Application de l'analyse technique financière à l'analyse de traces d'exécution de programmes

Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES

par :

David SENNHAUSER

Conseiller au travail de Bachelor :

(Philippe DUGERDIL, Professeur HES)

Carouge, le 1er décembre 2008

Haute École de Gestion de Genève (HEG-GE)

Filière Informatique de Gestion

Déclaration

Ce travail de Bachelor est réalisé dans le cadre de l'examen final de la Haute école de gestion de Genève, en vue de l'obtention du titre de Bachelor of Science. L'étudiant accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et recommandations formulées dans le travail de Bachelor, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de Bachelor, du juré et de la HEG.

« J'atteste avoir réalisé seul le présent travail, sans avoir utilisé des sources autres que celles citées dans la bibliographie. »

Fait à Carouge, le 1er décembre 2008

David Sennhauser

Remerciements

Je remercie toutes les personnes qui ont participé de près ou de loin à la réalisation de ce projet, tout particulièrement Monsieur Philippe Dugerdil qui m'a suivi et conseillé tout au long de ce travail.

Je remercie aussi Monsieur Peter Daehne qui, grâce à ses précieux conseils, a pu m'aider à avancer dans ce projet.

Je tiens également à remercier Karine Pasquier m'ayant soutenu pendant la réalisation de ce mandat et qui a relu avec attention mon travail.

Pour terminer, un grand merci à Odorico von Susani, de Pictet & Cie, qui a accepté d'être mon juré.

Sommaire

Dans le cycle de vie des systèmes d'information, la maintenance est ce qui coûte le plus cher. L'architecture évolue avec les besoins des utilisateurs, les besoins changent et de nombreuses modifications sont effectuées. Avec le temps l'architecture peut se révéler de plus en plus inadaptée parce qu'elle est de plus en plus complexe pour le développeur.

De plus, la documentation décrit rarement de manière complète tout le système d'information. D'où le besoin de procéder au réengineering[1] du système, en général lorsque les besoins non-fonctionnels [2] ne sont plus respectés.

Lors du processus réengineering, on est amené à retrouver l'architecture du système. On a besoin de mapper des éléments [3]du code source avec des business use-cases. C'est l'objectif principal de ce travail.

Pour réaliser ceci, on enregistre l'exécution d'un use-case métier. On obtient une trace d'exécution qui contient tous les éléments du code source qui sont intervenus dans l'exécution du système.

A partir des informations de cette trace et à l'aide de l'analyse technique, je proposerai une technique permettant de retrouver les classes de la trace qui travaillent ensemble, avec une possibilité qu'elles représentent l'implémentation de la fonction logiciel analysée.

Je montre également l'application que j'ai réalisé qui implémente la technique de corrélation dynamique que je propose. Je procèderai ensuite à l'expérimentation de cette technique pour montrer les résultats qu'elle produit.

J'aborderai aussi les considérations d'implémentation de mon application. Et pour finir, je conclurai et présenterai de nouvelles idées pour apporter des améliorations à ce qui a été réalisé lors de ce travail.

---

1 DUGERDIL Ph. - *Architecture-Based Software Reengineering*. Technical report, HEG Geneva, February 2006

2 La portabilité, la maintenabilité, la performance, etc.

3 Des classes java ou des éléments de même granularité dans d'autres langages.

Table des matières

Liste des Figures

Introduction

Afin dans le cadre des recherches sur le Réengineering, il est nécessaire dans certains cas de devoir retrouver l'architecture d'un système. Se basant sur l'analyse de traces d'exécution des fonctions d'une application, on essaye de retrouver les classes qui implémentent ces fonctions.

Ce travail est très proche des préoccupations du groupe de recherche de Philippe Dugerdil et se focalise sur la corrélation dynamique de classes de manière à retrouver l'architecture de l'application. Il expliquera comment appliquer l'analyse technique à l'analyse dynamique de trace d'exécution.

Hormis le fait que ce travail est basé sur l'analyse dynamique des traces d'exécution, le but est comprendre quels sont les éléments du code source qui travaillent ensemble et implémentent un ou plusieurs « use cases » du système à analyser. Il suffira de laisser l'application se définir elle-même à travers une exécution que nous enregistrerons puis analyserons. Nous pouvons faire une analyse ciblée en exécutant le(s) « use case(s) » qui contiennent les fonctions dont on veut retrouver l'implémentation.

Le problème à résoudre

1.1 Réengineering

Actuellement, beaucoup d'entreprises ont des processus métier critiques et complètement automatisés les rendent complètement dépendantes de leur système d'information.

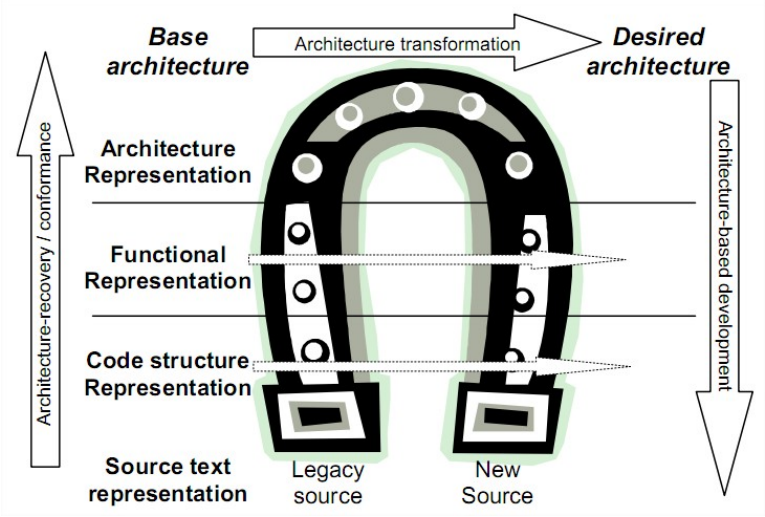
Un système d'information doit d'évoluer en fonction des besoins de l'entreprise. Pour que les changements nécessaires soient appliqués le vite possible, la documentation du système n'est pas mise à jour correctement, étant donné que le système évolue constamment et que sa structure devient de plus en plus complexe.

Un projet de réengineering peut diviser en trois étapes :

1. La première étape consiste en l'analyse du système existant pour s'en construire une représentation fidèle en mettant à jour la documentation le décrivant.
2. La deuxième étape consiste à déterminer les changements dans l'architecture existante.
3. La troisième étape consiste en le développement du nouveau système qui correspond aux besoins.

Figure 1

Le model de fer à cheval du SEI



[Tiré de : Bergey J., Smith D., Weiderman N., Woods S. – Options Analysis for Reengineering (OAR) : Issues and Conceptual Approach. Sofware Engineering Institute, Carnegie

]

Sur la figure 1, on peut voir le processus de réengineering et les différents niveaux d'analyse.

Ce sujet a déjà été couvert en détails par Philippe Dugerdil dans le rapport « Architecture-based software reengineering »[4].

1.2 Indentification des classes corrélées

Durant la première étape d'un projet de réengineering, il est nécessaire de retrouver l'architecture du système existant pour l'analyser et comprendre quelles sont les transformations qu'il faudra lui apporter.

Cela signifie qu'il faut retrouver pour une ou plusieurs fonctions logicielles du système, quels sont les composants fonctionnels[5] qui les implémentent. Sans ces informations, il serait impossible de faire le lien entre les fonctions logicielles décrites dans la documentation du système et leur implémentation concrète.

Pour ce faire, on enregistre l'exécution d'une fonction métier. On obtient une trace d'exécution. Puis on analyse les informations de cette trace pour trouver les classes qui sont corrélées[6] durant toute l'exécution, car ces classes sont l'implémentation la fonction exécutée.

Dans mon cas, les éléments que je cherche à associer sont des classes java. Cela n'a pas d'importance, car selon la plupart des langages le principe reste le même.

Le problème à résoudre est de proposer une technique basé sur l'analyse technique qui permet d'identifier les classes qui sont corrélées dans une trace d'exécution.

---

4 DUGERDIL Ph. - *Architecture-Based Software Reengineering*. Technical report, HEG Geneva, February 2006

5 Un ensemble de classes java

6 Ce sont les classes qui travaillent ensemble.

2. La problématique de l'analyse dynamique

Dans ce chapitre, j'explique tout d'abord ce qu'est que l'analyse dynamique et quelle est la différence avec l'analyse statique. J'explique également pourquoi est ce que l'on utilise l'analyse dynamique pour résoudre le problème d'identification des classes corrélées.

Pour bien comprendre ce sur quoi on va travailler, il faut bien comprendre ce qu'est une trace d'exécution, c'est pourquoi j'expliquerai également ce que c'est.

2.1 L'analyse dynamique

Dans cette partie, je décris l'analyse dynamique en général dans le contexte du développeur. Je décris aussi l'analyse statique pour montrer qu'elles sont les forces et les faiblesses de chacune de ces méthodes. J'explique aussi pourquoi ce travail se base sur une analyse dynamique et non statique.

2.1.1 Qu'est-ce qu'une analyse dynamique ou statique d'une application

L'analyse dynamique d'un programme consiste à analyser son exécution. On va récupérer des informations extraites de son exécution et les exploiter.

Au chapitre 1.2, j'expliquerai que l'on va identifier les classes corrélées à l'aide de la trace d'exécution. C'est de l'analyse dynamique.

Toutefois, il faut savoir que lors de l'analyse dynamique, d'autres informations que la trace d'exécution auraient pu être extraites, telles que :

- Surveiller les performances de l'application
- Vérifier l'allocation de mémoire et détecter les éventuelles fuites
- Observer la valeur de certaines variables
- Voir les objets instancier pendant l'exécution

Complémentaire à l'analyse dynamique, l'analyse statique d'un programme consiste à analyser uniquement le code source de ce dernier, afin d'obtenir des informations sur le comportement d'un programme sans vraiment l'exécuter, pour trouver une erreur de programmation ou de conception. On choisira l'analyse dynamique ou statique en fonction de ce que l'on cherche à faire et de la manière dont on souhaite s'y prendre.

Un exemple d'utilisation bien connu de l'analyse statique est le reverse engineering[7] du code source d'une application en diagramme de classe.

Dans l'optique où l'on étudie les interactions entre les éléments du code source qui composent l'application, l'analyse dynamique est intéressante, car on sait exactement quels sont les éléments qui ont été utilisés lors de l'exécution.

2.2 Les traces d'exécution

2.2.1 Qu'est-ce qu'une trace d'exécution

Une trace d'exécution représente la séquence des évènements (appels de fonctions, procédures ou méthodes) appelés pendant l'exécution d'un programme.

Cependant, un programme ne s'exécute jamais de la même manière. L'utilisateur du programme et les données que le programme traite peuvent modifier l'exécution du programme tout au long de son exécution.

Imaginons que l'utilisateur lance deux fois un même programme en utilisant des fonctionnalités complètement différentes à chaque exécution. Les deux traces résultantes seront probablement similaires au niveau de l'initialisation du programme, mais seront complètement différentes par la suite.

C'est est une structure assez complexe, mais surtout très longues. Elle peut être représentée sous forme d'un arbre d'arité [8] X, où X représente le nœud ayant le plus grand nombre de fils. Voici un petit exemple de ce à quoi peut ressembler une trace :

Figure 2

Description trace d'exécution

```
f1()
|---> f2()
|     |---> f5()
|     |---> f6()
|
|---> f3()
|     |---> f7()
|     |---> f8()
|     |     |---> f9()
|     |     |     |---> f10()
|
|---> fx()…
```

7 Transformation du code source en diagramme de classe.

8 Nombre de fils que possède un arbre.

L'exécution d'un programme va générer un grand nombre d'évènements. Pour un programme de type industriel (gros logiciels d'entreprise) on parle de plusieurs millions d'évènements pouvant être générés par l'exécution d'un seul « use-case ».

2.2.2 Comment récupérer une trace d'exécution

Comment récupérer les évènements d'un programme qui s'exécute ? L'une des techniques est l'instrumentation du code source[9].

On instrumente le code source du programme cible à l'aide d'un logiciel qu'on appelle simplement l'instrumenteur. L'instrumenteur va ajouter des lignes de code supplémentaires au début et à la fin de chaque fonction, méthode ou procédure du programme cible.

Ces lignes supplémentaires vont être exécutées en même temps que le programme instrumenté de façon à écrire dans un fichier des informations sur l'exécution du programme, en particulier les noms des fonctions appelées et les classes auxquelles elles appartiennent.

9 Hamou-Lhadj A., Lethbridge T.C. - A Survey of Trace Exploration Tools and Techniques. Proc of the Conference of the Centre for Advanced Studies on Collaborative Research CASCON 2004, October 5-7, 2004,Markham, Canada.

3. L'analyse technique en finances[10]

Ce chapitre, est une synthèse de ce que j'ai pu apprendre sur le domaine de l'analyse technique en finances. Je vais donner une définition de l'analyse technique et de ses principes fondamentaux. Ensuite, je présenterai les certains outils[11] incontournables qui sont utilisés par les analystes technique. Je donnerai une description simplifié pour chacun des outils, car il n'est pas nécessaire de rentrer dans les détails spécifiques à l'analyse technique et qui n'apporterai rien de plus pour la compréhension du reste de travail.

Ce chapitre présente les idées dont je me suis inspiré pour réaliser ce travail.

3.1 Qu'est-ce que l'analyse technique ?

Les praticiens de l'analyse technique lui ont donné de nombreuse définition. Elles sont toutes plus ou moins équivalentes. Toutefois, la définition de John J. Murphy est considérée comme la meilleure :

« L'analyse technique est l'étude de l'évolution d'un marché, principalement sur la base de graphiques, dans le but de prévoir les futures tendances ».

Dans cette définition, on peut développer trois éléments :

- « L'évolution d'un marché » : On peut appliquer l'analyse technique à tous les types de marchés[12].
- « Une étude graphique » : L'analyse technique a longtemps été considéré comme équivalente à une étude graphique. Ce qui avec le temps s'est révélé de plus en plus inapproprié. Il existe aujourd'hui de nombreuses applications informatiques qui traitent les données d'un graphique à l'aide d'outils statistiques. Il n'en reste pas moins que l'outil phare de l'analyse technique est véritablement le graphique et ses multiples interprétations possibles.

---

10 Ce chapitre a été rédigé à l'aide des ouvrages suivants :
BECHU, Thierry, BERTRAND, Eric, NEBENZAHL, Julien. *L'analyse technique : théories et méthodes*. Paris : Economica, 2008. (Finance).
BARON, François. Le chartisme : méthodes et stratégies pour gagner en bourse. [Paris] : Eyrolles, 2008. 490 p. (Analyse technique).

11 Ensemble de techniques, astuces de l'analyse technique

12 Marchés d'actions, taux d'intérêt, matières premières, etc.

- « Prévoir les tendances futures » : C'est le but ultime de l'analyse technique, prévoir l'évolution d'un marché en se basant sur l'histoire de ce dernier. C'est l'un des points sur lesquelles elle est souvent critiquée, car certains considèrent l'évolution d'un marché comme aléatoire.

3.2 Les principes de l'analyse technique

L'analyse technique ne se base pas sur les facteurs externes autres que l'évolution du marché (le cours, les volumes et la courbe), car elle considère que tous les facteurs qui interviennent sur le marché sont représentés par l'évolution de ce dernier.

L'analyse technique est basée sur trois principes fondamentaux:

1. Tout est pris en compte par le marché : « Pour l'analyste technique, tout ce qui peut influencer la valeur d'un bien est à tout moment reflété par le prix de ce bien sur le marché. Dès lors, il suffit de s'intéresser à l'évolution du prix, puisque celui-ci est la résultante de tout le reste».[13]

2. Les cours évoluent selon des tendances : « L'analyste technique ne croit pas beaucoup (pour ne pas dire pas du tout) à une évolution erratique des cours. Au contraire, il constate que :

- abstraction faite des fluctuations mineures, les cours évoluent en tendances ;
- ces tendances durent toujours un certain temps avant d'être modifiées.»[14]

3. L'histoire peut se répéter : A partir du moment où l'évolution des marchés est lié aux comportements de spéculateurs humaines, on peut voir dans histoire de ces marchés des comportements prévisibles qui se répètent. En analyse technique, le passé est la clé du futur.

13 BECHU, Thierry, BERTRAND, Eric, NEBENZAHL, Julien. L'analyse technique : théories et méthodes. Paris : Economica, 2008. (Finance).

14 BARON, François. Le chartisme : méthodes et stratégies pour gagner en bourse. [Paris] : Eyrolles, 2008. 490 p. (Analyse technique).

3.3 ***Les outils***

3.3.1 Les moyennes mobiles

Les fonctions de filtrage numérique sont très utilisées en analyse technique. Mais tout d'abord, quel est l'intérêt de ces fonctions ?

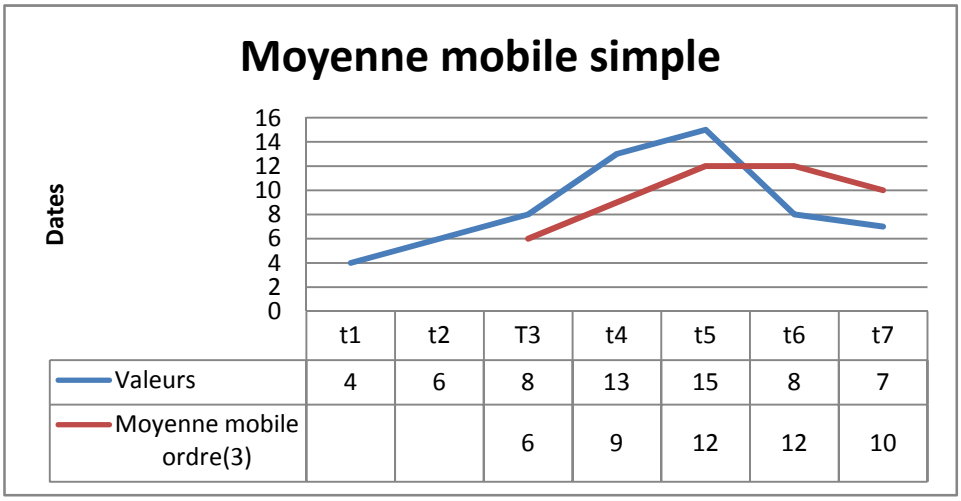
Les analystes techniques utilisent ces fonctions pour éliminer le bruit se manifestant sous forme de petites oscillations sur la courbe. Ainsi on filtre les valeurs de la courbe pour qu'elle reflète une tendance plus nette. Ca reviendrait à retirer les informations parasites de la courbe, pour qu'elle puisse exprimer les tendances du marché.

La moyenne mobile est le filtre le plus utilisé pas les analystes pour sa simplicité. Il y a trois principales sortes de moyennes mobiles : la moyenne mobile simple[15], la moyenne mobile pondérée et la moyenne mobile exponentielle. Dans la durée limitée de ce travail, je me suis concentré sur la technique de moyenne mobile.

La moyenne mobile simple comme son nom l'indique est la plus simple. Je vais utiliser la série suivante pour illustrer son fonctionnement.

Figure 3

Moyenne mobile simple



	t1	t2	T3	t4	t5	t6	t7
Valeurs	4	6	8	13	15	8	7
Moyenne mobile ordre(3)			6	9	12	12	10

15 « En bourse, la moyenne mobile est un indicateur qui montre la valeur d'un prix sur une certaine période. En d'autres termes, la moyenne mobile est une analyse mathématique de la valeur moyenne d'un prix sur une durée prédéterminée […] La moyenne mobile simple est la plus utilisée et la plus populaire des moyennes mobiles. La première raison est la relative facilité à laquelle les moyennes mobiles simples sont calculées. Une moyenne mobile simple est calculée en additionnant les valeurs d'un certain nombre de périodes et ensuite divisées par la somme du total des nombres de la valeur. » [tiré de : www.realtimeforex.fr]

Sur la figure 3, on peut voir une moyenne mobile simple. Elle est d'ordre 3, ce qui signifie que l'on calcule la valeur moyenne sur une période glissante de 3. C'est pour cela qu'on ne peut calculer la première valeur qu'en T3.

