esegue l'esplorazione dei metodi tramite le librerie *bcel*;

- ***ClassVisitor***: esegue l'esplorazione delle classi tramite le librerie bcel e chiamando a sua volta *MethodVisitor*;

- ***ClassMetrics***: ogni istanza di questa classe tiene traccia dei risultati dei calcoli (le metriche *Chidamber and Kemerer*) che l'applicazione esegue;

- ***ClassMetricsContainer***: è un contenitore di istanze della classe ClassMetrics;

- ***PrintPlainResults***: gestisce la stampa in output dei risultati.



Ovviamente, per raggiungere i nostri fini, nel nostro fork abbiamo modificato parzialmente alcune delle classi originali e ne abbiamo introdotte delle nuove. Possiamo suddividere il lavoro svolto per realizzare il nostro programma in quattro componenti principali:

- Raccolta informazioni
- Visione globale del sistema analizzato
- Conteggio metodi per categoria
- Calcolo fragm parallelo

3.1 Raccolta informazioni

Innanzitutto abbiamo bisogno dell'elenco dei metodi chiamati da ogni classe analizzata dal nostro programma. Ciò si ricava tramite *ClassVisitor,* infatti durante l'esplorazione di una classe, essa aggiorna una variabile *signature* ogni volta che incontra un nuovo metodo chiamato dalla classe analizzata. La variabile *signature* conterrà un'informazione riguardante il package, la classe e il nome del metodo, nel seguente formato:

package.classe.metodo()

La nostra applicazione salverà, per ogni classe analizzata, l'elenco di tutte le *signature*, ottenendo un elenco del tipo:

package1.classeA.metodo1()
package1.classeA.metodo2()
package2.classeC.metodo6()
...

A questo punto abbiamo realizzato la classe ***CalledClassPath*** per memorizzare queste informazioni, ovvero la lista dei metodi chiamati da ogni classe analizzata, organizzandole nella seguente maniera: più precisamente, una "*CalledClassPath*" rappresenta l'entità classe che viene "chiamata" (ovvero che contiene uno o più metodi chiamati dalla classe che stiamo analizzando).

Il suffisso "path" serve ad indicare che, in questo caso, la classe è identificata dall'intero percorso più il nome effettivo della classe (ad esempio *package1.classeA*), tale informazione viene memorizzata nella variabile *className* di *CalledClassPath*.

Ogni istanza di *CalledClassPath* contiene un elenco di tutti i metodi che \ 1appartengono alla classe che essa rappresenta e, per ciascun metodo, il numero di volte in cui esso viene chiamato. Ovviamente non contiene nomi di metodi che non vengono mai chiamati.

Il metodo ***getCalledMethodsCount()*** della classe *CalledClassPath* restituisce la somma del numero di volte in cui viene chiamato ogni metodo appartenente alla classe che *CalledClassPath* rappresenta.

Per chiarire meglio il funzionamento della classe *CalledClassPath* supponiamo che una classe che stiamo analizzando chiami una serie di metodi, espressi in un elenco (ricavato tramite la variabile *signature* come spiegato precedentemente) nella forma:

package1.classeA.metodo1()
package1.classeA.metodo1()
package1.classeA.metodo1()
package1.classeA.metodo2()
package1.classeB.metodo4()
package1.classeB.metodo4()
package2.classeC.metodo6()
package2.classeC.metodo8()
package1.classeB.metodo5()

Allora, in questo caso, i dati verranno organizzati creando tre istanze della classe CalledClassPath, ovvero:

package1.classeA
package1.classeB
package2.classeC

L'istanza *package1.classeA* conterrà i seguenti valori:

metodo1() = 3
metodo2() = 1

getCalledMethodsCount() → 4

L'istanza *package1.classeB* conterrà i seguenti valori:

metodo4() = 2
metodo5() = 1

getCalledMethodsCount() → 3

L'istanza *package2.classeC* conterrà i seguenti valori:

metodo6() = 1
metodo8() = 1

getCalledMethodsCount() → 2

Ad ogni classe analizzata viene quindi associato un array di *CalledClassPath*, nel

caso del precedente esempio questo array avrebbe lunghezza 3 (in quanto contiene tre classi: ***package1.classeA***, ***package1.classeB*** e ***package2.classeC***). Tale array viene memorizzato all'interno dell'istanza di *ClassMetrics* corrispondente alla classe analizzata.

È possibile scegliere di far stampare nella prima parte dell'output del nostro programma, per ogni classe analizzata dal programma, i dati appena descritti, seguiti da un ulteriore informazione aggiuntiva opzionale riguardante il numero di metodi chiamati da ogni classe analizzata appartenenti ad un particolare package scelto dall'utente all'avvio del programma, tale informazione è una metrica detta ***fan-in***. Questo viene ovviamente calcolato, per ogni classe che l'applicazione prende in input, a partire dai dati memorizzati negli array di *CalledClassPath*.

3.2 Visione globale del sistema analizzato

Per avere una visione globale del sistema analizzato, abbiamo deciso di implementare una sorta di struttura ad alberi che indichi quante volte, in totale, ogni metodo viene chiamato considerando **tutte** le classi analizzate. In questa struttura abbiamo un albero per ogni package contenente almeno un metodo che viene chiamato almeno una delle classi analizzate.

In ogni albero il package rappresenta la radice, i figli della radice rappresentano le classi e i figli delle classi, le foglie, rappresentano i metodi. A questo fine, abbiamo introdotto le classi:

- ***PathNode*** le cui istanze rappresentano i nodi non-foglie dell'albero;
- ***MethodNode*** le cui istanze sono le foglie dell'albero;
- ***DataHandler*** che gestisce l'intera struttura dati.

La classe *DataHandler* è progettata utilizzando *Singleton*, un design pattern creazionale che garantisce che di tale classe esista una sola istanza.

Per mostrare come funziona la struttura dati appena descritta, prendiamo come esempio il caso in cui il programma prende in input due classi, *C1* e *C2*. Supponiamo che la classe *C1* chiama i seguenti metodi:

package1.classeA.metodo1()
package1.classeA.metodo1()
package1.classeA.metodo2()
package1.classeB.metodo4()
package1.classeB.metodo4()
package2.classeC.metodo6()

e che la classe *C2* chiama i metodi:

package1.classeA.metodo1()
package1.classeA.metodo2()
package1.classeA.metodo2()
package1.classeA.metodo2()
package2.classeC.metodo6()
package3.classeD.metodo9()

La struttura finale sarà l'unione di tutte le informazioni riguardanti le classi analizzate (nel nostro esempio due, C1 e C2). In questo caso abbiamo 3 alberi, visto che tutti i metodi chiamati dalle classi *C1* e *C2* provengono, in totale, da tre package diversi.

Il primo albero ad esempio, corrispondente a *package1*, avrà due nodi figli: *classeA* e *classeB*. A loro volta *classeA* avrà due foglie come figli, cioè *metodo1()* e *metodo2()*, invece la *classeB* ha solo una foglia, *metodo4()*.

La foglia corrispondente a *metodo1()*, conterrà il valore 3, poiché metodo1() viene chiamato in totale 3 volte (sommando le chiamate di C1 con quelle di C2). Analogamente *metodo2()* e *metodo4()* avranno rispettivamente valori 4 e 2.

Passando alle classi, *classeA* avrà come valore la somma dei valori dei propri figli *metodo1()* e *metodo2()*, ovvero 7. Invece *classeB* avendo un solo figlio, ha il medesimo valore di quest'ultimo, cioè 2.

Applicando lo stesso procedimento anche per gli alberi aventi come radici *package2* e *package3*, con i dati del precedente esempio otteniamo in output un risultato del tipo:

```
package1 [9]
{
        classeA [7]
        {
                metodo1 [3]
                metodo2 [4]
        }
        classeB [2]
        {
                metodo4 [2]
        }
}

package2 [2]
{
        classeC [2]
        {
                metodo6 [2]
        }
}

package3 [1]
{
        classeD [1]
        {
                metodo9 [1]
        }
}
```

Questa visione del sistema ci permette di apprendere rapidamente informazioni riguardanti il numero di volte in cui l'intero sistema effettua chiamate verso metodi di altre classi e package.

3.3 Conteggio metodi per categoria

Per organizzare le classi che ricevono chiamate ai propri metodi da parte delle classi che l'applicazione analizza, abbiamo realizzato le classi:

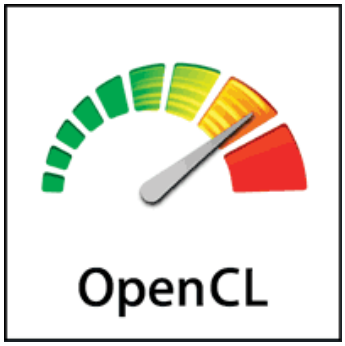
- ***Category***, le cui istanze rappresentano le singole categorie. È caratterizzata da un nome e dall'insieme di classi che le appartengono. Fornisce dei metodi che consentono di aggiungere nuove classi a se stessa e verificare se una data classe appartiene o meno alla categoria che rappresenta.

- ***CategoryHandler***, crea e gestisce le categorie (istanze di Category). Processa le classi analizzate assegnando ad ognuna di esse il valore di appartenenza ad ogni categoria, cioè il valore corrispondente al numero di chiamate che la classe analizzata effettua verso metodi appartenenti a una data categoria. Tramite questi dati calcola il *fragm*. Di questa classe ci serve un'unica istanza, pertanto anch'essa è *Singleton*.

3.4 Calcolo fragm parallelo

Grazie al calcolo parallelo possiamo eseguire simultaneamente il nostro codice sorgente su più microprocessori (o su più core dello stesso processore) allo scopo di aumentare le prestazioni del nostro sistema di elaborazione e quindi ridurre i tempi del calcolo del fragm, per esempio destinando il calcolo del fragm di ogni classe in input ad un core differente.

A questo fine, ci siamo serviti di **OpenCL**, una libreria basata sul linguaggio di programmazione C che può esser eseguito su una molteplicità di piattaforme (CPU, GPU, e altri tipi di processori …).



Ma come abbiamo potuto servirci di OpenCL se la nostra applicazione è scritta in linguaggio Java? La risposta a questa domanda è **Aparapi**: una API che consente di effettuare il calcolo parallelo in Java convertendo a runtime il Java bytecode in OpenCL.

Nella versione seriale, il calcolo della metrica *fragm* avviene mediante un doppio ciclo *for*: il ciclo più esterno scorre tutte le classi in input su cui vogliamo calcolare il valore di fragm; mentre quello più interno scorre tutte le categorie e raccoglie tutte le informazioni necessarie per calcolare *fragm* (il numero di metodi che la classe chiama appartenenti ad ogni categoria).

Il seguente frammento di codice mostra quanto appena descritto:

```
for (int inputClassIdx = 0; inputClassIdx < fragm.length; inputClassIdx++)
{
    itc = itc_sqr = 0;

    for (int k = 0; k < categories.length; k++)
    {
        itc += matrix[inputClassIdx][k];
        itc_sqr += Math.pow(matrix[inputClassIdx][k], 2);
    }

    fragm[inputClassIdx] = coeff * (itc/((float)Math.sqrt(itc_sqr)) - 1);
}
```

Nella versione parallela invece, ogni thread si occupa di calcolare il fragm di una class. È quindi presente un unico ciclo *for*, in quanto quello più esterno non è necessario.

Di seguito riportiamo il codice del metodo parallelFragm(), all'interno del quale è definito il kernel del calcolo parallelo, ovvero la parte di codice che sarà condivisa da ogni thread distinto.

```java
public float[] parallelFragm()
{
    final int n_$costant$ = fragm.length;
    final int[] matrix2_$costant$ = new int[matrix.length*matrix[0].length];
    final float coeff2_$costant$ = coeff;
    final float fragm2[] = new float[matrix.length];
    final int colLen_$costant$ = matrix[0].length;

    for (int i = 0; i < matrix.length; i++)
        for (int j = 0; j < matrix[i].length; j++)
            matrix2_$costant$[i*matrix[0].length+j] = matrix[i][j];

    Kernel kernel = new Kernel()
    {
        @Override public void run()
        {
            int inputClassIdx = getGlobalId();

            if (inputClassIdx >= n_$costant$)
                return;

            int itc = 0, itc_sqr = 0;

            for (int k = 0; k < colLen_$costant$; k++)
            {
                int curr = matrix2_$costant$[inputClassIdx*colLen_$costant$+k];

                itc += curr;
                itc_sqr += (curr*curr);
            }

            fragm2[inputClassIdx] = coeff2_$costant$ * ((itc/FloatMath.sqrt(itc_sqr)) - 1);
        }
    };

    kernel.setExecutionMode(EXECUTION_MODE.CPU);
    kernel.execute(Range.create(n_$costant$));

    kernel.dispose();
    return fragm2;
}
```

Questa volta il valore di *inputClassIdx*, che come nella versione seriale identificava la classe su cui si sta calcolando il *fragm*, dipende non più da un ciclo *for* ma dall'indice del thread.

Ad ogni passaggio, abbiamo confrontato i tempi di esecuzione dell'algoritmo in versione parallela su GPU con quello seriale. A questo fine eseguivamo entrambi i metodi n volte, conteggiando, per ognuno di essi, il tempo impiegato e dividendolo per n.

Inizialmente, in ognuno degli *n* lanci del metodo parallelo, nel conteggio del tempo erano prese in considerazione anche le fasi (iniziali e finali) di scambio di dati tra host e device. Ai nostri fini è stato utile invece conteggiare solo una volta il tempo impiegato per scambiare i dati tra host e device.

Prima:

```
for(int i = 0; i < N ; i++)
{
        // invio informazioni di input dall'host verso il device
        // ...

        // esecuzione del kernel
        kernel.execute(fragm.length);

        // invio informazioni di output dal device verso l'host
        // ...
}
```

Dopo:

```
// invio informazioni di input dall'host verso il device
// ...

// esecuzione del kernel N volte
kernel.execute(fragm.length, N);

// invio informazioni di output dal device verso l'host
// ...
```

Un secondo tentativo di miglioramento è stato fatto riducendo le dimensioni dei tipi di dati utilizzati e utilizzando le memorie *local* e *constant*. Dopo tali cambiamenti, ripetendo i test 100000 volte si è potuto evincere che:

- I due array sulla quale vengono fatti le operazioni più significative sono stati spostati nella memoria locale hanno ottenuto un miglioramento di circa 50 ms.
- Per tutte le singole variabili che sono state spostate nella costant memory si è riscontrato un peggioramento di 10 ms circa.

Attraverso il solo utilizzo della local memory siamo scesi da 260 ms a 105 ms. La costant memory è stata rimossa a causa del degrado dei tempi di esecuzione.

L'utilizzo della memoria locale si trattava, in realtà, in uno spreco. In particolare, non ha senso assegnare alla memoria riservata ad ogni WorkGroup, l'intera matrice di dati, in quanto la maggior parte di essi verrebbero inutilizzati. Pertanto abbiamo scelto di non utilizzare la memoria locale.

Un terzo tentativo di miglioramento ha visto un refactoring del metodo parallelo parallelFragm(), in particolare adottando come tipi di dato i float al posto dei double per garantire la compatibilità dell'algoritmo anche su architetture che non supportano i double e inoltre adottando gli int al posto degli short per prevenire eventuali sforamenti e perdite di precisione nel caso di input molto grandi.

