Université de Rennes 1

Rapport de stage de Master

OpenSource et Modularité à l'INSEE : Exemple avec l'implémentation d'une méthode d'échantillonage.

Etudiant :
Antoine Brunetti

Tuteur de stage :
Abdou Diaw

Encadrant :
Rémi Pépin

15 novembre 2020

Remerciements

Je tenais à remercier toutes les personnes du service national de développement informatique d'Orléans pour l'accueil au sein de la structure et pour les nombreux échanges que l'on a au quotidien avec l'équipe.

Je tenais à remercier personnellement Olivier Levitt pour sa détermination et tous les précieux conseils qu'il a pu m'apporter au cours de ma formation et pendant ce stage. Mon intérêt pour le sujet de l'open source n'étant probablement pas indépendant des discussions que l'on a pu avoir à ce sujet.

Je remercie tout particulièrement Abdou Diaw pour tout l'encadrement et tout le soutien au cours de ce stage qui a ainsi pu se dérouler dans les meilleures conditions grâce à ses conseils et à son aide.

Résumé

Le stage se présente comme une étude de bonnes pratiques du passage à l'open source pour un projet classique INSEE et un nouveau projet, l'application Nautile et un projet d'intégration d'une méthode du cube développée dans le cadre du stage.

En effet, lors de la mise en place d'un projet opensource, les risques sur le code source sont multiples, il faut alors prendre des dispositions pour les éviter. Il est également possible de développer afin de minimiser l'adhérence aux environnements et donc préparer, par la qualité logicielle, un passage à l'open source.

Enfin la mise en place d'un projet open source offre une organisation au niveau projet, certaines propositions pour réduire les coûts d'entrées sont en effet des standards dans le monde open source. Un travail de maquettage en réponse à un besoin métier sur les tirages d'échantillons d'unités primaires dans les territoires de la France d'outremer a pu être réalisé selon les concepts présentés.

Introduction

A l'heure des modèles contributifs libres et de l'open data, il peut sembler pertinent de s'intéresser à la mise à disposition des outils en développement à l'INSEE pour d'autres usages. Et cela pour plusieurs raisons, tout d'abord plus d'utilisateurs implique une meilleure qualité du produit, puisqu'il est potentiellement plus soumis à des usages dont la mutabilité entraîne donc plus de visibilité sur les bogues existants. Ensuite parce que le partage d'outils est également un gage de meilleure correspondance au besoin exprimé. Il peut en effet arriver que lors de projets, l'erreur étant humaine, l'on constate a posteriori des divergences entre les spécifications fonctionnelles et l'existant technique par exemple. On pourrait également penser aux enjeux de conformité avec l'informatique contemporaine et la mise en concurrence des développeurs avec l'extérieur, ce qui permettrait effectivement de valoriser le travail des développeurs au sein de la structure interne et de tendre a une amélioration générale du niveau des services informatiques. De fait, est considéré open source toute solution logicielle pour laquelle le code source est soumis a une licence[1] qui permet l'échange, la modification, la libre distribution et l'étude du code source, dans un environnement ouvert décentralisé. Il devient donc logique d'associer les aspirations a l'ouverture du code avec des projets offrant une collaboration. Il peut par exemple s'agir, dans des cadres plus restreints, à la formation de projets européens communs à plusieurs organisations, comme l'est le projet Accueil-Réception-Contrôle des fichiers administratifs (ARC) au niveau de l'ESSnet, projet qui s'est étendu lors d'échanges avec l'organisme italien public ISTAT.

Mais la question de l'open source ne se limite pas seulement aux sujets d'un partage des outils vers un public plus étendu. L'économie sur les coûts de développement d'une solution déjà existante s'inscrit également dans ce vaste sujet. On a pu voir émerger en 2018 la plate-forme Onyxia au sein de la cellule innovation de l'INSEE (DIIT), par l'intégration et la configuration de solutions open source permettant une réponse à de nombreux besoins de différents utilisateurs, développeurs, intégrateurs, statisticiens ou communicants en compléments des outils existants.

En cela, les méthodologies statistiques fournies par l'INSEE sont déjà partagées à une grande échelle et permettent d'assurer une image à l'institut. Le partage de l'implémentation des méthodes et également des composants logiciels en serait une consécration puisque permettant d'assurer une place de l'INSEE à un niveau plus global. Ce stage est une exploration en ce sens, avec l'exemple de l'implémentation d'une méthode de tirage.