Le calcul pour T3 est le suivant : (4 + 6 + 8) / 3 = 6.

Et le calcul pour t4 est : (6 + 8 + 13) / 3 = 9.

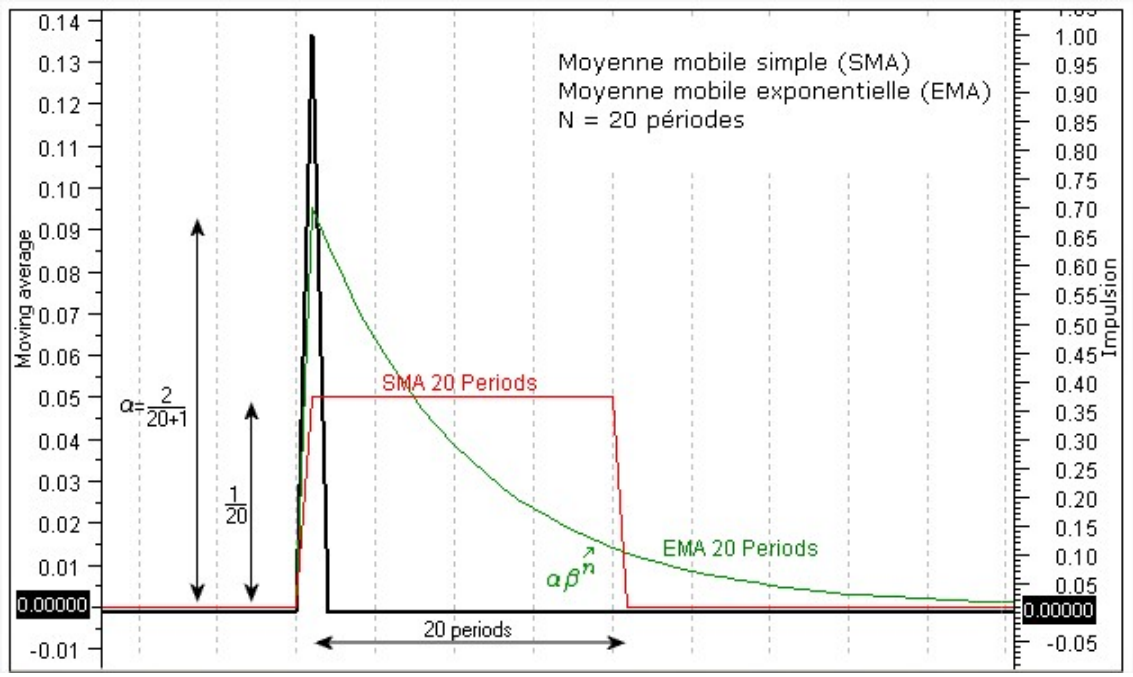
Si on observe la courbe de la moyenne mobile simple sur le graphique, on constate qu'elle est plus nuancée et surtout qu'elle réagit en retard par rapport à la courbe d'origine. C'est tout à fait normal, car les valeurs utilisées dans le calcul ont toutes la même pondération. Les autres moyennes mobile corrigent un peu ce retard.

La moyenne mobile pondérée, utilise des coéficients pour donner un poids plus important aux valeurs les plus récentes dans le calcul.

La moyenne mobile exponentielle, utilise pour le calcul une pondération sur les valeurs qui décroît exponentiellement. Elle suit encore mieux la courbe d'origine.

Figure 4

Moyenne mobile simple et exponentielle



[Tiré de : http://fr.wikipedia.org/wiki/Moyenne_mobile]

On voit sur la figure 4 que la moyenne mobile exponentielle croît beaucoup plus vite et qu'elle atténue beaucoup plus vite l'impact des valeurs les plus anciennes.

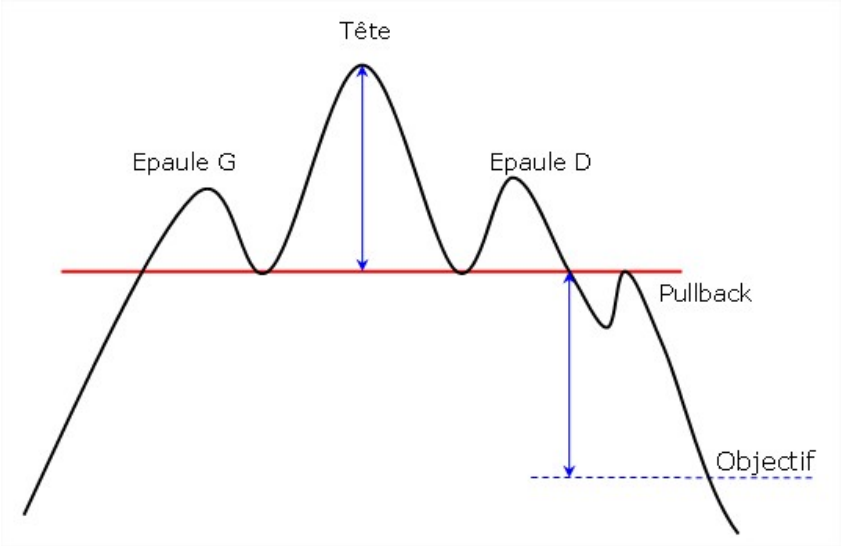
Pour toutes les moyennes mobiles, plus l'ordre est élevé plus il y aura de valeurs utilisées pour le calcule ce qui entrainera un lissage de plus en plus prononcé de la courbe.

3.3.2 Les figures chartistes

L'analyse technique traditionnelle considère qu'il est possible de prévoir l'évolution des cours grâce à des configurations graphiques[16] qui se répètent dans le temps. Et que ces configurations graphiques sont une manifestation du comportement des intervenants[17]. Leur apparition a une influence sur le comportement des intervenants et permet d'établir des prédictions sur la base de ces configurations graphiques et du volume des échanges à ce moment là.

Figure 5

Epaule Tête Epaule (ETE)



[Tiré de : http://www.trading-school.eu/glossaire-bourse/fiche-Epaule-Tete-Epaule-ETE--47]

Sur la figure 5, on peut voir un petit exemple de figure chartiste (configuration graphique) l'épaule-tête-épaule (ETE), caractérisée par trois sommets successifs. Le deuxième sommet est la tête, elle doit avoir une hauteur une fois et demi à deux fois cette des épaules. De plus, les épaules doivent être plus ou moins de même proportion.

16 Ce sont des figures chartistes qui se dessinent lors l'évolution de la valeur d'une action.

17 Les personnes qui spéculent sur le marché.

3.3.3 Support, résistance et canal

3.3.3.1 Qu'est qu'une tendance ?

Selon l'ouvrage sur le chartisme de François Baron[18], une tendance peut être considérée comme :

« Il s'agit d'un mouvement sur les cours assez régulier et soutenu dans un sens déterminé, sur une certaine période de temps. Il peut être soit à la hausse, on parlera de tendance haussière, soit à la baisse pour une tendance baissière. »

Le cours peut évoluer dans un intervalle réduit de prix pendant un certain temps. On verra alors apparaitre une tendance à l'aspect horizontal de la courbe, qu'on appelle dérive latérale.

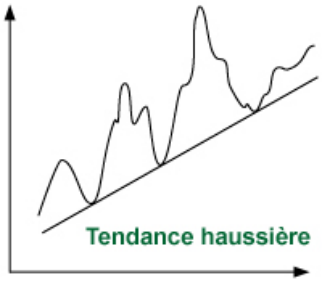
3.3.3.2 Lignes de tendance

Lorsque sur la courbe dessine une tendance haussière, on prendra les points les plus bas pour tracer une ligne de tendance sur laquelle l'action va rebondir un certain temps. Cette ligne de tendance s'appelle un support haussier (voir Figure 6).

Tant que le prix de l'action ne descend pas de façon persistante en dessous, la tendance devrait continuer.

Figure 6

Support haussier



[Tiré de : http://www.bnpparibas.net/banque/portail/particulier/Fiche?type=fiche&identifiant=NOT_Tendances_supports_et_resistances_20060403115328]

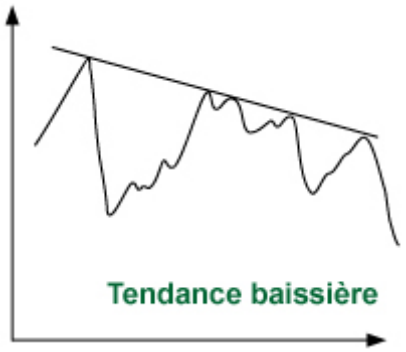
Dans le cas d'une tendance baissière, on peut appliquer le même principe en traçant une ligne de tendance avec cette fois-ci les points les plus hauts, c'est la ligne de

18 BARON, François. Le chartisme : méthodes et stratégies pour gagner en bourse. [Paris] : Eyrolles, 2008. 490 p. (Analyse technique).

tendance baissière. On devrait constater que la courbe rebondit sur cette ligne à l'image d'une boule sauteuse qui rebondirait sur le sol (voir figure 7).

Figure 7

Ligne de tendance baissière

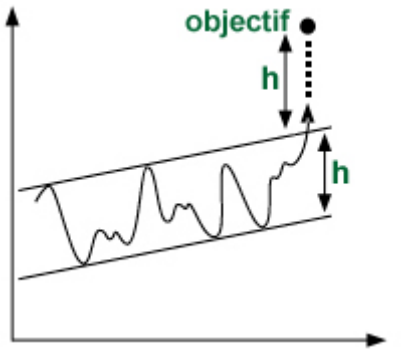


[Tiré de : http://www.bnpparibas.net/banque/portail/particulier/Fiche?type=fiche&identifiant=NOT_Tendances_supports_et_resistances_20060403115328]

On peut encore tracer une ligne parallèle à la ligne de tendance, pour former un canal haussier ou baissier. Pour que cette dernière ligne soit valide, il faut aussi qu'elle puisse s'aligner sur des pics de la courbe. On n'arrive pas toujours à définir un canal haussier qui soit parfait.

Figure 8

Canal



[Tiré de : http://www.bnpparibas.net/banque/portail/particulier/Fiche?type=fiche&identifiant=NOT_Tendances_supports_et_resistances_20060403115328]

3.3.3.3 Supports et résistances horizontales

« Les supports horizontaux sont des niveaux de cotation qui vont freiner la baisse de la valeur. Ils agissent comme des planchers plus ou moins solides, sur lesquels la valeur peut rebondir et reprendre une évolution haussière »

« Une résistance horizontale est un niveau de cours où la progression haussière de la valeur est freinée »[19].

On peut voir sur la figure 9 un exemple réel de mise en application des supports et résistances horizontales. Et qu'elles[20] sont plutôt bien respectées par l'évolution du prix de l'action.

Figure 9

Support et résistance horizontale

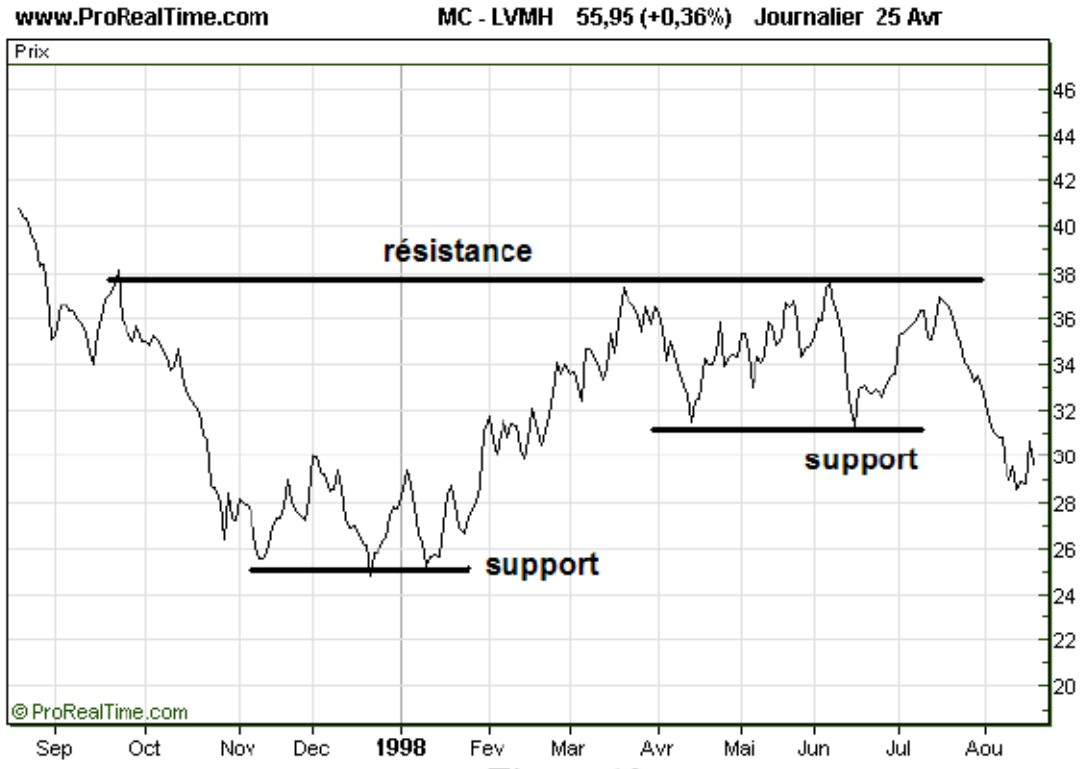


Figure 10

[Tiré de : http://www.daily-bourse.fr/NOTIONS-DE-SUPPORT-ET-DE-RESISTANCE-vtptc-1578.php]

19 BARON, François. Le chartisme : méthodes et stratégies pour gagner en bourse. [Paris] : Eyrolles, 2008. 490 p. (Analyse technique).

20 Les lignes

4. Application des idées de l'analyse technique à l'analyse dynamique de programme

M. Philippe Dugerdil m'a donné quelques indices sur ce que l'on pourrait essayer de faire pour appliquer des idées de l'analyse technique à l'analyse dynamique. Finalement, après réflexion, j'ai pu proposer une solution que je présenterai dans cette partie.

4.1 Technique de segmentation

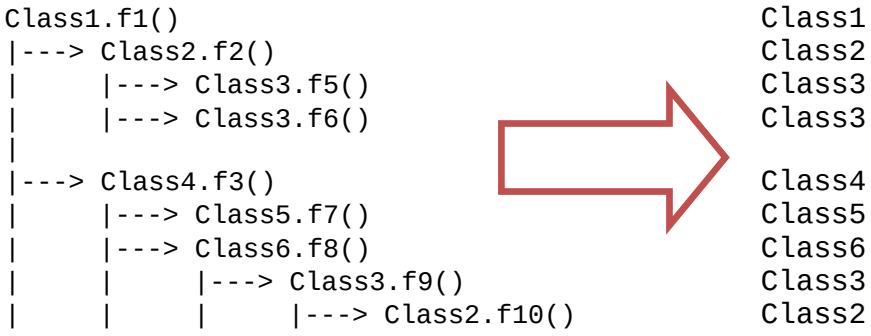
Avant d'aborder la segmentation de la trace, je vais décrire la constitution de la trace.

4.1.1 Description de la trace

Comme nous l'avons vu au Chapitre 2.2, la trace contient toutes fonctions (méthodes) qui ont été appelées par le programme dont on analyse l'exécution. Pour chaque appel de fonction, on connait la Class à laquelle appartient la fonction appelée (voir la partie gauche de la figure 10).

Figure 10

Description de l'information utilisée dans la trace



Dans notre analyse, nous nous intéresseront uniquement à la classe à laquelle appartient chacune des fonctions appelées, ainsi qu'à la séquence de ces appels. Ce sont ces informations sur lesquelles nous effectuerons notre analyse. Voici l'information de la trace que nous allons exploiter (voir la partie droite de la figure 10).

A présent, chaque fois je ferai référence à la trace, il faudra se la représenter comme une séquence de classes dans l'ordre établie par la trace.

4.1.2 Segmentation de la trace

La technique de segmentation consiste à découper toute la trace en segments contigus de taille égale. Chaque segment peut ainsi être analysé individuellement et apporter des informations localisées[21] dans la trace, sur les classes qu'il contient.

Figure 11

Représentation de la trace

	Trace																				
Séquence	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Classes	C0	C2	C4	C2	C0	C0	C2	C2	C6	C2	C2	C2	C2	C0	C3	C2	C0	C5	C1	C7	C7	C0

Sur la figure 11, je donne l'exemple d'une trace qui contient 22 appels et qui contient 7 classes (C0, C1, C2, C3, C4, C5, C6 et C7).

Maintenant, je vais diviser la trace en 7 segments contigus de taille égale à raison de 3 classes par segment. Le dernier segment contient une classe car il ne reste qu'une classe et le dernier segment contient minimum 3 classes dans ce cas (voir figure 12).

Figure 12

Segmentation de la trace

	Trace																				
Séquence	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Classes	C0	C2	C4	C2	C0	C0	C2	C2	C6	C2	C2	C2	C2	C0	C3	C2	C0	C5	C1	C7	C7	C0
Segments	1			2			3			4			5			6			7		

J'exprimerai toujours le nombre de segments en nombre de fois (f) le nombre d'éléments [22] (n) dans la trace, selon la formule suivante:

$$Ns = f \times n$$

n est alors considéré comme nombre d'éléments (classes) distincts dans la trace et f une valeur qui sera multiplié par n, pour obtenir le nombre de segment (Ns). n est donc déjà défini en fonction de la trace.

---

21 La position du segment dans la trace.

22 Les classes

On considère que le nombre de segments (Ns) doit être supérieur d'au moins 2 fois le nombre des éléments (n) qui composent la trace.

Donc Ns = 2 x n, où f = 2

La figure 12 est segmentée selon Ns = 1 x n, ou n = 7, ce qui donne bien Ns = 7.

Un problème récurrent à la technique de segmentation est de définir le Ns optimal pour obtenir le juste milieu entre précision (avec un Ns élevé) et quantité de calcul (qui augmente avec Ns). Une technique de réduction du bruit[23] et une technique de corrélation dynamique basée sur la segmentation de traces ont été présentées par Philippe Dugerdil[24].

Ce qui faut retenir de la segmentation de la trace, c'est qu'elle nous permettra d'analyser la présence des classes pour chaque segment.

Ces deux techniques ne sont pas utilisées dans ce travail, cependant ce sont des techniques de segmentation intéressantes. Je propose d'autres techniques basées sur la segmentation qui sont au cœur de ce travail.

4.1.3 Omniprésence temporelle

Pour analyser les traces d'exécution et faire face à l'énorme quantité d'information les composant, il faut réduire cette information.

L'objectif de la technique d'omniprésence temporelle est de repérer les éléments apparaissant dans un grand nombre de segments de la trace. Une fois repérés, il faut considérer qu'apparaissant trop fréquemment ils ne sont pas spécifiques au use case[25] analysé.

Un seuil doit être défini, au-delà duquel un élément sera considéré comme temporellement omniprésent. Une fois les éléments temporellement omniprésents repérés, on peut les exclure des éléments à analyser, pour réduire la quantité d'information de la trace.

---

23 Le bruit représente les classes qui n'ont pas d'intérêt.

24 IBM CAS SOFTWARE AND SYSTEMS ENGINEERING SYMPOSIUM (2007, Dublin). Using trace sampling techniques to identify dynamic clusters of classes : proceedings of the 2007 conference of the center for advanced studies on Collaborative research. Hamilton auditorium of Dublin, 24 octobre 2007. 9 p.

25 Ou la fonction logiciel

4.1.4 Corrélation binaire

La corrélation binaire a pour but de tester la corrélation des éléments[26] de la trace. Pour cela, elle utilise un vecteur d'occurrence par élément (ou classe) qui permet de savoir, quelles sont les segments dans lesquelles l'élément est présent (1) ou absent (0).

