Etude Empirique Test de Robustesse

Master 2 ILA – ISTIC – Rennes 1

Abdallah LACHAAL – Mounir BOUYAMINE

Introduction

Dans le cadre du Master 2 ILA, et sous l'encadrement de Monsieur Mathieu Acher, nous avons eu l'opportunité de réaliser un projet d'ingénierie dirigée par les modèles. Ce projet a été effectué après avoir assimilé la partie théorique du cours et également la partie pratique durant les séances de TP.

En outre, le résultat de notre projet devait être déployé sur une interface web qu'il a fallu développer, afin de donner à l'utilisateur la possibilité de tester les différentes possibilités de l'outil. Pour avoir une idée de cette interface, on s'est inspiré du site de la chaîne CANAL+.

Mais le défi le plus intéressant de ce projet est de mettre à l'épreuve notre générateur en le testant sur un certain nombre de vidéos et de fichiers de spécifications afin d'avoir une idée sur la robustesse de notre programme. Les vidéos qui seront utilisées pour ce test proviendront des autres membres de la promotion du Master ILA, ces vidéos seront accompagnés par les fichiers de spécification «. videogen » grâce auxquels notre générateur va pouvoir générer des playlists qui décriront la structure des vidéos générées. Nous expliquerons dans les paragraphes suivants la méthode utilisée pour effectuer nos tests, par la suite, nous exposerons des résultats.

Outils utilisés :

Java : Le langage de programmation orienté objet qu'on a utilisé pour réaliser le projet

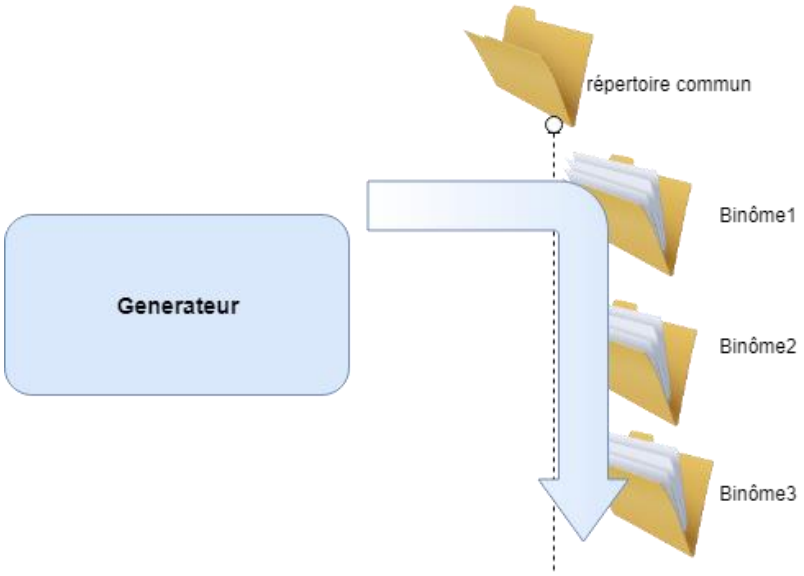
FFMPEG : une collection de logiciels libres destinés au traitement de flux audio ou vidéo. Cette bibliothèque est utilisée par de nombreux autres logiciels ou services comme VLC, iTunes ou YouTube. Elle peut être compilée sur la plupart des systèmes d'exploitation, y compris Windows.

Xtext : un outil libre pour décrire des langages de programmation et des langages dédiés

