Ecole Publique d'Ingénieurs en 3 ans

Rapport à mi-parcours

PROJET AUTOMIX SOFTWARE

Le 21 janvier 2018,

Niels HECQUARD,
Chef de projet
Raphaël HERBERT, Cédric MASSON,
Richard NICOLAS, Anaëlle TROADEC,
Développeurs

Tuteur école : **Baptiste HEMERY**
Client : **Maxime STEVENOT**

ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

www.ensicaen.fr

TABLE DES MATIERES

INTRODUCTION

La version 2017-2018 du projet Automix Software est la reprise de l'application imaginée et développée l'année précédente par le groupe de Maxime Stevenot. L'idée de ce logiciel est de permettre à un public le plus vaste possible de créer des mix musicaux intelligents et personnalisés très facilement.

Ainsi, l'utilisateur sélectionnera des musiques de sa propre bibliothèque, et le logiciel s'occupera de générer un mix. Pour ce faire, il commencera par analyser les musiques fournies en récupérant un certain nombre de caractéristiques. L'utilisateur aura alors la possibilité de trier les musiques. Ce tri se base sur les caractéristiques obtenues afin d'obtenir un mix harmonieux. Enfin, la génération du mix utilisera une intelligence artificielle pour établir des transitions agréables à écouter entre les musiques, à la manière d'un DJ. Pour un résultat encore plus personnalisé, l'utilisateur pourra paramétrer le logiciel. Parmi les paramètres réglables, on retrouve entre autre l'ordre des musiques, la durée du mix, la durée des transitions, la priorité des caractéristiques des musiques prises en compte pour le tri et les transitions…

La première version du logiciel développée par l'équipe précédente permet d'importer des musiques, de les trier et de générer un mix simple qui concatène toutes les musiques choisies par l'utilisateur et applique une transition en fondu entre chaque musique. Notre travail consiste donc principalement à ajouter les fonctionnalités de paramétrage du mix et à améliorer la qualité des transitions et les performances de l'application.

Nous allons voir dans un premier temps le contexte qui a fait naître l'idée de ce projet ainsi que l'état actuel des travaux. Puis nous nous attarderons sur les méthodes et outils utilisés pour mener à bien ce projet. Enfin, nous détaillerons les objectifs atteints jusqu'à maintenant et ceux à venir.

1. Présentation du projet

1.1. Présentation générale du projet et du contexte

L'idée du projet est partie du constat qu'il existe aujourd'hui différentes façons d'écouter de la musique en continu, mais qu'aucune ne rassemble toutes les qualités qu'un amateur de musique attend. Le premier média auquel on peut penser est la radio qui est facilement accessible et gratuite mais on ne choisit pas la musique et il y a des publicités. On peut également profiter de playlists créées par des amateurs, des labels ou générées automatiquement par des plateformes de streaming. Ce moyen est probablement celui qui sera le plus proche des goûts de l'utilisateur puisque, avec un peu de recherche, il peut trouver des playlists avec ses musiques favorites. Toutefois, bien souvent il n'y a pas d'effort de fait sur les transitions qui sont assez brutes. La 3ème solution est de faire appel à un DJ. Celui-ci pourra s'adapter aux goûts du client et devrait offrir un mix avec des transitions de qualité. Cette option n'est bien sûr pas adaptée au quotidien mais à des évènements particuliers qui nécessitent un investissement non négligeable pour faire venir un DJ et avoir le matériel adéquat.

Le but d'Automix Software est de réunir tous les avantages cités précédemment en une seule solution, utilisable via une interface simple et intuitive, accessible à tout utilisateur sans compétence informatique ou musicale.

1.2. Objectifs

L'objectif final de ce projet est de proposer un logiciel de mix plus performant, avec de meilleures transitions, un tri des musiques de qualité et contenant plus de fonctionnalités pour l'utilisateur et le développeur, notamment pour le paramétrage du mix (configurer l'ordre des musiques, la durée du mix, la durée des transitions, la priorité des caractéristiques des musiques prises en compte pour le tri et les transitions…). Il faut également que le logiciel soit portable sous MacOS et Linux, si la portabilité est possible, afin de toucher plus d'utilisateurs.