Table des matières

1 - Environnement du stage

1.1 Le Service d'accueil

Le stage s'est déroulé au Service National de Développement Informatique de l'Insee d'Orléans (SNDIO), au sein de l'équipe du projet Nautile (Nouvelle Application Utilisée pour le Tirage des Individus et Logements pour les Enquêtes ménages). Nautile permet de créer les bases de sondage, d'effectuer les tirages d'échantillons logements et individus, pour les enquêtes à partir de la source FIDELI (FIchier DEmographique sur les Logements et les Individus) et de les mettre à disposition. Le SNDIO est avec ceux de Paris, Nantes et Lille, l'un des 4 Services Nationaux de Développement Informatique de l'Insee. Rattaché à la Direction Régionale de l'Insee Centre - Val de de Loire et fonctionnellement dépendant de la Direction du Système d'Information de l'Insee (la DSI), il compte une quarantaine d'informaticiens exerçant dans 3 grandes catégories d'activités :

- la maintenance d'applications informatiques qui consiste à assurer le bon fonctionnement d'une vingtaine d'applications utilisées par des agents de l'Insee (corrections de bugs, assistance utilisateurs, réalisation des évolutions demandées par les MOA)

- le développement de projets informatiques : 6 projets sont actuellement en cours de développement au SNDIO

- la démarche Qualité, Expertise et Innovation qui a pour mission la promotion et la diffusion d'actions qualité, d'expertise ou d'innovation (ateliers thématiques, séances de communication, coding dojo, diffusion de la méthodologie et des pratiques de tests, participation à l'innovation, expertises, formations,)

Ces activités sont regroupées au sein de 5 groupes fonctionnels qui ont pour missions d'assurer assure la maintenance et le renouvellement du parc applicatif d'une maîtrise d'ouvrage ou d'un groupe restreint de maîtrises d'ouvrages.

1.2 Le contexte organisationnel

Le stage s'est effectué au Service National de Développement de l'INSEE d'Orléans (SNDIO), au sein de l'équipe du projet Nautile (Nouvelle Application Utilisée pour le Tirage des Individus et Logements pour les Enquêtes ménages). Le contexte global induit par la crise sanitaire et en particulier la période de confinement, avait placé le télétravail comme mode d'organisation principal. Par question, la question de l'utilisation de plateformes externes et de mise au carré des projets pour un développement en externe a pu se poser pour de nombreux projets, puisque les contraintes matérielles au niveau du réseau peuvent parfois nuire à la disponibilité / productivité des développeurs au sein des équipes.

Dans le cas particulier du projet dans lequel je suis analyste développeur, une prestation était prévue pour la période estivale, sans poste interne et donc il fallait envisager des solutions pour sortir le code source du projet Nautile hors de l'infrastructure INSEE. Cela s'est effectué en partie par l'appui de l'équipe en charge du passage au Gitlab externe et de l'équipe du projet ATHENES (projet du SNDIO) dans les vérifications finales.

La stratégie a donc été de minimiser les risques (le code passant dans un cloud privé, il y a donc des risques de sécurité et juridiques), et l'on a donc procédé à l'ouverture du code, par 3 grandes étapes, la réduction de l'adhérence à l'environnement INSEE, la suppression des données sensibles du projet et enfin la documentation ainsi que l'apport d'une infrastructure de test par l'anonymisation de données ainsi que la création d'une documentation pour la contribution externe.

En ce qui concerne l'échantillonnage équilibré, une description d'un besoin réel a été discuté au sein de l'équipe Nautile, notamment pour l'élaboration d'échantillons d'unités primaires pour le tirage des enquêtes ménages. Le sujet m'intéressant j'ai décidé de proposer ce sujet pour le stage de cette année dans le cadre d'un sujet plus vaste, le passage à l'open-source et comment il s'inscrit dans une dynamique globale du développement actuel.

2 - Introduction à l'open source

Il convient, avant de commencer un argumentaire sur les points de convergence d'un projet à une ouverture de code de présenter les principaux concepts qui viennent avec la terminologie opensource.