On obtient par exemple ces vecteurs pour les éléments Class1 et Class2 (voir figure 13).

Figure 13

Description des vecteurs d'occurrences

	Trace														
Segments	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Vecteur Occ Class1	0	0	1	1	0	0	0	0	1	0	1	1	1	1	0
Vecteur Occ Class2	0	1	0	1	0	1	0	0	1	1	1	1	1	0	1

Sur la figure 13, on voit que Class 1 est présent dans les segments 3, 4, 9, 11, 12, 13 et 14.

Une fois tous les vecteurs d'occurrence générés, à raison d'un vecteur par élément, on compare tous les vecteurs d'occurrence à l'aide d'une matrice de corrélation[27] pour déterminer l'indice de corrélation entre les éléments. L'indice obtenu nous permet de savoir si deux éléments sont présents dans les mêmes échantillons. Un fort couplage entre deux éléments se situe à partir de 80 % de correspondance et un couplage moyen plutôt entre 60 % et 79 %.

Cette technique de corrélation est performante. La recherche d'un élément dans un échantillon est terminée dès qu'on a trouvé la première occurrence de celui-ci.

Il est toutefois important de noter qu'il suffit d'une seule occurrence d'un élément dans un segment pour que cet élément soit considéré comme présent, et ceci, de manière équivalente à un autre élément qui apparaitrait 500 fois dans le même segment.

---

26 Les classes

27 Voir chapitre 4.1.5

4.1.5 Matrice de corrélation

En effectuant les comparaisons définies dans la matrice de corrélation, on compare tous les éléments entre eux. Toutes les corrélations entre les éléments dans ce travail sont effectuées de cette manière.

Sur la figure 14, on voit un exemple de matrice de corrélation qui définies tous les comparaisons pour 7 éléments.

Figure 14

Matrice de corrélation pour 7 éléments

	C1	C2	C3	C4	C5	C6	C7
C1	-	1	1	1	1	1	1
C2	-	-	1	1	1	1	1
C3	-	-	-	1	1	1	1
C4	-	-	-	-	1	1	1
C5	-	-	-	-	-	1	1
C6	-	-	-	-	-	-	1
C7	-	-	-	-	-	-	-

4.1.6 Comptage des fréquences

Ce sont les fréquences d'occurrences[28] des classes dans chaque segment de la trace qui vont me servir de base pour appliquer les outils de l'analyse technique. Je m'inspire directement de la technique de corrélation binaire définie plus haut, sauf que je ne me limite pas à déterminer la présence ou l'absence des classes dans un segment, mais plutôt la fréquence d'occurrence de chacune des classes pour chacun des segments comme illustré dans la figure 15.

28 On peut dire les fréquences d'apparitions

Figure 15

Comptage des fréquences d'occurrences

	Trace																				
Séquence	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Classes	C0	C2	C4	C2	C0	C0	C2	C2	C6	C2	C2	C2	C2	C0	C3	C2	C0	C5	C1	C7	C7	C0
Segments	1			2			3			4			5			6			7		
C0	1			2			0			0			1			1			1		
C1	0			0			0			0			0			0			1		
C2	1			1			2			3			1			1			0		
C3	0			0			0			0			1			0			0		
C4	1			0			0			0			0			0			0		
C5	0			0			0			0			0			1			0		
C6	0			0			1			0			0			0			0		
C7	0			0			0			0			0			0			2		

La différence est conséquente, car à présent une classe qui apparait une seule fois dans un segment n'est plus du tout équivalent à une classe qui apparait 500 fois dans ce segment. Cette donnée importante nous permettra d'être beaucoup plus précis dans les tentatives de corrélation. De plus, la précision de nos futures analyses sera bien moins affectée par le nombre de segment (Ns), grâce à la fréquence d'occurrence. Elle permet, de rendre compte dans quelle mesure une classe est présente dans un segment.

On peut se représenter nos données comme des séries de valeurs où chaque série représente les fréquences d'occurrences d'une classe (voir figure 15).

On pourrait essayer d'exploiter directement toutes ces fréquences et les afficher sur un graphique où les valeurs en abscisse représenteraient les segments dans la trace et l'ordonnée représenterait les fréquences d'occurrences.

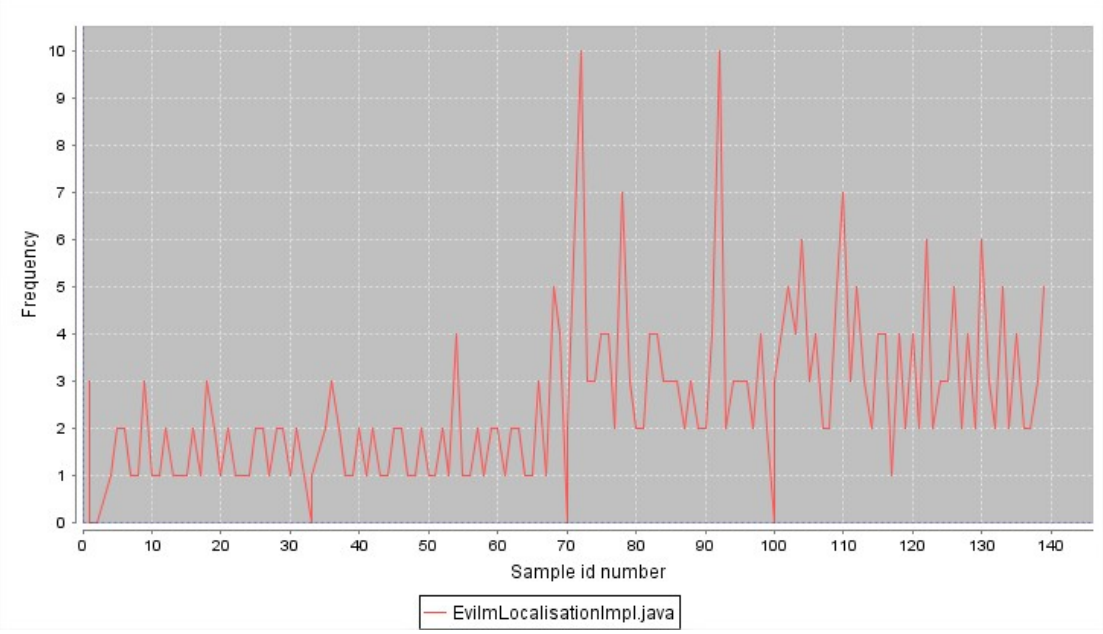
4.1.7 Exemples de segmentation

Si l'on considère une trace d'environ 600 000 évènements, de 139 éléments distincts. En affichant, ces fréquences d'occurrences par segment sur un graphique :

Avec un Ns égal à 1n, soit 139 segments - ce qui fait environ 4300 appels [29]par segment - on distingue correctement la courbe et on constate alors que la fréquence oscille beaucoup d'un segment à un autre (voir figure 16).

Figure 16

Fréquences d'occurrences d'un élément avec Ns = 1n

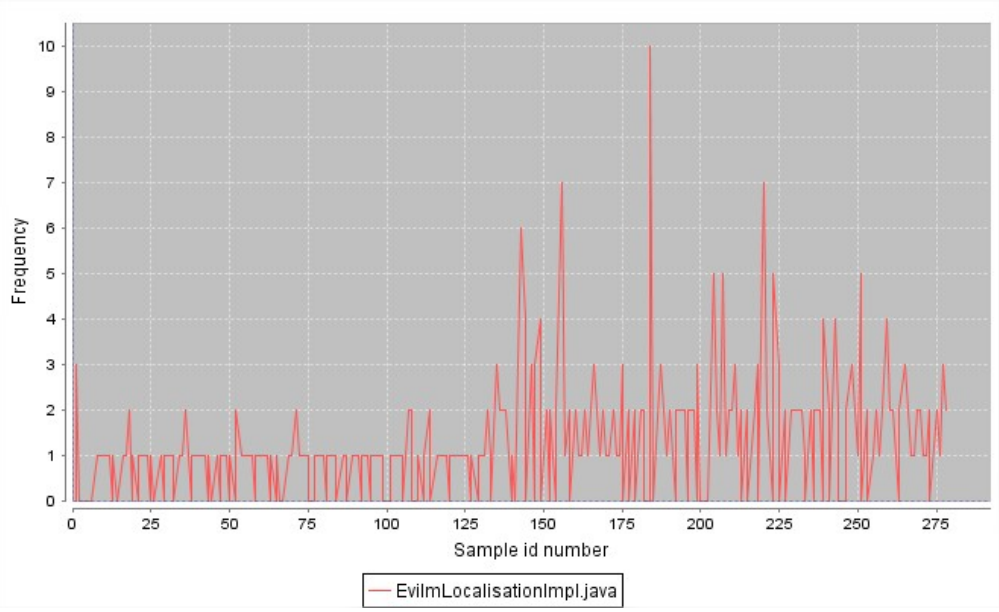


Avec un Ns égal à 2n, soit 278 segments - ce qui fait environ 2150 appels par segment - on distingue correctement la courbe, mais les oscillations sont plus nombreuses étant donné que l'on a un Ns plus grand et donc plus d'échantillons (voir figure 17).

29 Appels aux classes dont une fonction à été appelé. Voir Chapitre 2.2

Figure 17

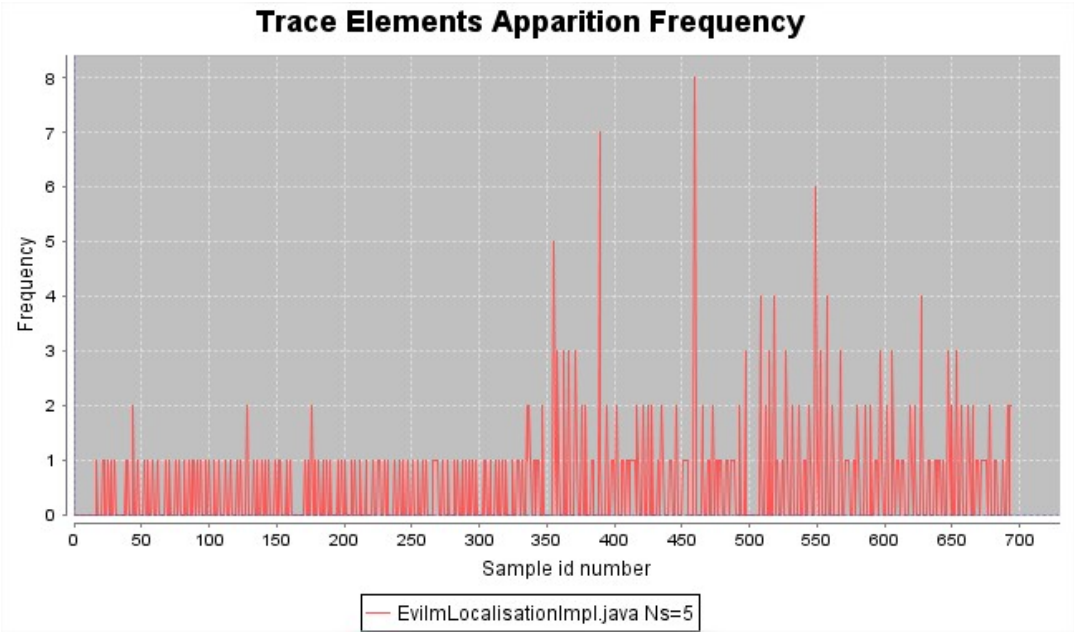
Fréquences d’occurrences d’un élément avec NS = 2n



Avec un Ns égal à 5n, soit 695 segments - ce qui fait environ 863 appels par segment - on ne distingue plus grand-chose. A l’œil, il serait impossible de comparer les fréquences d’occurrences de deux classes de la trace et d’affirmer avec certitude qu’elles sont similaires (voir figure 18).

Figure 18

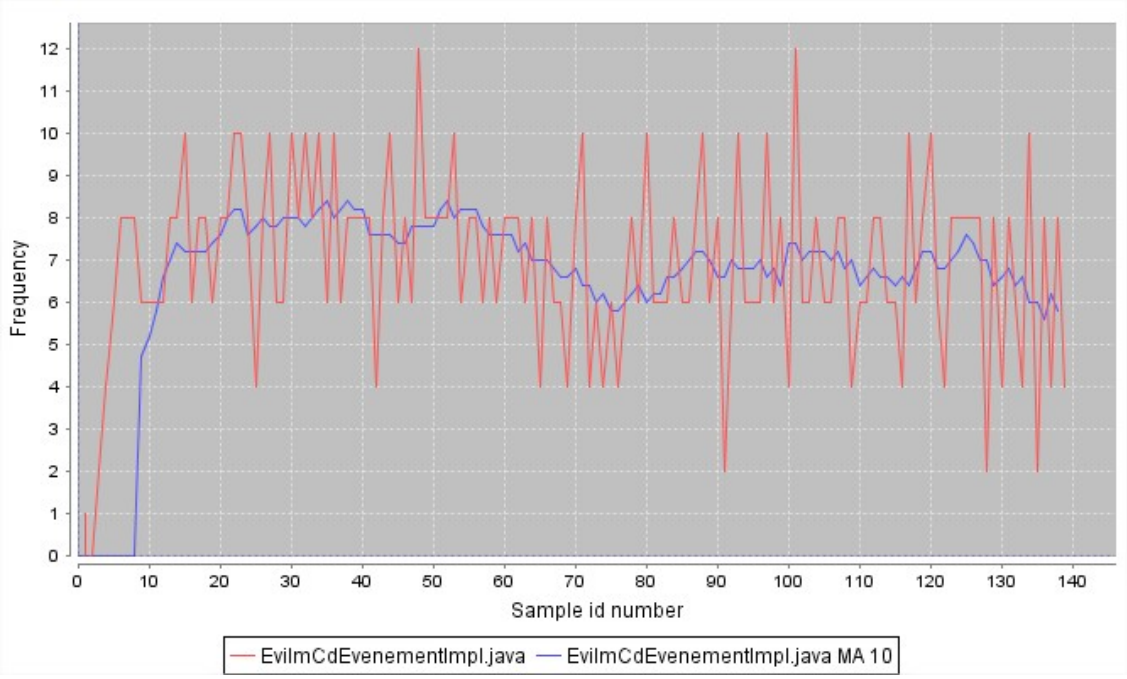
Fréquences d’occurrences d’un élément avec NS = 5n



Plus le Ns est élevé plus le phénomène s'accentue. Pour ce problème, comme pour les courbes des indices boursiers, on peut appliquer une moyenne mobile simple (voir figure 19).

Figure 19

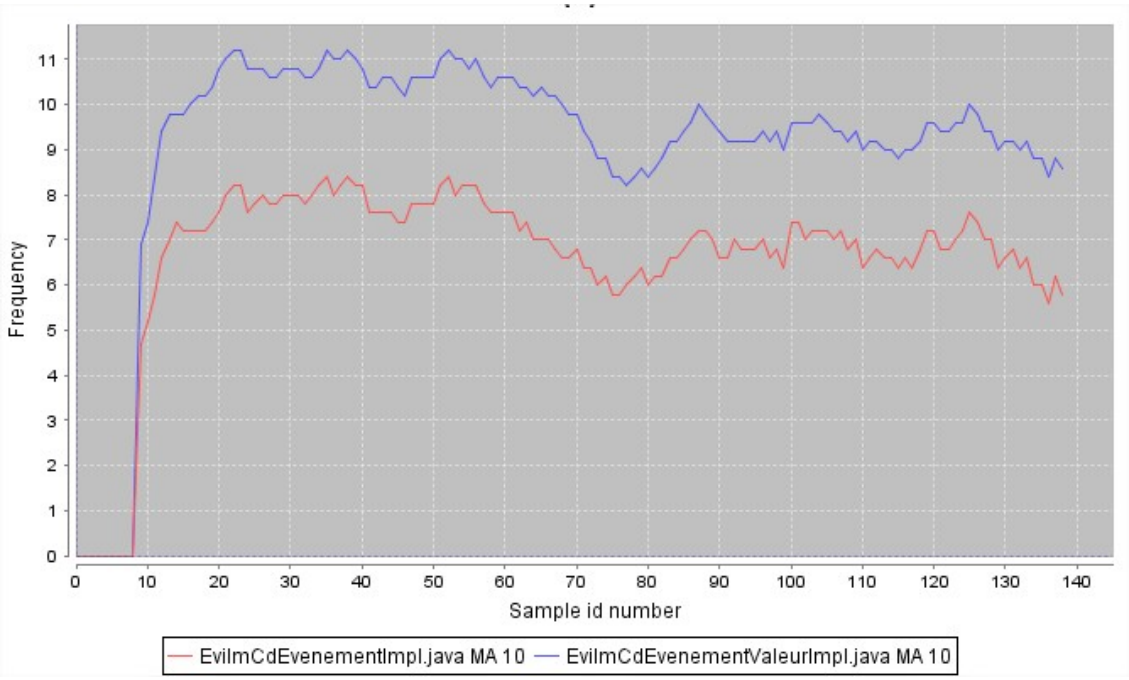
Fréquences d'occurrences et moyenne mobile simple (10) avec Ns = 1n



Grâce à la moyenne mobile simple (voir figure 19), on peut facilement adoucir la courbe et repérer les grandes tendances.

Figure 20

Moyennes mobiles simples (10) et Ns = 1n

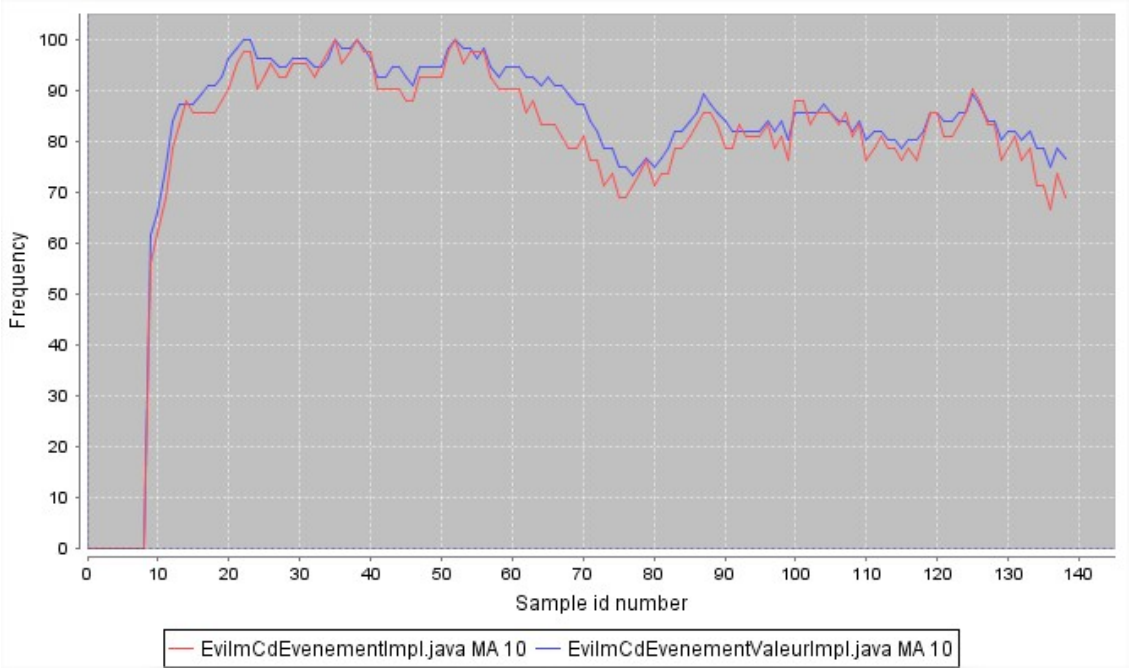


On constate visuellement qu'il est beaucoup plus facile de comparer l'implication de deux classes dans une trace si l'analyse fréquentielle est filtrée par les moyennes mobiles.