Pour nos objectifs à mi-parcours, il nous fallait étudier la portabilité sous MacOS/Linux du logiciel (étant développé sous Windows) afin de savoir si cela était possible et d'avoir une idée de la méthode à adopter pour ce faire. Nous nous sommes également fixés d'ajouter de nouvelles fonctionnalités pour l'utilisateur (concernant la base de données avec la liste des musiques analysées, le dossier temporaire où mettre les musiques ou le paramétrage du mix) et le développeur (concernant l'accès à l'historique des événements pour analyser les erreurs plus facilement). Nous devons également améliorer le tri des musiques, afin d'assurer un enchaînement logique, satisfaisant et agréable des musiques dans le mix. Une partie des bugs devait être traitée. Le site internet doit être réactualisé et fournir une nouvelle version du logiciel tous les 15 jours. De même la notice utilisateur et développeur doit être mise à jour.

2. Réalisation du projet

2.1. Méthodologie agile

Il nous a été préconisé par le client de travailler en méthodologie agile. Ainsi, nous fonctionnons en sprints (itérations) de 2 semaines de travail, durant lesquels nous nous fixons des stories à réaliser (ie des tâches décrivant des fonctionnalités à développer). Ces sprints sont intercalés par des réunions où nous présentons et faisons un point avec le client et/ou le tuteur sur le travail réalisé, les problèmes qui peuvent être rencontrés ainsi que sur ce qui doit être effectué dans le sprint suivant. Afin de mieux quantifier la charge de travail, les stories sont notées lors de scrum session avec un système de points définis (1, 2, 3, 5, 8, 13 ou 20). Chaque story est analysée : un premier vote sur la difficulté de la story est fait, puis il y a une phase de justification et discussion, puis un 2ème vote est réalisé et on affecte les points à la story (à la majorité des votes). Grâce à ces points nous pouvons définir la courbe burn-up contenant 4 courbes : une courbe minimum avec les fonctionnalités indispensables, une courbe cible à laquelle on ajoute d'autres fonctionnalités importantes, une courbe idéale avec toutes les autres stories et la courbe réelle correspondant à l'évolution du projet.

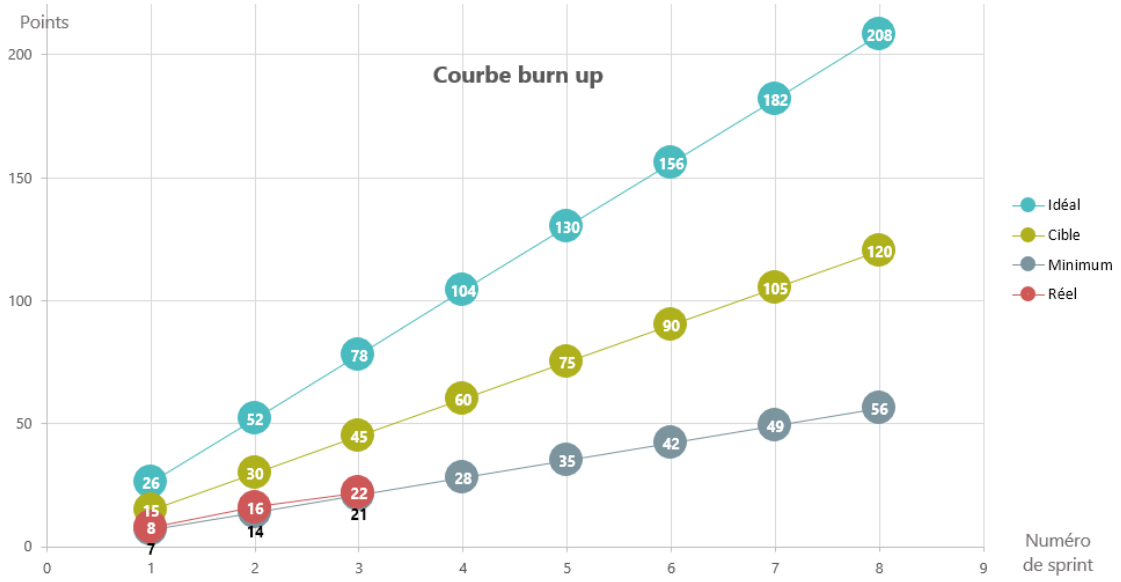

Figure 1: Courbe burn-up après le sprint 3

2.2. Gestionnaire de système de version

Afin d'avoir une évolution du projet en parallèle pour tous les membres, le gestionnaire de système de version git est utilisé sous un serveur GitLab. Il existe deux branches fixes : une branche master avec une version stable du logiciel et une branche develop avec une version actuelle du logiciel. Pour chaque fonctionnalité, une branche feature est créée et ajoutée au projet lorsque celle-ci est fonctionnelle.

