Ecole Polytechnique de l'Université François Rabelais de Tours
Département Informatique
64 avenue Jean Portalis
37200 Tours, France
Tél. +33 (0)2 47 36 14 14
polytech.univ-tours.fr

Projet de programmation et génie logiciel

2016-2017

Diffusion open source d'une bibliothèque en c++, pour la mise en correspondance de séquences

Tuteur académique
Nicolas Ragot

Étudiants
Houda boutbib (DI4)
Mohammed elmoutaraji (DI4)

11 janvier 2017

Liste des intervenants

Nom	Email	Qualité
Houda boutbib	houda.boutbib@etu.univ-tours.fr	Étudiant DI4
Mohammed elmoutaraji	mohammed.elmoutaraji@etu.univ-tours.fr	Étudiant DI4
Nicolas Ragot	nicolas.ragot@univ-tours.fr	Tuteur académique, Département Informatique

Avertissement

Ce document a été rédigé par Houda boutbib et Mohammed elmoutaraji susnommés les auteurs.

L'Ecole Polytechnique de l'Université François Rabelais de Tours est représentée par Nicolas Ragot susnommé le tuteur académique.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

Les auteurs reconnaissent assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

Les auteurs attestent que les propos du document sont sincères et assument l'entière responsabilité de la véracité des propos.

Les auteurs attestent ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

Les auteurs attestent que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

Les auteurs reconnaissent qu'ils ne peuvent diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable du tuteur académique et de l'entreprise.

Les auteurs autorisent l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.

Pour citer ce document

Houda boutbib et Mohammed elmoutaraji, *Diffusion open source d'une bibliothèque en c++, pour la mise en correspondance de séquences*, Projet de programmation et génie logiciel, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2016-2017.

```
@mastersthesis{
    author={boutbib, Houda and elmoutaraji, Mohammed},
    title={Diffusion open source d'une bibliothèque en c++,
       pour la mise en correspondance de séquences},
    type={Projet de programmation et génie logiciel},
    school={Ecole Polytechnique de l'Université François Rabelais de Tours},
    address={Tours, France},
    year={2016-2017}
}
```

Document mis en forme sous LaTeX
grâce à la classe de document polytech.cls V2.1.13 (2016/12/14

Table des figures

Table des figures

1 Remerciements

Au terme de ce travail, ont saisi l'occasion pour adresser nos remerciements à tous ceux qui, de près ou de loin, ont contribué à sa réussite. On tient à exprimer nos plus vifs remerciements à notre encadrant M.Nicolas ragot, pour son encadrement et son orientation tout au long du semestre. On remercie également bastien menier , Antoine Fonfria et Nicolas Gougeon qui ont étaient source de documentation pour nous à l'aide de leurs rapport et code sources.

2 Introduction

Ce projet rentre dans le cadre de la validation du semestre 7 de la 4 ème année en cycle d'ingénieur spécialité informatique. L'objectif est de faire un travail de recherche d'analyse et d'amélioration sur un projet existant ainsi que trouver des solutions adéquates pour corriger les failles et toute structures qui ne répond pas aux règles de la qualité logiciel. Le projet se décompose aux phases suivantes :

- Comprendre l'objectif de la librairie fourni.
- Compréhensibilité des spécifications des méthodes principales.
- Analyse des diagrammes utilisées.
- Vérification de la qualité du code.
- Amélioration du code.
- Tester la librairie.

3 Chapitre 1 : Étude de l'existant

3.1 Le projet

La Matching Tool Box est un outil permettant de comparer des séquences. L'idée est d'avoir une séquence cible et une séquence référence, puis de déterminer si la seconde apparaît dans la première. Cette correspondance peut prendre plusieurs Forme, mais généralement, on fait état d'une distance. Cette bibliothèque a été développée en C++ et intègre un certain nombre d'algorithmes qui assurent la recherche de correspondance.

3.2 Lecture des rapports

Après la lecture attentive des différents rapports on a bien compris les différentes méthodes de correspondances utilisées ainsi que les différents types de séquences qu'on vas aborder dans le chapitre suivant. - On a compris les entrées, sorties et les buts des méthodes sans passer par le fonctionnement mathématique de chaque méthode parce que nous voulions voir le code d'un point de vue de génie logiciel.

3. Compréhension des spécifications des méthodes principales

Parmi les algorithmes principaux :

- *CDP (Continous Dynamic Programming)*

But : comparer une séquence référence avec une séquence source plus grande

Input : Référence R[..T] et Source C[..M] avec T et M les tailles des séquences Output : tableau qui contient les cout CDP l'indice du tableau nous donne la place de correspondance dans la séquence d'entrée

- *Algorithme FSM (Flexible Sequence Matching)*

But : trouver la correspondance de type un à un, un à plusieurs et plusieurs à un
Input : Référence R[..T] et Source C[..M] avec T et M les tailles des séquences sachant que la taille de la séquence source inférieur à une séquence référence

- *Algorithme ESC (Exemplary Sequence Cardinality)*

But : pouvoir sauter des éléments de la séquence référence pour pouvoir éliminer certains bruits de la séquence.