Pourtant, il subsiste encore un détail avant l'application d'une technique de corrélation sur les classes de la trace. J'ai constaté que certaines courbes oscillaient à peu près de la même manière et aux mêmes emplacements de la trace. Par contre, elles étaient complètement décalées verticalement (voir figure 20). Certaines classes n'ont pas les mêmes fréquences d'occurrences par segment, mais on constate visuellement sur le graphique qu'en les exprimant de manière relative, elles sont comparable.

Figure 21

Moyennes mobiles simples (10) relatives et Ns = 1n



Exprimer de manière relative, la fréquence d'occurrence la plus élevée d'une classe devient 100% et la fréquence la plus faible devient 0%. Les fréquences relatives nous permettent de ne prendre en compte que le comportement de la courbe : l'augmentation et la diminution de la fréquence d'apparition par rapport à la valeur maximale et minimale. On constate que les courbes ont un comportement similaire l'une par rapport à l'autre même si elles ne sont pas parfaitement superposées.

En l'état actuel, mes données peuvent être soumises à différentes techniques de corrélation. La partie suivante m'a permis d'utiliser des concepts issus de l'analyse technique et de les exploiter pour proposer des solutions plus ou moins originales.

4.2 Idées possibles d'application de l'analyse technique

Premièrement, je vais expliquer la notion d'empreinte dans ce travail qui est essentielle pour comprendre les techniques inspirées de l'analyse technique que j'ai envisagées.

4.2.1 La notion d'empreinte

L'empreinte d'une classe est réalisée à partir des observations faites sur les fréquences d'occurrences d'une et seule classe. Son contenu dépend complètement du type observation que l'on enregistre. Elle sera capable de révéler la position (segment) où une certaine observation a été faite, dans le but d'être comparé à

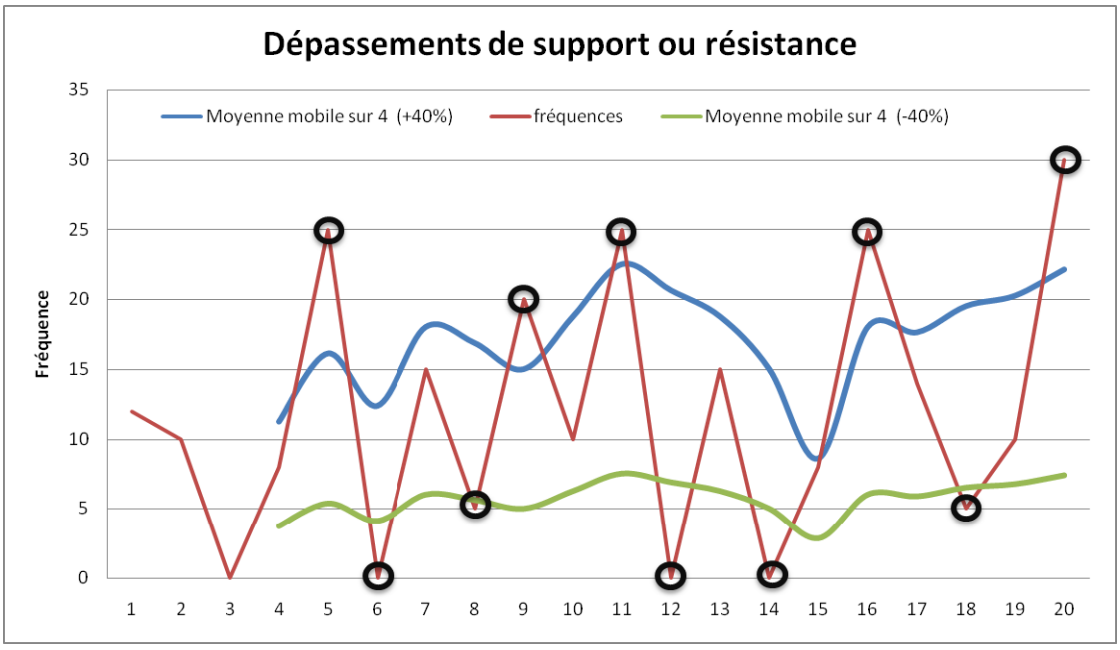
d'autres empreintes du même type et de déterminer rapidement, sur une ensemble réduit d'observations, si elles présente des caractéristiques similaires.

4.2.2 Points de dépassements de support ou résistance

Cette technique que j'ai nommée « Points de dépassements de support ou résistance » a pour objectif de créer une empreinte pour chaque élément de la trace et de comparer toutes les empreintes entre elles (voir figure 22).

Figure 22

Dépassement support ou résistance



Pour réaliser cette empreinte, on a besoin de la courbe des fréquences[30] d'une classe, puis on génère deux moyennes mobiles simples modifiées. L'une dont les valeurs en ordonné sont augmentées, par exemple de 40%[31], et l'autre dont les valeurs en ordonné sont réduites de 40 %. En affichant les trois courbes, on obtient par exemple le graphique ci-dessus (figure 22). La courbe jaune représente le support, tandis que la courbe bleu foncé représente la résistance. L'ensemble des segments pour lesquels la

30 On peut aussi utiliser des fréquences filtrées si les oscillations de la courbe sont trop fortes.

31 Le pourcentage d'augmentation et de réduction est défini de manière que les deux moyennes mobiles soient au niveau des valeurs extrêmes de la courbe des fréquences. Sinon on risque de prendre une empreinte sur tous les segments de la classe, ce qui ne sert à rien. Cette valeur doit donc être adaptée en fonction du cas.

fréquence brise le support ou la résistance forme l'empreinte de la classe représentée par la courbe des fréquences.

Évidemment, la moyenne mobile simple est d'un ordre qui devra rester être le même pour réaliser toutes les empreintes à comparer. Le pourcentage appliqué en augmentation ou en réduction de la moyenne mobile devra lui aussi être constant. Et la comparaison des empreintes consistera tester si les dépassements du support et de la résistance ont lieu sur les mêmes segments.

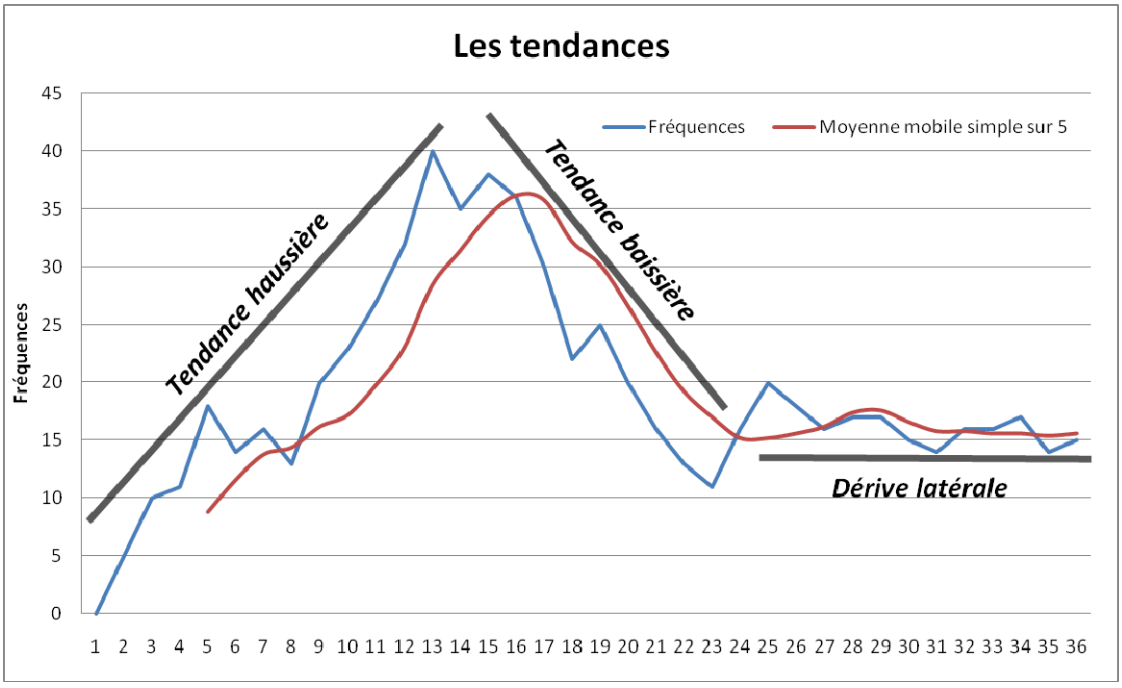
4.2.3 Analyse des tendances

Comme la technique « Pattern de dépassements de support ou résistance », la technique d'analyse des tendances à pour but de créer une empreinte pour chaque classe de la trace, puis de comparer[32] les classes les unes aux autres à l'aide de leur empreinte pour révéler celles qui pourraient être corrélés.

Cependant, l'empreinte utilisée par cette technique est différente. Pour générer l'empreinte d'une classe, on a besoin des fréquences d'occurrences filtré avec une moyenne mobile.

Figure 23

Tendances d'une courbe



32 Selon la matrice de corrélation (Chapitre 4.1.5)

On peut observer trois types de tendances sur un graphique, comme:

La tendance haussière, qui peut être identifiée par le fait que sur un ensemble de segments (axe des X) parcouru dans l'ordre croissant, la fréquence est de plus en plus élevée.

La tendance baissière, qui peut être identifiée par le fait que sur un ensemble segments (axe des X) parcouru dans l'ordre croissant, la fréquence est de plus en plus faible.

La dérive latérale, qui peut être identifiée par le fait que sur un ensemble segments (axe des X) parcouru dans l'ordre croissant, la fréquence reste constante ou évolue dans un intervalle réduit.

Il est difficile de déterminer les tendances d'une courbe, car on peut repérer des macro-tendances, elles-mêmes constituées de micro-tendances. Tout dépend de l'échelle à laquelle on effectue l'analyse. Et en appliquant une moyenne mobile, on va lisser la courbe et faire disparaitre les micro-tendances. La puissance de cet effet dépend complètement du nombre de segments (Ns), et de l'ordre[33] de la moyenne mobile. Plus l'ordre sera grand plus la courbe sera lissée.

Comme on peut le voir sur la figure 23 ci-dessus, une moyenne mobile simple d'ordre 5 suffit à lisser la courbe et faire disparaitre les micro-tendances. Ainsi, il est plus facile de détecter les macro-tendances.

L'empreinte contient la position[34] de chaque tendance sur courbe. Une fois que toutes les empreintes ont été construites, on peut finalement lancer le processus de corrélation pour trouver tous les éléments qui ont une empreinte similaire. La comparaison des empreintes consiste à tester si les mêmes tendances ont lieu sur la même série de segments contigus.

Deux courbes qui ont la même empreinte nous assurent les tendances des deux courbes seront similaires. Par contre, rien ne nous garantit que les deux courbes soient superposées. En effet, une tendance haussière localisée aux mêmes endroits sur deux courbes n'implique pas que ces deux tendances évoluent d'une même pente. Une tendance pouvant être plus forte que l'autre.

---

33 Nombre de valeurs avec lesquelles la moyenne mobile est calculée.

34 Une série de segments contigus

On peut perfectionner la technique pour la rendre plus efficace, en incluant dans l'empreinte des informations supplémentaires sur la pente de la courbe pour les tendances haussières et baissières.

4.2.4 Empreinte sur les patterns de courbe

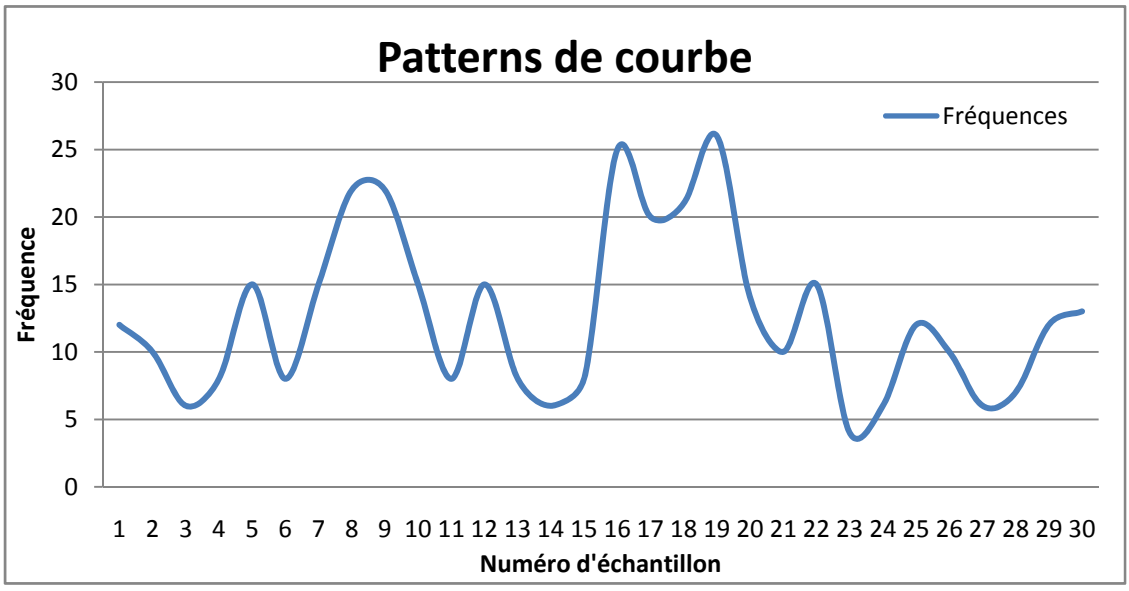
Cette technique est celle qui se rapproche le plus à l'analyse technique. C'est sur cette piste que j'ai commencé mes recherches sur ce travail.

Le principe est d'analyser une courbe pour y trouver des patterns de courbe[35]. L'empreinte de cette courbe consistera en une succession de patterns. Pour cette technique comme pour la précédente, le point sensible est de définir l'échelle à laquelle on va effectuer la recherche. On pourrait trouver des macro-patterns[36], dans lesquels on pourrait aussi trouver des micro-patterns[37].

Analysons la courbe de la figure 24 à la recherche de patterns de courbe. Elle est essentiellement composée de patterns de courbe que nous allons visuellement mettre en évidence.

Figure 24

Patterns de courbe



35 On les appels aussi les figures chartistes.

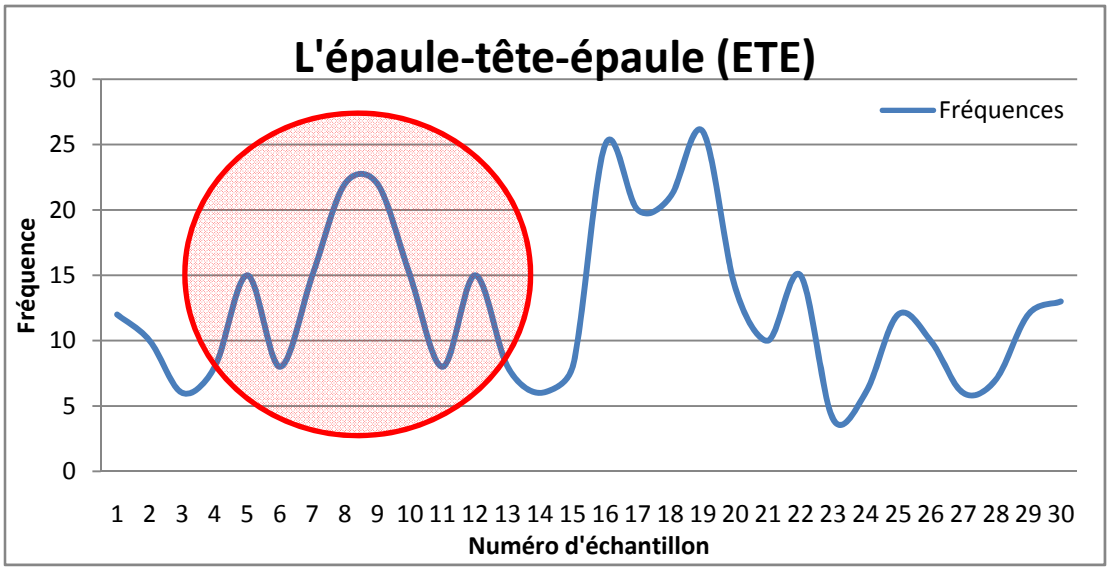
36 Ce sont des figures chartistes qui sont visibles à grande échelle.

37 Ce sont des figures chartistes qui sont visibles à petite échelle.

Le premier pattern est une ETE[38], on le voit très nettement sur la figure 25.

Figure 25

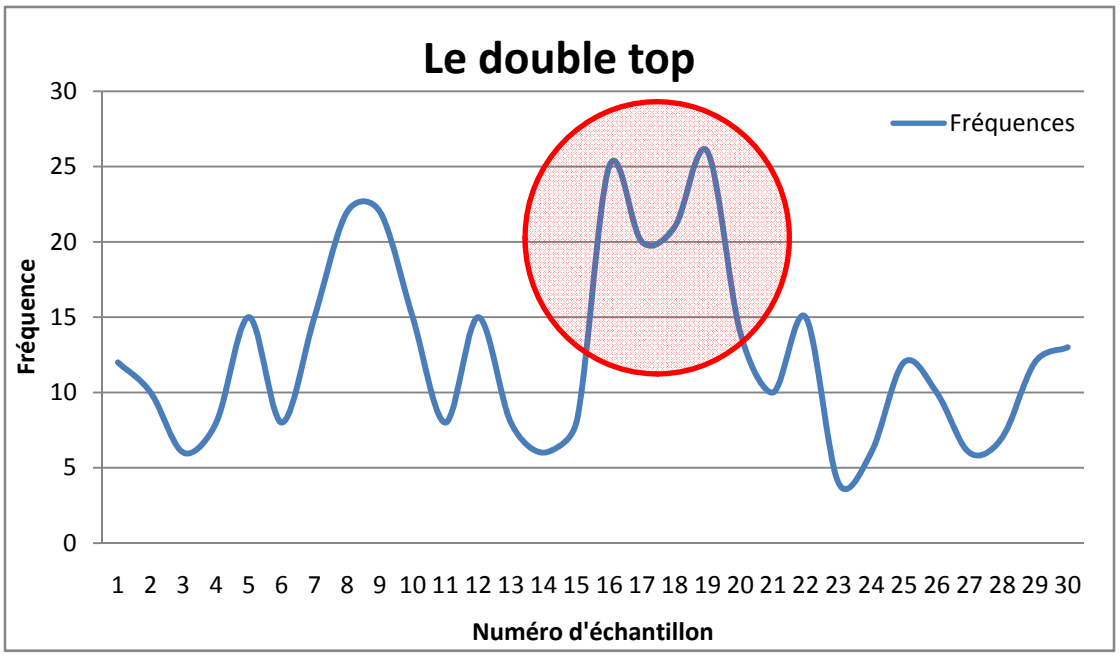
Pattern ETE



Le deuxième pattern est un double top. Il est mis en évidence sur la figure 26.