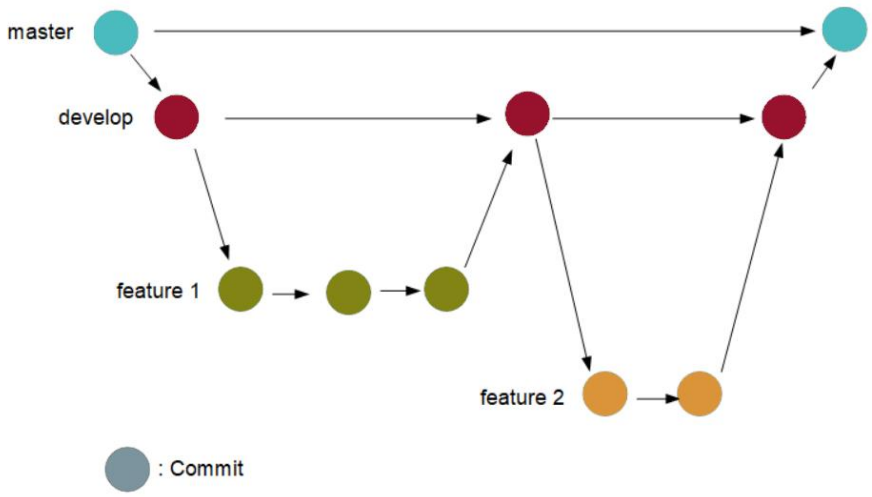

Figure 2: Fonctionnement de la gestion de version avec git

2.3. Présentation des outils utilisés

- Afin de faciliter l'utilisation de git sous Windows, nous utilisons le logiciel Sourcetree qui permet une visualisation des branches et un basculement rapide sur n'importe quel commit effectué. Il est utilisé en parallèle du GitLab de l'école où le code de l'application est stocké.

- Pour coder, nous utilisons l'IDE Visual Studio qui est optimisé pour le développement en C# et son framework Microsoft.NET. Il nous permet de construire l'interface facilement et offre de nombreuses options pour obtenir de l'aide sur les fonctions et naviguer dans le code.

- Enfin, pour une bonne visibilité des tâches et de leur avancement nous utilisons un Trello. Les tâches sont organisées en plusieurs catégories : backlog général (stories non faites), backlog du sprint courant (stories à faire pour le sprint), tâches en cours de développement, tâches faites et tâches validées par le client/tuteur.

2.4. Présentation des sprints et des objectifs réalisés

A mi-parcours de ce projet, les sprints terminés sont au nombre de 3. Le sprint courant à l'heure où ce rapport est écrit est donc le 4ème. Dans cette partie sera donc présenté uniquement le travail qui a été validé lors des réunions de fin de sprints ainsi que des remarques d'ordre général. Les tâches qui ont été réalisées lors du 4ème sprint n'étant pas encore validées, on considérera qu'il s'agit de travaux pour la suite du projet (traité dans la partie perspective du rapport).

→ Sprint 1 (10 au 23 novembre 2017) :

[8 points] Utilisateur - peut choisir son dossier temporaire : Le dossier temporaire du logiciel se trouvait dans le dossier temporaire de windows par défaut. A l'issue de la réalisation de cette tâche, l'utilisateur peut donc changer ce dossier en sachant que le logiciel se charge de vérifier l'existence du dossier ou de sa création. Un travail sur l'interface graphique était donc implicitement de mise afin que l'utilisateur puisse appeler cette fonctionnalité dans les onglets présents en haut du logiciel.

Plusieurs autres tâches étaient déjà en cours de réalisation à la fin du sprint 1 mais ont été reportées puis validées lors des sprints suivants. Le sprint 1 a été en grande partie de la découverte du code existant et de la prise en main des outils de développement pour l'équipe.