- *LEVENHSTEIN*

BUT : permet de calculer le nombre des modifications a effectué pour passer d'une séquence A à une séquence B.
INPUT : référence A[1..N] et source B[1..M]
OUTPUT : la distance

- *LCS (Longest Commun Subsequence)*

BUT : calcule la plus longue sous séquence commune entre deux séquences
INPUT : référence A[1..N] et source B[1..M]
OUTPUT : longueur de là sous séquence

- *DTW (Dynamic Time Warping)*

BUT : calcule la similitude entre deux séquences temporelles permet d'accélérer le temps d'exécution des DTW
INPUT : référence A[1..N] et source B[1..M]
OUTPUT :DTW forme un chemin linéaire

- *MVM (Minimum Variance Matching)*

BUT : trouver le chemin optimal entre deux séquence, fusion de LCS et DTW
INPUT : référence A[1..N] et source B[1..M]
OUTPUT : MVM

3.4 Type de formatage de fichiers

La lecture des séquences références et cibles se fait à travers différents types de formatage de fichiers le programme lit chaque fichier comme un fichier de vecteurs. On peut dire que les fichiers peuvent être aussi simple que :

	A	B	C
1	1,2,3,49,64,34,9,22		
2			

Figure 1 – *fichiers texte référence*

Ou des fichiers complexe pour une séquence longue comme :

	A	B	C	D	E	F	G
1	0.25835,0.025862,0.034483,0.93966,0.017241,0.1,0.051724,0						
2	0.26771,0.034483,0.025862,0.93966,0.025862,0.1,0.047414,0						
3	0.28583,0.043103,0.017241,0.93966,0.034483,0.1,0.043103,0						
4	0.30078,0.051724,0.017241,0.93103,0.043103,0.1,0.047414,0						
5	0.31479,0.11207,0.017241,0.31034,0.66379,0.2,0.33621,0						
6	0.3252,0.11207,0.017241,0.31034,0.66379,0.2,0.33621,0						
7	0.34419,0.11207,0.017241,0.31034,0.66379,0.2,0.33621,0						
8	0.38069,0.18966,0.017241,0.31034,0.66379,0.3,0.34444,0						
9	0.4313,0.2069,0.017241,0.31034,0.66379,0.3,0.33549,0						

Figure 2 – *type de fichier complexe*

Ainsi que les fichiers xml

sequences.xml

```
<Sequences></Sequences>
<Type id="0 | 1 | 2" />
<Sequence idS="string">
  <!-- if type id = 0 -->
  <Element id="string" value="character" />
  <!-- if type id = 1 -->
  <Element id="string" value="float" />
  <!-- if type id = 2 -->
  <Element id="string">
    <VectorElement id="float" />
    ... <!-- more vector elements -->
  </Element>
  ... <!-- more elements of the same type -->
</Sequence>
... <!-- more sequences -->
```

Figure 3 – *type de fichier complexe*

Et les fichiers EXT

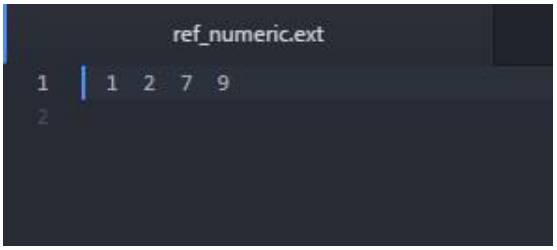

Figure 4 – *type de fichier complexe*

Les fichiers des résultats sont différés en fonction de l'algorithme utilisé :
Pour les algorithmes : Levenshtein, LCS,

z

```
Result
Distance = 5
Correspondance

f c a b a c c
o = + + + = +
t c       c
```

Figure 5 – *Résultat levenshtein et LCS*

Pour les algorithmes : DTW, MVM, CDP, FSM, ESC

```
5.43611: 0->0,1->1,2->1,3->1,4->2,4->3,4->4,5->5,6->6,7->7,8->8,9->9,10->10,11->11,12->12,13->13,14->13,15->13,16->14,17->
6,127->127,128->128,129->129,130->130,131->131,132->132,133->133,134->134,135->135,136->136,137->137,138->138,139->139,140
231,232->232,233->233,234->234,235->235,236->236,237->237,238->238,239->239,240->240,241->241,242->242,243->243,244->244,2
->338,339->339,340->340,341->341,342->342,343->343,344->344,345->345,345->346,346->347,347->348,348->349,349->349,350->350
```

Figure 6 – *DTW,MVM,CDP,FSM,ESC*

- **Explication du premier résultat**

  La première séquence est la séquence cible, la deuxième est la séquence référence
  Les symboles entre les deux séquences montrent le degré de similitude entre les deux éléments
  Si o : donc les deux éléments ne sont pas pareil faut échanger un des deux pour avoir le même élément
  Si = : les deux éléments sans égaux
  Si + : manque d'élément pour comparer
- **Explication du deuxième résultat :**

  Le résultat affiche la distance puis il regroupe les deux indices des éléments similaires entre la séquence cible et référence.