In questo passaggio sono state introdotte inoltre varie ottimizzazioni del codice che hanno portato a notevoli miglioramenti, come ad esempio l'eliminazione di funzioni come `Math.pow(x, 2)`, rimpiazzate dalla semplice moltiplicazione `x*x`; o la riduzione degli accessi in memoria salvando, nella memoria privata, valori che vengono utilizzati più volte in modo tale da effettuare un unico accesso alla memoria globale.

Un ulteriore tentativo è stato fatto scegliendo manualmente la dimensione dei WorkGroup, piuttosto che affidare tale compito ad aparapi, i risultati però non sono stati così soddisfacenti come ci aspettavamo: aparapi offre già di base un ottimo metodo automatico di decisione riguardo alla grandezza dei WorkGroup, e i test con la scelta manuale della dimensione dei WorkGroup non erano migliori di quelli effettuati in precedenza.

Capitolo 4. Risultati

4.1 Tempi di esecuzione: seriale vs parallelo

Uno dei primi test, effettuato su un MacBook Pro con processore dual core da 2,53 GHz e una scheda video NVIDIA GeForce 9400M ha dimostrato come la CPU batte la GPU con input di piccole dimensioni, mentre il risultato si inverte con input di dimensioni più grandi.

In particolare, ecco i tempi impiegati dall'applicazione nel *calc*olare fragm, passando come input i suoi stessi *.class files, ovvero **14 classi** in tutto:

Parallel GPU time: **0.2803** ms

Serial time: **0.0049** ms

Ecco invece i tempi impiegati dall'applicazione nel calcolare il *fragm* di **3300 classi**:

Parallel GPU time: **0.4877** ms

Serial time: **0.7459** ms

Altri test sono stati effettuati considerando anche la versione parallela OpenCL eseguita su CPU, in un ASUS con processore Intel core i7 2630QM 2 GHz con scheda video nVidia GeForce GT 520MX.

Qui per misurare il tempo abbiamo usato diversi approcci. Per quanto riguarda la versione seriale abbiamo utilizzato la funzione ***System.nanoTime()*** offerta da Java.

Per le versioni OpenCL, sia CPU che GPU, abbiamo fatto uso di due diverse funzioni:

1) ***getExecutionTime()*** che restituisce il tempo di esecuzione, compreso il tempo di caricamento e scaricamento dati host-device;

2) ***Kernel.getProfileInfo()***, che contiene al suo interno informazioni separate riguardanti i singoli tempi di accesso in memoria e di esecuzione del kernel.

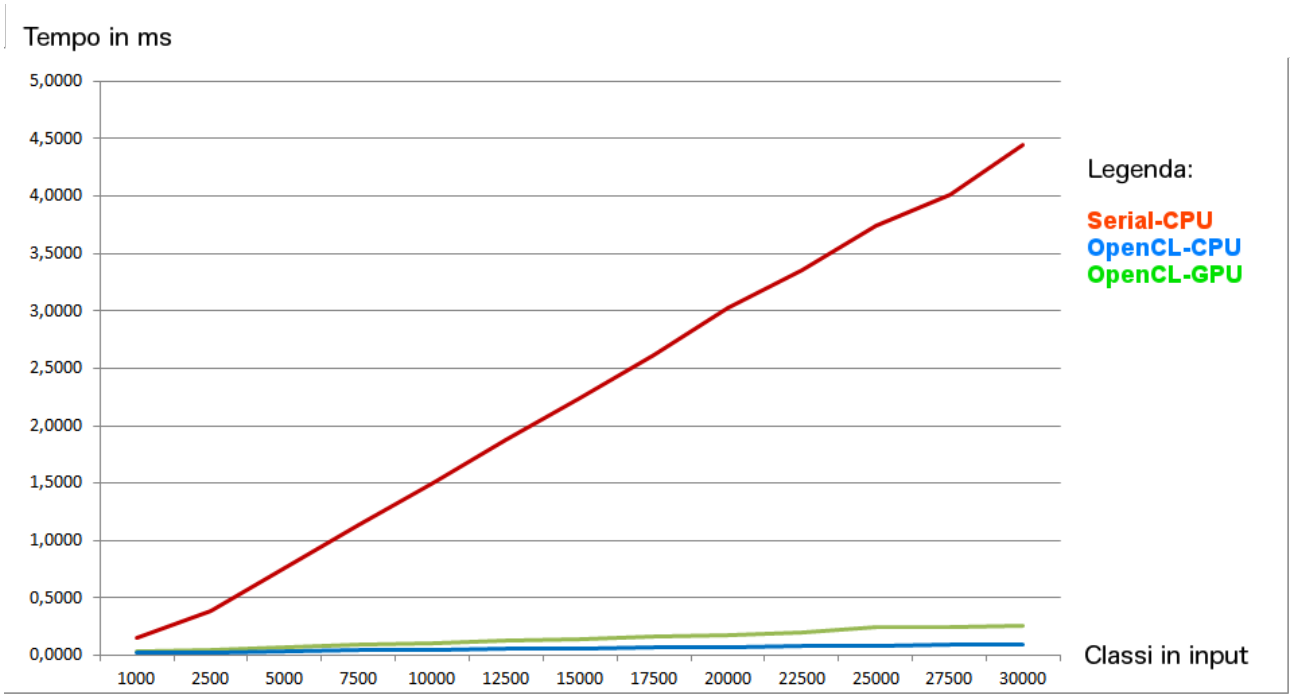
Inizialmente ci siamo affidati esclusivamente alla prima funzione, ma notando alcune discrepanze tra i valori da noi attesi e quelli effettivi, abbiamo eseguito i test misurando il tempo anche con la seconda funzione. Essa è presumibilmente più attendibile, in quanto i valori che restituisce si avvicinano di più a quelli da noi attesi. Nonostante questo, mettendo a confronto i valori da noi attesi con quelli restituiti dalle due funzioni di conteggio del tempo impiegato, abbiamo il dovere di premettere, prima di mostrare interamente i risultati, che quest'ultimi potrebbero non essere molto attendibili a causa di possibili imprecisioni nelle funzioni di conteggio del tempo. Di seguito riportiamo i risultati ottenuti:

1) **1000 Classi** Max Carico GPU= 21%	8) **17500 Classi** Max Carico GPU= 81%
Serial CPU time = 0.1567 ms	Serial CPU time = 2.6130 ms
getExecutionTime()	***getExecutionTime()***
OpenCL GPU time = 0.0396 ms	OpenCL GPU time = 0.1629 ms
OpenCL CPU time = 0.0260 ms	OpenCL CPU time = 0.0690 ms
Kernel.getProfileInfo()	***Kernel.getProfileInfo()***
OpenCL GPU Execution Time = 17µs	OpenCL GPU Execution Time = 173µs
OpenCL GPU R/W Time = 13µs	OpenCL GPU R/W Time = 47µs
OpenCL CPU Execution Time = 52µs	OpenCL CPU Execution Time = 144µs
OpenCL CPU R/W Time = 2µs	OpenCL CPU R/W Time = 13µs
2) **2500 Classi** Max Carico GPU= 44%	9) **20000 Classi** Max Carico GPU= 83%
Serial CPU time = 0.3845 ms	Serial CPU time = 3.0289 ms
getExecutionTime()	***getExecutionTime()***
OpenCL GPU time = 0.0528 ms	OpenCL GPU time = 0.1816 ms
OpenCL CPU time = 0.0297 ms	OpenCL CPU time = 0.0753 ms
Kernel.getProfileInfo()	***Kernel.getProfileInfo()***
OpenCL GPU Execution Time = 26µs	OpenCL GPU Execution Time = 186µs
OpenCL GPU R/W Time = 15µs	OpenCL GPU R/W Time = 50µs
OpenCL CPU Execution Time = 69µs	OpenCL CPU Execution Time = 123µs

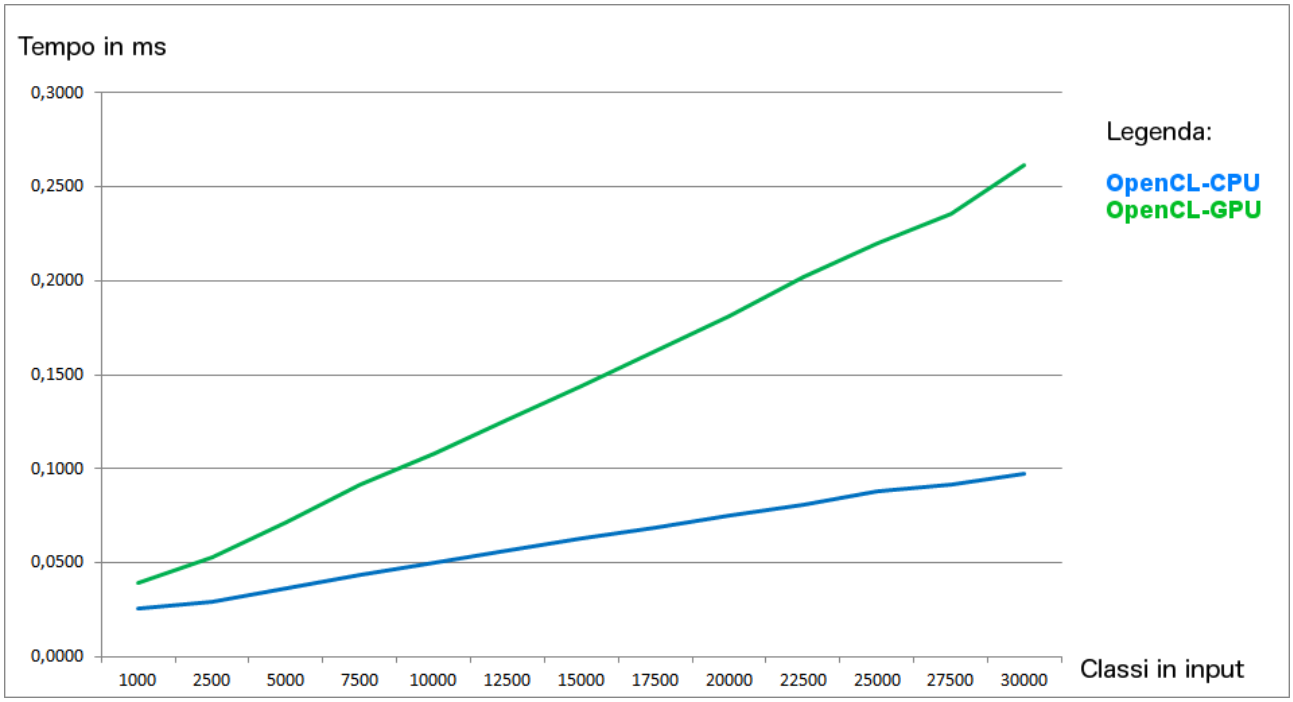
OpenCL CPU R/W Time = 2µs	OpenCL CPU R/W Time = 23µs
3) **5000 Classi** Max Carico GPU= 59%	10) **22500 Classi** Max Carico GPU= 83%
Serial CPU time = 0.7585 ms	Serial CPU time = 3.3599 ms
getExecutionTime()	***getExecutionTime()***
OpenCL GPU time = 0.0719 ms	OpenCL GPU time = 0.2020 ms
OpenCL CPU time = 0.0365 ms	OpenCL CPU time = 0.0812 ms
Kernel.getProfileInfo()	***Kernel.getProfileInfo()***
OpenCL GPU Execution Time = 45µs	OpenCL GPU Execution Time = 210µs
OpenCL GPU R/W Time = 19µs	OpenCL GPU R/W Time = 53µs
OpenCL CPU Execution Time = 92µs	OpenCL CPU Execution Time = 129µs
OpenCL CPU R/W Time = 6µs	OpenCL CPU R/W Time = 19µs
4) **7500 Classi** Max Carico GPU= 66%	11) **25000 Classi** Max Carico GPU= 83%
Serial CPU time = 1.1346 ms	Serial CPU time = 3.7385 ms
getExecutionTime()	***getExecutionTime()***
OpenCL GPU time = 0.0915 ms	OpenCL GPU time = 0.2212 ms
OpenCL CPU time = 0.0435 ms	OpenCL CPU time = 0.0882 ms
Kernel.getProfileInfo()	***Kernel.getProfileInfo()***
OpenCL GPU Execution Time = 61µs	OpenCL GPU Execution Time = 192µs
OpenCL GPU R/W Time = 22µs	OpenCL GPU R/W Time = 55µs
OpenCL CPU Execution Time = 80µs	OpenCL CPU Execution Time = 144µs
OpenCL CPU R/W Time = 9µs	OpenCL CPU R/W Time = 19µs
5) **10000 Classi** Max Carico GPU= 72%	12) **27500 Classi** Max Carico GPU= 83%
Serial CPU time = 1.5019 ms	Serial CPU time = 4.0165 ms
getExecutionTime()	***getExecutionTime()***
OpenCL GPU time = 0.1080 ms	OpenCL GPU time = 0.2526 ms
OpenCL CPU time = 0.0499 ms	OpenCL CPU time = 0.0918 ms
Kernel.getProfileInfo()	***Kernel.getProfileInfo()***
OpenCL GPU Execution Time = 82µs	OpenCL GPU Execution Time = 209µs
OpenCL GPU R/W Time = 25µs	OpenCL GPU R/W Time = 59µs
OpenCL CPU Execution Time = 105µs	OpenCL CPU Execution Time = 142µs
OpenCL CPU R/W Time = 10µs	OpenCL CPU R/W Time = 22µs

6) **12500** Classi Max Carico GPU= 76% Serial CPU time = 1.8748 ms ***getExecutionTime()*** OpenCL GPU time = 0.1262 ms OpenCL CPU time = 0.0564 ms ***Kernel.getProfileInfo()*** OpenCL GPU Execution Time = 100µs OpenCL GPU R/W Time = 27µs OpenCL CPU Execution Time = 103µs OpenCL CPU R/W Time = 13µs 7) **15000 Classi** Max Carico GPU= 79% Serial CPU time = 2.2450 ms ***getExecutionTime()getExecutionTime()*** OpenCL GPU time = 0.1440 ms OpenCL CPU time = 0.0631 ms ***Kernel.getProfileInfo()*** OpenCL GPU Execution Time = 118µs OpenCL GPU R/W Time = 31µs OpenCL CPU Execution Time = 147µs OpenCL CPU R/W Time = 13µs	13) **30000 Classi** Max Carico GPU= 86% Serial CPU time = 4.4512 ms ***getExecutionTime()*** OpenCL GPU time = 0.2613 ms OpenCL CPU time = 0.0977 ms ***Kernel.getProfileInfo()*** OpenCL GPU Execution Time = 223µs OpenCL GPU R/W Time = 61µs OpenCL CPU Execution Time = 156µs OpenCL CPU R/W Time = 26µs