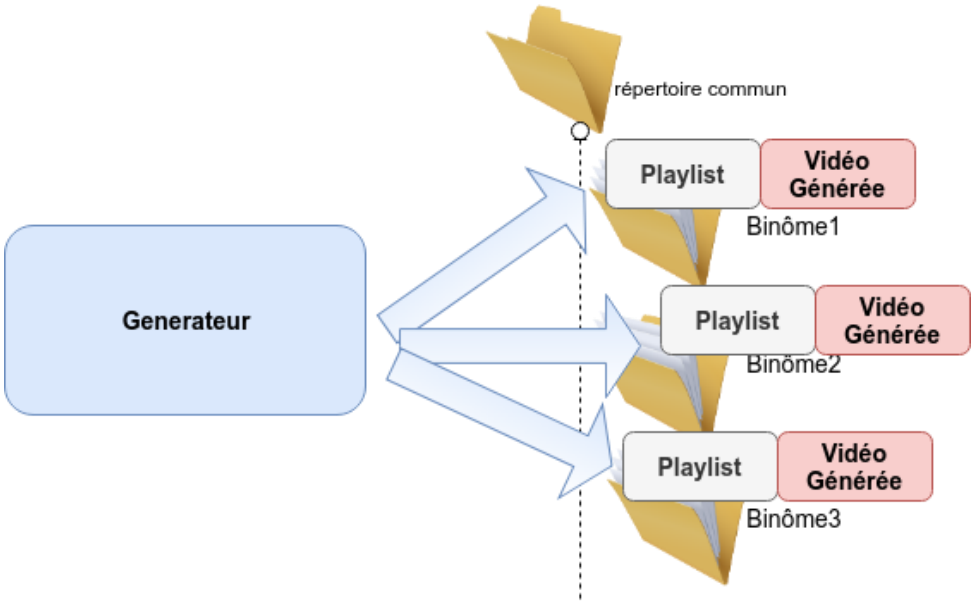
Xtend : un langage de programmation orienté objet pour la machine virtuelle Java. Il se concentre sur une syntaxe plus concise et quelques fonctionnalités supplémentaires telles que l'inférence de type, les méthodes d'extension et la surcharge des opérateurs. Il est compilé en code Java et s'intègre donc parfaitement à toutes les bibliothèques Java existantes.

Méthode

Pour cette étude, nous avons ajouté à notre générateur une fonctionnalité qui permet de parcourir l'ensemble de tous les dossiers (chaque dossier correspondant à un binôme de la promotion) dans lesquels il va récupérer le fichier de spécification « .videogen » ainsi que les échantillons de séquences vidéo qui seront concaténés selon la structure décrite dans le fichier de spécification. La figure suivante illustre cette étape :



Par la suite, si tout se passe comme prévu, une vidéo générée sera présente dans chaque dossier de binôme ainsi qu'un fichier txt contenant la playlist générée comme le montre la figure ci-après.



Nous avons constaté trois niveaux de résultats :

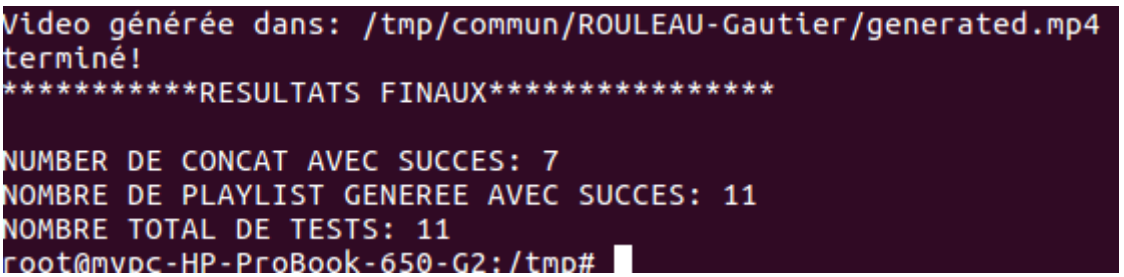
- Le premier niveau c'est le succès complet, à savoir, la génération de la playlist plus la génération de la vidéo résultante de la concaténation en respectant la structure décrite dans le fichier de spécification.
- Le second niveau c'est le demi succès (ou demi échec) à savoir que la génération de la playlist à bien été effectuée mais la vidéo finale non générée dût à des erreurs d'exécution coté FFMPEG.
- Le troisième niveau qui est tout simplement un échec total, c'est-à-dire, pas de génération de playlist ni de vidéo.

Résultat :

Tout d'abords nous vérifions que notre générateur à bien parcouru l'ensemble des répertoires correspondant à chaque binôme, on peut récupérer cette information à partir les logs comme le montre la figure suivante :

On constatera que notre binôme ne figure pas sur la liste car on a jugé inutile de refaire les tests sur notre propre jeu de données.