La syntaxe des fichiers paramètres

```
<lists>
  <Comparaison size1="integer" size2="integer" isVector="integer" sizeVector="integer" weightDistance="float">
    <LevenshteinParam addCost="float" delCost="float" transCost="float" />
    <CdpParam threshold="float" />
    <MvmParam divElast="integer" />
    <FsmParam Elasticity="integer" Weight="float" SkipCost="float" SmallSkipCost="float" TypeOfResult="integer" ResultFirstColumn="integer"
      NbMinPerLine="integer" DefaultSkipCost="float" StandardDeviation="integer" />
    <WeightNode>
      <sequenceWeight1>
        <sequenceElement value="float" />
        <sequenceElement value="float" />
        ...
        <!-- number of sequenceElement nodes should be equal to size1 (attribute of Comparaison) -->
      </sequenceWeight1>
      <sequenceWeight2>
        <sequenceElement value="float" />
        <sequenceElement value="float" />
        ...
        <!-- number of sequenceElement nodes should be equal to size2 (attribute of Comparaison) -->
      </sequenceWeight2>
      <vectorWeights>
        <sequenceVectorElement value="float" />
        <sequenceVectorElement value="float" />
        ...
        <!-- number of sequenceVectorElement nodes should be equal to sizeVector (attribute of Comparaison) -->
      </vectorWeights>
      <sequenceMatrix>
        <sequenceMatrixElement value="float" />
        <sequenceMatrixElement value="float" />
        ...
        <!-- number of sequenceMatrixElement nodes should be equal to
        size1 * size2 (attributes of Comparaison) -->
      </sequenceMatrix>
    </WeightNode>
  </Comparaison>
</lists>
```

Figure 7 – *syntaxe des fichiers paramètres*

4 Chapitre 2 : Modélisation du logiciel

4.1 Analyse des diagrammes

On a constaté un manque de diagramme qui identifie les différents liens entre les packages utilisé pour cela on a conçu un digramme de package qui montre bien les liens entre les différents modules.

4.1.1 Ancien diagramme de packages

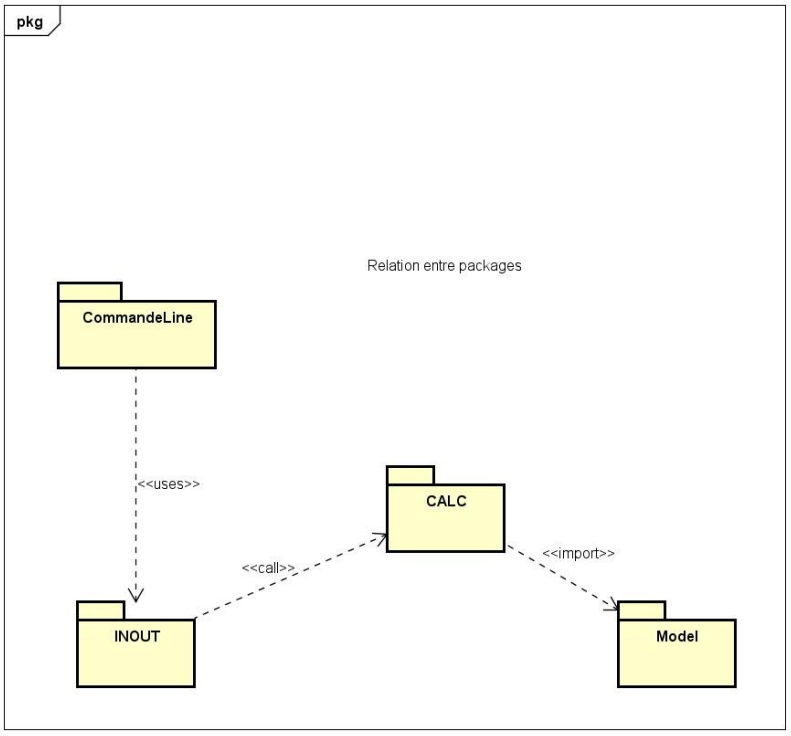

Figure 8 – *Ancien diagramme de packages*

4.1.2 Nouveau diagramme de packages

Nouveau diagramme de packages

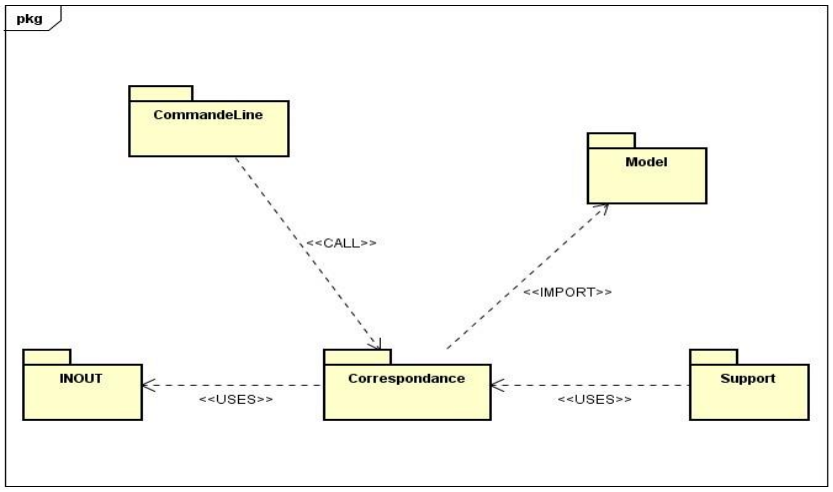

Figure 9 – *Nouveau diagramme de package*

- Le package CommandeLine a pour objectifs d'exécuter ce qui sera rentrer en ligne de commande par l'utilisateur.
- Le package INOUT a pour objectifs de lire et récupérer les séquences et les paramètres à partir de différents types de fichiers : CSVParser ,XMLParser ,ParamParser..
- Le package Model c'est au niveau de ce package que les différents types de séquence sont gérer : vecteur, simple, caractère
- Le package Correspondance a pour objectifs de traiter les résultats de correspondance après l'utilisation d'un ou plusieurs algorithme
- Le package Support au niveau de ce package on trouve les différents algorithmes utilisées pour comparer les séquences LCScorrespondance, CDPcorrespondance, MVMcorrespondance. . .