→ Sprint 2 (23 novembre au 7 décembre 2017) :

[8 points] Utilisateur/Développeur - peut consulter un fichier de log : La réalisation de cette tâche devait permettre l'écriture automatique dans un fichier texte de l'historique des événements lors de l'exécution dans un but de maintenance du logiciel. L'outil recommandé était le paquet "log4net", dont il a fallu adapter l'utilisation au projet au travers de l'écriture d'un fichier de configuration. Du côté utilisateur, le fichier de log répertorie les messages importants qu'il est utile de récupérer lors d'une demande de support par exemple. Du côté développeur, le logger a plutôt un but d'aide au développement.

La bonne installation de l'outil de logging ayant été validée en réunion, la fin de cette tâche impliquait bien sûr un travail sur le long terme : logger au fûr et à mesure les nouvelles lignes de codes qui allaient être écrites lors de la suite du développement.

→ Sprint 3 (7 au 19 décembre 2017) :

[3 points] Améliorer la gestion des exceptions : Pour cette story, le travail consistait à trouver les différentes parties du code qui seraient susceptible de conduire à une exception pour la faire remonter et la gérer sans que l'application ne se crashe. Il fallait également pouvoir avoir une trace de l'état de l'application lors de la levée de ces exceptions afin de pouvoir mieux comprendre pourquoi ces exceptions étaient générées dans certains cas à l'avenir. Il se trouve que l'outil de log possède justement une fonctionnalité qui permettait de tracer les exceptions.

[3 points] Utilisateur - peut importer / exporter la base de données : Cette fonctionnalité permet à l'utilisateur d'ajouter du contenu à la base de données existante ou de l'exporter dans le but de partager des bases entre utilisateurs. Nous rappelons qu'une base de données est présente dans le logiciel afin de stocker des informations nécessaires à la génération du mix qui sont déterminées lors de l'analyse. L'analyse étant assez longue, stocker les

informations résultantes de celle-ci permet d'éviter d'avoir à analyser à chaque utilisation. L'import/export permet donc à un utilisateur de supprimer le temps de l'analyse déjà effectuée par d'autres membres de la communauté d'Automix Software.

2.5. Répartition des tâches

Les différents sprints et les objectifs réalisés ont nécessité une mise en place en amont, au début de chaque sprint, pour décider de quel développeur effectuerait une tâche précise. Bien que les tâches effectuées soient de natures variées (résolution de bugs, ajout d'une nouvelle fonctionnalité, etc.), l'affectation des tâches et leur réalisation suivent toujours le schéma suivant :