2.1 Concepts généraux

Le terme opensource[6] a d'abord émergé dans le développement de logiciels informatiques encourageant la collaboration entre différents acteurs indépendants. Il s'agit d'un modèle décentralisé par nature, puisque le coeur du développement est délégué à une communauté plus ou moins grande. Ce modèle a fait ses preuves à une plus grande échelle, une technologie pouvant devenir standard par la contribution de tous les utilisateurs qui s'emparent ainsi des moyens de production d'une valeur sur le produit qu'ils utilisent.
Ainsi vient l'importance d'une documentation accessible par tous du projet concerné, documentation qui s'intègre à la fois à l'extérieur du code : reproductibilité des environnements, détails sur les objectifs du projet et points sur les règles concernant les modes de contribution (utilisation de git, pull requests) . Il est à noter que la norme est à l'utilisation de l'anglais par défaut, le partage de code étant standardisé sous cette forme. A l'intérieur du code source, pour une meilleure compréhension des enjeux du projet, une documentation des classes métiers principales est fortement recommandée, la description des fonctionnalités attendues à prévoir mais également des bugs et, par le biais d'éventuel tests, de scénarios permettant de mieux comprendre l'utilisation de l'applicatif.

2.2 Étude de la méthodologie interne

La littérature sur une bonne méthodologie d'ouverture des codes source est très dense de par l'engouement du sujet qu'est l'ouverture du code à tous (exemple : [14]). En interne, des pistes sont proposées par un groupe de travail sur le sujet. Il faut savoir que la loi pour une République numérique (2016) donne certaines pistes sur les directives de l'open source au sein des infrastructures publiques. Notamment au niveau de l'INSEE, il y est fait mention d'une "Ouverture et gratuité des données de l'INSEE" ainsi que d'une "Mention explicite de l'utilisation d'un traitement algorithmique dans le cadre d'une décision administrative". C'est dans ce cadre qu'il parait pertinent de partager, avec les données, le code source qui sert de parangon pour comprendre les données résultantes des divers traitements. Il paraît donc cohérent d'insister sur l'ouverture du code des projets actuels à l'INSEE par défaut et non comme un investissement nouveau pour les projets, étant donné la qualité induite par ce processus (cf partie 3).
En interne, la méthodologie proposée offre un cadre sur les parties d'anonymisation du dépôt pour ne pas divulguer d'informations sur les plateformes internes, le croisement d'informations de plusieurs dépôts mal anonymisés étant par nature une des failles de sécurité

pour l'infrastructure de production. Elle offre également des exemples de documentation sur la description des applicatifs par processus métiers, par référence aux dépôts INSEE existant en externe mais également du conseil sur certains points. Par exemple, au cours de mon stage, j'ai pu poser la question sur la nécessité du développement d'un composant implémentant le pattern d'injection de dépendances pour certains modules INSEE, un broker de message interne : le Service de Soumission des Tâches et l'autorité de certification : Keycloak . Cela permettant ainsi de travailler dans un environnement simplifié sans contraintes de sécurité, ce qui est acceptable pour un environnement de développement en local.

Une autre solution aurait pu également être l'ajout d'un module spécifique configurant un keycloak identique à l'environnement interne, mais cela a été abandonné par la suite, l'injection de dépendance par ajout de propriété étant bien plus confortable pour des nouveaux arrivants sur le projet.
La DIIT propose également des prestations de passage a l'Opensource par du consulting sur les projets en demande d'appui ou simplement de consultation sur le travail attendu en interne. Une note (disponible en annexe), précise en effet les attendus minimaux relatifs principalement à la sécurité du passage à l'open source et à l'élaboration d'une documentation accessible par tous. Dans ce contexte minimal, l'ouverture constitue donc une évaluation par les pairs au sein de l'administration.
Un contexte étendu est également à prévoir, notamment sur la documentation des classes métiers en langue de Shakespeare mais également sur la constitution d'un processus de mise en déploiement par la documentation sur l'utilisation du projet et la création d'une architecture logique par l'utilisation de docker, maintenant standard dans le milieu du développement. Le déploiement sur une infrastructure de test externe comme par exemple le cluster Kubernetes de l'innovation consiste en une étape supplémentaire permettant la simplification du workflow de revue de code pour l'intégration de nouvelles fonctionnalités mais également à des fins de compréhension des enjeux du projet.
Enfin se pose la question de où et comment est organisée l'externalisation du code. Pour l'instant, au sein de l'infrastructure de développement interne, cela s'organise principalement sur le groupe INSEEFR sur github, groupe de l'organisation regroupant donc différents projets ayant passé le pas de l'ouverture du code. Récemment, et par harmonisation des processus d'intégration continue et d'accessibilité pour tous, un projet est celui du passage a Gitlab pour les projets en les regroupant par domaines d'activités au sein de la vaste urbanisation du SI de l'INSEE. J'ai notamment pu participer à cette expérience dans le cadre de mon stage, l'ouverture totale étant prévue pour la fin d'année 2020.