4.1.3 Nouveau diagramme de classe

Nouveau diagramme classe

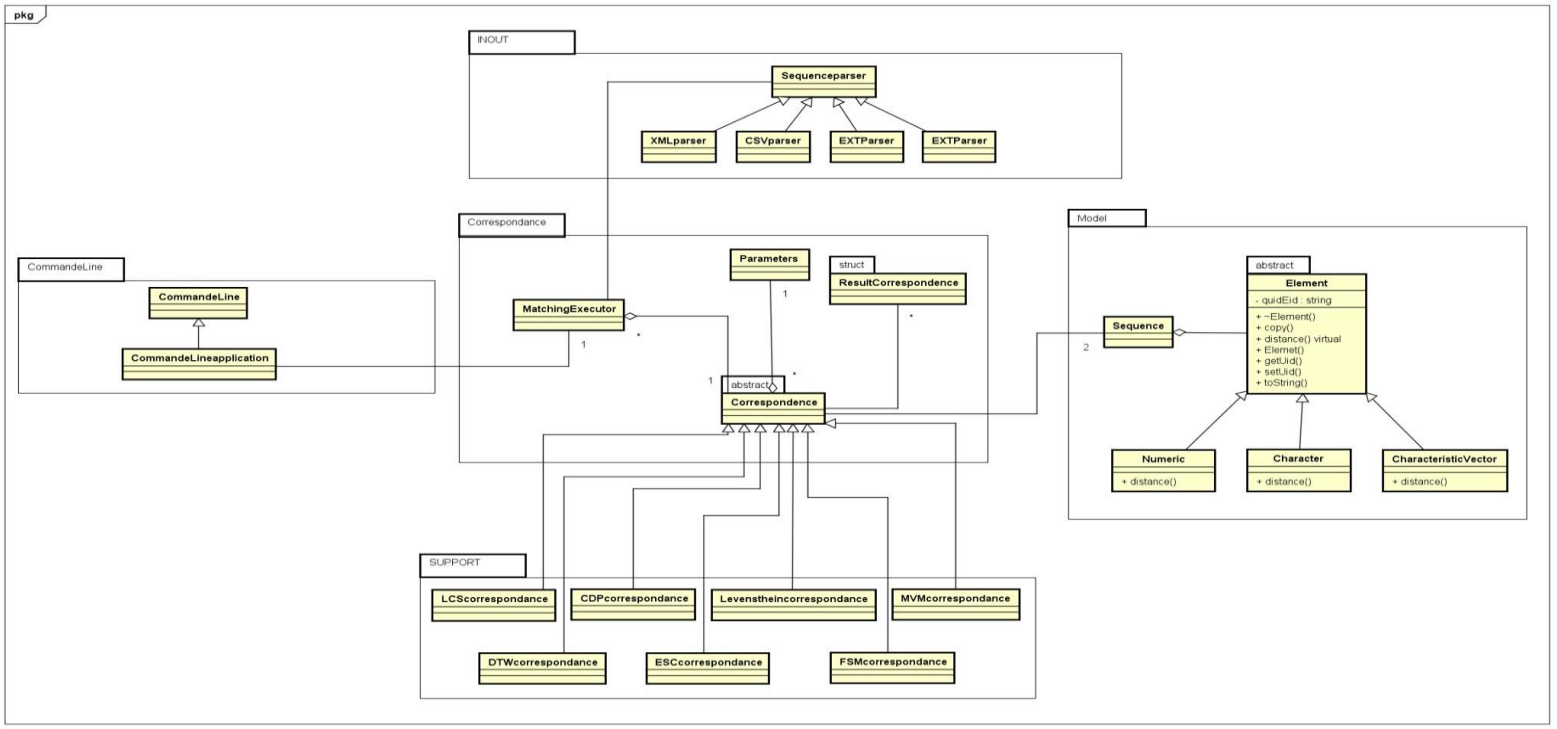

Figure 10 – *Nouveau diagramme de classe*

4.1.4 Diagramme d'objet

Pour enrichir le rapport on a ajouté un diagramme d'objet qui décrit les objets créer, voici un scénario explicatif à-propos de la trace des objets :
1. l'utilisateur entre une séquence donc un objet séquence est stocké, ce dernier peut être de type XML, CSV, EXT ou paramètre
2. Le programme parse la séquence est en sort les éléments constituant cette séquence soit sous Forme numérique, caractère ou vecteur.
3. Le résultat de correspondance est stocké puis affichée