Figure 26

Pattern double top

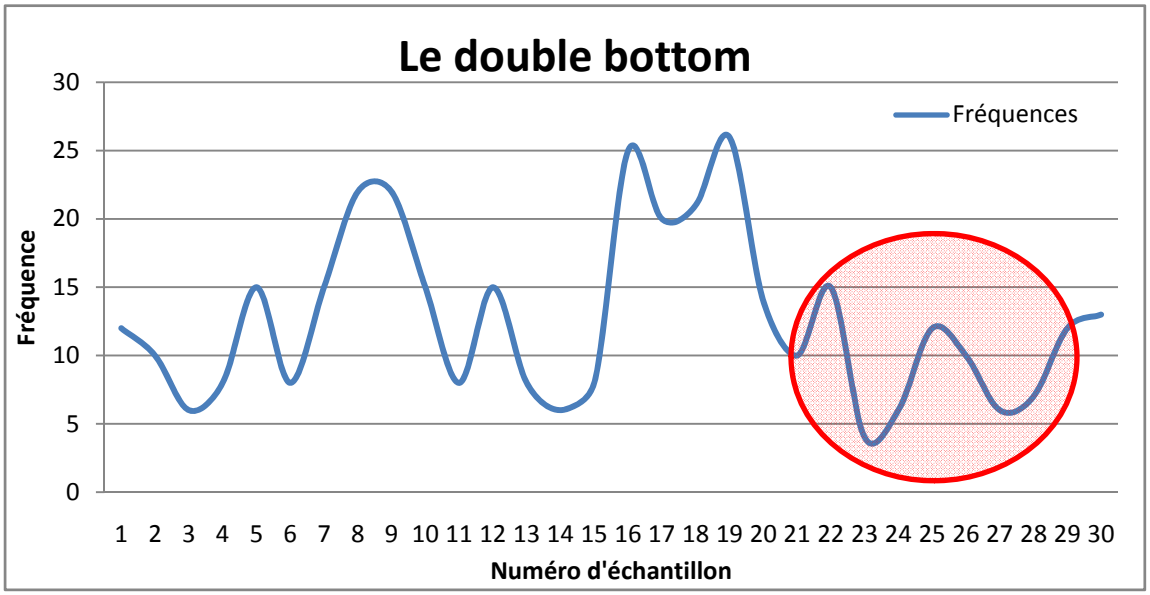


38 Pattern épaule-tête-épaule (ETE)

Le troisième et dernier pattern est un double bottom, comme on peut le voir sur la figure 27.

Figure 27

Pattern double bottom



Cette technique est intéressante, car réaliser une empreinte sur la base de pattern de courbe, nous permet de prendre un peu plus en compte la forme de la courbe.

4.2.5 Technique de la différence entre courbes relatives et filtrées par les moyennes mobiles

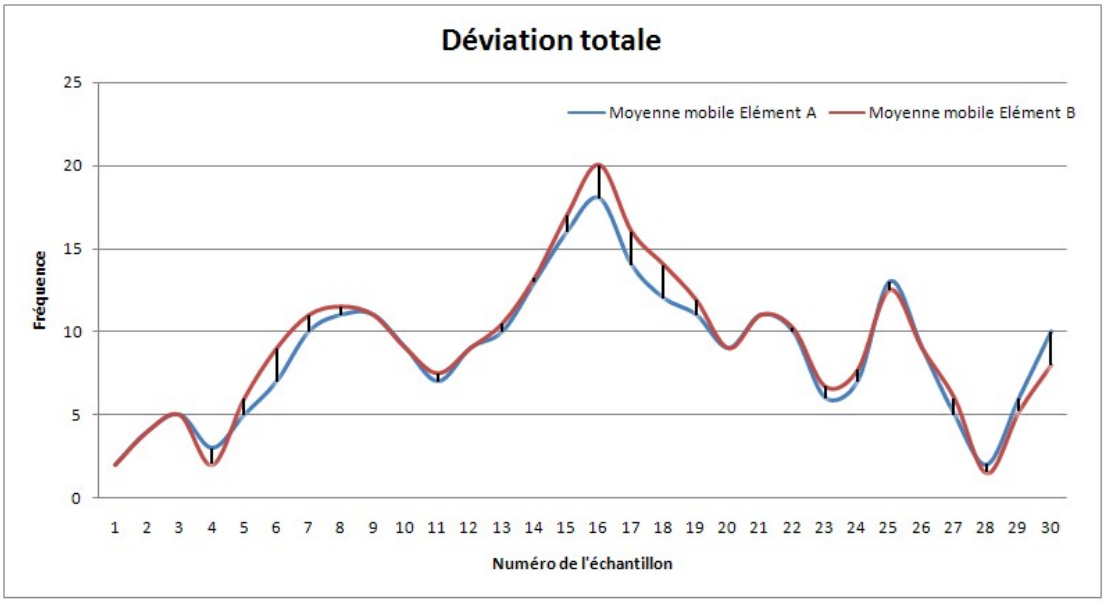
Son principe de fonctionnement est simple et ne nécessite aucune empreinte préalable à la comparaison de deux courbes. Pour chaque segment, on récupère la fréquence de chacune des deux courbes. On soustrait les deux fréquences, puis on prend la valeur absolue du résultat de la soustraction. Et on répète cette opération pour tous les échantillons. Pour finir, on additionne la déviation de tous les échantillons. On obtient le score de déviation[39].

Sur la figure 28, on voit deux courbes et la zone hachurée représente la déviation que l'on calcule.

[39] C'est le total de la déviation entre deux courbes.

Figure 28

Déviation de deux courbes



Plus le score de déviation est faible plus la corrélation entre les deux courbes est forte et inversement. Le score de déviation minimum est zéro, car les deux courbes sont strictement identiques, elles ne dévient pas l'une par rapport à l'autre.

Pour que le score de déviation reflète le plus le comportement des deux courbes comparées, on doit absolument utiliser les fréquences relatives de chaque courbe.

Un petit problème subsiste, car si les deux courbes n'ont pratiquement que des valeurs nulles, le score de déviation sera injustement faible. Pour corriger ce problème, il suffit de diviser le score de déviation par le nombre de valeurs différent de zéro sur l'ensemble des deux courbes.

Ainsi, plus il y aura de valeurs non nulles, plus le diviseur du score de déviation sera grand et plus le résultat sera petit par rapport aux courbes ayant beaucoup de valeurs nulles. En résumé, plus les courbes auront de valeurs nulles, plus le score de déviation sera sensible à la plus petite déviation.

La représentation mathématique de cette technique est la suivante :

e = le nombre de segments total

$$x_k = \underbrace{\frac{\sum_{\substack{V=k-N \\ V \geq N}}^{k} frequ_{C_i}(V)}{e - N}}_{\text{Moyenne mobile d'ordre N}}$$

Soit $S_{C_i} = x_1 \dots x_m$: valeurs filtrées pour C_i
$S_{C_j} = y_1 \dots y_m$: valeurs filtrées pour C_j

On a :

$$S_{C_i,C_j} = \left\{ (x_k, y_k) \mid x_k \in S_{C_i} \wedge y_k \in S_{C_j} \wedge (x_k \neq 0 \wedge y_k \neq 0) \right\}$$

$$N_f = \left| S_{C_{i,C_j}} \right|$$

$$correlation(C_i, C_j) = \frac{\sum_{k=1}^{m} |x_k - y_k|}{N_f}$$

5. Choix d'une technique

Un des objectifs de ce travaille est mettre en œuvre une application pour tenter de corrélés les classes d'une trace d'exécution. Je vais critiquer les techniques de corrélation dynamique proposées au chapitre précédent et présenter la technique que j'ai choisi d'implémenter.

5.1 Analyse de la pertinence des techniques proposées

5.1.1 Points de dépassements de support ou résistance

Cette technique pourrait permettre de trouver des éléments qui sont corrélés comme des éléments qui ne le sont pas du tout, mais qui partagent la même empreinte. Il n'a aucun lien concret entre une empreinte et la forme de la courbe. On peut donc avoir deux courbes qui n'ont pas la même forme et pourtant la même empreinte. De plus, en fonction du Ns choisi, on peut avoir plus ou moins de points qui constituent l'empreinte. Le choix du facteur utilisé pour calculer le support et la résistance dépendra donc deux choses : le Ns et le comportement de la courbe (si la courbe oscille beaucoup ou pas).

Cette technique ne semble pas très solide déjà au niveau théorique. Plus le Ns est élevé plus la fréquence de la courbe oscille, je ne crois pas que cette technique fonctionnerait.

Selon moi, cette technique serait difficile à implémenter dans une application qui l'utiliserait de manière efficiente, c'est-à-dire chercher le facteur (pour calculer le support et la résistance) idéal pour générer les empreintes de tous les éléments de la trace, ceci affectant aussi les performances.

Faute de temps, je n'ai pas pu faire de tests.

5.1.2 Analyse des tendances

La mise en pratique de cette technique nécessite cependant de mettre en œuvres une analyse, puis une corrélation coûteuse en ressources, sans offrir une mesure permettant d'exprimer la qualité de la corrélation. Si, sur 9000 comparaisons d'empreinte, 500 sont positives. Il reste à savoir qu'elles sont celles qui ont qualité de corrélation suffisante.

Mais quel sera l'impact sur les performances, sachant que chaque opération supplémentaire rallonge le temps de traitement global.

5.1.3 Empreinte sur les patterns de courbe

La mise en œuvre de cette technique de corrélation dynamique serait complexe. Un algorithme de recherche de pattern rapide et peu couteux en ressources me semble difficile à réaliser.

5.1.4 Technique de la différence entre courbes relatives et filtrées par les moyennes mobiles

C'est la dernière technique de corrélation dynamique, que je critique. Et contrairement aux techniques précédentes, je l'ai implémenté et testé dans l'application que j'ai réalisée. Car c'est la technique qui semble être la plus prometteuse du lot.

Cette technique est simple et performance. On peut comparer les courbes directement sans faire d'empreintes. Il faudra tout même transformer les fréquences filtrées [40] ou non en valeur relatives avant de calculer le score de déviation entre deux courbes. Cette technique fonctionne très bien d'après les tests que j'ai effectués.

Elle donne une mesure de la qualité de la corrélation entre tous les couples d'éléments comparés, on peut donc les classer de la corrélation la plus forte à la plus faible. Cette mesure est très importante, car elle nous permet de définir un seuil au-dessus duquel la qualité de la corrélation est considérée comme insuffisante. Donc si deux éléments sont corrélés le score de déviation de leur courbe doit être inférieur au seuil défini.

Je montre le test de cette technique au chapitre 6 « Expérimentation et analyse de sensibilité par rapport au nombre de segments».

[40] La moyenne mobile simple est un type de filtre et il y en a d'autres.

6. Expérimentation et analyse

Dans ce chapitre, je présente l'application que j'ai développée. Cette application implémente la technique de la différence entre courbes relatives et filtrées par les moyennes mobiles. Pour essayer de déterminer dans une trace quelles sont les classes corrélées et confirmer visuellement à l'aide des graphiques si effectivement les courbes des fréquences d'occurrences se ressemblent.

Je ferai quelques tests pour montrer l'effet de la moyenne mobile simple sur les résultats de la corrélation. Pour finir, je vais aussi montrer quelle est la sensibilité des résultats de corrélation par rapport au nombre de segments définis.

6.1 Description de l'application

6.1.1 Objectifs de l'outil de mise en œuvre

Les objectifs de l'outil se divisent en deux groupes :

- Les objectifs inhérents à la consultation des éléments de la trace. Cela signifie que l'utilisateur a liberté de consulter les informations qu'il désire.
- Les objectifs nécessaires à la corrélation de tous les éléments de la trace. Lorsque l'utilisateur a besoin de paramétrer l'outil et lancer le processus de corrélation sur tous les éléments de la trace.

Commençons par le premier groupe d'objectifs, celui qui est lié à la consultation des données. Les objectifs sont les suivants :

1. Afficher la liste de tous les éléments distincts qui apparaissent dans la trace d'exécution.
2. Afficher dans un graphique les fréquences d'apparition d'un ou plusieurs des éléments de la trace, avec le Ns désiré.
3. Afficher dans un graphique les fréquences filtrées d'un ou plusieurs éléments de la trace, en ayant la liberté de choisir un des filtres disponible (implémenté),

et si nécessaire choisir le nombre de valeurs[41] que le filtre utilisera (échantillons / Axe des X).

4. Afficher dans un graphique la courbe de corrélation entre deux éléments de la trace, avec la technique de corrélation dynamique de son choix.
5. Être capable d'afficher dans un seul graphique les trois objectifs précédent.

Le deuxième groupe qui concerne le processus de corrélation sur toute la trace, contient les objectifs suivants :

1. Permettre de paramétrer :
    a. La valeur de Ns.
    b. Le filtre à appliquer sur les fréquences et le nombre de valeur utilisé par le filtre.
    c. La technique de corrélation dynamique que l'outil utilisera pour effectuer la corrélation entre les éléments de la trace.
2. Afficher les résultats de la corrélation dans une liste.
3. Si possible trier les résultats de la corrélation par paires d'éléments dont la corrélation est la plus forte vers ceux dont la corrélation est plus faible.
4. Afficher le résultat de la corrélation de deux classes depuis la liste des résultats.

6.1.2 Présentation générale de l'interface graphique

Je vais présenter l'interface graphique de l'application. Au chapitre 6.1, j'ai montré les deux groupes d'objectifs que je me suis fixés. On va voir maintenant que l'interface de l'application se compose de deux parties.

La première partie est constituée d'onglet d'analyse dans lequel on lance le processus de corrélation sur l'ensemble de la trace sélectionnée avec les paramètres désirés.

La deuxième partie de l'interface est un onglet de visualisation des classes. On peut donc choisir la ou les classes que l'on désire afficher sur un graphique et les paramètres désirées. Les deux parties sont indépendantes.

41 Utilisé par les filtres de type moyen mobile.

6.1.3 Onglet d'analyse des corrélations

Cette partie de l'interface graphique est la plus importante de l'application et aussi la plus simple. On peut la voir sur la figure 29.

Pour lancer une corrélation, il faut :

- Sélectionner une trace dans la liste déroulante « Trace »
- Choisir le nombre de segment Ns[42]
- La moyenne mobile simple est le seul filtre, il est automatiquement appliqué.
- Choisir l'ordre de la moyenne mobile simple « Range ».
- On n'a pas d'influence sur la technique de corrélation, il y en a une seule.

On peut ensuite cliquer sur « Start Correlation » et le résultat de toutes les corrélations apparaîtra dans la zone de texte « Result », dans l'ordre du meilleur résultat vers le plus mauvais.

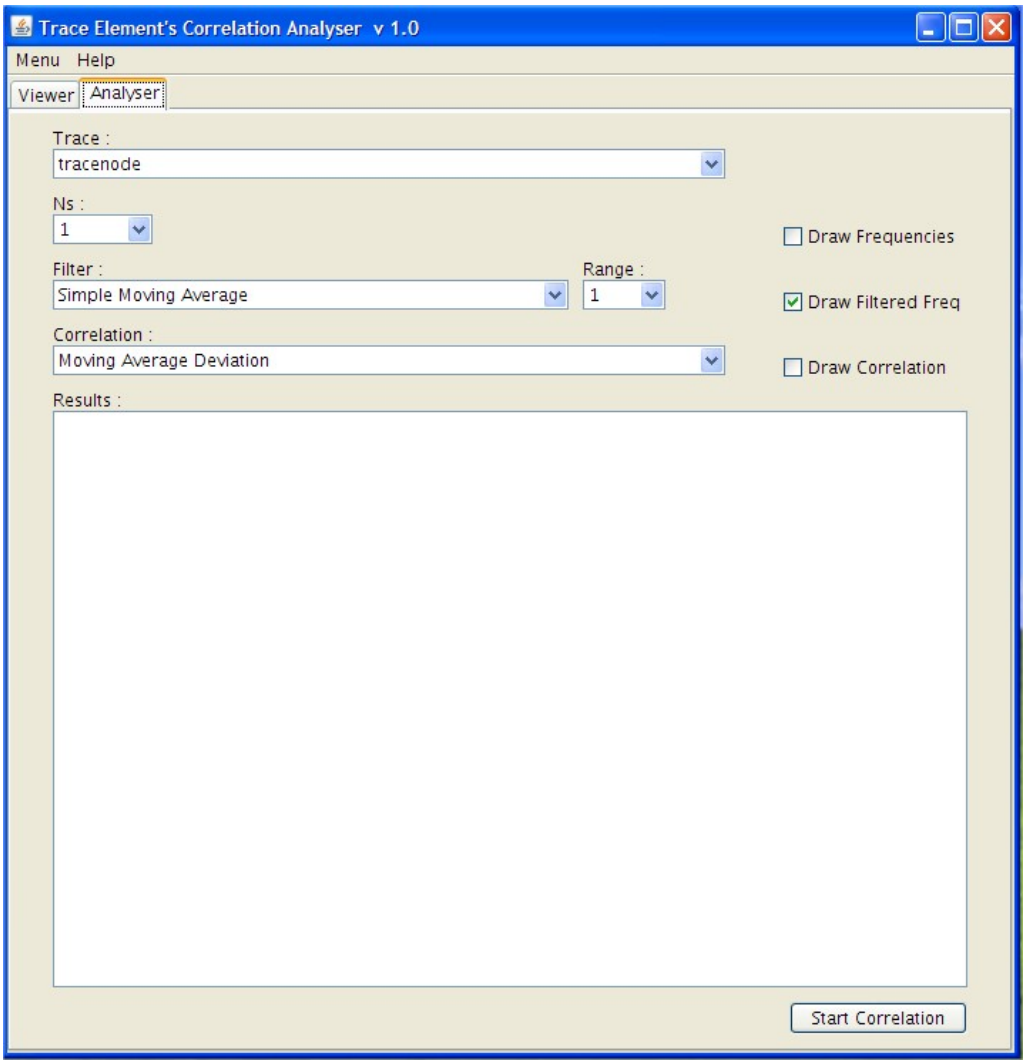
On peut choisir d'afficher les courbes pour n'importe lesquelles des classes corrélées, en double-cliquant sur un des résultats dans « Result ».

Les trois cases à cocher « Draw Frequencies », « Draw Filtered Freq » et « Draw Correlation » permettent respectivement d'afficher les courbes des fréquences d'occurrences, les fréquences d'occurrences filtrées par une moyenne mobile et la courbe de déviation des moyennes mobiles.

42 IMPORTANT : Ns est exprimé ici sous forme d'une valeur x. Ns = xn, où n est le nombre de classes dans la trace. On n'a pas d'influence sur n. Il reste donc à définir x et le calcul du Ns se fait automatiquement dans l'application.

Figure 29

Panneau d'analyse des corrélations



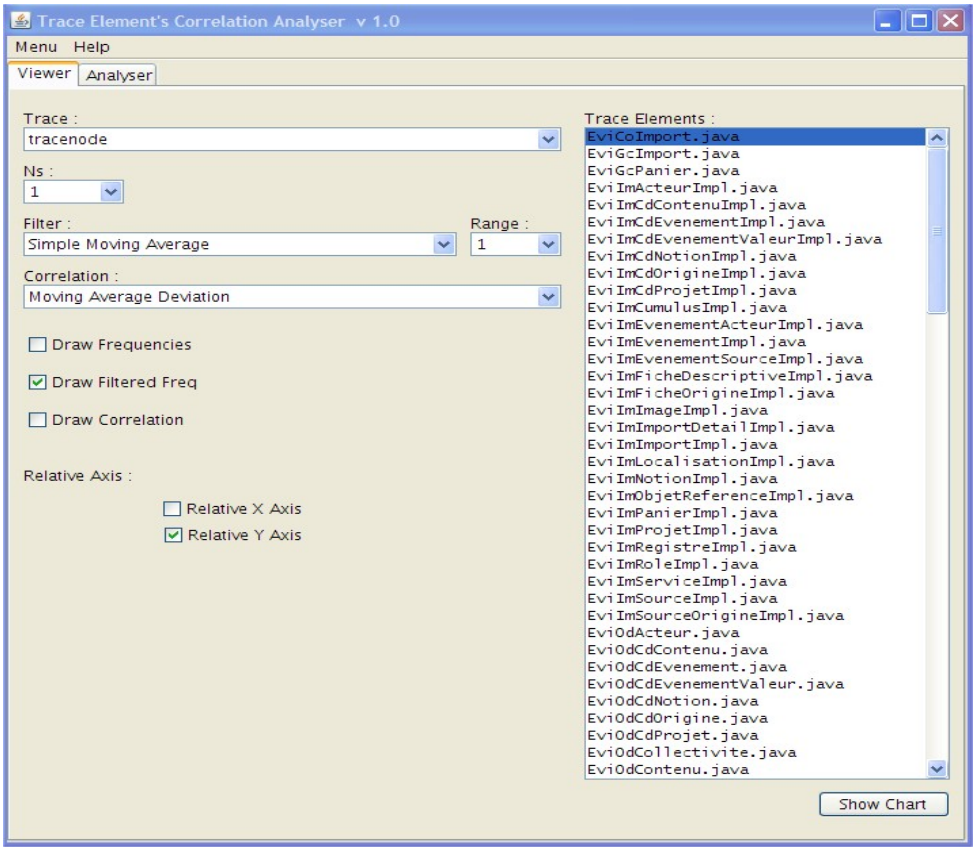
6.1.4 Onglet de visualisation des classes

Le fonctionnement de cette onglet est le même que le précédant pour la sélection des paramètres. Dans la zone de liste « Trace Elements », on peut sélectionner une ou plusieurs classes et toutes les afficher sur un même graphique, toujours en fonction des paramètres définis. On appuie sur « Show Chart » pour afficher le graphique.