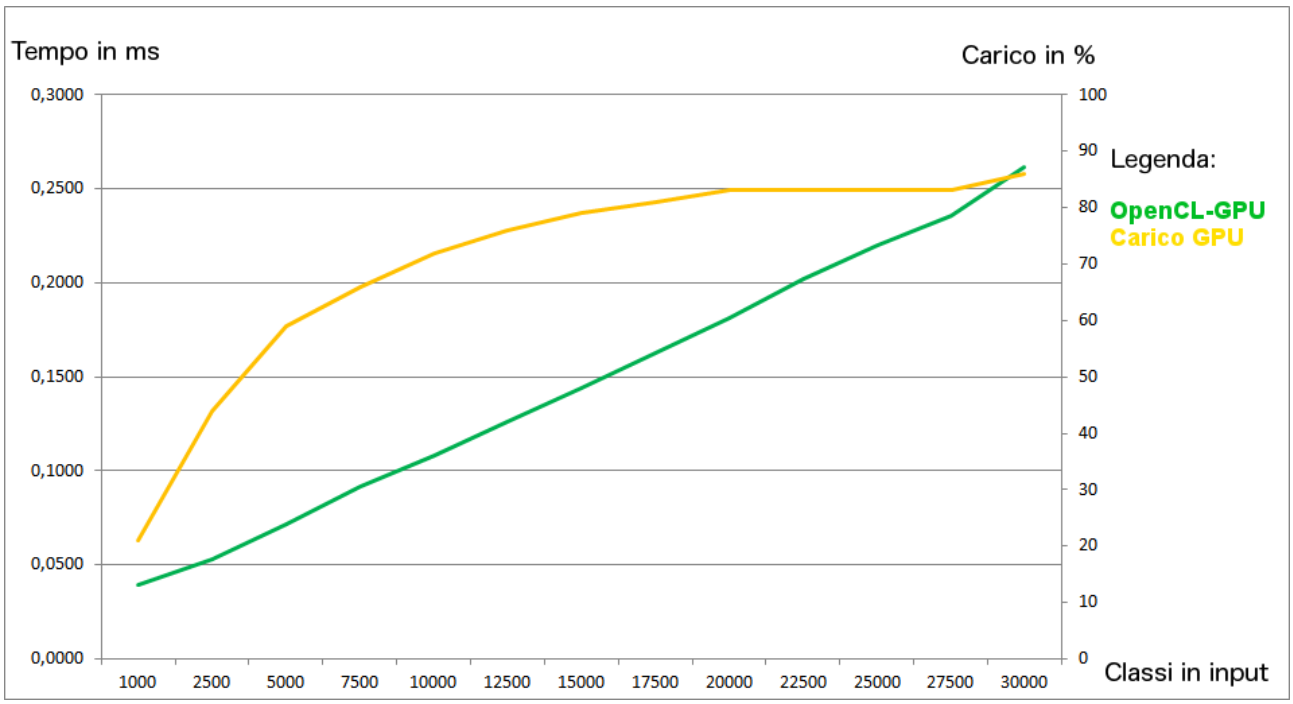
I precedenti dati ottenuti tramite la funzione ***getExecutionTime()*** sono illustrati nei seguenti grafici.



In questo grafico si evidenzia notevolmente il vantaggio di utilizzare OpenCL, sia usando la CPU che la GPU, rispetto al normale calcolo seriale.

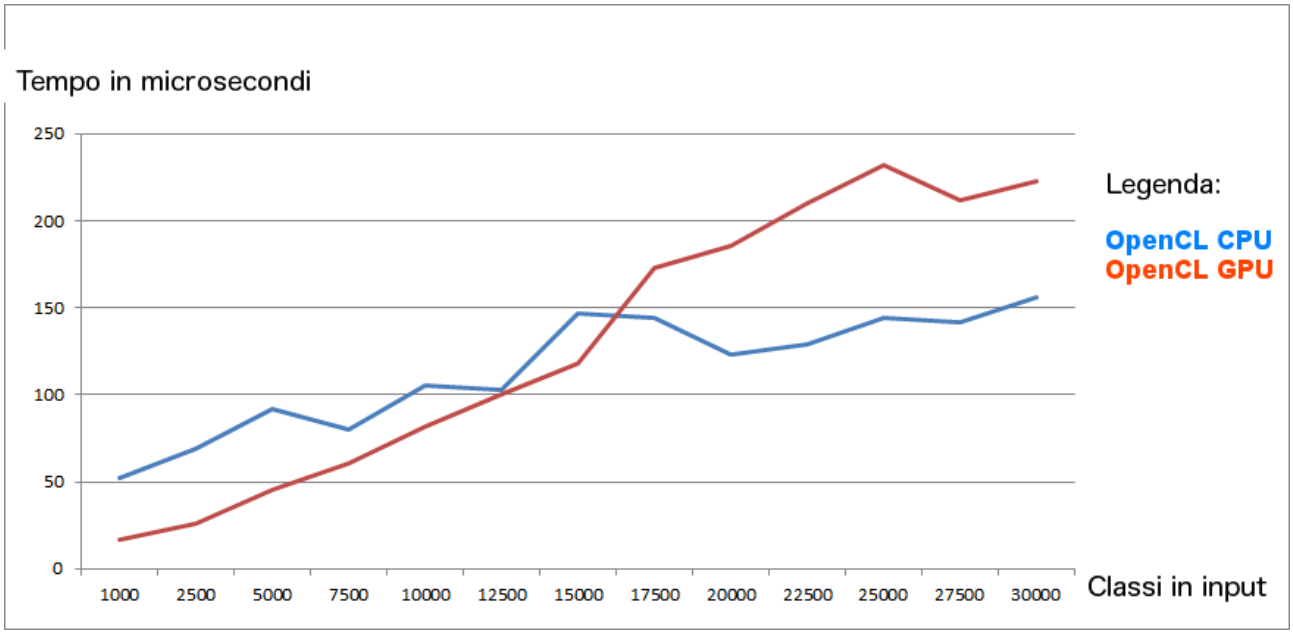


Qui evidenziamo la differenza di utilizzo OpenCL-CPU e OpenCL-GPU. Probabilmente la CPU ha la meglio sulla GPU a causa dei costi di trasferimento dati host-device che quest'ultima richiede.



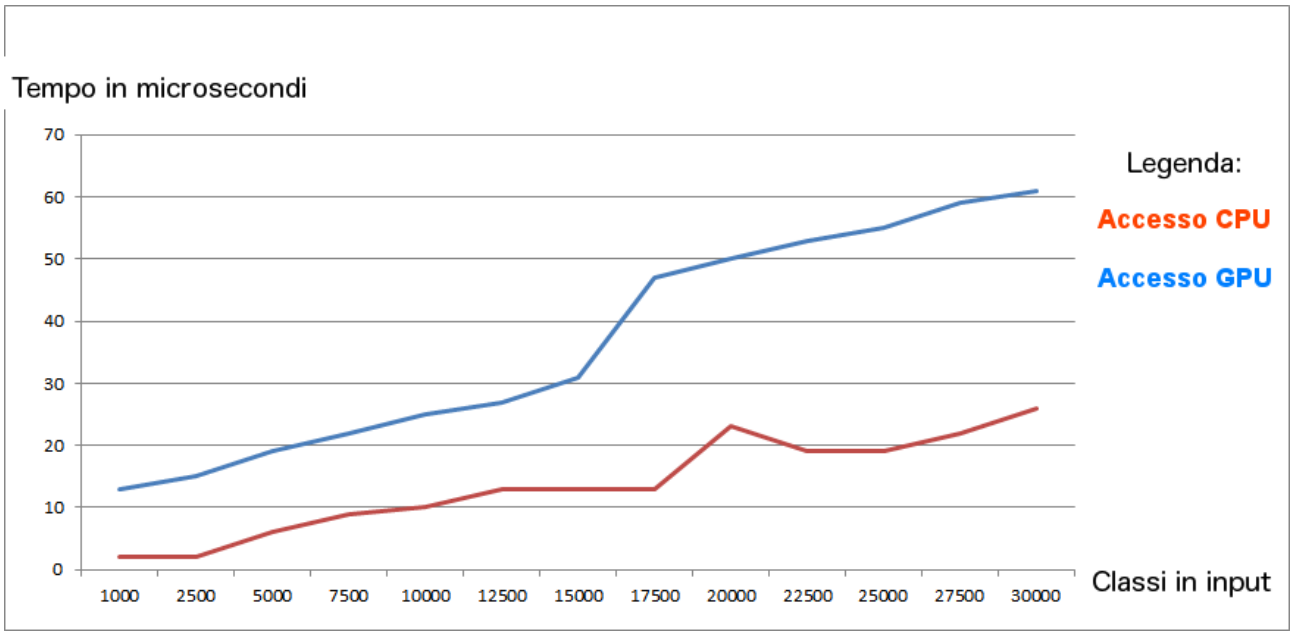
Quest'ultimo grafico illustra il carico della GPU al variare del numero di classi in input.

Di seguito sono invece riportati i grafici ottenuti tramite la funzione ***Kernel.getProfileInfo()***.



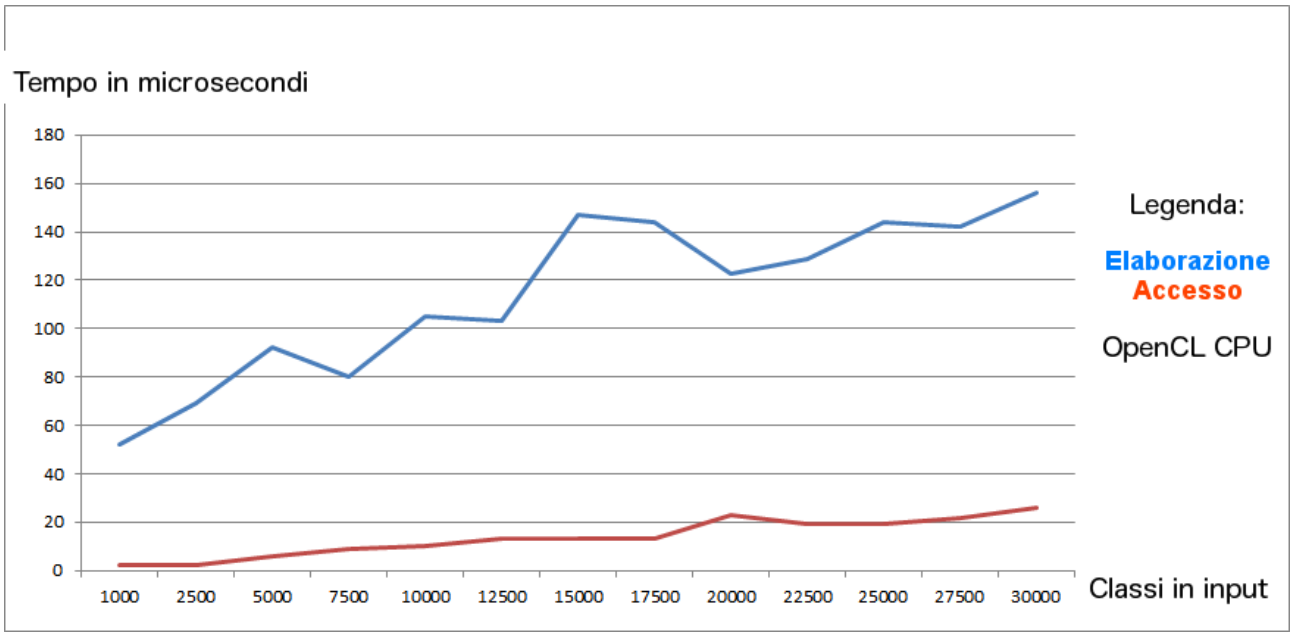
Dal grafico sovrastante si evince che con input inferiori alle 15000 classi la GPU ha tempi di esecuzione inferiori rispetto alla CPU, mentre superando questo valore i tempi della GPU crescono rispetto a quelli della CPU.

Possiamo spiegarne le cause attraverso il grafico che confronta i tempi di accesso tra la versione OpenCL della CPU e quella GPU:

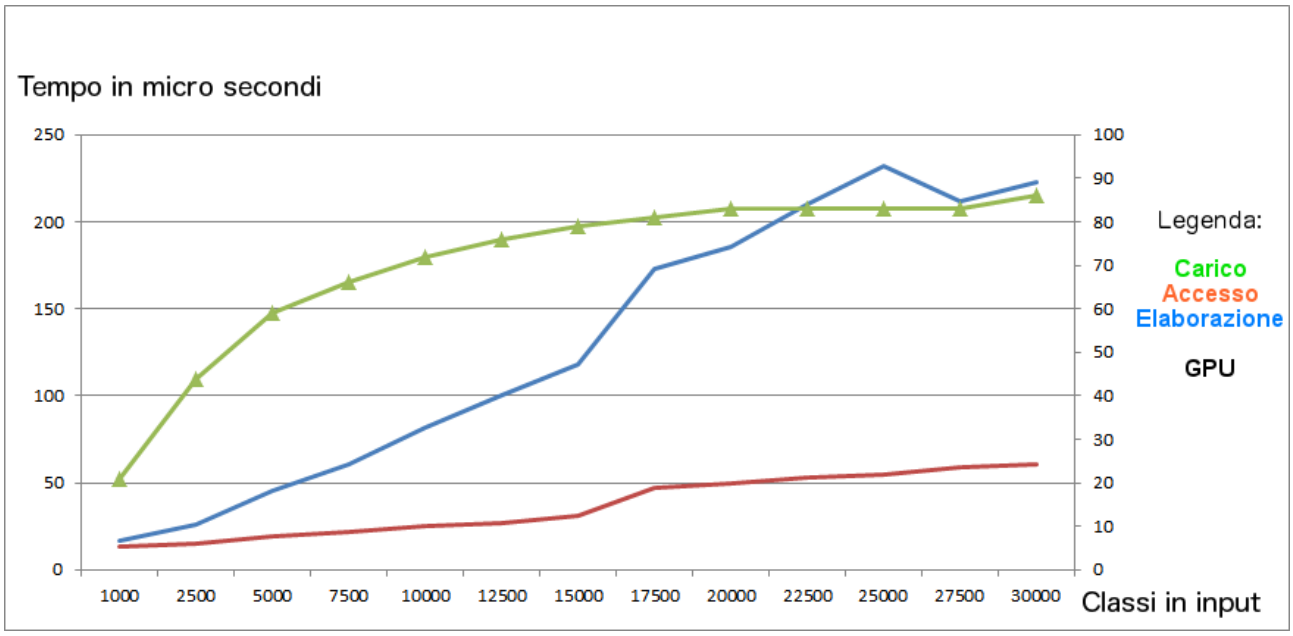


Infatti, a partire da input di 15000 classi, i tempi di accesso della GPU aumentano notevolmente rispetto a quelli della CPU.

Nel successivo grafico vengono comparati i tempi di elaborazione con i tempi di accesso alla memoria dell'OpenCL-CPU:



Il successivo grafico invece mette a confronto i tempi di carico, accesso alla memoria ed elaborazione dell'OpenCL-GPU:



è importante notare che finchè il carico non arriva quasi alla saturazione, i tempi di elaborazione e accesso rimangono approssimativamente lineari.

Per comprendere il comportamento dei grafici appena illustrati, è necessario andare ad analizzare più in dettaglio le caratteristiche dell'hardware su cui sono stati effettuati i test.

La potenza di calcolo del processore Intel core i7 2630QM è circa **48 GFLOP**, mentre quella della scheda video nVidia GeForce GT 520MX è circa **173 GFLOP**.

Apparentemente quest'ultima dovrebbe essere circa il triplo più potente rispetto al processore. Il fatto che la versione OpenCL della CPU batta la GPU è dovuto probabilmente alla differenza tra la RAM della CPU (la cui banda è di **15 GBps**) e quella della GPU (**13 GBps**). Questa differenza spicca soprattutto per i pochi effettivi calcoli che vengono eseguiti, i cui tempi sono meno rilevanti rispetto a quelli del trasferimento dati.