2.3 Licences

Un autre point clé du passage d'un projet à l'Opensource est de définir contractuellement les droits des intervenants du projet. Cela est un point assez capital, puisqu'à l'heure du partage de code, certaines entreprises peuvent, un peu à la manière du projet Onyxia à

l'INSEE mais a des fins de rentabilité, intégrer des solutions opensource en un seul grand projet, lui, sous licence non libre, pour ainsi vendre des solutions clés en main, à des tarifs pouvant être élevés, à des organismes divers. Cela peut poser problème et l'on peut par exemple, au moyen d'une licence favorisant le copyleft (impossibilité d'imposer un copyright sur le code source), limiter ce genre de situations. Cela peut par contre s'accompagner d'une certaine défiance des potentiels contributeurs du projet quant à l'appropriation du projet par la contribution de ceux-ci.

En cela il y a des licences pour chaque besoin, et le non choix de licence rend le projet non libre de droits par défaut. Cette condition conservatrice préserve les projets non soumis à licences d'une certaine forme de vol, mais encore une fois la logique étant au partage, il peut s'avérer très utile de choisir une licence privilégiant un partage libre avec contrôle sur les versions filles comme la licence General Public licence qui est une licence "Strong copyleft", au sens où elle impose au code source des projets fils d'être ouvert. Il y a également des licences plus ouvertes comme la licence MIT, pour laquelle les contrôles sont moins forts puisque n'attendant rien des projets fils. Il semble toutefois pertinent de définir le cadre d'ouverture du projet à son lancement, en imposant par exemple une licence de type GPL par nature, ou en choisissant des licences "standards" en fonction du type de projet opensource envisagé (cf choosealicence.com). La communauté open source est parfois considérée comme élitiste, mais cela fait partie également de l'intérêt du partage du code, la discussion technique prenant une place encore plus importante dans la vie du projet.

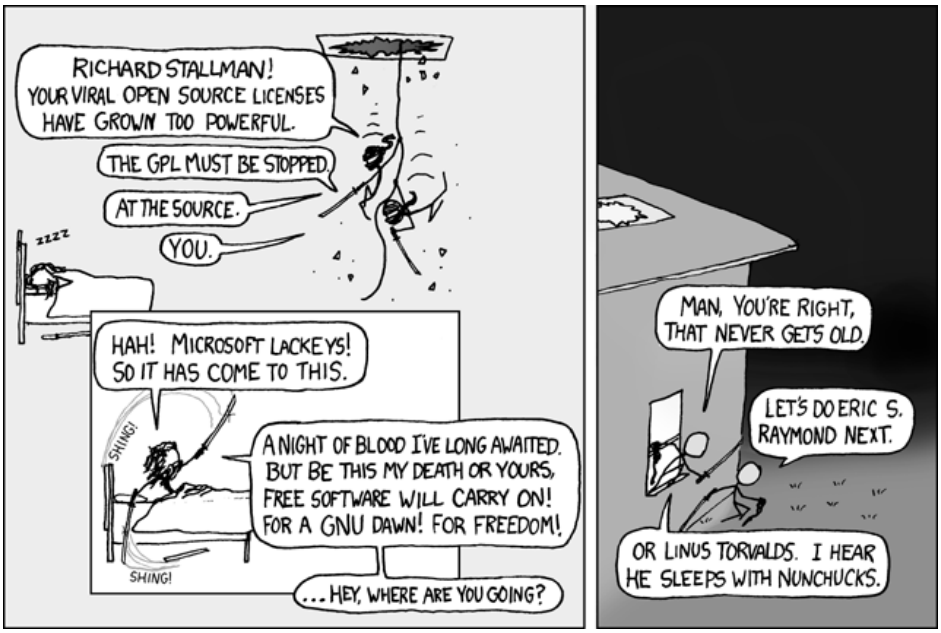

FIGURE 2.1 – Caricature du purisme dans l'open source
Source : xkcd - Opensource : https ://xkcd.com/225/

3 - Modularité et ouverture du code

Lorsque l'on parle de bonnes pratiques dans les normes de développement, la recherche de la qualité d'un produit par son gain en modularité est souvent une question intéressante, surtout sur des produits dont la taille peut être importante (50000 lignes de codes) par exemple.