À la fin de l'exécution, notre programme affiche un bilan des tests :



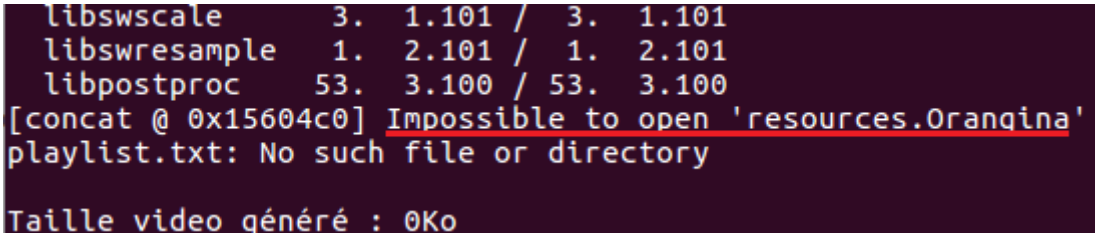
On a donc le nombre de concaténation avec succès (ici 7) qui correspondent au nombre de binôme avec lesquels la concaténation s'est bien faite (et donc à fortiori la génération de la playlist) ce qui correspond au premier niveau de résultat (cf paragraphe précédent).

On a également le nombre de playlist générée avec succès (ici 11) ce qui implique que le nombre de tests à demi succès sont en nombre de $11 - 7 = 4$.

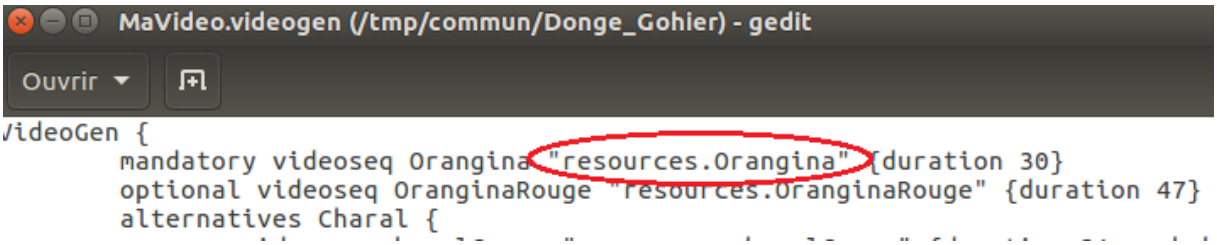
Enfin on a le nombre total de test effectués qui correspond évidemment au nombre de binômes.

En regardant de plus près l'origine des problèmes, nous avons constaté que les informations précisées dans les fichiers de spécification sont erronés.

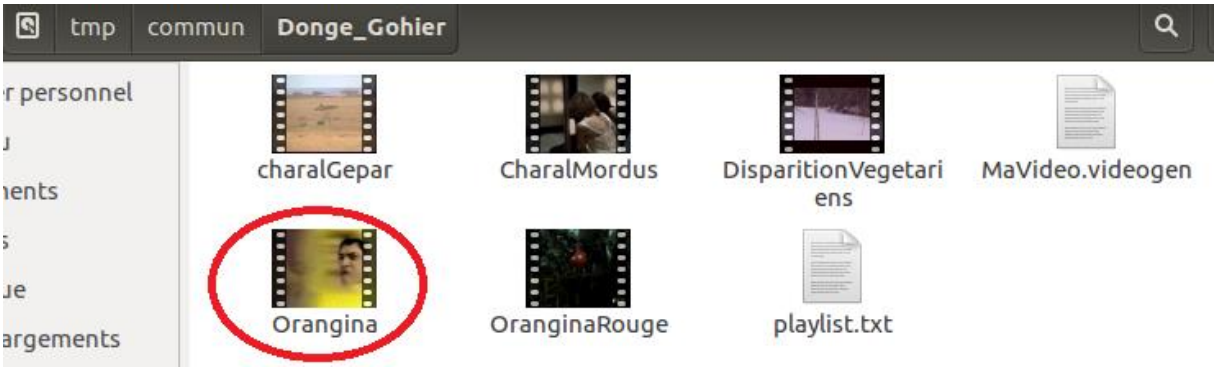
Nous repérons dans les logs l'origine du problème comme le montre la figure suivante :