4.2 Lancio dell'applicazione

L'applicazione originale *ckjm* può essere eseguita in diverse modalità, a secondo del del parametro passato a riga di comando durante il lancio dell'applicazione (ad esempio *-s* o *-p*). Noi abbiamo introdotto un nuovo parametro ***-c*** che serve ad avviare la nostra versione di ckjm, cioè quella per calcolare la metrica ***fragm*** (oltre alle metriche che *ckjm* calcola di base).

Per avviare la nostra applicazione bisogna quindi digitare:

java -jar ckjm.jar -c esempio.class

in questo caso l'applicazione analizzerà la classe *esempio.class*. Come già detto in precedenza, è anche possibile far analizzare più classi alla nostra applicazione, digitando ad esempio:

java -jar ckjm.jar -c /path/folder/.class*

Per semplicità nell'output standard abbiamo omesso i dettagli riguardanti le informazioni raccolte dal programma descritte nei paragrafi **3.1** e **3.2** (incluso il calcolo del fan-in per un dato package). È possibile avviare il programma scegliendo di visualizzare tali dettagli concatenando a "-c" i seguenti parametri:

- **D "***details*" mostra, per ogni singola classe, le informazioni descritte nel paragrafo 3.1;

- **G** "*global details*" mostra le informazioni globali descritte nel paragrafo 3.2;

- **F** "*fan-in*" chiede all'utente di inserire il nome di un package, calcola e stampa, per ogni classe, il numero di chiamate verso quel package.

Ad esempio per abilitare sia i dettagli, i dettagli globali che il fan-in verso un dato package, bisogna digitare:

java -jar ckjm.jar -cDGF /path/folder/.class*

4.3 Test su sistemi software noti

Abbiamo preso in esame tre sistemi software scritti in Java su cui abbiamo calcolato il *fragm* delle loro classi:

- ***JUnit***, un framework di unit testing per il linguaggio di programmazione Java (**www.junit.org**);

- ***JHotDraw***, un framework grafico per applicazioni Java (**www.jhotdraw.org**);

- ***JRefactory***, un utility di refactoring per applicazioni Java (**jrefactory.sourceforge.net**).

Di seguito sono riportati i risultati per tutte le classi dei sistemi software appena descritti che hanno un valore di *fragm* non nullo e maggiore di 0:

JUnit

fragm(org.junit.runners.Parameterized$TestClassRunnerForParameters):	0.40369606	{CBO: 17}
fragm(org.junit.internal.runners.MethodRoadie$1):	0.36602545	{CBO: 16}
fragm(org.junit.internal.runners.statements.FailOnTimeout):	0.31649655	{CBO: 10}
fragm(org.junit.experimental.theories.internal.AllMembersSupplier):	0.30295503	{CBO: 24}
fragm(org.junit.runners.BlockJUnit4ClassRunner):	0.30004436	{CBO: 30}
fragm(org.junit.experimental.max.MaxHistory):	0.2949984	{CBO: 24}
fragm(org.junit.runner.JUnitCore):	0.28582525	{CBO: 26}
fragm(org.junit.runners.Parameterized):	0.27151674	{CBO: 17}
fragm(org.junit.internal.runners.rules.RuleFieldValidator):	0.26076746	{CBO: 11}
fragm(org.junit.experimental.max.MaxCore):	0.25592893	{CBO: 28}
fragm(org.junit.internal.runners.MethodValidator):	0.25041705	{CBO: 16}
fragm(org.junit.internal.ArrayComparisonFailure):	0.22199482	{CBO: 9}
fragm(org.junit.internal.TextListener):	0.21767694	{CBO: 13}
fragm(junit.runner.BaseTestRunner):	0.2105298	{CBO: 35}
fragm(org.junit.experimental.results.FailureList):	0.20710677	{CBO: 10}
fragm(org.junit.runner.manipulation.Filter$2):	0.20710677	{CBO: 7}
fragm(org.junit.internal.requests.FilterRequest):	0.20710677	{CBO: 11}
fragm(org.junit.experimental.results.PrintableResult):	0.20710677	{CBO: 13}
fragm(org.junit.runners.Suite):	0.20710677	{CBO: 15}
fragm(org.junit.experimental.theories.Theories):	0.20278192	{CBO: 18}
fragm(org.junit.experimental.theories.Theories$TheoryAnchor):	0.19999999	{CBO: 26}
fragm(junit.textui.ResultPrinter):	0.19358987	{CBO: 14}
fragm(org.junit.experimental.max.MaxHistory$RememberingListener):	0.19337523	{CBO: 13}
fragm(org.junit.experimental.theories.internal.ParameterizedAssertionError):	0.18041384	{CBO: 10}
fragm(org.junit.internal.runners.InitializationError):	0.17082036	{CBO: 7}
fragm(org.junit.internal.runners.JUnit38ClassRunner):	0.17082036	{CBO: 23}
fragm(org.hamcrest.StringDescription):	0.17082036	{CBO: 12}
fragm(org.junit.internal.runners.MethodRoadie):	0.16033816	{CBO: 22}
fragm(org.junit.experimental.categories.Categories$CategoryFilter):	0.15997499	{CBO: 14}

fragm(org.junit.runners.model.FrameworkMethod): 0.15791619 {CBO: 16}

fragm(org.junit.runners.model.NoGenericTypeParametersValidator): 0.14888567 {CBO: 12}

fragm(junit.textui.TestRunner): 0.14888567 {CBO: 17}

fragm(junit.framework.TestSuite): 0.13965714 {CBO: 26}

fragm(junit.framework.Assert): 0.13474685 {CBO: 15}

fragm(org.hamcrest.core.DescribedAs): 0.13245553 {CBO: 10}

fragm(org.junit.runner.notification.Failure): 0.13245553 {CBO: 11}

fragm(org.junit.experimental.ParallelComputer$1): 0.13245553 {CBO: 13}

fragm(junit.framework.TestFailure): 0.13245553 {CBO: 11}

fragm(org.junit.experimental.theories.suppliers.TestedOnSupplier): 0.13245553 {CBO: 13}

fragm(org.junit.internal.AssumptionViolatedException): 0.1063391 {CBO: 10}

fragm(junit.extensions.RepeatedTest): 0.1063391 {CBO: 8}

fragm(org.junit.internal.runners.SuiteMethod): 0.1063391 {CBO: 12}

fragm(org.junit.runners.model.RunnerBuilder): 0.07539642 {CBO: 12}

fragm(org.junit.internal.runners.TestClass): 0.06568545 {CBO: 13}

fragm(org.hamcrest.BaseDescription): 0.05215758 {CBO: 16}

fragm(org.junit.internal.runners.statements.ExpectException): 0.0432145 {CBO: 10}

fragm(junit.framework.TestCase): 0.030215442 {CBO: 15}

JHotDraw

fragm(org.jhotdraw.xml.JavaxDOMOutput): 0.40728754 {CBO: 38}

fragm(nanoxml.XMLElement): 0.3524111 {CBO: 28}

fragm(org.jhotdraw.xml.NanoXMLLiteDOMInput): 0.32681394 {CBO: 27}

fragm(org.jhotdraw.draw.TextAreaFigure): 0.32323194 {CBO: 45}

fragm(org.jhotdraw.xml.DefaultDOMFactory): 0.32197762 {CBO: 16}

fragm(org.jhotdraw.io.ExtensionFileFilter): 0.31114745 {CBO: 12}

fragm(org.jhotdraw.xml.NanoXMLDOMInput): 0.3090195 {CBO: 33}

fragm(org.jhotdraw.xml.NanoXMLLiteDOMOutput): 0.2941876 {CBO: 26}

fragm(org.jhotdraw.xml.NanoXMLDOMOutput): 0.28110284 {CBO: 30}

fragm(org.jhotdraw.app.DefaultMDIApplication): 0.27615052 {CBO: 75}

fragm(org.jhotdraw.samples.pert.figures.TaskFigure): 0.26542026 {CBO: 46}

fragm(org.jhotdraw.app.DefaultSDIApplication): 0.26063883 {CBO: 62}

fragm(org.jhotdraw.xml.JavaxDOMInput): 0.24762493 {CBO: 34}

fragm(org.jhotdraw.app.DefaultOSXApplication): 0.24409741 {CBO: 76}

fragm(org.jhotdraw.draw.TextFigure): 0.24074376 {CBO: 40}

fragm(net.n3.nanoxml.XMLWriter): 0.23940378 {CBO: 13}

fragm(org.jhotdraw.samples.net.NetApplicationModel): 0.2352941 {CBO: 38}

fragm(org.jhotdraw.samples.pert.PertApplicationModel): 0.2352941 {CBO: 41}

fragm(net.n3.nanoxml.XMLException): 0.23363245 {CBO: 9}

fragm(org.jhotdraw.app.action.FocusAction): 0.20710683 {CBO: 24}

fragm(org.jhotdraw.draw.CreationTool): 0.20710677 {CBO: 28}

fragm(org.jhotdraw.samples.pert.figures.DependencyFigure): 0.20710677 {CBO: 15}

fragm(org.jhotdraw.app.action.FocusAction$1): 0.20710677 {CBO: 8}

fragm(org.jhotdraw.draw.ConnectionTool): 0.20278192 {CBO: 32}

fragm(org.jhotdraw.draw.BezierFigure): 0.19999999 {CBO: 41}

fragm(org.jhotdraw.app.AbstractProject): 0.19999999 {CBO: 20}

fragm(net.n3.nanoxml.StdXMLParser): 0.18764734 {CBO: 23}

fragm(org.jhotdraw.draw.AbstractFigure): 0.18219101 {CBO: 44}

fragm(org.jhotdraw.draw.TransformEdit): 0.18219101 {CBO: 17}

fragm(org.jhotdraw.app.DefaultMDIApplication$3): 0.17082036 {CBO: 12}

fragm(org.jhotdraw.app.DefaultOSXApplication$2): 0.17082036 {CBO: 11}

fragm(org.jhotdraw.samples.net.NetProject$2): 0.17082036 {CBO: 9}

fragm(org.jhotdraw.app.AbstractApplication): 0.16766727 {CBO: 24}

fragm(org.jhotdraw.app.action.LoadRecentAction): 0.16212225 {CBO: 23}

fragm(org.jhotdraw.app.action.OpenRecentAction): 0.16212225 {CBO: 24}

fragm(org.jhotdraw.draw.AttributeKeys): 0.14524156 {CBO: 31}

fragm(org.jhotdraw.draw.action.GenericListener$DefaultInvoker): 0.13245553 {CBO: 15}

fragm(org.jhotdraw.app.DefaultSDIApplication$2): 0.13012606 {CBO: 12}

fragm(org.jhotdraw.app.action.OpenAction): 0.115457416 {CBO: 27}

fragm(org.jhotdraw.samples.svg.SVGUtil): 0.11079848 {CBO: 37}

fragm(org.jhotdraw.samples.draw.DrawApplet$1): 0.1063391 {CBO: 25}

fragm(org.jhotdraw.samples.net.NetApplet$1): 0.1063391 {CBO: 25}

fragm(org.jhotdraw.samples.pert.PertApplet$1): 0.1063391 {CBO: 25}

fragm(org.jhotdraw.samples.svg.SVGApplet$1): 0.1063391 {CBO: 25}

fragm(org.jhotdraw.undo.UndoRedoManager$RedoAction): 0.1063391 {CBO: 12}

fragm(org.jhotdraw.undo.UndoRedoManager$UndoAction): 0.1063391 {CBO: 12}

fragm(org.jhotdraw.draw.action.ZoomEditorAction): 0.1063391 {CBO: 11}

fragm(org.jhotdraw.draw.action.VerticalGridLayout): 0.10633904 {CBO: 13}

fragm(org.jhotdraw.app.action.SaveAction): 0.10370344 {CBO: 25}

fragm(org.jhotdraw.draw.action.ToolBarButtonFactory): 0.09954375 {CBO: 105}

fragm(org.jhotdraw.samples.svg.SVGDrawing): 0.081377685 {CBO: 13}

fragm(org.jhotdraw.draw.action.GenericListener): 0.07021445 {CBO: 13}

fragm(org.jhotdraw.app.action.ExitAction): 0.0676226 {CBO: 33}

fragm(org.jhotdraw.geom.Dimension2DDouble): 0.05815631 {CBO: 8}

fragm(org.jhotdraw.draw.GridConstrainer): 0.05815631 {CBO: 15}

fragm(org.jhotdraw.draw.TriangleFigure): 0.05815631 {CBO: 28}

fragm(org.jhotdraw.app.action.ExportAction): 0.05215758 {CBO: 28}

fragm(org.jhotdraw.app.action.SaveBeforeAction): 0.047270477 {CBO: 24}

fragm(org.jhotdraw.draw.AttributedFigure): 0.045837104 {CBO: 29}

fragm(org.jhotdraw.draw.AbstractAttributedCompositeFigure): 0.0432145 {CBO: 30}

fragm(org.jhotdraw.util.Methods): 0.041435122 {CBO: 16}

fragm(org.jhotdraw.geom.Insets2DDouble): 0.039795637 {CBO: 13}

fragm(org.jhotdraw.util.ResourceBundleUtil): 0.039795637 {CBO: 22}

fragm(nanoxml.XMLParseException): 0.039795637 {CBO: 5}

fragm(net.n3.nanoxml.NonValidator): 0.03828025 {CBO: 18}

fragm(org.jhotdraw.draw.DelegationSelectionTool): 0.023216665 {CBO: 29}

fragm(org.jhotdraw.draw.AbstractCompositeFigure): 0.0208053 {CBO: 35}

fragm(net.n3.nanoxml.XMLUtil): 0.010634661 {CBO: 12}

fragm(net.n3.nanoxml.StdXMLReader): 0.010305941 {CBO: 24}

JRefactory

fragm(edu.umd.cs.findbugs.gui.AnalysisRun): 0.41287088 {CBO: 27}

fragm(org.acm.seguin.tools.install.RefactoryStorage): 0.40098614 {CBO: 19}

fragm(org.acm.seguin.pmd.PMD): 0.38593662 {CBO: 38}

fragm(org.acm.seguin.ide.common.options.SelectedRules): 0.38456625 {CBO: 29}

fragm(org.acm.seguin.awt.ExceptionDialog): 0.38345218 {CBO: 26}

fragm(edu.umd.cs.findbugs.CountBugs): 0.3705715 {CBO: 20}

fragm(org.acm.seguin.pmd.rules.XPathRule): 0.36602545 {CBO: 20}

fragm(org.acm.seguin.refactor.Refactoring): 0.3660254 {CBO: 27}

fragm(org.jaxen.util.IdentityHashMap): 0.3616404 {CBO: 28}

fragm(org.acm.seguin.pmd.cpd.CPD): 0.35895568 {CBO: 27}

fragm(org.acm.seguin.ide.common.PackageSelectorPanel): 0.35895568 {CBO: 57}

fragm(org.acm.seguin.summary.FileSummary): 0.35590148 {CBO: 35}

fragm(edu.umd.cs.findbugs.Project): 0.3502946 {CBO: 26}

fragm(org.dom4j.DocumentException): 0.34852815 {CBO: 9}

fragm(edu.umd.cs.findbugs.DetectorFactoryCollection): 0.33906078 {CBO: 20}

fragm(org.acm.seguin.refactor.undo.DefaultUndoAction): 0.33666003 {CBO: 12}

fragm(org.acm.seguin.ide.common.MultipleDirClassDiagramReloader): 0.3365019 {CBO: 11}