3.1 Modularity in design

Il est important de bien définir ce qui fait d'une application une application modulaire.On entend par module le regroupement d'un nombre restreint de fonctionnalités, interconnecté en son sein, et isolé de l'extérieur. La programmation fonctionnelle est par essence génératrice de modules, la signature de la dite fonction exprimant un contrat et son effet étant limité dans son impact qu'en son sein, quand on parle de manipulation de données. En cela, la recherche de l'idempotence dans le cadre de développement de fonctions constitue un nécessaire permettant de définir des transactions logiques qui doivent être acceptées dans leur entièreté sans option. L'on pourrait ainsi comprendre que dans le cadre d'une fonctionnalité de tirage d'un échantillon par rapport à une règle, il s'agirait donc de définir un contrat d'interface qui à une série d'unités de tirage définirait ou non une résultante sur leur statut : tiré ou rejeté. Et que cela se produise pour toute taille d'échantillon, tout type d'unités. C'est ici que la programmation objet apporte beaucoup, en définissant des classes abstraites, comme pré-requis pour ce qu'on entend être une unité de tirage, et son contrat d'interface, sur les manipulations qui peuvent lui être faites. Tout cela apporte donc de bonnes propriétés pour l'externalisation du code, puisqu'il est donc plus malléable. Il est également plus testable puisqu'il suffit de tester implémentation par implémentation, avec l'utilisation de "Mocks" et "Spy", deux types d'objets permettant d'isoler d'une interface les comportements attendus par les implémentations et leur utilisation. Cela permet donc une isolation complète des tests en tests unitaires, et permet une couverture de tests maximale, permettant l'élaboration d'un projet plus cohérent et plus structuré. L'injection de dépendance apporte, en effet, beaucoup de qualité pour les développements.

3.2 Principe d'injection de dépendance

Le principe de l'injection de dépendance est l'isolation des dépendances entre plusieurs classes en définissant leurs dépendances dynamiquement plutôt que statiquement. Il peut donc être assez pertinent d'utiliser les propriétés du polymorphisme de l'objet par l'élaboration d'interfaces. En effet, les classes ne seront reliées que par références aux spécifications attendues de l'interface, et l'on pourra donc dynamiquement remplacer une instance de classe par une autre (voir NOOP [1]).

1. NOOP est un standard qui signifie "no operation", implémentaion n'effectuant aucune opération.

L'intérêt de cette pratique est donc de déclarer plusieurs implémentations pour les ressources non accessibles dans tous les environnements et de surcharger par fichier de configuration (injection par mutateur) et par déploiement partiel (dépendances considérées comme disponibles dans l'environnement) pour ainsi rendre indépendant un applicatif d'un module adhérent.

Par exemple a l'INSEE, l'on pourrait effectuer ce genre d'injection pour les services d'envoi de mail, de soumission de taches et plus généralement pour les modules de sécurité (pour développer en local) ou des fonctions de génération de l'aléa ou d'envoi de mail par exemple.

A l'INSEE l'inversion de contrôle se fait généralement par l'utilisation de fonctionnalités avancées du catalogue de frameworks Spring Boot de Spring.io, framework d'inversion de contrôle très utilisé dans l'univers Java Entreprise Edition. En effet celui ci implémente l'injection de dépendance par injection de mutateurs, soit plus vulgairement l'injection d'implémentations d'un comportement bien connu (par les mécanismes d'interface fonctionnelle ou d'abstraction objet notamment). Cela permet d'élever au niveau le plus élevé les dépendances d'un projet et donc d'apporter beaucoup sur les possibilités de réécriture du code source, de maintenabilité et d'ajout de fonctionnalités qui sont marqueurs de qualité et de bonnes propriétés pour des projets effectués en agile comme au sein de l'organisation. L'utilisation de Spring Boot par exemple permet notamment de définir librement des choix d'implémentations paramétrables au sein d'un fichier de configuration lié à l'environnement et non au packaging de l'application, permettant à une même version d'une application de fonctionner sur des environnements pourtant bien différents, en respectant des contrats d'interface communs.

3.3 Risques du passage a l'open source

Avant de s'attaquer à l'externalisation du code source d'un projet, il est important de comprendre les risques de l'ouverture du code à l'échelle d'une entreprise.