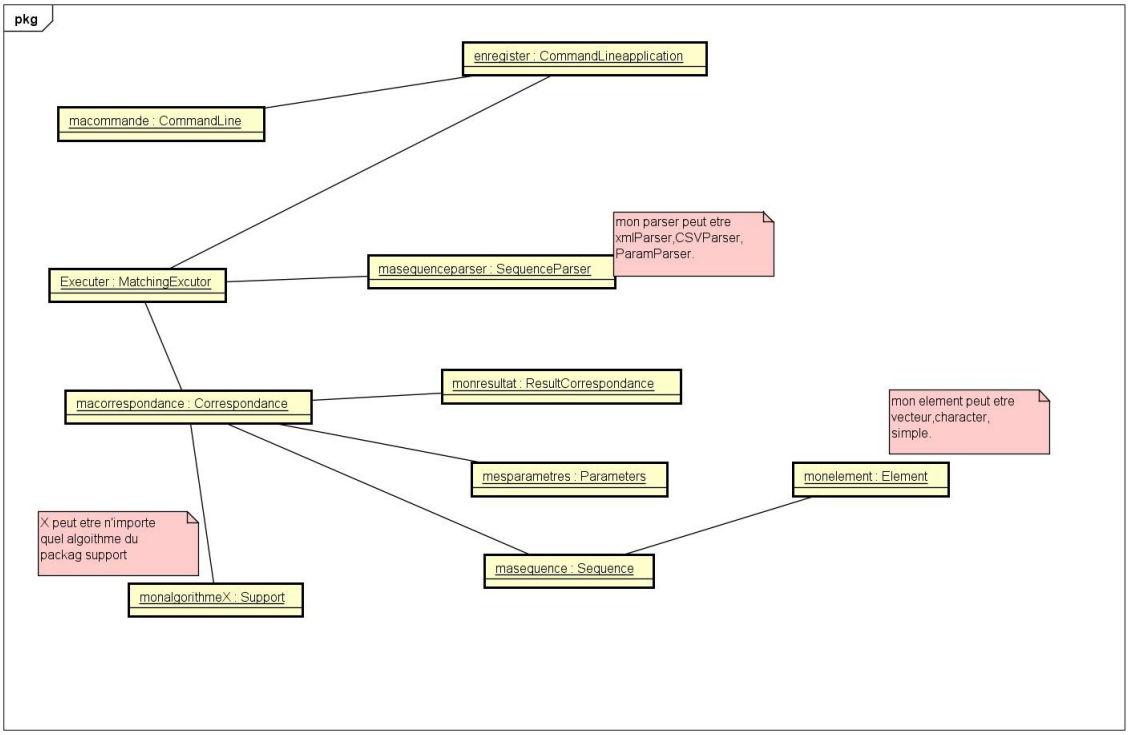

Figure 11 – *diagramme d'objet*

4.1.5 Diagramme de séquence

- Scénario de ce diagramme de séquence : Correspondance entre les séquences de :2 fichiers à partir d'une commande

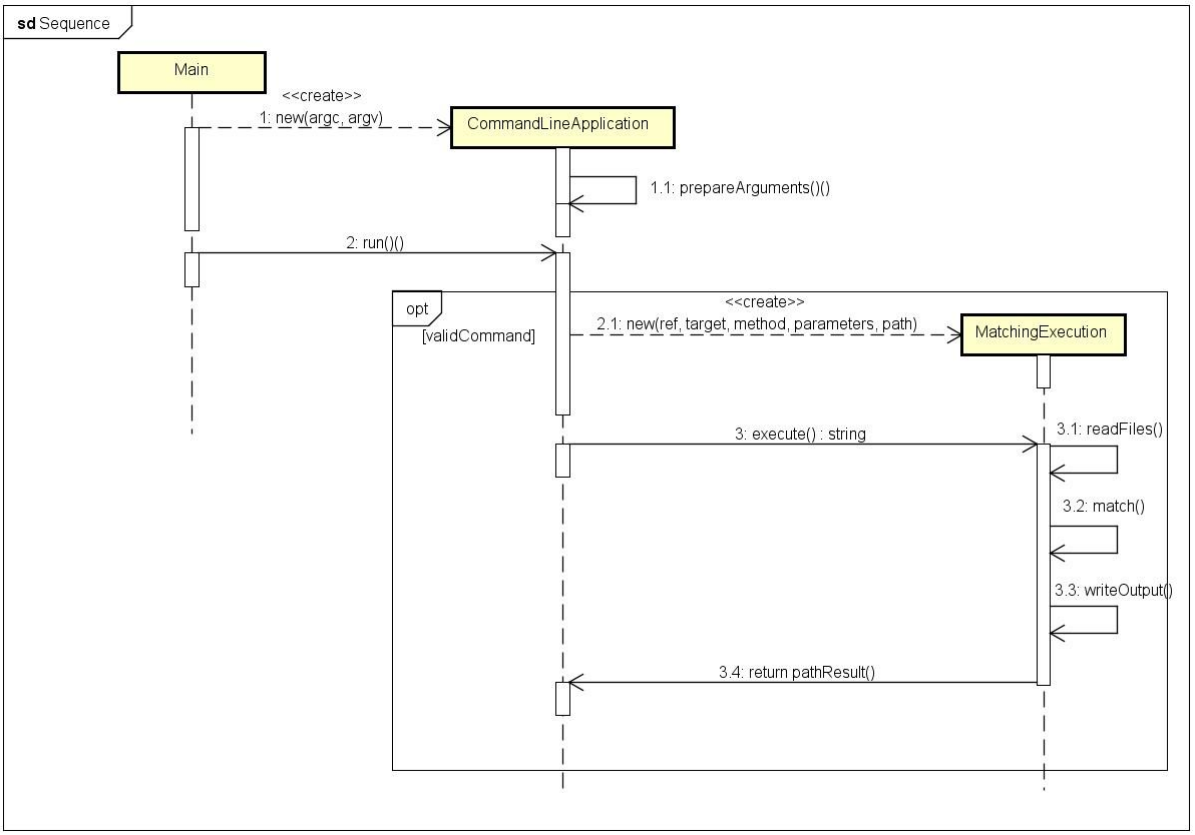

Figure 12 – *Correspondance entre les séquences de 2 fichiers*

- Scénario : Lire les fichiers pour extraire les séquences.