fragm(org.acm.seguin.ide.common.CodingStandardsViewer): 0.33449197 {CBO: 57}

fragm(edu.umd.cs.findbugs.gui.FindBugsFrame$ArchiveAndDirectoryFilter): 0.3333333 {CBO: 8}

fragm(org.acm.seguin.refactor.undo.UndoCleaner): 0.3333333 {CBO: 9}

fragm(org.apache.bcel.generic.FieldGenOrMethodGen): 0.31649655 {CBO: 15}

fragm(org.acm.seguin.pmd.PMDDirectory): 0.31649655 {CBO: 35}

fragm(edu.umd.cs.findbugs.ba.ValueNumberAnalysis): 0.31649655 {CBO: 34}

fragm(org.acm.seguin.io.ExtensionFileFilter): 0.3111071 {CBO: 7}

fragm(edu.umd.cs.findbugs.CategorizeBugs): 0.30391014 {CBO: 24}

fragm(net.sourceforge.jrefactory.uml.PackageLoader): 0.299671 {CBO: 46}

fragm(edu.umd.cs.findbugs.detect.FindOpenStream): 0.29410142 {CBO: 36}

fragm(org.acm.seguin.tools.build.CleanClassFiles): 0.2912566 {CBO: 15}

fragm(org.acm.seguin.ant.CVSUtil): 0.29018366 {CBO: 17}

fragm(org.acm.seguin.ant.Pretty): 0.28671777 {CBO: 25}

fragm(org.apache.bcel.util.JavaWrapper): 0.28571427 {CBO: 15}

fragm(org.acm.seguin.refactor.type.MoveClass): 0.28072518 {CBO: 19}

fragm(edu.umd.cs.findbugs.ba.PruneInfeasibleExceptionEdges): 0.2790547 {CBO: 38}

fragm(org.acm.seguin.tools.install.SettingPanel): 0.27702868 {CBO: 16}

fragm(org.acm.seguin.pmd.RuleSetFactory): 0.2760384 {CBO: 40}

fragm(org.dom4j.io.aelfred.XmlParser): 0.27596736 {CBO: 26}

fragm(net.sourceforge.jrefactory.uml.UMLPackage): 0.27482593 {CBO: 63}

fragm(org.apache.bcel.util.ClassLoaderRepository): 0.27272725 {CBO: 14}

fragm(org.acm.seguin.tools.build.JavadocBuilder): 0.2634799 {CBO: 15}

fragm(edu.umd.cs.findbugs.BugCollection): 0.2631579 {CBO: 40}

fragm(edu.umd.cs.findbugs.ProjectStats): 0.2620008 {CBO: 36}

fragm(edu.umd.cs.findbugs.detect.FindInconsistentSync2): 0.2613303 {CBO: 53}

fragm(org.dom4j.rule.RuleManager): 0.26063877 {CBO: 14}

fragm(org.dom4j.DocumentFactory): 0.25377834 {CBO: 47}

fragm(org.jaxen.function.StringLengthFunction): 0.25377834 {CBO: 11}

fragm(org.acm.seguin.summary.PackageSummary): 0.25192064 {CBO: 17}

fragm(org.acm.seguin.refactor.type.ExtractInterfaceRefactoring): 0.24983317 {CBO: 26}

fragm(org.acm.seguin.pmd.cpd.FileFinder): 0.24838328 {CBO: 9}

fragm(edu.umd.cs.findbugs.InstructionScannerDriver): 0.24627793 {CBO: 16}

fragm(org.apache.bcel.verifier.Verifier): 0.2425152 {CBO: 21}

fragm(edu.umd.cs.findbugs.detect.LockedFields): 0.24104178 {CBO: 24}

fragm(edu.umd.cs.findbugs.ba.AbstractDataflowAnalysis): 0.24074376 {CBO: 17}

fragm(edu.umd.cs.findbugs.ba.ValueNumberCache): 0.24074376 {CBO: 11}

fragm(edu.umd.cs.findbugs.detect.DroppedException): 0.23905712 {CBO: 37}

fragm(org.dom4j.io.DOMReader): 0.23854893 {CBO: 20}

fragm(org.acm.seguin.pmd.symboltable.AbstractScope): 0.23827165 {CBO: 21}

fragm(org.acm.seguin.refactor.type.RenameClassVisitor): 0.23827165 {CBO: 21}

fragm(org.acm.seguin.tools.build.JarDirectoryBuilder): 0.23786479 {CBO: 15}

fragm(edu.umd.cs.findbugs.ba.ValueNumberFactory): 0.23786479 {CBO: 11}

fragm(org.acm.seguin.pmd.rules.design.PositionalIteratorRule): 0.2372098 {CBO: 15}

fragm(org.apache.bcel.generic.InstructionHandle): 0.23470801 {CBO: 17}

fragm(org.acm.seguin.pmd.cpd.GUI$SaveListener): 0.23276299 {CBO: 21}

fragm(org.apache.bcel.generic.FieldGen): 0.23192507 {CBO: 27}

fragm(edu.umd.cs.findbugs.detect.UnreadFields): 0.23179173 {CBO: 26}

fragm(org.apache.bcel.verifier.structurals.Subroutines): 0.23053056 {CBO: 31}

fragm(org.acm.seguin.pmd.swingui.MessageDialog): 0.23029673 {CBO: 34}

fragm(edu.umd.cs.findbugs.ba.Frame): 0.22691458 {CBO: 19}

fragm(org.acm.seguin.util.FileSettings): 0.22597164 {CBO: 32}

fragm(org.acm.seguin.tools.build.CodeStoreBuilder): 0.22505236 {CBO: 15}

fragm(edu.umd.cs.findbugs.detect.FindRefComparison): 0.22392279 {CBO: 42}

fragm(org.apache.bcel.verifier.structurals.Subroutines$SubroutineImpl): 0.22392279 {CBO: 21}

fragm(edu.umd.cs.findbugs.detect.Naming): 0.22280413 {CBO: 23}

fragm(org.apache.bcel.util.InstructionFinder): 0.22019207 {CBO: 20}

fragm(org.dom4j.tree.NamespaceStack): 0.21918023 {CBO: 13}

fragm(org.apache.tools.ant.taskdefs.optional.javastyle.JavaStyle): 0.21739924 {CBO: 13}

fragm(edu.umd.cs.findbugs.FindBugs): 0.21685964 {CBO: 70}

fragm(org.acm.seguin.pmd.renderers.IDEAJRenderer$SourcePath): 0.216039 {CBO: 10}

fragm(org.dom4j.tree.AbstractElement): 0.21592575 {CBO: 44}

fragm(org.apache.bcel.util.ClassPath): 0.21458703 {CBO: 24}

fragm(listclass): 0.21346766 {CBO: 19}

fragm(org.apache.bcel.verifier.structurals.OperandStack): 0.21242636 {CBO: 13}

fragm(edu.umd.cs.findbugs.detect.InitializationChain): 0.2111513 {CBO: 18}

fragm(edu.umd.cs.findbugs.EmacsBugReporter): 0.21110016 {CBO: 14}

fragm(edu.umd.cs.findbugs.detect.MutableStaticFields): 0.2096777 {CBO: 16}

fragm(org.jaxen.function.CountFunction): 0.20710683 {CBO: 9}

fragm(org.acm.seguin.refactor.DefaultComplexTransform): 0.20710683 {CBO: 18}

fragm(org.apache.bcel.generic.LDC): 0.20710683 {CBO: 29}

fragm(org.apache.bcel.generic.LDC2_W): 0.20710683 {CBO: 20}

fragm(org.jaxen.pattern.PatternHandler): 0.20710683 {CBO: 21}

fragm(org.jaxen.function.SubstringAfterFunction): 0.20710683 {CBO: 9}

fragm(org.jaxen.function.SumFunction): 0.20710683 {CBO: 11}

fragm(org.dom4j.swing.XMLTableModel): 0.20710683 {CBO: 14}

fragm(edu.umd.cs.findbugs.AbstractBugReporter): 0.20710677 {CBO: 16}

fragm(org.acm.seguin.pmd.swingui.DirectoryTable$TableSortComparator): 0.20710677 {CBO: 7}

fragm(org.acm.seguin.pmd.swingui.DirectoryTreeModel): 0.20710677 {CBO: 16}

fragm(net.sourceforge.jrefactory.action.EmptySelectedFileSet): 0.20710677 {CBO: 8}

fragm(org.acm.seguin.pmd.rules.strictexception.ExceptionSignatureDeclaration): 0.20710677 {CBO: 17}

fragm(edu.umd.cs.findbugs.FindBugs+1): 0.20710677 {CBO: 6}

fragm(edu.umd.cs.findbugs.detect.FindUninitializedGet): 0.20710677 {CBO: 14}

fragm(org.acm.seguin.ide.common.options.JSSpacingOptionPane): 0.20710677 {CBO: 14}

fragm(org.jaxen.function.LastFunction): 0.20710677 {CBO: 8}

fragm(org.acm.seguin.summary.MethodSummary): 0.20710677 {CBO: 10}

fragm(edu.umd.cs.findbugs.ba.OtherLockCountAnalysis): 0.20710677 {CBO: 19}

fragm(org.acm.seguin.pmd.swingui.PMDViewer): 0.20710677 {CBO: 31}

fragm(org.jaxen.function.PositionFunction): 0.20710677 {CBO: 9}

fragm(org.acm.seguin.pretty.PrettyPrintString): 0.20710677 {CBO: 10}

fragm(org.acm.seguin.pmd.ProjectFile): 0.20710677 {CBO: 15}

fragm(RenameType): 0.20710677 {CBO: 13}

fragm(edu.umd.cs.findbugs.ResourceTrackingDetector): 0.20710677 {CBO: 25}

fragm(org.acm.seguin.pmd.symboltable.TypeSet$VoidResolver): 0.20710677 {CBO: 7}

fragm(org.apache.bcel.classfile.Utility$JavaReader): 0.20710677 {CBO: 8}

fragm(edu.umd.cs.findbugs.Version): 0.20710677 {CBO: 7}

fragm(org.dom4j.util.XMLErrorHandler): 0.20710677 {CBO: 11}

fragm(org.acm.seguin.pretty.JavaDocableImpl): 0.20678234 {CBO: 14}

fragm(org.acm.seguin.io.InplaceOutputStream): 0.20673913 {CBO: 11}

fragm(org.dom4j.tree.AbstractBranch): 0.20654172 {CBO: 22}

fragm(org.acm.seguin.pmd.cpd.Match): 0.2058866 {CBO: 9}

fragm(org.acm.seguin.pmd.rules.UnusedImportsRule): 0.2058866 {CBO: 17}

fragm(org.dom4j.io.SAXHelper): 0.20554066 {CBO: 18}

fragm(org.acm.seguin.tools.install.SortSettingPanel): 0.20554066 {CBO: 19}

fragm(org.acm.seguin.ide.common.Navigator$Navigation): 0.20527446 {CBO: 19}

fragm(org.jaxen.pattern.PatternParser): 0.205024 {CBO: 44}

fragm(org.acm.seguin.print.PrintingSettings): 0.205024 {CBO: 13}

fragm(org.acm.seguin.pmd.RuleSetList): 0.20479387 {CBO: 18}

fragm(edu.umd.cs.findbugs.ba.ValueNumberFrame): 0.20452344 {CBO: 22}

fragm(edu.umd.cs.findbugs.TextUIBugReporter): 0.20420283 {CBO: 9}

fragm(org.acm.seguin.print.xml.XMLLinePrinter): 0.20420283 {CBO: 14}

fragm(org.dom4j.io.aelfred.SAXDriver): 0.20376545 {CBO: 38}

fragm(org.acm.seguin.pretty.PrettyPrintFile): 0.20278192 {CBO: 23}

fragm(net.sourceforge.jrefactory.action.RenameClassAction$RenameClassDialog): 0.20278192 {CBO: 15}

fragm(org.acm.seguin.uml.refactor.RenameClassDialog): 0.20278192 {CBO: 14}

fragm(org.acm.seguin.uml.refactor.RenameParameterDialog): 0.20278192 {CBO: 27}

fragm(org.jaxen.SimpleFunctionContext): 0.20278192 {CBO: 9}

fragm(org.acm.seguin.metrics.Summarize): 0.20278192 {CBO: 11}

fragm(org.dom4j.io.XPPReader): 0.20278192 {CBO: 26}

fragm(org.acm.seguin.pmd.rules.XPathRule$1): 0.20278192 {CBO: 8}

fragm(org.dom4j.io.HTMLWriter): 0.20133436 {CBO: 25}

fragm(org.jaxen.expr.DefaultUnionExpr): 0.19999999 {CBO: 16}

fragm(net.sourceforge.jrefactory.action.MoveClassAction$NewPackageDialog): 0.19999999 {CBO: 31}

fragm(org.acm.seguin.uml.refactor.NewPackageDialog): 0.19999999 {CBO: 29}

fragm(edu.umd.cs.findbugs.RecursiveFileSearch): 0.19999999 {CBO: 10}

fragm(org.acm.seguin.pmd.rules.StringToStringRule): 0.19999999 {CBO: 18}

fragm(org.acm.seguin.pmd.rules.BeanMembersShouldSerializeRule): 0.19808489 {CBO: 25}

fragm(org.jaxen.expr.DefaultPredicated): 0.1974858 {CBO: 8}

fragm(net.sourceforge.jrefactory.uml.line.AssociationRelationship): 0.1974858 {CBO: 13}

fragm(org.acm.seguin.pmd.swingui.AnalysisViewer$SaveSaveAs): 0.1974858 {CBO: 19}

fragm(org.acm.seguin.pmd.swingui.SearchViewer$SaveSaveAs): 0.1974858 {CBO: 19}

fragm(org.jaxen.SimpleVariableContext): 0.1974858 {CBO: 10}

fragm(org.acm.seguin.summary.SummaryTraversal): 0.1974858 {CBO: 24}

fragm(org.jaxen.function.TranslateFunction): 0.1974858 {CBO: 13}

fragm(com.werken.saxpath.XPathReader): 0.1974858 {CBO: 16}

fragm(org.acm.seguin.pretty.PrintData): 0.1969412 {CBO: 28}

fragm(edu.umd.cs.findbugs.MergeResults): 0.19675028 {CBO: 23}

fragm(org.acm.seguin.pmd.ant.PMDTask): 0.19582844 {CBO: 38}

fragm(edu.umd.cs.findbugs.ba.PruneUnconditionalExceptionThrowerEdges): 0.19387525 {CBO: 36}

fragm(net.sourceforge.jrefactory.query.EqualTree): 0.19337529 {CBO: 60}

fragm(org.jaxen.function.ConcatFunction): 0.19337523 {CBO: 11}

fragm(org.dom4j.dom.DOMElement): 0.19337523 {CBO: 26}

fragm(org.acm.seguin.pmd.swingui.DirectoryTableModel$FilesFilter): 0.19337523 {CBO: 7}