3.3.1 Copyright by default

Il faut savoir que le code, dans le monde opensource, est considéré comme sous copyright s'il n'est pas sous licence par défaut. On retrouve ainsi la question de la propriété du code présent, dans le cas de la copie du code par un développeur. Il est donc nécessaire pour les développeurs, s'ils souhaitent utiliser des ressources existantes par ailleurs de tenir compte de ces licences.

3.3.2 Risques sur les projets de l'entreprise

Un des premiers risques pour un projet est la définition commune en externe du versionning de l'application, le passage à l'open source entraîne donc potentiellement des délais sur

la mise à disposition de certaines fonctionnalités clés pour le projet. Un exemple se trouve dans l'utilisation d'une bibliothèque de composants javascript pour lesquels l'accessibilité ne serait pas permise au regard des contraintes des administrations publiques. Ou a l'inverse, le développement d'une fonctionnalité de tirage pour laquelle le degré de qualité ne serait pas acceptable dans le cadre d'un vaste projet et la revue de code par les pairs (qui est souvent définie dans la gestion de projet opensource) ne serait pas effectuée alors que le besoin métier, serait lui pourtant bien inscrit dans le temps.
Il peut aussi y avoir des enjeux d'image de l'entreprise, puisque le code source ouvert s'inscrit dans l'image de l'entreprise, engageant d'autant plus l'équipe projet dans la responsabilité quant à l'efficacité et à la qualité du développement. (cela peut effectivement s'inscrire dans l'obligation de dignité des fonctionnaires travaillant sur le projet)

3.3.3 Reverse engineering

L'ouverture du code implique naturellement l'accès à l'étude du code par un nombre plus important d'utilisateurs, et donc cela accroît la probabilité d'études à portée malveillante de l'existant. Cela induit donc des problèmes aussi bien sur la recherche de points d'entrées pour les attaques (injections sql, injection dans des jsons, authentification..), mais également l'étude d'une infrastructure et de ses logiques (dans le cas où certaines données d'environnement persisteraient dans le dépôt, et donc dans un instant de sa vie.)

Toutefois, cela est souvent bénéfique sur le long terme puisque ces problèmes sont à nuancer une fois contrecarrés, augmentant donc le niveau de qualité dans les projets. Dans le cadre du stage, j'ai donc pu étudier certaines solutions pour éviter ces problèmes.

3.4 Anonymisation des données

Une étape importante pour l'ouverture de code à des fins de sécurité et de mise en place de scénarios de tests est la conception / anonymisation d'un jeu de données. Cette étape s'accompagne souvent d'une définition au sein de l'équipe métier d'une liste de variables qu'il faut anonymiser. Une donnée anonymisée étant une donnée sans corrélation avec la donnée initiale ([7]), il a été fait le choix de s'intéresser aux solutions de génération de données [10] et de masquage[2] de données.

3.4.1 A la recherche d'un outil pour la génération de données

L'anonymisation des données est un vaste sujet, plusieurs solutions externes existent afin de générer des jeux de données de diverses volumétries et dans différents formats. Une solution initiale pour la réalisation de jeu de tests pour l'application Nautile a été d'utiliser un utilitaire externe permettant la génération de champs selon des règles pré-établies pour la cohérence des champs. Par exemple : une date générée aurait un format de date classique, et un nom aurait un nom tiré aléatoirement parmi une liste de noms. Cette solution a donc été conservée pour la génération des données des tables FIDELI issues des fichiers fiscaux

pour une anonymisation complète des données au sein même des jeux de tests disponibles dans le code source, dans la partie des tests d'intégration fonctionnels.[2]

3.4.2 Le problème de la volumétrie

Un problème a ensuite été rencontré lors de l'ouverture du code à une prestation externe. En effet, le prestataire avait besoin, pour le traitement d'un jeu de données en volumétrie réelle, les enjeux du développement se trouvant principalement dans la constitution d'une solution pérenne et fiable vis a vis des performances en base de données. J'ai pour cela d'abord étudié les solutions existantes par contact de la division Recueil et Traitement de l'Information (RTI) qui m'a aiguillé vers la génération des données. En fait, l'anonymisation relève souvent de la destruction de données puis du remplacement de ces données par des données