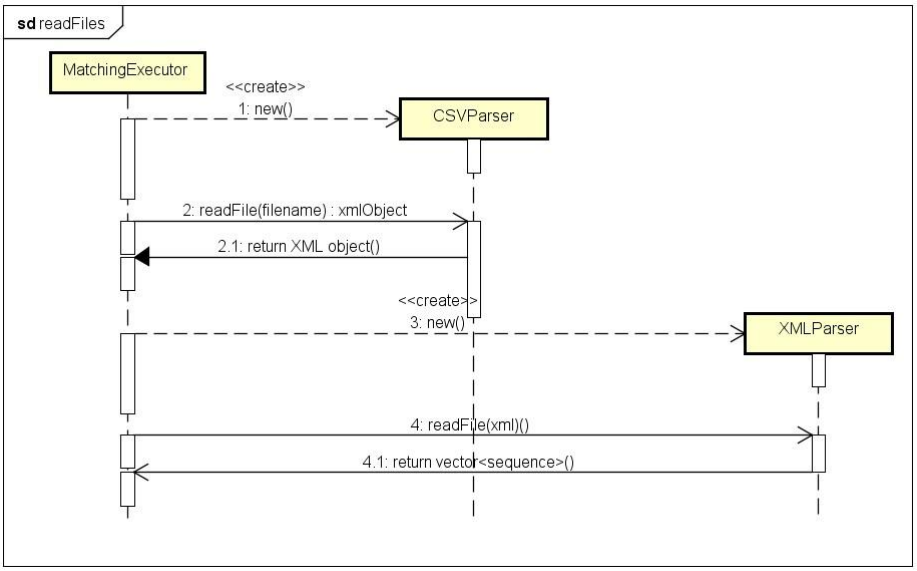

Figure 13 – *lecture des fichiers pour extraire les séquences*

- Scénario : Correspondance entre les séquence d'un fichier cible et un fichier référence

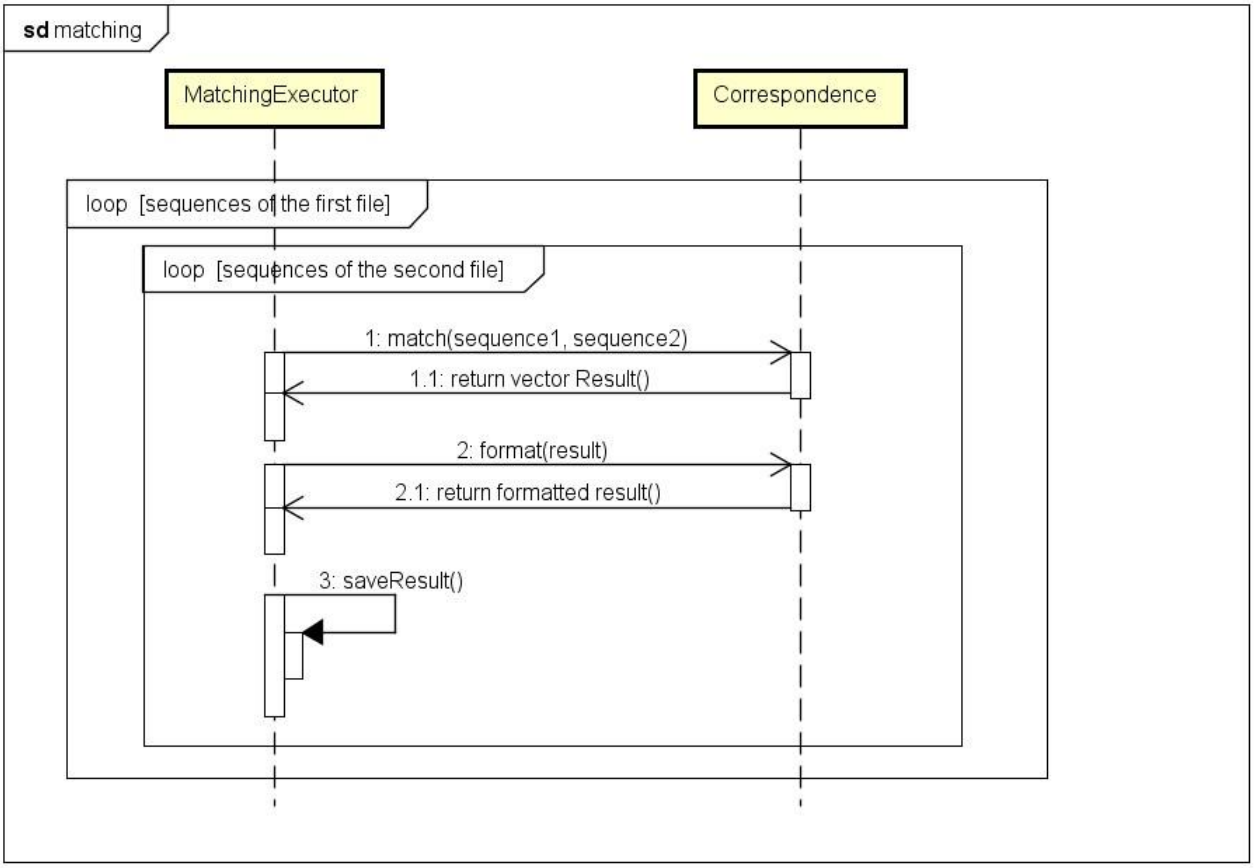

Figure 14 – *Correspondance entre les séquence d'un fichier cible et un fichier référence*

- Scénario : écriture des résultats.