fragm(org.jaxen.util.IdentityHashMap$Entry): 0.19337523 {CBO: 5}

fragm(org.jaxen.function.ext.LowerFunction): 0.19337523 {CBO: 10}

fragm(org.jaxen.function.RoundFunction): 0.19337523 {CBO: 12}

fragm(org.acm.seguin.tools.international.StringListTraversal): 0.19337523 {CBO: 15}

fragm(org.jaxen.function.SubstringBeforeFunction): 0.19337523 {CBO: 9}

fragm(org.acm.seguin.refactor.undo.UndoStack): 0.19337523 {CBO: 27}

fragm(org.acm.seguin.pmd.rules.UnusedPrivateFieldRule): 0.19337523 {CBO: 16}

fragm(org.jaxen.function.ext.UpperFunction): 0.19337523 {CBO: 10}

fragm(org.acm.seguin.pmd.rules.strictexception.AvoidCatchingThrowable): 0.19337523 {CBO: 18}

fragm(org.acm.seguin.pmd.rules.AccessorClassGenerationRule): 0.19096804 {CBO: 25}

fragm(org.acm.seguin.pmd.cpd.CPDTask): 0.19047618 {CBO: 25}

fragm(org.dom4j.xpath.DefaultXPath): 0.18899864 {CBO: 33}

fragm(org.acm.seguin.uml.refactor.BatchRename): 0.18899864 {CBO: 28}

fragm(net.sourceforge.jrefactory.action.BatchRenameAction$BatchRenameListener): 0.18899864 {CBO: 30}

fragm(Repackage): 0.18736595 {CBO: 18}

fragm(net.sourceforge.jrefactory.uml.render.SaveAdapter): 0.18599439 {CBO: 14}

fragm(edu.umd.cs.findbugs.FindExamples): 0.18524832 {CBO: 19}

fragm(org.acm.seguin.pmd.cpd.cppast.CPPParser): 0.18352073 {CBO: 22}

fragm(org.acm.seguin.tools.install.TagEditorSettingPanel): 0.18295401 {CBO: 30}

fragm(org.jaxen.expr.DefaultAbsoluteLocationPath): 0.18219101 {CBO: 12}

fragm(org.acm.seguin.pmd.swingui.RulePropertyEditingPanel): 0.18219101 {CBO: 36}

fragm(org.acm.seguin.summary.load.TextLoadStatus): 0.18219101 {CBO: 7}

fragm(org.dom4j.tree.DefaultDocument): 0.1809184 {CBO: 21}

fragm(org.apache.bcel.util.SyntheticRepository): 0.18041384 {CBO: 14}

fragm(org.acm.seguin.pmd.cpd.CPPTokenizer): 0.18041384 {CBO: 20}

fragm(org.acm.seguin.pmd.symboltable.TypeSet): 0.18030912 {CBO: 19}

fragm(org.acm.seguin.pmd.swingui.DirectoryTableModel): 0.17990011 {CBO: 13}

fragm(org.acm.seguin.pmd.symboltable.NameOccurrences): 0.17990011 {CBO: 15}

fragm(org.acm.seguin.pmd.rules.UnusedPrivateMethodRule): 0.17990011 {CBO: 23}

fragm(org.acm.seguin.pmd.RuleSet): 0.17958224 {CBO: 12}

fragm(org.acm.seguin.tools.install.SortOptionPanel): 0.1792587 {CBO: 16}

fragm(org.dom4j.tree.DefaultElement): 0.17906731 {CBO: 27}

fragm(org.acm.seguin.io.FileCopy): 0.17837381 {CBO: 15}

fragm(edu.umd.cs.findbugs.detect.FindUnsyncGet): 0.17728549 {CBO: 16}

fragm(edu.umd.cs.findbugs.ba.BetterCFGBuilder2$Subroutine): 0.17728549 {CBO: 23}

fragm(org.acm.seguin.pmd.Report): 0.17647058 {CBO: 16}

fragm(org.acm.seguin.pmd.jaxen.AttributeAxisIterator): 0.17618084 {CBO: 22}

fragm(org.dom4j.io.DispatchHandler): 0.17583781 {CBO: 10}

fragm(org.dom4j.io.SAXContentHandler): 0.17533231 {CBO: 36}

fragm(org.acm.seguin.project.ProjectClassLoader): 0.1750933 {CBO: 30}

fragm(org.apache.bcel.generic.MethodGen): 0.17432499 {CBO: 47}

fragm(org.acm.seguin.pmd.swingui.Preferences): 0.17240894 {CBO: 19}

fragm(org.acm.seguin.refactor.type.CreateNewInterface): 0.17082042 {CBO: 23}

fragm(org.acm.seguin.ide.common.CodingStandardsViewer$6): 0.17082042 {CBO: 20}

fragm(org.acm.seguin.pmd.cpd.JavaLanguage$JavaFileOrDirectoryFilter): 0.17082042 {CBO: 7}

fragm(org.acm.seguin.print.text.LineSet): 0.17082042 {CBO: 5}

fragm(org.acm.seguin.tools.install.ListOrderPanel): 0.17082042 {CBO: 14}

fragm(org.acm.seguin.ide.netbeans.NetbeansSelectedFileSet): 0.17082042 {CBO: 14}

fragm(org.jaxen.function.CeilingFunction): 0.17082036 {CBO: 12}

fragm(org.acm.seguin.pmd.util.ASTViewer$MyPrintStream): 0.17082036 {CBO: 7}

fragm(edu.umd.cs.findbugs.DetectorFactory): 0.17082036 {CBO: 10}

fragm(net.sourceforge.jrefactory.ast.ASTVariableDeclaratorId): 0.17082036 {CBO: 14}

fragm(org.acm.seguin.pmd.swingui.DirectoryTreeNode): 0.17082036 {CBO: 5}

fragm(net.sourceforge.jrefactory.factory.FileParserFactory): 0.17082036 {CBO: 12}

fragm(edu.umd.cs.findbugs.gui.FindBugsFrame$AuxClasspathEntryFileFilter): 0.17082036 {CBO: 6}

fragm(edu.umd.cs.findbugs.gui.FindBugsFrame$ProjectFileFilter): 0.17082036 {CBO: 6}

fragm(edu.umd.cs.findbugs.gui.FindBugsFrame$XMLFileFilter): 0.17082036 {CBO: 6}

fragm(org.acm.seguin.pmd.swingui.AnalysisViewer$HTMLFileFilter): 0.17082036 {CBO: 7}

fragm(edu.umd.cs.findbugs.detect.FindUnreleasedLock$LockFrameModelingVisitor): 0.17082036 {CBO: 24}

fragm(org.jaxen.function.FloorFunction): 0.17082036 {CBO: 12}

fragm(org.acm.seguin.pmd.symboltable.GlobalScope): 0.17082036 {CBO: 14}

fragm(org.acm.seguin.pmd.rules.IdempotentOperationsRule): 0.17082036 {CBO: 19}

fragm(org.apache.bcel.verifier.statics.IntList): 0.17082036 {CBO: 4}

fragm(org.acm.seguin.pmd.symboltable.LocalScope): 0.17082036 {CBO: 14}

fragm(org.jaxen.function.ext.LocaleFunctionSupport): 0.17082036 {CBO: 11}

fragm(org.acm.seguin.pmd.symboltable.MethodScope): 0.17082036 {CBO: 13}

fragm(org.acm.seguin.ide.common.Navigator$RunGotoNode): 0.17082036 {CBO: 11}

fragm(org.acm.seguin.tools.install.OptionPanel): 0.17082036 {CBO: 11}

fragm(org.acm.seguin.pmd.AbstractRule): 0.17082036 {CBO: 12}

fragm(edu.umd.cs.findbugs.PrintingBugReporter): 0.17082036 {CBO: 7}

fragm(org.acm.seguin.ide.common.RefreshDiagramThread): 0.17082036 {CBO: 8}

fragm(org.acm.seguin.pmd.swingui.RulesTree$RulesFileFilter): 0.17082036 {CBO: 6}

fragm(org.acm.seguin.pmd.swingui.SearchViewer$HTMLFileFilter): 0.17082036 {CBO: 7}

fragm(org.jaxen.function.StringFunction): 0.17082036 {CBO: 11}

fragm(org.acm.seguin.ide.common.CodingStandardsViewer$RunGotoViolation): 0.17082036 {CBO: 9}

fragm(org.acm.seguin.ide.common.ASTViewerPane$MyPrintStream): 0.17082036 {CBO: 7}

fragm(org.apache.bcel.classfile.Utility$JavaWriter): 0.17082036 {CBO: 8}

fragm(org.acm.seguin.ide.common.PackageSelectorArea): 0.16989386 {CBO: 30}

fragm(org.acm.seguin.pretty.LineQueue): 0.16937393 {CBO: 9}

fragm(org.dom4j.io.DOMWriter): 0.16910964 {CBO: 38}

fragm(org.apache.bcel.generic.ConstantPoolGen): 0.16724354 {CBO: 28}

fragm(edu.umd.cs.findbugs.ba.bcp.PatternMatcher$State): 0.16712439 {CBO: 29}

fragm(org.acm.seguin.ide.common.options.PropertiesFile): 0.16582769 {CBO: 19}

fragm(org.acm.seguin.metrics.LineCounter): 0.16562748 {CBO: 12}

fragm(edu.umd.cs.findbugs.ba.SourceFinder): 0.16530788 {CBO: 23}

fragm(edu.umd.cs.findbugs.gui.FindBugsFrame): 0.16431803 {CBO: 161}

fragm(org.acm.seguin.refactor.type.RemoveEmptyClassRefactoring): 0.16350877 {CBO: 17}

fragm(edu.umd.cs.findbugs.detect.BCPMethodReturnCheck): 0.16322136 {CBO: 31}

fragm(org.jaxen.expr.DefaultFunctionCallExpr): 0.1628598 {CBO: 13}

fragm(org.apache.bcel.verifier.VerifyDialog): 0.16212225 {CBO: 30}

fragm(org.apache.bcel.verifier.structurals.Pass3bVerifier): 0.16161406 {CBO: 43}

fragm(org.acm.seguin.tools.install.PrettySettingsParser): 0.16138166 {CBO: 20}

fragm(org.acm.seguin.pmd.util.ASTViewer): 0.15997499 {CBO: 32}

fragm(org.acm.seguin.pmd.util.ResourceLoader): 0.15997499 {CBO: 17}

fragm(org.acm.seguin.version.SourceSafe): 0.15719658 {CBO: 21}

fragm(org.acm.seguin.refactor.type.MoveClassVisitor): 0.15653217 {CBO: 21}

fragm(org.acm.seguin.pmd.cpd.PHPLanguage$PHPFileOrDirectoryFilter): 0.15653217 {CBO: 7}

fragm(org.acm.seguin.pmd.cpd.CPPLanguage$CPPFileOrDirectoryFilter): 0.15653217 {CBO: 7}

fragm(edu.umd.cs.findbugs.ba.TargetEnumeratingVisitor): 0.15653217 {CBO: 19}

fragm(edu.umd.cs.findbugs.PackageStats): 0.15648794 {CBO: 14}

fragm(org.dom4j.swing.XMLTableDefinition): 0.15527803 {CBO: 20}

fragm(net.sourceforge.jrefactory.ast.SimpleNode): 0.15504259 {CBO: 24}

fragm(edu.umd.cs.findbugs.ba.BetterCFGBuilder2): 0.15411907 {CBO: 49}

fragm(org.acm.seguin.ide.common.options.JSAlignmentOptionPane): 0.15384614 {CBO: 17}

fragm(net.sourceforge.jrefactory.action.RefactoringAction): 0.15384614 {CBO: 21}

fragm(org.dom4j.tree.AbstractDocument): 0.15319723 {CBO: 29}

fragm(org.jaxen.expr.DefaultLocationPath): 0.1499337 {CBO: 20}

fragm(org.acm.seguin.summary.FrameworkFileSummaryLoader): 0.1499337 {CBO: 25}

fragm(org.dom4j.tree.BackedList): 0.1499337 {CBO: 14}

fragm(org.jaxen.function.NormalizeSpaceFunction): 0.1499337 {CBO: 11}

fragm(edu.umd.cs.findbugs.PluginLoader): 0.1499337 {CBO: 26}

fragm(org.dom4j.swing.BranchTreeNode): 0.1499337 {CBO: 12}

fragm(org.acm.seguin.pmd.swingui.SearchResultsViewer$FilesFilter): 0.1499337 {CBO: 7}

fragm(org.acm.seguin.pmd.swingui.SearchResultsViewer): 0.1499337 {CBO: 10}

fragm(edu.umd.cs.findbugs.ba.DataflowTestDriver): 0.1499337 {CBO: 25}

fragm(edu.umd.cs.findbugs.ba.SourceFinder$DirectorySourceRepository): 0.1499337 {CBO: 8}

fragm(org.apache.bcel.verifier.exc.VerifierConstraintViolatedException): 0.1499337 {CBO: 5}

fragm(org.acm.seguin.ide.netbeans.JRefactory): 0.14974028 {CBO: 119}

fragm(org.acm.seguin.tools.install.PrettyPrinterConfigGUI): 0.14961427 {CBO: 48}

fragm(org.acm.seguin.pmd.symboltable.TypeSet$ExplicitImportResolver): 0.14888567 {CBO: 9}

fragm(edu.umd.cs.findbugs.detect.FindUnreleasedLock): 0.14699662 {CBO: 33}

fragm(org.acm.seguin.refactor.type.AddClassRefactoring): 0.14616233 {CBO: 16}

fragm(org.acm.seguin.pmd.swingui.HTMLResultRenderer): 0.14553261 {CBO: 11}

fragm(edu.umd.cs.findbugs.ba.ClassContext): 0.14549726 {CBO: 53}

fragm(org.apache.bcel.util.BCELifier): 0.14469326 {CBO: 25}

fragm(org.acm.seguin.refactor.method.EMParameterFinder): 0.14372635 {CBO: 21}

fragm(org.acm.seguin.pmd.swingui.AnalysisViewer$SaveAsActionListener): 0.14372635 {CBO: 24}

fragm(org.acm.seguin.ide.netbeans.NetbeansCurrentSummary): 0.14372635 {CBO: 19}

fragm(org.acm.seguin.pmd.swingui.SearchViewer$SaveAsActionListener): 0.14372635 {CBO: 24}

fragm(org.apache.bcel.classfile.Unknown): 0.14372635 {CBO: 17}

fragm(org.dom4j.bean.BeanMetaData): 0.14372635 {CBO: 19}

fragm(org.acm.seguin.ide.common.PluginSourceBrowser): 0.14076829 {CBO: 11}

fragm(org.acm.seguin.refactor.method.PushDownMethodRefactoring): 0.14076829 {CBO: 19}

fragm(net.sourceforge.jrefactory.action.ShowSourceAction$ShowSourceListener): 0.14076829 {CBO: 17}