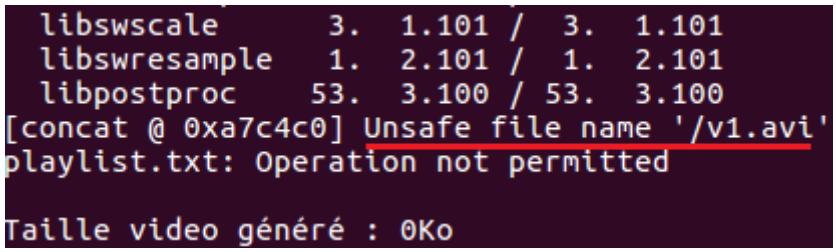
Ici on voit que le choix de nommage des fichiers de séquence vidéo est mauvais, voici un extrait du fichier de spécification correspondant :



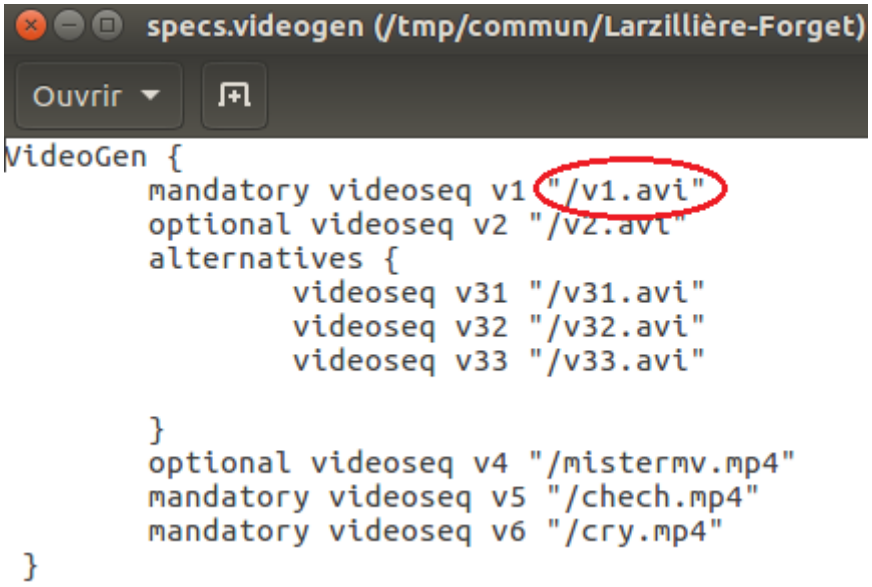
On constate une discordance entre les noms qui se trouvent dans les spécifications et les noms de fichiers dans le dossier dont on montre le contenu :



Un autre exemple sont les erreurs qui ont comme origine l'utilisation des « slash » dans les fichiers de spécifications, voici une trace de cette erreur dans les logs :



Et voici un extrait du fichier de spécification ainsi que la playlist généré :



playlist.txt (/tmp/commun/Larzillière-Forget)

Ouvrir

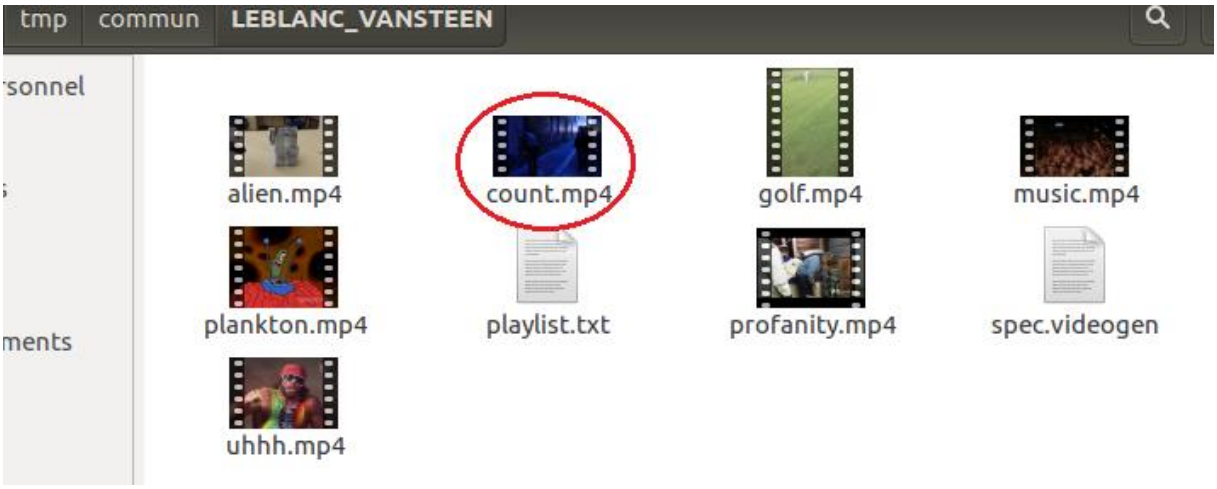
```
file '/v1.avi'
file '/v31.avi'
file '/mistermv.mp4'
file '/chech.mp4'
file '/cry.mp4'
```