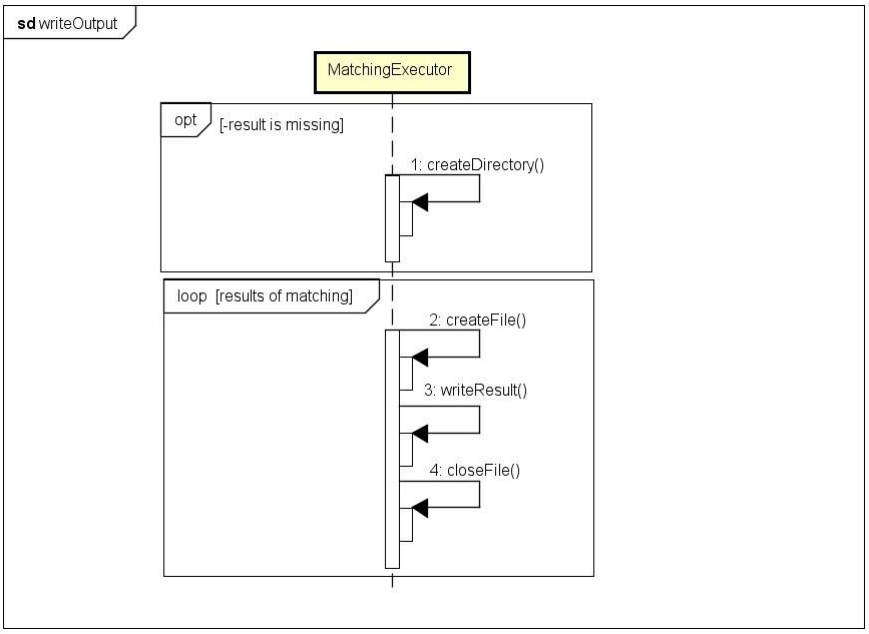

Figure 15 – *écriture des résultats de correspondance*

5 Chapitre 3 : Nouvelle architecture logicielle

Afin d'organiser le code entre les tests et la libraire on a choisi l'architecture suivante : on a généré une solution au niveau du Visual studio qui contient trois projets :

Le projet Matching Library :
Ce projet est constitué de classes de la bibliothèque, Il est également configuré comme une bibliothèque statique afin de produire un fichier .lib qui peut être utilisé avec d'autres projets.

Le projet MatchingToolBox :
Contient que le « main » qui pourrait être exécuté en mode console et qui fait appelle au package Matching Library

Le projet MatchingToolBoxTest :
Ce projet est consacré pour les tests unitaires effectués à l'aide de framework google test.

5.1 Bugs existant dans le code

Dans cette section on va aborder les différents bugs détectés puis corriger dont on va parler dans la prochaine section :

- Au niveau de la classe UCRparser : parce que le premier nombre de la ligne d'une séquence
- Manque des tests unitaires pour toutes les classes
- Au niveau de la classe ParamParser : renvoi un vecteur de paramètre même si elle parse un Fichier
- Au niveau de la classe Parameter : on ne peut pas rajouter d'autre paramètres à la classe sans modifier le code
- Au niveau de la classe CommandeLineapplication : beaucoup de traitement dans une seule classe

5.2 Modification du code et amélioration proposées

Modification au niveau de la qualité logicielle :

- Nous avons nettoyé le code en supprimant des fonctionnalités qui ne fonctionnaient pas.
- Commenté les fonctions et les classes qui n'étaient pas commentée
- Changer les noms de variables et de classes nom significatifs au traitement qui font
- Réorganisées la structure des packages pour que du code soit plus simple pour les développeurs de l'utiliser comme une bibliothèque.

Modification au niveau implémentation du code :

- Nous avons extrait un fichier de paramètres standard que le programme serait capable d'analyser, qui n'était pas disponible avant, la raison pour laquelle nous n'avons pas pu créer de fichier de paramètres pour tester le code.
- CSVParser était une classe qui analysait les fichiers csv mais avec un certain format, ce format était facile à lire pour de petites séquences mais plus difficile pour de longues séquences donc il n'était plus utilisé.
- La classe EXTParserClassMatching a été utilisée pour analyser les fichiers UCR que nous n'avons pas pu tester en raison de l'indisponibilité de ces fichiers, ce qui nous a amenés à le supprimer.
- La méthode ParamParser : : readFile (filename) a été modifiée pour renvoyer un objet Parameters au lieu d'un vecteur car il n'utilisait que le premier élément de ce vecteur et il n'est pas nécessaire de renvoyer plusieurs objets car la méthode analyse un fichier .
- Nous avons modifié la classe Parameters de manière à ce qu'elle soit utilisée comme dictionnaire de valeur-clé, la raison pour laquelle nous avons fait ceci est de permettre d'insérer d'autres paramètres qui ne sont pas déjà définis.
- La classe CommandLineApplicationClassMatching a été supprimée car elle utilise UCR parser qui n'a pas fonctionné et n'a donc pas été entièrement utilisé dans le programme.