fragm(org.acm.seguin.ide.netbeans.NetBeansPrettyPrinter): 0.14039445 {CBO: 16}

fragm(org.acm.seguin.pmd.rules.AvoidDuplicateLiteralsRule): 0.14019626 {CBO: 20}

fragm(org.acm.seguin.pmd.swingui.PrintAnalysisResults): 0.13980347 {CBO: 25}

fragm(edu.umd.cs.findbugs.ba.CFGPrinter): 0.1397028 {CBO: 27}

fragm(org.acm.seguin.pmd.ant.Formatter): 0.13948858 {CBO: 18}

fragm(edu.umd.cs.findbugs.detect.DumbMethods): 0.13790524 {CBO: 17}

fragm(edu.umd.cs.findbugs.I18N): 0.13604832 {CBO: 8}

fragm(org.acm.seguin.summary.ReflectiveSummaryLoader): 0.13433504 {CBO: 22}

fragm(org.dom4j.tree.AbstractNode): 0.13245553 {CBO: 22}

fragm(org.jaxen.function.ext.EndsWithFunction): 0.13245553 {CBO: 10}

fragm(org.acm.seguin.pmd.swingui.AboutPMD): 0.13245553 {CBO: 49}

fragm(org.jaxen.expr.DefaultRelationalExpr): 0.13245553 {CBO: 16}

fragm(edu.umd.cs.findbugs.ba.InnerClassAccessMap): 0.13245553 {CBO: 17}

fragm(LineNumberTool): 0.13245553 {CBO: 11}

fragm(org.acm.seguin.refactor.method.ObjectReference): 0.13245553 {CBO: 10}

fragm(org.acm.seguin.pmd.PMDException): 0.13245553 {CBO: 7}

fragm(org.jaxen.function.ContainsFunction): 0.13245553 {CBO: 10}

fragm(org.dom4j.persistence.PersitenceManager): 0.13245553 {CBO: 14}

fragm(net.sourceforge.jrefactory.action.RenameFieldAction): 0.13245553 {CBO: 19}

fragm(org.acm.seguin.refactor.field.RenameSystemTraversal): 0.13245553 {CBO: 27}

fragm(org.acm.seguin.pmd.swingui.RulesTreeNode): 0.13245553 {CBO: 18}

fragm(org.dom4j.io.SAXValidator): 0.13245553 {CBO: 15}

fragm(org.jaxen.expr.DefaultEqualityExpr): 0.13245553 {CBO: 18}

fragm(org.jaxen.function.StartsWithFunction): 0.13245553 {CBO: 10}

fragm(org.acm.seguin.ide.common.ClassDiagramReloader): 0.13245553 {CBO: 11}

fragm(org.dom4j.bean.BeanDocumentFactory): 0.13245553 {CBO: 19}

fragm(edu.umd.cs.findbugs.BugCodeMatcher): 0.13245553 {CBO: 9}

fragm(org.dom4j.xpath.XPathPattern): 0.13245553 {CBO: 27}

fragm(edu.umd.cs.findbugs.detect.FindNullDeref): 0.13089842 {CBO: 35}

fragm(org.acm.seguin.project.Path): 0.13011044 {CBO: 14}

fragm(org.acm.seguin.ide.common.options.JSTagsOptionPane$TagsTableModel): 0.12935573 {CBO: 16}

fragm(org.dom4j.io.SAXWriter):	0.12935573	{CBO: 41}
fragm(edu.umd.cs.findbugs.Filter):	0.12831211	{CBO: 29}
fragm(org.apache.bcel.util.MethodHTML):	0.12493241	{CBO: 19}
fragm(edu.umd.cs.findbugs.SelfCalls):	0.12319642	{CBO: 29}
fragm(org.acm.seguin.pmd.rules.AccessorClassGenerationRule$AllocData):	0.12275994	{CBO: 14}
fragm(org.acm.seguin.pmd.rules.DuplicateImportsRule):	0.122587085	{CBO: 16}
fragm(org.acm.seguin.tools.install.AlphabeticalOrderPanel):	0.122587085	{CBO: 9}
fragm(org.acm.seguin.tools.install.InitializerOrderPanel):	0.122587085	{CBO: 9}
fragm(edu.umd.cs.findbugs.ba.ResourceValueAnalysis):	0.122587085	{CBO: 35}
fragm(org.jaxen.expr.DefaultFilterExpr):	0.122587085	{CBO: 17}
fragm(org.acm.seguin.tools.install.BeanOrderPanel):	0.122587085	{CBO: 9}
fragm(org.acm.seguin.util.BackupTraversal):	0.12144321	{CBO: 10}
fragm(org.apache.bcel.verifier.structurals.ControlFlowGraph$InstructionContextImpl):	0.120318055	{CBO: 30}
fragm(org.acm.seguin.ide.command.CommandLineSourceBrowser):	0.11971164	{CBO: 13}
fragm(org.acm.seguin.pmd.rules.AtLeastOneConstructorRule):	0.11812252	{CBO: 17}
fragm(edu.umd.cs.findbugs.detect.FindRefComparison+1):	0.11812252	{CBO: 22}
fragm(org.apache.bcel.verifier.NativeVerifier):	0.11812252	{CBO: 11}
fragm(edu.umd.cs.findbugs.ba.AssignedFieldMap):	0.11812252	{CBO: 23}
fragm(org.acm.seguin.pmd.cpd.PHPTokenizer):	0.11812252	{CBO: 15}
fragm(net.sourceforge.jrefactory.action.SaveImageAction):	0.11812252	{CBO: 29}
fragm(org.acm.seguin.pmd.cpd.SourceCode):	0.11812252	{CBO: 7}
fragm(org.jaxen.function.SubstringFunction):	0.11812252	{CBO: 13}
fragm(org.apache.bcel.util.ClassPath$Dir):	0.11812252	{CBO: 10}
fragm(edu.umd.cs.findbugs.ba.AnalysisException):	0.11812252	{CBO: 9}
fragm(edu.umd.cs.findbugs.ba.ValueNumberFrameModelingVisitor):	0.11540228	{CBO: 42}
fragm(org.acm.seguin.tools.build.CreateVersion):	0.114798665	{CBO: 12}
fragm(edu.umd.cs.findbugs.ba.IsNullValueAnalysis):	0.1139406	{CBO: 42}
fragm(net.sourceforge.jrefactory.uml.line.SegmentedLine):	0.1139406	{CBO: 22}
fragm(edu.umd.cs.findbugs.detect.LazyInit):	0.11313933	{CBO: 46}
fragm(org.acm.seguin.metrics.LCTraversal):	0.11313617	{CBO: 10}
fragm(org.apache.bcel.util.AttributeHTML):	0.111218154	{CBO: 27}
fragm(edu.umd.cs.findbugs.BugInstance):	0.110783935	{CBO: 33}
fragm(org.acm.seguin.refactor.type.AddChildRefactoring):	0.1063391	{CBO: 14}
fragm(org.apache.bcel.generic.FCONST):	0.1063391	{CBO: 16}
fragm(edu.umd.cs.findbugs.detect.FindOpenStream$StreamFrameModelingVisitor):	0.1063391	{CBO: 20}
fragm(edu.umd.cs.findbugs.gui.Hello):	0.1063391	{CBO: 6}
fragm(org.apache.bcel.generic.ICONST):	0.1063391	{CBO: 16}
fragm(edu.umd.cs.findbugs.IntAnnotation$IntAnnotationXMLTranslator):	0.1063391	{CBO: 12}
fragm(org.dom4j.InvalidXPathException):	0.1063391	{CBO: 5}
fragm(org.acm.seguin.pmd.cpd.JavaTokenizer):	0.1063391	{CBO: 19}
fragm(org.apache.bcel.generic.LCONST):	0.1063391	{CBO: 16}
fragm(edu.umd.cs.findbugs.MethodAnnotation$MethodAnnotationXMLTranslator):	0.1063391	{CBO: 14}
fragm(edu.umd.cs.findbugs.gui.AboutDialog):	0.1063391	{CBO: 37}
fragm(org.jaxen.function.BooleanFunction):	0.1063391	{CBO: 10}
fragm(net.sourceforge.jrefactory.ast.ASTMultiplicativeExpression):	0.1063391	{CBO: 8}
fragm(org.acm.seguin.ide.common.PackageNameLoader):	0.1063391	{CBO: 8}
fragm(org.acm.seguin.ide.common.PackageSelectorPanel$1):	0.1063391	{CBO: 15}
fragm(org.acm.seguin.ide.common.PackageSelectorPanel$RemoveAction):	0.1063391	{CBO: 12}
fragm(net.sourceforge.jrefactory.factory.ParserFactory):	0.1063391	{CBO: 17}
fragm(org.acm.seguin.pretty.PrintSpecialDefault):	0.1063391	{CBO: 9}
fragm(net.sourceforge.jrefactory.ast.ASTRelationalExpression):	0.1063391	{CBO: 8}
fragm(org.apache.bcel.generic.DCONST):	0.1063391	{CBO: 16}
fragm(edu.umd.cs.findbugs.ba.PruneInfeasibleExceptionEdges+1):	0.1063391	{CBO: 8}

fragm(net.sourceforge.jrefactory.ast.ASTEqualityExpression): 0.1063391 {CBO: 8}

fragm(net.sourceforge.jrefactory.ast.ASTAdditiveExpression): 0.1063391 {CBO: 8}

fragm(net.sourceforge.jrefactory.action.RenameFieldAction$RenameFieldDialog): 0.1063391 {CBO: 16}

fragm(org.acm.seguin.uml.refactor.RenameFieldDialog): 0.1063391 {CBO: 15}

fragm(net.sourceforge.jrefactory.action.RenameMethodAction$RenameMethodDialog): 0.1063391 {CBO: 12}

fragm(org.acm.seguin.uml.refactor.RenameMethodDialog): 0.1063391 {CBO: 11}

fragm(org.acm.seguin.pmd.swingui.SearchViewer$SaveActionListener): 0.1063391 {CBO: 20}

fragm(net.sourceforge.jrefactory.ast.ASTShiftExpression): 0.1063391 {CBO: 8}

fragm(edu.umd.cs.findbugs.ba.StandardTypeMerger): 0.1063391 {CBO: 15}

fragm(org.acm.seguin.pmd.swingui.AnalysisViewer$SaveActionListener): 0.1063391 {CBO: 20}

fragm(org.acm.seguin.refactor.type.AddAbstractParent): 0.1063391 {CBO: 18}

fragm(edu.umd.cs.findbugs.XMLBugReporter): 0.1063391 {CBO: 10}

fragm(edu.umd.cs.findbugs.ba.CFG): 0.1063391 {CBO: 17}

fragm(edu.umd.cs.findbugs.ba.ZipSourceFileDataSource): 0.1063391 {CBO: 10}

fragm(org.acm.seguin.tools.install.IndexedPanel): 0.10633904 {CBO: 12}

fragm(org.acm.seguin.uml.line.SizableLabel): 0.10633904 {CBO: 21}

fragm(org.acm.seguin.metrics.CommaDelimitedReport): 0.10446787 {CBO: 8}

fragm(org.acm.seguin.ide.common.options.SelectedPanel): 0.1038596 {CBO: 27}

fragm(org.acm.seguin.ide.common.IDEPlugin): 0.10370344 {CBO: 16}

fragm(edu.umd.cs.findbugs.BugInstance$BugInstanceXMLTranslator): 0.10370344 {CBO: 16}

fragm(org.acm.seguin.pmd.cpd.GUI): 0.10189319 {CBO: 45}

fragm(org.apache.bcel.verifier.TransitiveHull): 0.1012283 {CBO: 14}

fragm(edu.umd.cs.findbugs.ByteCodePatternDetector): 0.09962535 {CBO: 21}

fragm(org.dom4j.XPathException): 0.09962535 {CBO: 7}

fragm(org.dom4j.io.XMLWriter): 0.097946346 {CBO: 44}

fragm(edu.umd.cs.findbugs.detect.FindSpinLoop): 0.09655875 {CBO: 14}

fragm(org.jaxen.function.NumberFunction): 0.09655875 {CBO: 14}

fragm(org.acm.seguin.pretty.PrintData$JavaOutputStreamWriter): 0.09655875 {CBO: 8}

fragm(org.apache.bcel.verifier.exc.AssertionViolatedException): 0.09655875 {CBO: 5}

fragm(org.acm.seguin.refactor.field.PushDownFieldRefactoring): 0.094331145 {CBO: 26}

fragm(org.apache.bcel.verifier.statics.Pass2Verifier): 0.0940541 {CBO: 30}

fragm(org.dom4j.IllegalAddException): 0.09366572 {CBO: 8}

fragm(org.dom4j.tree.AbstractProcessingInstruction): 0.09366572 {CBO: 18}

fragm(org.acm.seguin.summary.query.TopLevelDirectory): 0.09366572 {CBO: 18}

fragm(org.apache.bcel.verifier.statics.LocalVariableInfo): 0.09228009 {CBO: 8}

fragm(org.acm.seguin.pretty.line.NumberedLineQueue): 0.09146738 {CBO: 6}

fragm(org.apache.bcel.generic.InstructionFactory): 0.089244425 {CBO: 78}

fragm(org.jaxen.function.ext.MatrixConcatFunction): 0.08834845 {CBO: 13}

fragm(edu.umd.cs.findbugs.graph.AbstractGraph): 0.08834839 {CBO: 15}

fragm(org.apache.bcel.generic.BIPUSH): 0.08834839 {CBO: 18}

fragm(Metrics): 0.08834839 {CBO: 16}

fragm(org.apache.bcel.util.ClassLoader): 0.08834839 {CBO: 21}

fragm(org.dom4j.io.PruningElementStack): 0.08834839 {CBO: 11}

fragm(net.sourceforge.jrefactory.action.RenameFieldAction$RenameFieldListener): 0.08834839 {CBO: 16}

fragm(edu.umd.cs.findbugs.ba.ResourceValueAnalysisTestDriver): 0.08834839 {CBO: 24}

fragm(edu.umd.cs.findbugs.ba.ReturnPathAnalysis): 0.08834839 {CBO: 19}

fragm(net.sourceforge.jrefactory.uml.RoleHolder): 0.08834839 {CBO: 17}

fragm(edu.umd.cs.findbugs.ba.bcp.Binding): 0.08834839 {CBO: 6}

fragm(org.apache.bcel.generic.SIPUSH): 0.08834839 {CBO: 18}

fragm(org.acm.seguin.refactor.EliminatePackageImportVisitor): 0.08834839 {CBO: 16}