Enfin un troisième exemple d'erreur est dut au fait que le fichier de spécification fait référence à un dossier inexistant sensé se trouver dans le répertoire du binôme concerné. Voici un extrait du log signalant l'erreur :

```
  libswscale      3.  1.101 /  3.  1.101
  libswresample   1.  2.101 /  1.  2.101
  libpostproc    53.  3.100 / 53.  3.100
[concat @ 0xa334c0] Impossible to open 'videos/count.mp4'
playlist.txt: No such file or directory

Taille video généré : 0Ko
```

Et voici une capture du contenu du répertoire du binôme concerné ainsi que l'extrait de leur fichier de spécification :



spec.videogen (/tmp/commun/LEBLANC_VANSTEEN) - gedit

Ouvrir

```
@author "Adrien et Damien"
@version "1.0"
@creation "10/02/2019"
VideoGen {
        mandatory videoseq v01 "videos/count.mp4"
        optional videoseq v02 "videos/profanity.mp4" {probabilit
        alternatives v3 {
                videoseq v31 "videos/alien.mp4" {probability=15}
                videoseq v32 "videos/golf.mp4" {probability=85}
        }
        optional videoseq v043 "videos/music.mp4" {probability=6
```

Le 4ième binôme présente un cas d'erreur similaire.

Résultat après les corrections :

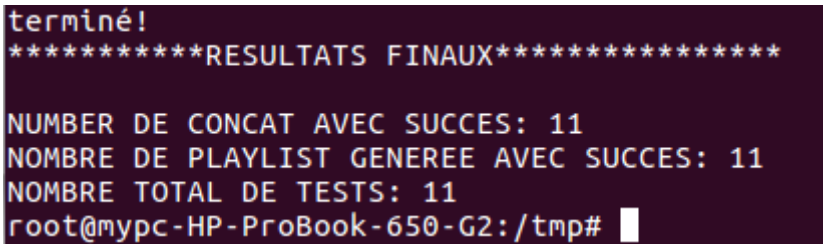
La curiosité nous à poussé à retenter un test après corrections des différents fichiers de spécifications présenté précédemment :

Pour le premier cas on remplace `resources.Orangina` par `Orangina` (on fait de même pour le reste des vidéos)

Pour le deuxième cas on remplace `/v1.mp4` par `v1.mp4`

Pour le troisième cas on remplace `videos/count.mp4` par `count.mp4`

Les résultats finaux ont été concluants puisqu'on a atteint un taux de réussite de 100% :



Conclusion :

Notre générateur est plutôt robuste car les seuls échecs qu'on a eu sont dût au fait de mauvaises écritures dans les fichiers de spécifications, il a juste fallu corriger ces erreurs pour aboutir à un tôt de réussite de 100%.

En ce qui concerne les acquis de ce projet, nous avons eu la possibilité de découvrir de près l'ingénierie dirigée par les modèles, ce qui nous motive encore plus à réaliser d'autres projets pour la maitriser encore plus. En outre, à travers la réalisation de l'interface, nous avons pu confirmer nos connaissances en Angular et les Web Services.

En revanche, l'une des difficultés qu'on a rencontrées pendant le développement de notre générateur est liée principalement à FFMPEG. En effet, la connexion entre Java et FFMPEG n'était pas simple au premier abord, il a fallu ajouter à notre programme plusieurs paramètres supplémentaires comme par exemple préciser le « working directory » c'est-à-dire le dossier sur lequel java doit faire exécuter FFMPEG. Ces difficultés ont heureusement été surmontées.