Les cases à cocher « Relative X Axis » et « Relative Y Axis » servent à définir si les valeurs sur l'axe qu'elles désignent doivent être exprimées de manière relative. (voir figure 30)

Figure 30

Panneau de visualisation graphique des classes



6.2 Expérimentation de l'outil

Toutes les expérimentations qui vont suivre ont été réalisées sur une trace d'une taille de 600 000 appels. Il faut environ 4 minutes de traitement pour analyser les classes corrélées sur cette trace, avec Ns = 5n. Le temps de traitement peut être réduit significativement (voir chapitre 7.3). Il est important de préciser que l'application dont provient notre trace est très bien architecturée, pour facilité la recherche des classes corrélées.

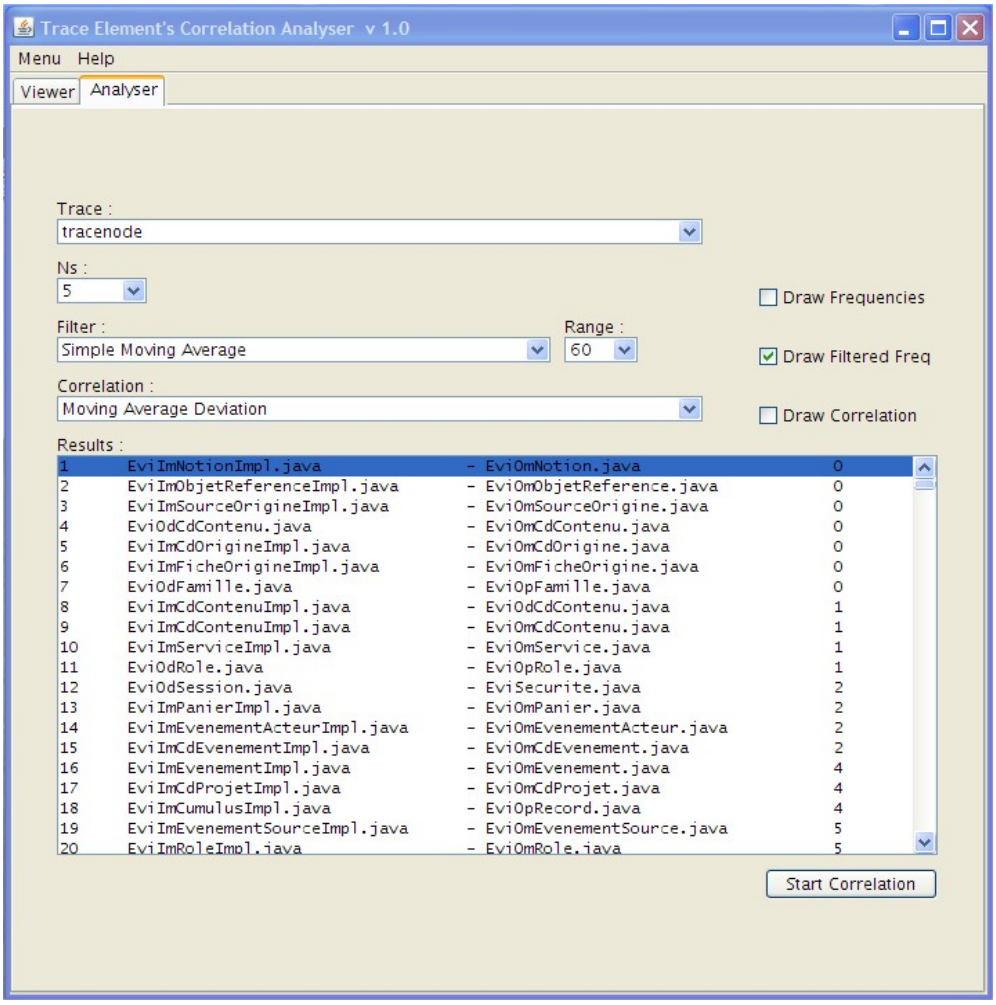
6.2.1 Résultats avec technique de corrélation implémentée

Je vais lancer le processus de corrélation des classes de la trace, avec les paramètres suivant :

- Segmentation de la trace avec Ns = 5n, filtrage des fréquences d'occurrences de toutes les classes avec une moyenne mobile simple d'ordre 60 (voir figure 39).

Figure 39

Corrélation des classes de la trace



Les résultats sont affichés dans l'ordre croissant du score de déviation entre les paires de classe comparées.

On constate pour toutes les paires de classes corrélées que les courbes sont parfaitement superposées pour les 200 meilleures corrélations. On peut en déduire que ces classes travaillent ensemble. En analysant visuellement les résultats de corrélations, on constate une ressemblance graphique[43] des paires de classes comparées pour les 1000 meilleures corrélations.

Voici un extrait plus étendu des résultats de la figure 39 (voir figure 40).

43 De la courbe

Figure 40

Extrait des corrélations des classes de la trace

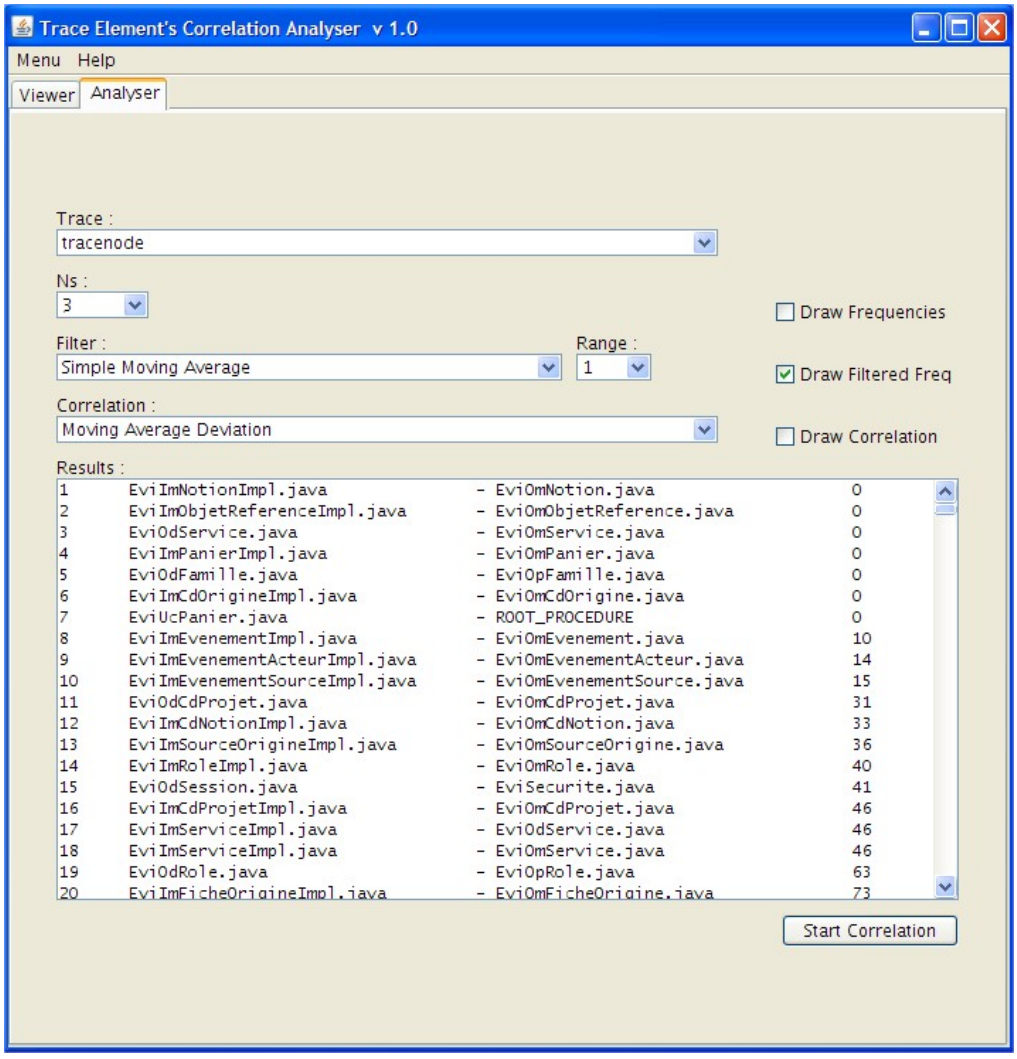
Ns 5n filtrage (Moyenne Mobile Simple) d'ordre 60			
R	C1	C2	RE
1	EviImNotionImpl.java	EviOmNotion.java	0
2	EviImObjetReferenceImpl.java	EviOmObjetReference.java	0
3	EviImSourceOrigineImpl.java	EviOmSourceOrigine.java	0
4	EviOdCdContenu.java	EviOmCdContenu.java	0
5	EviImCdOrigineImpl.java	EviOmCdOrigine.java	0
6	EviImFicheOrigineImpl.java	EviOmFicheOrigine.java	0
7	EviOdFamille.java	EviOpFamille.java	0
8	EviImCdContenuImpl.java	EviOdCdContenu.java	1
9	EviImCdContenuImpl.java	EviOmCdContenu.java	1
10	EviImServiceImpl.java	EviOmService.java	1
11	EviOdRole.java	EviOpRole.java	1
12	EviOdSession.java	EviSecurite.java	2
13	EviImPanierImpl.java	EviOmPanier.java	2
14	EviImEvenementActeurImpl.java	EviOmEvenementActeur.java	2
15	EviImCdEvenementImpl.java	EviOmCdEvenement.java	2
16	EviImEvenementImpl.java	EviOmEvenement.java	4
17	EviImCdProjetImpl.java	EviOmCdProjet.java	4
18	EviImCumulusImpl.java	EviOpRecord.java	4
19	EviImEvenementSourceImpl.java	EviOmEvenementSource.java	5
20	EviImRoleImpl.java	EviOmRole.java	5
21	EviOdCdOrigine.java	EviOmCdOrigine.java	5
22	EviImCdOrigineImpl.java	EviOdCdOrigine.java	5
23	EviOdCdOrigine.java	EviOmLocalisation.java	5
24	EviOdService.java	EviOmService.java	6
25	EviOdCdProjet.java	EviOmCdProjet.java	6
26	EviOmRole.java	EviOpRole.java	7
27	EviImCdNotionImpl.java	EviOmCdNotion.java	7
28	EviOdImage.java	EviOmImage.java	7
29	EviOdCdEvenement.java	EviOmCdEvenement.java	8
30	EviOmActeur.java	EviOpActeur.java	8
31	EviImServiceImpl.java	EviOdService.java	8
32	EviOdRole.java	EviOmRole.java	8
33	EviImCdEvenementImpl.java	EviOdCdEvenement.java	8
34	EviOdImage.java	EviOpRecord.java	8
35	EviOdCdProjet.java	EviOpProjetImage.java	9
36	EviImProjetImpl.java	EviOpProjet.java	9
37	EviImCumulusImpl.java	EviOdImage.java	10
38	EviOdFicheNotion.java	EviOdNotion.java	10
39	EviImCdOrigineImpl.java	EviOmLocalisation.java	11
40	EviOmCdOrigine.java	EviOmLocalisation.java	11
41	EviImCdProjetImpl.java	EviOdCdProjet.java	11
42	EviImCumulusImpl.java	EviOmImage.java	12
43	EviImRoleImpl.java	EviOpRole.java	12
44	EviImCdEvenementValeurImpl.java	EviOmCdEvenementValeur.java	12
45	EviOmImage.java	EviOpRecord.java	13
46	EviImImageImpl.java	EviOmImage.java	13
47	EviImRoleImpl.java	EviOdRole.java	14
48	EviOdSupport.java	EviOpContenu.java	14
49	EviOmCdProjet.java	EviOpProjetImage.java	16
50	EviOdCdNotion.java	EviOdNotion.java	16

6.2.2 Analyse de la sensibilité de la corrélation par rapport au nombre d'ordre de la moyenne mobile

Les résultats que je vais présenter montrent que filtrer les fréquences d'occurrence des classes, nous permet d'obtenir des meilleurs score de déviation[44] pour les classes corrélés, grâce à l'effet lissage de la courbe.

Figure 31

Résultat de corrélation avec Ns 3n et ordre 1



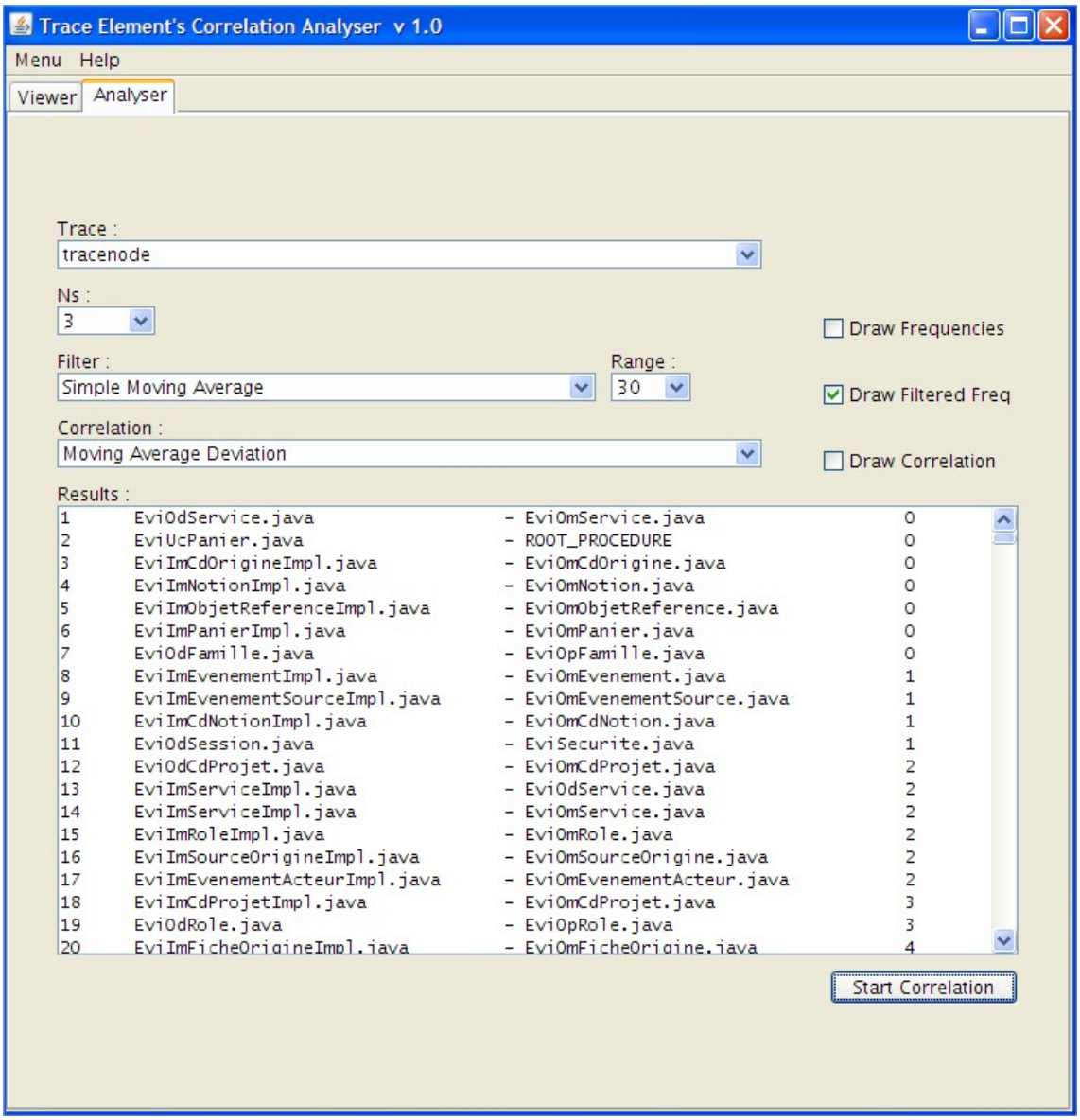
J'ai effectué une analyse de la trace avec Ns = 3n sans filtrage et j'ai obtenu les résultats de la figure 31. A présent, je refais la même analyse avec un filtrage d'ordre 30. J'obtiens les résultats de la figure 32.

44 Voir Chapitre 4.2.5

Si on compare les deux résultats on constate, pour la figure 33, que le score de déviation est plus faible et que les valeurs sont plus étalées.

Figure 32

Résultat de corrélation avec Ns 3n et ordre 30



Regardons maintenant sur le graphique l'impact de la moyenne mobile simple sur la courbe des fréquences deux classes parfaitement corrélées, sans filtrage (voir figure 33) et avec filtrage par une moyenne mobile simple d'ordre 30 (voir figure 34). L'effet est radical. On perçoit très nettement les tendances des deux courbes qui sont parfaitement superposées, alors que sur la courbe sans filtrage on ne voyait pas grand chose.