fragm(edu.umd.cs.findbugs.ba.TypeAnalysis): 0.08834839 {CBO: 30}

fragm(org.apache.bcel.generic.InstructionList): 0.08787137 {CBO: 31}

fragm(org.dom4j.datatype.SchemaParser): 0.08737844 {CBO: 32}

fragm(org.apache.bcel.generic.ClassGen): 0.08697736 {CBO: 31}

fragm(org.apache.bcel.generic.Instruction): 0.08608526 {CBO: 26}

fragm(org.apache.bcel.util.BCELFactory): 0.08539528 {CBO: 45}

fragm(org.acm.seguin.refactor.type.RenameClassRefactoring): 0.08472538 {CBO: 23}

fragm(edu.umd.cs.findbugs.detect.SerializableIdiom): 0.08472538 {CBO: 23}

fragm(org.apache.bcel.generic.Type): 0.08368212 {CBO: 30}

fragm(org.jaxen.pattern.LocationPathPattern): 0.083580375 {CBO: 15}

fragm(edu.umd.cs.findbugs.ba.DominatorsAnalysis): 0.083580375 {CBO: 26}

fragm(org.acm.seguin.pmd.swingui.RulesTreeModel): 0.083580375 {CBO: 26}

fragm(edu.umd.cs.findbugs.ba.LockAnalysis): 0.081377685 {CBO: 27}

fragm(org.acm.seguin.refactor.field.RenameFieldVisitor): 0.08010912 {CBO: 41}

fragm(org.acm.seguin.pmd.rules.CouplingBetweenObjectsRule): 0.07539648 {CBO: 21}

fragm(org.apache.bcel.classfile.ConstantUtf8): 0.07539642 {CBO: 11}

fragm(org.apache.bcel.classfile.ConstantLong): 0.07539642 {CBO: 12}

fragm(org.apache.bcel.classfile.ConstantFloat): 0.07539642 {CBO: 12}

fragm(org.jaxen.expr.DefaultXPathFactory): 0.07539642 {CBO: 65}

fragm(org.acm.seguin.print.PagePrinter): 0.07539642 {CBO: 19}

fragm(org.jaxen.expr.DefaultTruthExpr): 0.07539642 {CBO: 9}

fragm(org.apache.bcel.classfile.ClassParser): 0.07539642 {CBO: 18}

fragm(org.apache.bcel.classfile.ConstantDouble): 0.07539642 {CBO: 12}

fragm(org.apache.bcel.classfile.ConstantInteger): 0.07539642 {CBO: 12}

fragm(org.acm.seguin.refactor.type.CreateClass): 0.073539376 {CBO: 35}

fragm(org.acm.seguin.pmd.symboltable.TypeSet$ImportOnDemandResolver): 0.073539376 {CBO: 9}

fragm(org.acm.seguin.summary.PrintVisitor): 0.07186276 {CBO: 22}

fragm(org.acm.seguin.refactor.method.RenameMethodVisitor): 0.07119441 {CBO: 31}

fragm(org.dom4j.tree.AbstractDocumentType): 0.07021445 {CBO: 12}

fragm(org.acm.seguin.metrics.TextReport): 0.068297505 {CBO: 9}

fragm(org.acm.seguin.refactor.method.ExtractMethodRefactoring): 0.0682165 {CBO: 43}

fragm(org.acm.seguin.pmd.rules.junit.JUnitAssertionsShouldIncludeMessageRule): 0.06568545 {CBO: 17}

fragm(org.acm.seguin.pmd.cpd.LanguageFactory): 0.06568545 {CBO: 9}

fragm(org.apache.bcel.generic.LineNumberGen): 0.06568545 {CBO: 14}

fragm(edu.umd.cs.findbugs.ba.LineNumberMap): 0.06568545 {CBO: 16}

fragm(org.dom4j.util.NodeComparator): 0.06568545 {CBO: 22}

fragm(edu.umd.cs.findbugs.ClassMatcher): 0.06568545 {CBO: 10}

fragm(org.apache.bcel.classfile.PMGClass): 0.06568545 {CBO: 14}

fragm(org.jaxen.expr.PredicateSet): 0.06568545 {CBO: 17}

fragm(org.dom4j.xpp.ProxyXmlStartTag): 0.06568545 {CBO: 14}

fragm(Refactory): 0.06568545 {CBO: 21}

fragm(org.acm.seguin.tools.install.RefactoryInstaller): 0.06568545 {CBO: 25}

fragm(net.sourceforge.jrefactory.action.RenameMethodAction): 0.06568545 {CBO: 19}

fragm(org.apache.bcel.classfile.Synthetic): 0.06568545 {CBO: 13}

fragm(org.acm.seguin.refactor.type.TypeChangeVisitor): 0.06568545 {CBO: 28}

fragm(net.sourceforge.jrefactory.uml.UMLPopupMenu): 0.06568545 {CBO: 51}

fragm(net.sourceforge.jrefactory.uml.line.Vertex): 0.06568545 {CBO: 9}

fragm(org.dom4j.swing.XMLTableColumnDefinition): 0.06568545 {CBO: 16}

fragm(edu.umd.cs.findbugs.ba.BytecodeScanner): 0.06561273 {CBO: 13}

fragm(org.dom4j.datatype.DatatypeDocumentFactory): 0.061695814 {CBO: 21}

fragm(edu.umd.cs.findbugs.detect.FindInconsistentSync2+2): 0.0616017 {CBO: 43}

fragm(org.acm.seguin.pmd.RuleSetWriter): 0.061546564 {CBO: 13}

fragm(org.dom4j.tree.ContentListFacade): 0.060469806 {CBO: 11}

fragm(edu.umd.cs.findbugs.BugHistory): 0.06031555 {CBO: 16}

fragm(org.acm.seguin.refactor.method.RenameMethodRefactoring): 0.060230315 {CBO: 25}

fragm(net.sourceforge.jrefactory.uml.UMLType): 0.06007439 {CBO: 44}

fragm(org.jaxen.expr.DefaultUnaryExpr): 0.05815631 {CBO: 13}

fragm(edu.umd.cs.findbugs.ba.bcp.PatternMatcher): 0.05815631 {CBO: 30}

fragm(edu.umd.cs.findbugs.detect.FindDoubleCheck): 0.05815631 {CBO: 12}

fragm(edu.umd.cs.findbugs.ResourceTrackingDetector+1): 0.05815631 {CBO: 26}

fragm(org.acm.seguin.pmd.rules.strictexception.ExceptionTypeChecking): 0.05815631 {CBO: 23}

fragm(edu.umd.cs.findbugs.ba.AvailableLoad): 0.05815631 {CBO: 11}

fragm(edu.umd.cs.findbugs.HTMLBugReporter): 0.05815631 {CBO: 22}

fragm(org.acm.seguin.pmd.renderers.EmacsRenderer): 0.05815631 {CBO: 10}

fragm(org.apache.bcel.verifier.VerifierAppFrame): 0.05815631 {CBO: 46}

fragm(org.dom4j.tree.QNameCache): 0.05601102 {CBO: 13}

fragm(edu.umd.cs.findbugs.detect.FindOpenStream$StreamResourceTracker): 0.055600584 {CBO: 33}

fragm(edu.umd.cs.findbugs.ba.Dataflow): 0.05542648 {CBO: 17}

fragm(org.acm.seguin.pmd.rules.ConstructorCallsOverridableMethodRule): 0.05215764 {CBO: 41}

fragm(org.acm.seguin.summary.SummaryLoadVisitor): 0.05215764 {CBO: 67}

fragm(org.acm.seguin.ide.common.options.JSCommentOptionPane): 0.05215758 {CBO: 14}

fragm(org.acm.seguin.pmd.RuleProperties): 0.05215758 {CBO: 11}

fragm(edu.umd.cs.findbugs.ba.StackDepthAnalysis): 0.05215758 {CBO: 21}

fragm(org.acm.seguin.pmd.rules.UnnecessaryConversionTemporaryRule): 0.05215758 {CBO: 16}

fragm(org.dom4j.dom.DOMNodeHelper): 0.05143684 {CBO: 24}

fragm(org.apache.bcel.util.Class2HTML): 0.04964447 {CBO: 25}

fragm(edu.umd.cs.findbugs.ClassAnnotation): 0.047270477 {CBO: 15}

fragm(org.acm.seguin.pmd.renderers.CSVRenderer): 0.047270477 {CBO: 11}

fragm(edu.umd.cs.findbugs.gui.FindBugsFrame$BugTreeNode): 0.047270477 {CBO: 10}

fragm(edu.umd.cs.findbugs.detect.FindUnreleasedLock$LockResourceTracker): 0.047270477 {CBO: 26}

fragm(org.acm.seguin.pmd.CommandLineOptions): 0.045837104 {CBO: 16}

fragm(org.acm.seguin.JRefactoryVersion): 0.0432145 {CBO: 6}

fragm(net.sourceforge.jrefactory.parser.JavaParserTokenManager): 0.0432145 {CBO: 12}

fragm(edu.umd.cs.findbugs.detect.FindFinalizeInvocations): 0.0432145 {CBO: 16}

fragm(edu.umd.cs.findbugs.ba.SimplePathEnumerator): 0.0432145 {CBO: 19}

fragm(org.acm.seguin.tools.international.StringListVisitor): 0.0432145 {CBO: 13}

fragm(org.acm.seguin.pmd.symboltable.TypeSet$PrimitiveTypeResolver): 0.0432145 {CBO: 16}

fragm(org.acm.seguin.tools.install.MultilineSettingPanel): 0.041435122 {CBO: 17}

fragm(org.acm.seguin.pmd.symboltable.NameOccurrence): 0.041435122 {CBO: 14}

fragm(edu.umd.cs.findbugs.FindBugs$ZipClassProducer): 0.041435122 {CBO: 17}

fragm(org.apache.bcel.verifier.structurals.LocalVariables): 0.040728748 {CBO: 13}

fragm(org.apache.bcel.util.CodeHTML): 0.0399462 {CBO: 28}

fragm(org.dom4j.datatype.DatatypeAttribute): 0.039795637 {CBO: 17}

fragm(org.dom4j.datatype.DatatypeElement): 0.039795637 {CBO: 21}

fragm(org.dom4j.io.SAXReader): 0.036875486 {CBO: 31}

fragm(org.acm.seguin.tools.install.TagListModel): 0.036875486 {CBO: 7}

fragm(org.acm.seguin.ide.common.Navigator): 0.03599465 {CBO: 38}

fragm(org.apache.bcel.util.ConstantHTML): 0.03599465 {CBO: 20}

fragm(org.apache.bcel.classfile.Signature): 0.0351547 {CBO: 15}

fragm(org.acm.seguin.ide.common.options.JSHelpOptionPane): 0.0343529 {CBO: 34}

fragm(org.acm.seguin.pmd.ExternalRuleID): 0.0343529 {CBO: 5}

fragm(edu.umd.cs.findbugs.SourceLineAnnotation): 0.0343529 {CBO: 28}

fragm(org.acm.seguin.pretty.PrettyPrintVisitor): 0.033747554 {CBO: 151}

fragm(org.apache.bcel.classfile.JavaClass): 0.033366144 {CBO: 36}

fragm(org.dom4j.io.OutputFormat): 0.032152057 {CBO: 6}

fragm(org.acm.seguin.summary.TypeSummary): 0.032152057 {CBO: 12}

fragm(net.sourceforge.jrefactory.parser.ParseException): 0.031705797 {CBO: 8}

fragm(edu.umd.cs.findbugs.ba.bcp.PatternElement):	0.030215442	{CBO: 18}
fragm(org.acm.seguin.pmd.util.ASTViewer$XPathListener):	0.0269652	{CBO: 26}
fragm(org.acm.seguin.ide.common.ASTViewerPane$XPathListener):	0.0269652	{CBO: 27}
fragm(org.apache.bcel.classfile.StackMapType):	0.0269652	{CBO: 12}
fragm(net.sourceforge.jrefactory.parser.JavaParser):	0.025292277	{CBO: 130}
fragm(net.sourceforge.jrefactory.io.JavaCharStream):	0.02434498	{CBO: 11}
fragm(org.apache.bcel.generic.CodeExceptionGen):	0.02434498	{CBO: 17}
fragm(org.acm.seguin.pmd.rules.DoubleCheckedLockingRule):	0.023216665	{CBO: 32}
fragm(org.apache.bcel.classfile.Utility):	0.022252321	{CBO: 33}
fragm(org.saxpath.conformance.ConformanceXPathHandler):	0.021945	{CBO: 8}
fragm(edu.umd.cs.findbugs.FindBugsMessageFormat):	0.021246672	{CBO: 9}
fragm(org.apache.bcel.generic.LocalVariableGen):	0.020381808	{CBO: 19}
fragm(edu.umd.cs.findbugs.visitclass.DismantleBytecode):	0.019584477	{CBO: 31}
fragm(org.acm.seguin.summary.FieldAccessSummary):	0.018163264	{CBO: 16}
fragm(org.acm.seguin.pmd.renderers.TextRenderer):	0.017527223	{CBO: 11}
fragm(edu.umd.cs.findbugs.FieldAnnotation):	0.015860677	{CBO: 26}
fragm(org.apache.bcel.classfile.ConstantPool):	0.012814045	{CBO: 26}
fragm(edu.umd.cs.findbugs.ba.Edge):	0.0126520395	{CBO: 14}
fragm(org.acm.seguin.pmd.RuleSetReader$MainContentHandler):	0.011759758	{CBO: 22}
fragm(org.acm.seguin.ant.CVSUtil$CVSEntry):	0.010984957	{CBO: 14}
fragm(net.sourceforge.jrefactory.parser.TokenMgrError):	0.010748923	{CBO: 6}
fragm(org.apache.bcel.verifier.statics.Pass3aVerifier$InstOperandConstraintVisitor):	0.010578394	{CBO: 65}
fragm(edu.umd.cs.findbugs.MethodAnnotation):	0.010522783	{CBO: 21}
fragm(org.acm.seguin.pmd.renderers.HTMLRenderer):	0.009705961	{CBO: 11}
fragm(org.apache.bcel.verifier.statics.Pass3aVerifier):	0.009521186	{CBO: 42}
fragm(org.apache.bcel.verifier.statics.Pass2Verifier$CPESSC_Visitor):	0.009354353	{CBO: 59}
fragm(org.acm.seguin.pretty.ai.MethodAnalyzer):	0.007091105	{CBO: 28}
fragm(org.acm.seguin.summary.LineCountVisitor):	0.0031151772	{CBO: 117}
fragm(org.apache.bcel.verifier.structurals.InstConstraintVisitor):	0.0029916763	{CBO: 201}

Capitolo 5. Conclusioni

Il nostro sistema software offre un valido strumento tramite il quale il programmatore può monitorare, in ogni momento, il livello di modularità delle proprie applicazioni.

Un livello di *fragm* troppo alto indica scarsa modularità, tramite la nostra applicazione si può risalire facilmente verso quali metodi di quali classi e package vengono fatte delle chiamate per una data classe di input, ciò semplifica il lavoro di refactoring che il programmatore effettuerà ai fini di diminuire il valore di *fragm*.

Inoltre, nel caso in cui il programmatore necessita analizzare sistemi software di grandi dimensioni, può servirsi della tecnologia OpenCL per parallelizzare i calcoli (sia su GPU che sulla stessa CPU), ottenendo i risultati in tempi più brevi.

Abbiamo provato ad analizzare la modularità di tre noti sistemi software Java come *JUnit*, *JRefactory* e *JHotDraw* e abbiamo scoperto che ci sono parecchie classi che hanno valori di fragm maggiore di zero. Quindi, nonostante tali valori non siano troppo elevati (non vanno mai oltre 0.5), si potrebbero comunque effettuare dei lavori di refactoring mirati a ridurre tali valori per migliorare la modularità di questi sistemi software.

Oltre ai valori di fragm per ogni classe, abbiamo fatto stampare all'applicazione anche i valori di CBO (Coupling Between Object classes), cioè la metrica che mostra il numero di classi esterne con cui una classe interagisce: è interessante notare come non vi sia alcuna correlazione tra il valore di fragm e quello di CBO. Cioè non è detto che una classe con basso livello di CBO abbia un altrettanto basso livello di fragm e viceversa.

Capitolo 6. Bibliografia

[1] A. Di Stefano, M. Fargetta, G. Pappalardo, E. Tramontana. Metrics for evaluating concern separation and composition.
Proceedings of the 2005 ACM Symposium on Applied computing.

[2] E. Tramontana. Automatically Characterising Components with Concerns and Reducing Tangling. Proceedings of the 2013 IEEE COMPSAC International workshop QUORS.