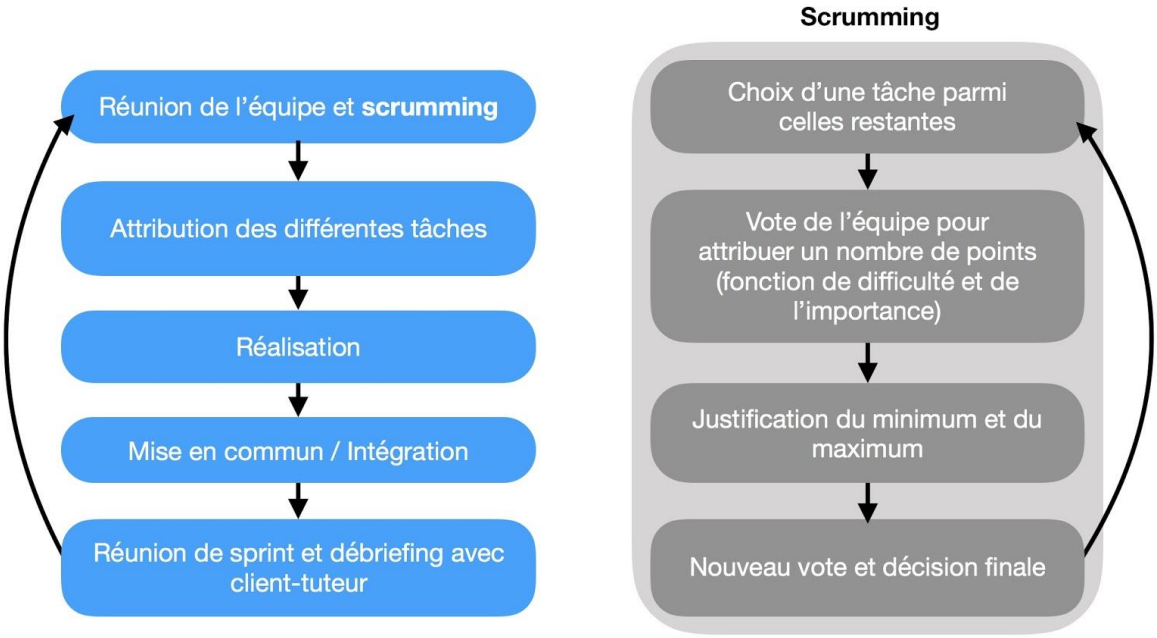

Figure 3: Processus de définition et d'affectation des tâches

Une fois les tâches décidées, chaque développeur s'acquitte d'une des tâches dont le nombre de points est défini. Dans la méthodologie agile, chaque développeur étant interchangeable, chacun peut alors choisir la tâche qui lui plaît/parle le plus. Cela permet de prévoir les imprévus, comme par exemple la méconnaissance d'un développeur sur un outil, une connaissance particulière, ou son absence. En fin de sprint, le travail de chaque développeur est mis en commun dans la branche « develop », puis si une nouvelle version stable est obtenue, on met le travail effectué dans la branche « master », qui sera alors la nouvelle version téléchargeable sur le site internet.

2.6. Présentation des sprints et des objectifs réalisés

Concernant les objectifs fixés pour le mi-parcours mais non atteints, nous n'avons pas mis à jour une nouvelle version du logiciel tous les 15 jours. En effet certaines tâches ont pris plus de temps à se développer, suite à des mauvaises compréhensions sur ce qu'il fallait réaliser

ou suite à des difficultés à appréhender le code, n'amenant ainsi pas assez d'évolution pour mettre à jour l'ancienne version du logiciel. De plus nous avons découvert de nouveaux bugs qui nous ont alors ralentit et ne nous permettaient pas alors de fournir une nouvelle version du logiciel propre et sans erreurs. Cela nous a ainsi beaucoup retardé sur l'avancement du projet. L'amélioration du tri est en cours de développement, mais pas finie, suite au retard expliqué précédemment et à la charge de travail plus conséquente en fin de semestre. De même que l'étude de portabilité sous MacOS/Linux, on sait désormais qu'il est possible de développer le logiciel sous ces systèmes d'exploitation, mais la manière pour le faire n'est pas encore complètement définie. Ainsi, le retard pris par le report de certaines tâches a eu un impact non négligeable sur l'aboutissement de tous les objectifs que nous nous étions fixés.

3. Développement du projet et organisation

3.1. Perspective

- Objectifs à court terme

Les objectifs de l'équipe à court terme pour la suite du projet sont bien établis. Il s'agit de réaliser les tâches qui ont été mises dans le backlog du sprint courant lors de la dernière réunion. Les tâches en cours sont donc décrites ci-après:

[13 pts] Utilisateur - peut verrouiller la position d'une musique : Avant de générer un mix, une opération de tri sur la liste courante des musiques est disponible afin de proposer un ordre plus harmonieux dans l'enchaînement des musiques. Cette fonctionnalité offre la possibilité à l'utilisateur de forcer une musique à être en première, deuxième, n-ième position mais nécessite d'implémenter une variation dans l'algorithme de tri. Cette tâche implique également le développement d'un moyen ergonomique pour l'utilisateur d'indiquer ce verrouillage sur l'interface du logiciel.

[13 pts] Utilisateur - peut bénéficier d'un tri des musiques d'excellente qualité : Lorsque l'utilisateur cherche à trier ses musiques, il peut arriver que l'ordre change jusqu'à ce que l'on ait appelé suffisamment de fois l'algorithme de recuit-simulé pour arriver à un ordonnancement stable et optimal. Le travail à réaliser consiste donc à trouver une manière d'améliorer la procédure de tri afin que deux appels consécutifs donnent le même résultat et qu'il soit optimal dès le premier appel.