Figure 33

Courbes des fréquences d'occurrences de deux classes avec Ns 3n ordre 1

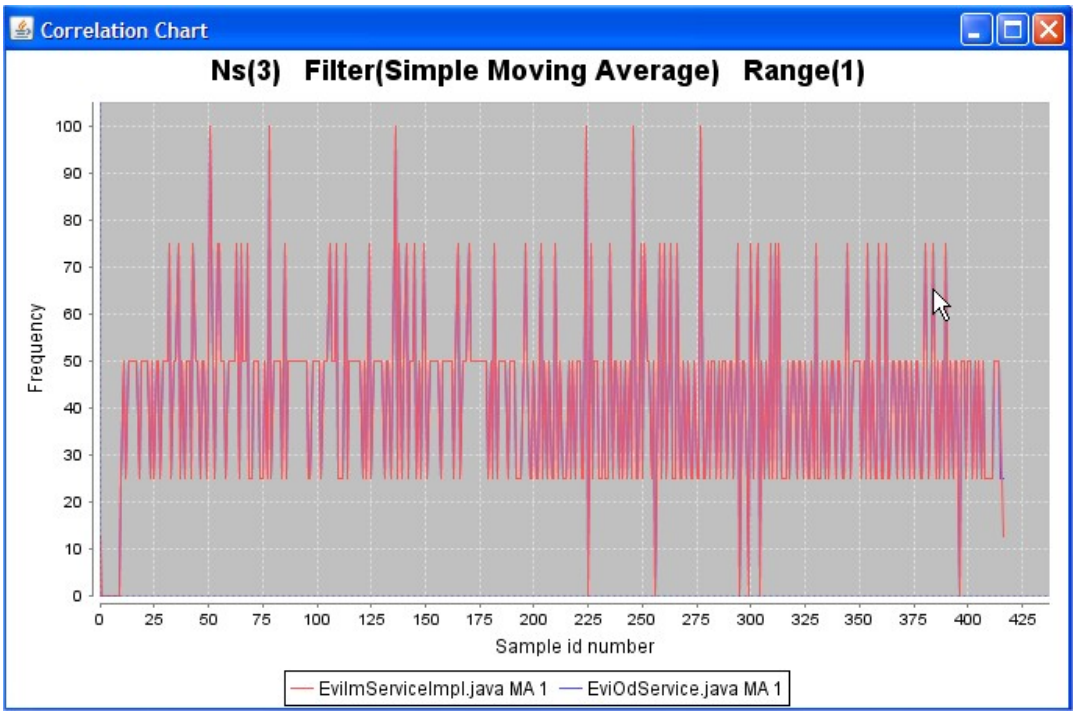
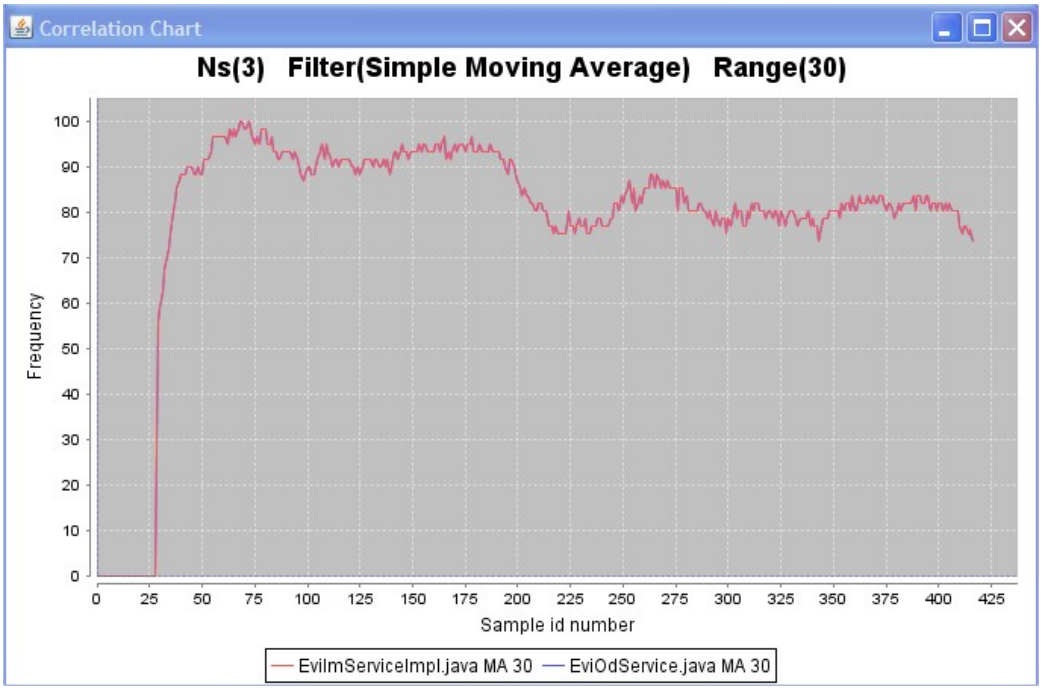


Figure 34

Courbes des fréquences d'occurrences de deux classes avec Ns 3n ordre 30



6.2.3 Analyse de la sensibilité de la corrélation par rapport au nombre de segment

J'ai effectué deux analyses de la trace avec la technique de la différence entre courbes relatives et filtrées par les moyennes mobiles et les paramètres suivant :

- Segmentation de la trace avec Ns = 1n, filtrage des fréquences d'occurrences de toutes les classes avec une moyenne mobile d'ordre 10.
- Segmentation de la trace avec Ns = 3n, filtrage des fréquences d'occurrences de toutes les classes avec une moyenne mobile d'ordre 30. J'ai défini un ordre de 3 fois supérieur à la première corrélation car on a 3 fois plus de segments avec Ns = 3n. J'espère ainsi isoler l'effet qu'apportera un Ns plus élevé.

J'ai récupéré les 50 paires de classe dont la corrélation est la plus forte pour chacune de ces deux analyses. Ces paires de classes sont triés par ordre alphabétique (voir figure 35). On cherche à voir comment le changement de Ns va affecter les résultats ou le classement des résultats. Donc si l'on prend les 50 meilleurs résultats pour les deux analyses, on pourrait regarder si on y retrouve les mêmes classes.

Sur la figure, la première colonne « R » est le rang[45] , « C1 » est une classe et « C2 » une autre classe à laquelle elle a été comparée C. C1 et C2 forment une paire de classe unique pour chaque ligne. « RE » est le score de déviation[46].

On constate pour les deux analyses, les 50 meilleures corrélations sont quasiment les mêmes. Cependant, on voit aussi que le Ns a un faible impact sur le classement et le score de déviation. On remarquera aussi qu'une partie des paires de classes manquantes sont parfois celles qui se trouvaient à un rang proche de 50.

---

45 C'est le classement de la corrélation entre deux classes. Pour cette trace le rang va jusqu'à 9591, parce qu'il y 9591 corrélations possible entre toutes les classes de la trace. Elles sont toutes classées selon leur résultat de 1 à 9591.

46 Voir au chapitre 4.2.5

Figure 35

Sensibilité entre Ns 1n filtrage d'ordre 10 et Ns 3n filtrage d'ordre 30

Ns 1n filtrage d'ordre 10					Ns 3n filtrage d'ordre 30			
R	C1	C2	RE		R	C1	C2	RE
42	EvilmCdContenuImpl.jav	EviOmCdContenu.java	16	=	31	EvilmCdContenuImpl.jav	EviOmCdContenu.java	8
26	**EvilmCdEvenementIr**	EviOdCdEvenement.jav	4		49	EvilmCdContenuImpl.jav	EviOdCdContenu.java	14
27	**EvilmCdEvenementIr**	EviOmCdEvenement.ja	4	=	23	**EvilmCdEvenementIr**	EviOmCdEvenement.java	6
37	**EvilmCdEvenementV**	EviOmCdEvenementVa	10		43	**EvilmCdEvenementIr**	EviOdCdEvenement.java	11
28	EvilmCdNotionImpl.java	EviOmCdNotion.java	5		45	**EvilmCdEvenementV**	EviOmCdEvenementVale	12
45	EvilmCdNotionImpl.java	EviOdNotion.java	18		10	EvilmCdNotionImpl.java	EviOmCdNotion.java	1
48	EvilmCdNotionImpl.java	EviOdFicheNotion.java	20		41	EvilmCdNotionImpl.java	EviOdCdNotion.java	10
1	**EvilmCdOrigineImpl.j**	EviOdCdOrigine.java	0		3	**EvilmCdOrigineImpl.j**	EviOmCdOrigine.java	0
3	**EvilmCdOrigineImpl.j**	EviOmCdOrigine.java	0		25	**EvilmCdOrigineImpl.j**	EviOdCdOrigine.java	7
4	**EvilmCdOrigineImpl.j**	EviOmLocalisation.java	0		18	EvilmCdProjetImpl.java	EviOmCdProjet.java	3
30	EvilmCdProjetImpl.java	EviOpProjetImage.java	6		21	EvilmCdProjetImpl.java	EviOdCdProjet.java	5
35	EvilmCdProjetImpl.java	EviOdCdProjet.java	9		40	EvilmCdProjetImpl.java	EviOpProjetImage.java	10
36	EvilmCdProjetImpl.java	EviOmCdProjet.java	9		32	**EvilmCumulusImpl.ja**	EviOpRecord.java	8
44	**EvilmCumulusImpl.ja**	EviOmImage.java	18		46	**EvilmCumulusImpl.ja**	EviOdImage.java	12
49	**EvilmCumulusImpl.ja**	EviOpRecord.java	21		48	**EvilmCumulusImpl.ja**	EviOmImage.java	13
23	EvilmEvenementImpl.java	EviOmEvenement.java	3		17	EvilmEvenementActeurIn	EviOmEvenementActeur.	2
29	EvilmEvenementSourceIr	EviOmEvenementSour	5		8	EvilmEvenementImpl.java	EviOmEvenement.java	1
5	EvilmFicheOrigineImpl.jav	EviOmFicheOrigine.java	0		9	EvilmEvenementSourceIr	EviOmEvenementSource	1
2	EvilmNotionImpl.java	EviOmNotion.java	0		20	EvilmFicheOrigineImpl.jav	EviOmFicheOrigine.java	4
10	EvilmObjetReferenceImpl	EviOmObjetReference.j	0		4	EvilmNotionImpl.java	EviOmNotion.java	0
11	EvilmPanierImpl.java	EviOmPanier.java	0		5	EvilmObjetReferenceImpl	EviOmObjetReference.jav	0
18	**EvilmRoleImpl.java**	EviOpRole.java	1		6	EvilmPanierImpl.java	EviOmPanier.java	0
24	**EvilmRoleImpl.java**	EviOdRole.java	4		38	EvilmPanierImpl.java	EviOdPanier.java	10
25	**EvilmRoleImpl.java**	EviOmRole.java	4	=	15	**EvilmRoleImpl.java**	EviOmRole.java	2
40	EvilmServiceImpl.java	EviOdService.java	15		29	**EvilmRoleImpl.java**	EviOpRole.java	7
41	EvilmServiceImpl.java	EviOmService.java	15		42	**EvilmRoleImpl.java**	EviOdRole.java	10
6	EvilmSourceOrigineImpl.j	EviOmSourceOrigine.ja	0		13	EvilmServiceImpl.java	EviOdService.java	2
32	EviOdCdContenu.java	EviOmCdContenu.java	8		14	EvilmServiceImpl.java	EviOmService.java	2
34	EviOdCdEvenement.java	EviOmCdEvenement.ja	9		16	EvilmSourceOrigineImpl.j	EviOmSourceOrigine.java	2
33	EviOdCdNotion.java	EviOdNotion.java	9		22	EviOdCdContenu.java	EviOmCdContenu.java	5
38	EviOdCdNotion.java	EviOdFicheNotion.java	14		30	EviOdCdEvenement.java	EviOmCdEvenement.java	8
46	EviOdCdNotion.java	EviOmCdNotion.java	20	=	34	EviOdCdNotion.java	EviOmCdNotion.java	9
7	EviOdCdOrigine.java	EviOmCdOrigine.java	0		36	EviOdCdNotion.java	EviOdNotion.java	9
8	EviOdCdOrigine.java	EviOmLocalisation.java	0		50	EviOdCdNotion.java	EviOdFicheNotion.java	14
9	**EviOdCdProjet.java**	EviOmCdProjet.java	0		26	EviOdCdOrigine.java	EviOmCdOrigine.java	7
21	**EviOdCdProjet.java**	EviOpProjetImage.java	3		12	**EviOdCdProjet.java**	**EviOmCdProjet.java**	2
12	EviOdFamille.java	EviOpFamille.java	0		28	**EviOdCdProjet.java**	**EviOpProjetImage.ja**	7
31	EviOdFicheNotion.java	EviOdNotion.java	7		7	EviOdFamille.java	EviOpFamille.java	0
43	EviOdFicheNotion.java	EviOmCdNotion.java	17		27	EviOdFicheNotion.java	EviOdNotion.java	7
50	EviOdImage.java	EviOpRecord.java	21	=	37	EviOdImage.java	EviOpRecord.java	10
39	EviOdNotion.java	EviOmCdNotion.java	14		47	EviOdImage.java	EviOmImage.java	13
47	EviOdNotion.java	EviOpNotion.java	20		39	EviOdPanier.java	EviOmPanier.java	10
13	EviOdRole.java	EviOmRole.java	0		19	EviOdRole.java	EviOpRole.java	3
19	EviOdRole.java	EviOpRole.java	2		33	EviOdRole.java	EviOmRole.java	8
14	EviOdService.java	EviOmService.java	0	=	1	EviOdService.java	EviOmService.java	0
16	EviOdSession.java	EviSecurite.java	0	=	11	EviOdSession.java	EviSecurite.java	1
15	EviOmCdOrigine.java	EviOmLocalisation.java	0		44	EviOmActeur.java	EviOpActeur.java	11
22	EviOmCdProjet.java	EviOpProjetImage.java	3	=	35	EviOmCdProjet.java	EviOpProjetImage.java	9
20	EviOmRole.java	EviOpRole.java	2	=	24	EviOmRole.java	EviOpRole.java	7
17	EviUcPanier.java	ROOT_PROCEDURE	0	=	2	EviUcPanier.java	ROOT_PROCEDURE	0

Maintenant, je vais comparer deux nouveaux résultats analyses avec les paramètres suivant et selon les mêmes principes que précédemment :

- Segmentation de la trace avec Ns = 3n, filtrage des fréquences d'occurrences de toutes les classes avec une moyenne mobile d'ordre 30.
- Segmentation de la trace avec Ns = 32n, filtrage des fréquences d'occurrences de toutes les classes avec une moyenne mobile d'ordre 320.

Sur la figure 36, on constate qu'avec Ns = 32n, on n'a pas non plus d'impact majeur sur les résultats des corrélations. On retrouve plus ou moins les mêmes paires de classes des 2 cotés.

Figure 36

Sensibilité entre Ns 3n filtrage d'ordre 30 et Ns 32n filtrage d'ordre 320

Ns 3n filtrage d'ordre 30					Ns 32n filtrage d'ordre 320				
R	C1	C2	RE		R	C1	C2	RE	
31	EvilmCdContenuImpl.java	EviOmCdContenu.jav	8		1	EviGcPanier.java	EviUcPanier.java	0	
49	EvilmCdContenuImpl.java	EviOdCdContenu.java	14		2	EviGcPanier.java	EviUiPanier.java	0	
23	**EvilmCdEvenementIm**	EviOmCdEvenement.j	6		19	EvilmCdContenuImpl.java	EviOmCdContenu.java	4	
43	**EvilmCdEvenementIm**	EviOdCdEvenement.ja	11		34	EvilmCdContenuImpl.java	EviOdCdContenu.java	9	
45	**EvilmCdEvenementVa**	EviOmCdEvenementV	12		7	**EvilmCdEvenementImp**	EviOmCdEvenement.ja	2	
10	EvilmCdNotionImpl.java	EviOmCdNotion.java	1		40	**EvilmCdEvenementImp**	EviOdCdEvenement.jav	11	
41	EvilmCdNotionImpl.java	EviOdCdNotion.java	10		29	**EvilmCdEvenementVal**	EviOmCdEvenementV	7	
3	**EvilmCdOrigineImpl.ja**	EviOmCdOrigine.java	0		13	EvilmCdNotionImpl.java	EviOmCdNotion.java	3	
25	**EvilmCdOrigineImpl.ja**	EviOdCdOrigine.java	7		23	**EvilmCdOrigineImpl.jav**	EviOmCdOrigine.java	5	
18	EvilmCdProjetImpl.java	EviOmCdProjet.java	3		49	**EvilmCdOrigineImpl.jav**	EviOdCdOrigine.java	15	
21	EvilmCdProjetImpl.java	EviOdCdProjet.java	5		20	EvilmCdProjetImpl.java	EviOmCdProjet.java	4	
40	EvilmCdProjetImpl.java	EviOpProjetImage.jav	10		33	EvilmCdProjetImpl.java	EviOdCdProjet.java	9	
32	**EvilmCumulusImpl.jav**	EviOpRecord.java	8		43	EvilmCdProjetImpl.java	EviOdService.java	12	
46	**EvilmCumulusImpl.jav**	EviOdImage.java	12		47	EvilmCdProjetImpl.java	EviOpProjetImage.java	14	
48	**EvilmCumulusImpl.jav**	EviOmImage.java	13		26	**EvilmCumulusImpl.jav**	EviOpRecord.java	6	
17	**EvilmEvenementActe**	EviOmEvenementActe	2		32	**EvilmCumulusImpl.jav**	EviOdImage.java	9	
8	**EvilmEvenementImpl.j**	EviOmEvenement.java	1		45	**EvilmCumulusImpl.jav**	EviOmImage.java	12	
9	**EvilmEvenementSour**	EviOmEvenementSou	1		12	**EvilmEvenementActeu**	EviOmEvenementActe	3	
20	EvilmFicheOrigineImpl.java	EviOmFicheOrigine.jav	4		15	**EvilmEvenementImpl.j**	EviOmEvenement.java	3	
4	EvilmNotionImpl.java	EviOmNotion.java	0		11	**EvilmEvenementSourc**	EviOmEvenementSour	3	
5	EvilmObjetReferenceImpl.j	EviOmObjetReference	0		14	EvilmFicheOrigineImpl.java	EviOmFicheOrigine.jav	3	
6	EvilmPanierImpl.java	EviOmPanier.java	0		39	EvilmImageImpl.java	EviOmImage.java	11	
38	EvilmPanierImpl.java	EviOdPanier.java	10		5	EvilmNotionImpl.java	EviOmNotion.java	1	
15	**EvilmRoleImpl.java**	EviOmRole.java	2		4	EvilmObjetReferenceImpl.ja	EviOmObjetReference.	0	
29	**EvilmRoleImpl.java**	EviOpRole.java	7		6	EvilmPanierImpl.java	EviOmPanier.java	2	
42	**EvilmRoleImpl.java**	EviOdRole.java	10		41	EvilmProjetImpl.java	EviOpProjet.java	11	
13	EvilmServiceImpl.java	EviOdService.java	2		16	**EvilmRoleImpl.java**	EviOmRole.java	3	
14	EvilmServiceImpl.java	EviOmService.java	2		28	**EvilmRoleImpl.java**	EviOpRole.java	6	
16	EvilmSourceOrigineImpl.jav	EviOmSourceOrigine.j	2		37	**EvilmRoleImpl.java**	EviOdRole.java	10	
22	EviOdCdContenu.java	EviOmCdContenu.jav	5		9	EvilmServiceImpl.java	EviOmService.java	2	
30	EviOdCdEvenement.java	EviOmCdEvenement.j	8		31	EvilmServiceImpl.java	EviOdService.java	8	
34	EviOdCdNotion.java	EviOmCdNotion.java	9		17	EvilmSourceOrigineImpl.jav	EviOmSourceOrigine.ja	3	
36	EviOdCdNotion.java	EviOdNotion.java	9		21	EviOdCdContenu.java	EviOmCdContenu.java	5	
50	EviOdCdNotion.java	EviOdFicheNotion.jav	14		36	EviOdCdEvenement.java	EviOmCdEvenement.ja	10	
26	EviOdCdOrigine.java	EviOmCdOrigine.java	7		50	EviOdCdNotion.java	EviOdNotion.java	16	
12	**EviOdCdProjet.java**	**EviOmCdProjet.jav**	2		35	EviOdCdOrigine.java	EviOmCdOrigine.java	10	
28	**EviOdCdProjet.java**	**EviOpProjetImage**	7		22	**EviOdCdProjet.java**	EviOmCdProjet.java	5	
7	EviOdFamille.java	EviOpFamille.java	0		25	**EviOdCdProjet.java**	EviOpProjetImage.java	5	
27	EviOdFicheNotion.java	EviOdNotion.java	7	=	46	EviOdFicheNotion.java	EviOdNotion.java	13	
37	**EviOdImage.java**	EviOpRecord.java	10		30	**EviOdImage.java**	EviOmImage.java	7	
47	**EviOdImage.java**	EviOmImage.java	13		44	**EviOdImage.java**	EviOpRecord.java	12	
39	EviOdPanier.java	EviOmPanier.java	10		18	EviOdRole.java	EviOpRole.java	3	
19	EviOdRole.java	EviOpRole.java	3		27	EviOdRole.java	EviOmRole.java	6	
33	EviOdRole.java	EviOmRole.java	8		24	EviOdService.java	EviOmService.java	5	
1	EviOdService.java	EviOmService.java	0		48	EviOdService.java	EviOmCdProjet.java	15	
11	**EviOdSession.java**	EviSecurite.java	1	=	8	**EviOdSession.java**	EviSecurite.java	2	
44	**EviOmActeur.java**	EviOpActeur.java	11	=	42	**EviOmActeur.java**	EviOpActeur.java	11	
35	**EviOmCdProjet.java**	EviOpProjetImage.jav	9	=	38	**EviOmCdProjet.java**	EviOpProjetImage.java	11	
24	**EviOmRole.java**	EviOpRole.java	7	=	10	**EviOmRole.java**	EviOpRole.java	3	
2	**EviUcPanier.java**	ROOT_PROCEDURE	0		3	**EviUcPanier.java**	EviUiPanier.java	0	

7. Considérations d'implémentation

Ce chapitre décrirait le modèle de données avec une description des tables et leur rôle ainsi que l'architecture de l'application avec une description des classes de celle-ci.