Amélioration ajoutée :

- Nous avons ajouté la possibilité d'intégrer de nouvelles méthodes de correspondence à la classe CommandLineApplication qui n'était pas possible avant sans changer le code de la classe, ceci fonctionne en appelant la méthode de classe CommandLineApplication : addMethod (nom, Correspondance), l'utilisateur peut désormais utiliser la méthode "Nom" pour appeler la méthode matching.
- Nous avons créé la classe MatchingExecutor, elle prend les fichiers de référence et de cible et l'objet Correspondence qui serait utilisé pour faire correspondre les séquences. Cette classe a une méthode appelée execute () qui va lire les fichiers et les correspondances de séquences et enregistre la sortie, il nous a beaucoup aidé pour simplifier le processus de correspondance de deux fichiers.
- Nous avons ajouté la possibilité de pouvoir ajouter d'autre parser à l'aide de l'argument -parser
- Nous avons ajouté la possibilité de pouvoir ajouter d'autre type àl'aide de l'argument -type
- Nous avons ajouté la fonctionnalité de voir les parsers supporté par la librairie

5.3 Les tests unitaires

Nous avons intégré le framework de test de Google pour créer des cas de test, nous avons choisi ce framework au lieu de cppUnit car il permet les tests unitaires de C ++ avec une modification de source minimale et est plus facile à utiliser que l'autre.

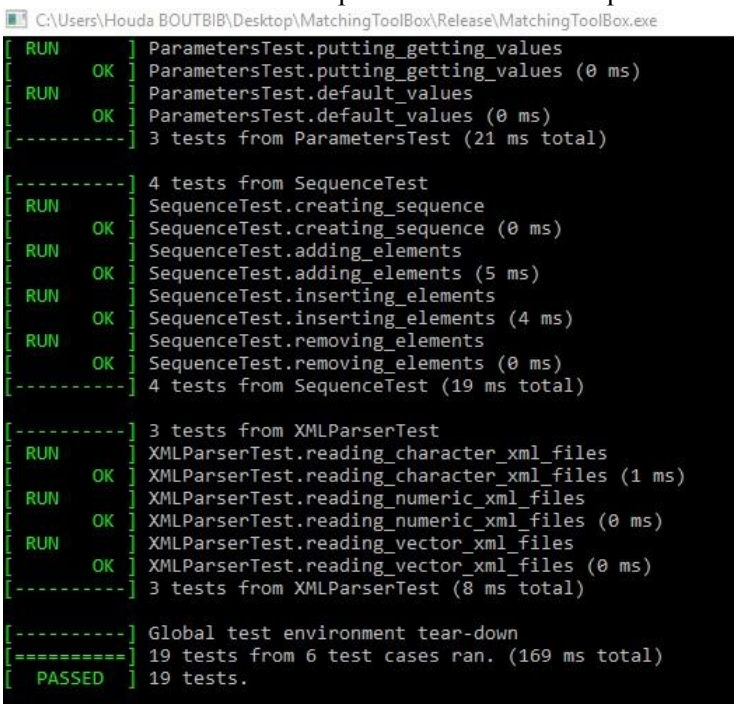

Figure 16 – *teste unitaire*

6 Conclusion

Le départ été assez difficile. Ayant à reprendre de l'existant tout en tentant de comprendre les différents algorithmes utilisés, il nous a fallu du temps avant de pouvoir produire la moindre ligne de code et cela fut assez déstabilisant. Finalement, nous pensons avoir contribué de manière positive à la Matching Tool Box. Depuis les améliorations du code et des améliorations de la modélisation nous espérons avoir répondu aux attentes qui nous ont été formulées. En réalité, et malgré un démarrage compliqué, ce projet a été pour nous intéressant, contrairement à notre sentiment initial. L'aspect analytique et critique dans ce projet est diffèrent de ce que nous avons l'habitude de réaliser.

Glossaire

- CDP : Continuous Dynamic Programming
- FSM : Flexible Sequence Matching
- ESC : Exemplary Sequence Cardinality
- LCS : Longest Common Subsequence
- DTW : Dynamic Time Warping
- MVM : Minimum Variance Matching

Bibliographie

[1] Auteur :*Bastien MEUNIER,Recherche par mots-clés "image" dans une masse d'image de documents : méthode*,adresse :bastien.meunier@gmail.fr année :2014 - 2015

[2] Auteur :*Antoine Fonfria and Nicolas Gougeon ,science de la décision*,année 2014

Diffusion open source d'une bibliothèque en c++, pour la mise en correspondance de séquences

Résumé
Ce projet rentre dans le cadre du module *qualité logiciel* a été pour nous l'occasion de découvrir l'aspect critique et analytique d'un projet qui n'est pas fait par nous. C'est au travers du PFE de Bastien Meunier, en 5ème année à Polytech Tours, que nous Avons apporté notre contribution à un projet plus vaste : la Matching Tool Box. Notre Travail a été d'améliorer la librairie existante en y apportant des options Supplémentaires pour assurer l'optimisation et l'amélioration de cette. Dans ce rapport, nous Détaillerons les phases qu'on a réalisé tout au long de la durée du projet ainsi que les solutions et les nouvelles idées apporter afin d'améliorer la librairie.
Mots-clés
Génie logiciel ,Correspondance entre séquences

Abstract
This project comes within the framework of the module textit qualité logiciel was for us the opportunity to discover the critical and analytical aspect of a project that is not done by us. It is through the PFE of Bastien Meunier, in 5th year at Polytech Tours, that we We have contributed to a larger project: the Matching Tool Box. Our work has been to improve the existing library with options Additional to ensure optimization and improvement of this. In this report we will detail the phases that have been realized throughout the duration of the project as well as the solutions and new ideas to improve the library
Keywords
génie logiciel ,matching between sequences

Tuteur académique
Nicolas Ragot

Étudiants
Houda boutbib (DI4)
Mohammed elmoutaraji (DI4)