[8 pts] Annuler l'analyse ne nettoie pas la liste des musiques : Le travail de cette story consiste en la correction d'un bug qui intervient au niveau de l'interface graphique du logiciel. Les musiques non analysées après une annulation doivent être automatiquement enlevées de l'interface.

[3 pts] Etude préliminaire pour le passage sur macOS / Linux : Le but de cette story est de répondre par oui ou par non à la question suivante "Le logiciel peut il être porté sur macOS /

Linux ?" et d'expliquer pourquoi. Et si oui quels risques et moyens seront liés à la réalisation du portage afin de le préparer au mieux.

- Objectifs à long terme (finaux)

A l'issue de l'étude préliminaire pour le portage sur macOS/Linux, l'un des objectifs principaux sera de réaliser ce portage afin de respecter les livrables établis du projet car une version du logiciel pour les utilisateurs de ces deux systèmes d'exploitation était promise lors du kick-off.

Un autre des objectifs est de se pencher sur les problèmes liés à l'algorithme de transition entre les musiques. Maintenant que le code est relativement bien pris en main, aborder les tâches liées aux transitions en général serait l'occasion d'apporter une plus-value supplémentaire non négligeable au travail de l'équipe.

Enfin l'ajout de filtres et d'effets serait également des fonctionnalités importantes à implémenter pour cette deuxième moitié de parcours car elles permettraient de rendre le logiciel plus paramétrable et ainsi offrir une expérience utilisateur plus agréable.

3.2. Sentiment de l'équipe

Tout d'abord, un des aspects positifs du projet pour l'ensemble de l'équipe a été de se confronter à un sujet et des méthodes de gestion de projet proches du milieu professionnel. Le développement en mode agile, la reprise d'un projet en cours et la mise en place de deadlines sont des points récurrents du travail en entreprise et même si ce dernier point a été une source de stress pour l'ensemble des membres de l'équipe, nous en retirons une bonne expérience pour le moment. De plus, l'agilité nous a paru une gestion de projet prometteuse notamment parce qu'elle nous a permis un contact fréquent avec le client, nécessaire dans le cas de tâches mal définies. Ensuite, ce projet a été également l'occasion d'approfondir nos connaissances. Bien que reprendre le code existant fut dans certains cas difficile et que notre connaissance en C# au commencement de ce projet était minime, nous avons pu profiter de celui-ci pour consolider notre bagage technique et ainsi prendre conscience de ce qu'implique plus généralement le métier d'ingénieur.

CONCLUSION

Cette première phase de développement a été l'occasion, dans un premier temps, de nous familiariser avec l'architecture du projet (qui utilise le patron d'architecture Modèle Vue Présentation), le langage C# et son framework .NET qui étaient nouveaux pour nous, l'IDE Visual Studio, la méthodologie de projet Agile et enfin une utilisation rigoureuse de git. Toutes ces tâches ont pris du temps et ont réduit notre productivité pour ajouter de nouvelles fonctionnalités au logiciel mais étaient nécessaires pour appréhender sereinement la poursuite du projet. Les premières stories que nous avons eu à développer étaient d'ailleurs prévues pour que nous parcourions le code et que nous appréhendions les différents aspects du développement du projet : les ajouts sur l'interface avec l'outil proposé par Visual Studio, les classes du modèle, représentant la logique métier, et de la présentation, les différents algorithmes et intelligences artificielles utilisés pour réaliser les actions les plus complexes.

Aujourd'hui, nous pouvons dire que nous avons une vision claire du fonctionnement de l'application ainsi que de son évolution puisque nous avons terminé d'évaluer toutes les stories qui constitueraient le produit parfait. De ce fait, nous serons en mesure de mener à bien plus efficacement les tâches que nous aurons à réaliser dans les sprints à venir et ajouter une réelle plus-value à ce projet.

ENSI CAEN

ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

Ecole Publique d'Ingénieurs en 3 ans

6 boulevard Maréchal Juin, CS 45053
14050 CAEN cedex 04

ministère éducation nationale enseignement supérieur recherche

Normandie Université

Cti Commission des titres d'ingénieur

cdefi Conférence des Directeurs des Écoles Françaises d'Ingénieurs

European Accreditation of Engineering Programmes EUR-ACE®

CONFÉRENCE DES GRANDES ÉCOLES