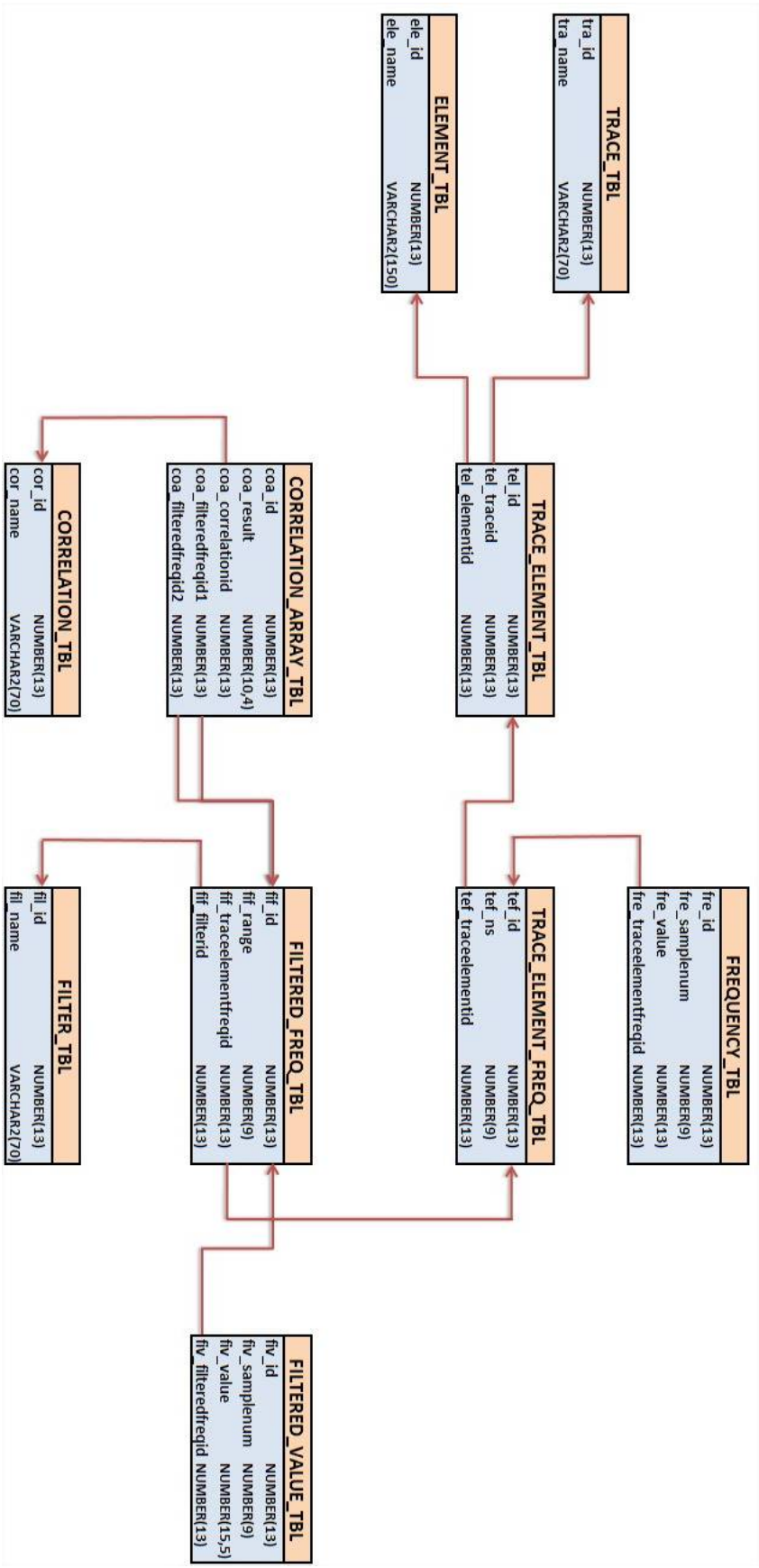
7.1 Modèle de données

Je vais décrire pour le modèle de données, les tables et leur rôle (voir figure 37) :

- ***Trace_tbl*** est la table qui contient les noms des traces qu'il est possible d'analyser. Elle permet de distinguer les informations générées pour chaque trace.
- ***Element_tbl*** contient le nom de tous les éléments, toutes traces confondues.
- ***Trace_element_tbl*** permet de distinguer les classes de chaque trace. Chacun des tuples de cette table est unique. On n'a pas deux fois la même classe pour la même trace.
- ***Frequency_tbl*** est la table qui contiendra toutes les valeurs des fréquences d'occurrences.
- ***Trace_element_freq*** permet faire le lien entre les fréquences d'occurrences d'une classe avec une segmentation de la trace définie par l'attribut tef_ns et les fréquences d'occurrences contenues dans la table Frequency_tbl.
- ***Filter_tbl*** définit le nom des types de filtres disponible dans l'application.
- ***Filtered_value_tbl*** contient toutes les valeurs des fréquences d'occurrences filtrées.
- ***Filtered_freq_tbl*** permet stocker les fréquences filtrées des fréquences d'occurrences contenues dans Trace_element_freq. Le type de filtrage est défini par l'attribut fif_filterid.
- ***Correlation_tbl*** définit les techniques de corrélation applicable par l'application.
- ***Correlation_array_tbl*** permet d'enregistrer le résultat de la corrélation entre les fréquences d'occurrences filtrées des deux classes comparées. L'attribut coa_correlationid nous renseigne sur la technique de corrélation employée.

Figure 37

Modèle de donnée de l’application



7.2 Architecture de l'application

Maintenant nous allons voir pour la couche domaine quels sont les packages de l'application et les classes importantes de chacun d'entre eux (voir figure 38):

- Le package « pd » contient des classes qui sont nécessaires à la persistance des données. Elle contient une classe qui gère la connexion à la base de données. Elle a été écrite par Peter Daehne[47].

- Le package « persistance » contient les classes de persistance. Il y a deux classes par table. L'une permet de récupérer automatiquement une collection d'objets en interrogeant sa table et l'autre est l'objet persistant dont est composée la collection.

- Le package « jfreechart [48]» est une librairie qui permet d'afficher les graphiques de l'application.

- Le package « chart » contient les classes qui permettent de préparer les données afin de les afficher à l'aide de la librairie « jfreechart ».

    - CustomDataset permet de créer une source de données. Elle va aussi transformer les séries de données à afficher de manière relative (ex. : les fréquences relatives).

    - ChartGen est la classe qui crée une fenêtre et affiche un graphique à avec les données contenues dans CustomDataset.

- Le package « core » est le cœur de l'application, elle contient les contrôleurs, et des classes de traitements.

    - TraceAnalyserControl permet de lancer les processus de corrélation avec tous les paramètres désirés.

    - TraceTableLoader permet d'analyser une table qui contient la trace d'exécution. Elle évalue le nombre de classes dans la trace (n). Elle récupère et insère la fréquence d'occurrence de chaque segment pour toutes les classes, si elles n'ont pas déjà été insérées dans la base. Elle ne fait rien si les fréquences pour le Ns demandé existent déjà.

---

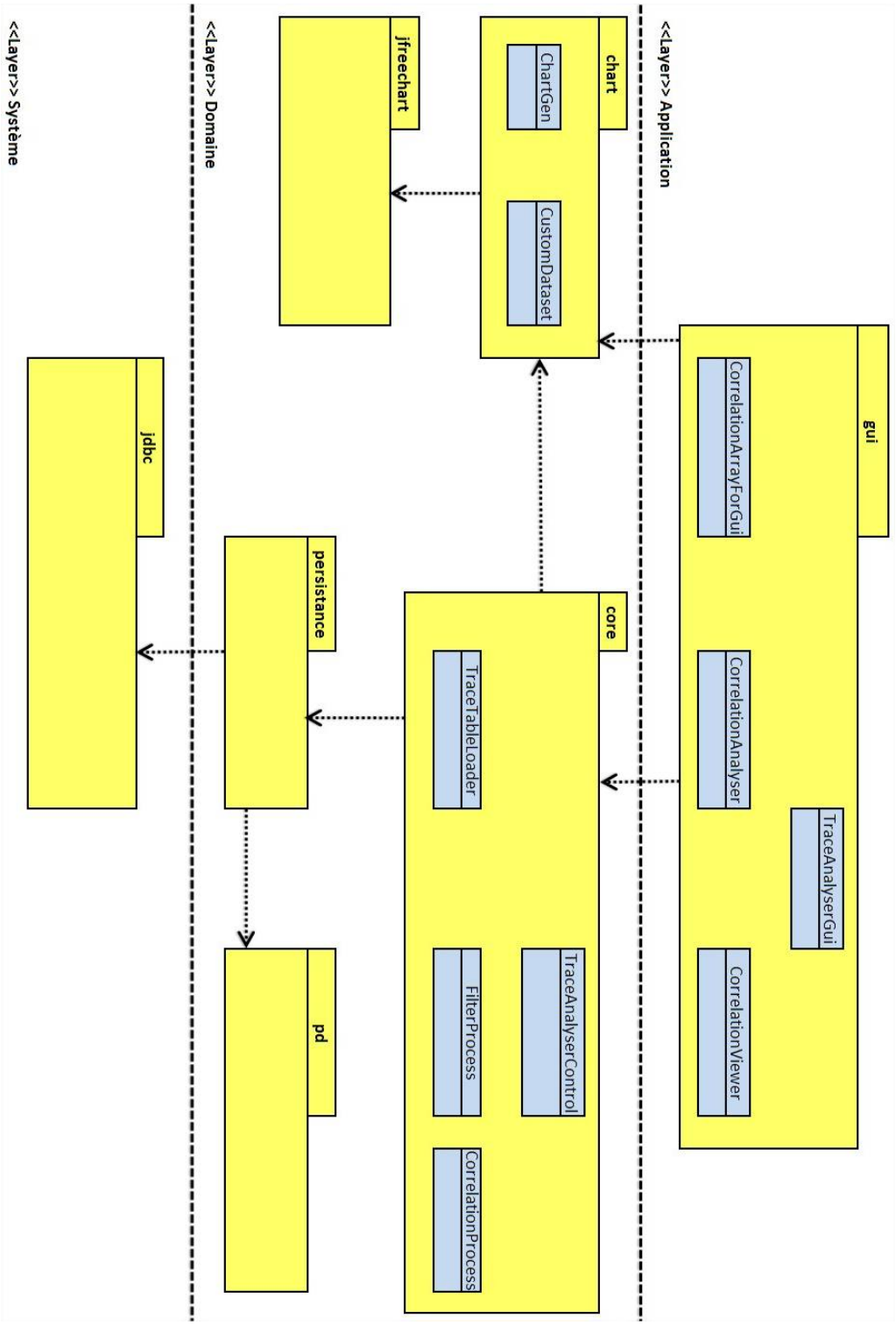
47 Professeur HES

48 OBJECT REFINERY LIMITED. JFreeChart [en ligne]. http://www.jfree.org/jfreechart/

- FilterProcess parcourt les fréquences d'occurrences d'une classe pour un Ns donné puis calcule et insère les fréquences d'occurrences filtrées.
- CorrelationProcess effectue le processus de corrélation des classes de la trace.

Figure 38

Architecture de l’application



Sur la figure 38 on peut voir les packages et leurs dépendances.

7.3 Travailler avec les traces d'exécution

Les traces d'exécutions peuvent être énormes. C'est pourquoi on doit très vite exclure de travailler sur des fichiers plats (XML, TXT, etc.), et passer à une base de données performante. En effet, même les petites bases de données sont à exclure car elles seront vite dépassées par la quantité d'information que représentent certaines traces.

Il serait donc plus recommandé de travailler avec la base de données comme Oracle [49].

Dans la mesure où une application doit travailler sur des traces d'exécutions avec une base de données, il est préférable de bien optimiser ces requêtes SQL afin de ne récupérer que l'information utile à l'application cliente, d'autant plus si la base de données est très performante. De plus, il faut absolument les longues séries de requêtes[50], si on faire la même opération en une seule. Je l'ai constaté dans mon travail : les performances chutent dramatiquement.

Pour améliorer les performances, on peut déléguer à la base de données certains traitements afin de ne pas envoyer des milliers de requêtes SQL via le réseau en utilisant par exemple, des procédures stockées[51]. Ainsi, on effectue un appel à une procédure stockée qui va lancer le traitement directement dans la base.

Il y a un autre facteur important qui permet d'améliorer les performances. Les résultats des opérations sur les traces peuvent eux aussi générer beaucoup d'informations et surtout on peut éviter de les refaire systématiquement pour une même trace. Si l'on considère qu'une trace est immuable, alors on ne doit pas refaire les mêmes opérations. On enregistre aussi les résultats dans notre base de données. Ça peut paraître évident, mais il ne faut pas l'omettre pour autant.

Créer des indexes dans la base de données après avoir inséré de grande quantité de données permet d'avoir des meilleurs temps de réponse sur les requêtes SQL.

---

49 Sauf Oracle Express Edition qui est gratuit, mais limité en stockage (4 Go de donnés maximum) et en mémoire RAM (1Go de mémoire maximum)

50 Des centaines ou des milliers de requêtes

51 Ce sont des procédures qui s'exécutent directement dans la base. Elles sont très rapides pour traiter certaines opérations nécessitant des requêtes paramétrées.

Conclusions

La technique de la différence entre courbes relatives et filtrées par les moyennes mobiles donne de très bons résultats. Elle est performante par sa simplicité de calcul, et la stabilité de ses résultats même avec une faible segmentation. Un facteur important qui lui permet de repérer les classes corrélées est le filtrage numérique qui, en lissant les courbes et en révélant les tendances des courbes, permet une comparaison de meilleure qualité. Le score de déviation permet ensuite de montrer les classes les mieux corrélées.

Avec mon application et ma technique de corrélation dynamique, j'ai pu analyser une trace de 600 000 appels, et une autre d'environ 7 millions d'appels. On remarque donc que la performance de l'application peut être améliorée de façon significative.

Pour les futures recherches, ma technique peut être enrichie, pour tenir compte de situations plus spécifiques, notamment lorsque des classes ne sont corrélées que sur une partie de la trace. J'ai pu constater ce phénomène en observant certaines courbes et le résultat de leur score de déviation. Une idée pour palier à cette situation, serait de calculer l'écart type entre deux moyennes mobiles exprimées de manière relatives sur des intervalles de segments contigus. En incluant la valeur de l'écart type dans le score de déviation, cela permettrait de rendre les corrélations encore plus pertinentes qu'elles ne le sont.

Certaines courbes[52] filtrées que j'ai pu observer sur les classes fortement corrélées, présentent des tendances similaires pour les mêmes segments. Une technique basée sur l'analyse des tendances (voir Chapitre 4.2.3) pourrait être mise en œuvre pour exploiter ces similitudes et tenter de corréler les classes de la trace.

Pour conclure, l'application de l'analyse technique financière à l'analyse de traces d'exécution de programme apporte une approche nouvelle et prometteuse qui permettra, je l'espère, de faire avancer les recherches actives dans ce domaine de la recherche.

52 Courbes des fréquences d'occurrences filtrées

Bibliographie

Livres

BARON, François. Le chartisme : méthodes et stratégies pour gagner en bourse. [Paris] : Eyrolles, 2008. 490 p. (Analyse technique).

BECHU, Thierry, BERTRAND, Eric, NEBENZAHL, Julien. *L'analyse technique : théories et méthodes*. Paris : Economica, 2008. (Finance).

Articles

HAMOU-LHADJ, A., LETHBRIDGE, T.C - *A Survey of Trace Exploration Tools and Techniques*. Proc of the Conference of the Centre for Advanced Studies on Collaborative Research CASCON 2004, October 5-7, 2004,Markham, Canada.

DUGERDIL PH. - *Architecture-Based Software Reengineering*. Technical report, HEG Geneva, February 2006

DUGERDI PH., JOSSI S. - Empirical assessment of execution traces segmentation in reverse-engineering. Porto, ICSOFT, 2008. 8 p.

ZAIDMAN A., DEMEYER S. - Managing trace data volume through a heuristical clustering process based on event execution frequency. EUROPEAN CONFERENCE ON SOFTWARE MAINTENANCE AND REENGINEERING (CSMR) (2004, Tampere) : proceedings of the eight CSMR. Tampere, 24-25-26 mars 2004. [10 p.]

DUGERDIL PH. - Using trace sampling techniques to identify dynamic clusters of classes. IBM CAS SOFTWARE AND SYSTEMS ENGINEERING SYMPOSIUM (2007, Dublin). : proceedings of the 2007 conference of the center for advanced studies on Collaborative research. Hamilton auditorium of Dublin, 24 octobre 2007. 9 p.

KUHN A., GREEVY O. - Exploiting the analogy between traces and signal processing. INTERNATIONAL CONFERENCE ON SOFTWARE MAINTENANCE (2006, Philadelphie) : processing of IEEE ICSM. Philadélphie, 24-25-26-26 septembre 2006. 9 p.

CORNELISSEN B., MOONEN L. - Visualizing similarities in execution traces. PROGRAM COMPREHENSION THROUGH DYNAMIC ANALYSIS (2007, Vancouver). : proceedings ot the 3rd international workshop on program comprehension through dynamic analysis. Vancouver, 29 octobre 2007. 10 p.

BENNETT C., MYERS D., STOREY M., GERMAN D. - Working with "Monster" Traces: Building a Scalable, Usable Sequence Viewer. PROGRAM COMPREHENSION THROUGH DYNAMIC ANALYSIS (2007, Vancouver). : proceedings of the 3rd international workshop on program comprehension through dynamic analysis. Vancouver, 29 octobre 2007. 5 p.

Bergey J., Smith D., Weiderman N., Woods S. – Options Analysis for Reengineering (OAR) : Issues and Conceptual Approach. Sofware Engineering Institute, Carnegie

Sites web

BNP PARIBAS. *Tendances, supports et résistances* [en ligne]. http://www.bnpparibas.net/banque/portail/particulier/Fiche?type=fiche&identifiant=NOT_Tendances_supports_et_resistances_20060403115328 (consulté le 30.11.2008)

CLUB D'ENTRAIDE DES DEVELOPPEURS FRANCOPHONES. *Le SQL de A à Z : Fonctions SQL* [en ligne]. http://sqlpro.developpez.com/cours/sqlaz/fonctions/ (consulté le 30.11.2008)

DAILY BOURSE. *Apprendre la bourse : notions de supports et de résistances* [en ligne]. http://www.daily-bourse.fr/NOTIONS-DE-SUPPORT-ET-DE-RESISTANCE-vtptc-1578.php (consulté le 30.11.2008)

OBJECT REFINERY LIMITED. *JFreeChart* [en ligne]. http://www.jfree.org/jfreechart/ (consulté le 30.11.2008)

PROREALTIME. *Logiciel analyse technique temps réel* [en ligne]. http://www.prorealtime.com/fr/ (consulté le 30.11.2008)

RYDE, Kevin. *Endpoint moving average* [en ligne]. http://www.geocities.com/user42_kevin/chart/manual/Endpoint-Moving-Average.html (consulté le 30.11.2008)

TRADING SCHOOL. *Epaule tête épaule (ÉTÉ) : les figures chartistes, glossaire de la bourse* [en ligne]. http://www.trading-school.eu/glossaire-bourse/fiche-Epaule-Tete-Epaule-ETE--47 (consulté le 30.11.2008)

WIKIMEDIA FOUNDATION. *Wikipédia* [en ligne]. http://fr.wikipedia.org/ (consulté le 30.11.2008)