fictives, sans lien de reproduction entre les premières et les secondes, auquel cas il n'y aurait plus d'anonymisation. Les solutions internes pour de grosses volumétries n'existant pas et les solutions externes n'étant que limitées dans le volume ou les coûts éventuels, j'ai donc cherché une solution personnelle à cette question. Pour cela, j'ai décidé de réaliser un script général, en base de données postgresql, permettant pour chaque champ défini en amont, de tirer parmi une liste de possibilités ou de simplement hasher un nombre entre 0 et 1 (pour une donnée aléatoire chaotique) pour générer des données. Et ensuite de régénérer pour les tables, les indexes permettant jointures pour garder la structure initiale des données avant extraction. Cela m'a permis de générer des jeux de données comparables (dans la structure et dans le volume) aux attendus réels des données fiscales, permettant ainsi un travail sécurisé sur le cloud privé du prestataire. J'ai ensuite envisagé de généraliser ce concept en développant une solution de génération de contraintes sur des tables pour la génération de données fictives par la création d'un projet opensource mettant un peu d'abstraction aux méthodes de génération initiales (construction d'une liste des valeurs a tirer, création de liens et de méta-données de tables).

3.5 Suppression des adhérences aux environnements

Un point critique pour sortir tout code source d'une infrastructure, est la vérification qu'il n'y ait pas de données sensibles en son sein et qu'il puisse fonctionner en externe.

3.5.1 Réécriture de l'historique du code source

Le sujet final avant l'ouverture du code était avant tout la suppression de toute trace d'un passé liant le dépôt à des dépendances internes, ou à des documents de travail confidentiels. J'ai donc suivi le document de travail sur l'open source proposé par la division en charge de l'ouverture du code pendant la période du stage. Ainsi, le choix s'est porté sur l'utilisation d'un utilitaire opensource (sous licence GNU), bfg-repo-cleaner (https ://rtyley.github.io/bfg-repo-cleaner/), il permet la réalisation par filtration de suppression récursive dans un dépôt

local Git de toutes les données qui correspondent à ce filtre, permettant par exemple de filtrer tous les documents volumineux, documents de travail en .ods .odt .pdf ou encore les données d'environnement en .properties par exemple. Ce travail s'est fait avec l'appui en recette de l'équipe du projet Athènes, également très investie sur le sujet de l'ouverture du code à l'extérieur à des fins de qualité.

3.5.2 Dépendances

Ensuite, il a fallu séparer, pour le projet Nautile, les parties adhérentes à l'interne en différents sous-modules pouvant s'interchanger en fonction de l'environnement par l'utilisation de fonctionnalités avancées de maven : les dépendances "disponibles" ou "provided", permettant donc d'ouvrir le code avec une version simplifiée et contractualisée du Service de soumission des tâches, spécifiant une file d'attente et une planification des tâches de manière fonctionnelle.

3.5.3 Qualimétrie

Enfin, une étude du code a été utilisée par un outil de qualimétrie opensource. Certains outils dont l'intégration est facilitée par des plugins maintenus par la communauté sont mis a disposition de tous, permettant notamment, par le biais de l'établissement de règles et par l'ajout d'un glossaire de règles pertinentes en fonction du projet d'évaluer la qualité du code. Pour le projet Nautile, le choix s'est porté sur l'utilisation de Sonarqube[4], en adaptant les règles notamment sur la détection de l'utilisation de méthodes pouvant induire des injections sql, mais également de la couverture de test. Ces outils seront très utiles pour établir une organisation du projet externe permettant ainsi d'instruire des règles objectives de non acceptation des contributions. Par exemple l'on peut retrouver dans certains projets, les règles sur la non duplication de code, de non réduction de la couverture de test globale ou même du non ajout de dépendances. Cela a permis d'isoler certains bugs en amont du passage en externe sur gitlab.

4 - Organisation d'un projet et portabilité

Une fois le code mis en externe, il est nécessaire pour l'intégration et la vie du projet de définir des normes de développement pour les ajouts de nouvelles fonctionnalités mais également de documenter et créer des environnements de tests accessibles pour tous, les développeurs pouvant donc prendre en main au plus vite l'applicatif et ses enjeux. Pour cela, beaucoup de documentation existe en ligne, puisque la communauté opensource a pu construire des points clés d'un bon passage à l'open source (exemple : [9])

4.1 Documentation

En ce qui concerne la documentation, elle doit être faite dans une langue accessible à tous, c'est-à-dire, dans le milieu des projets opensource, l'anglais. Par documentation on entend bien entendu les documents permettant d'accéder au code, c'est-à-dire les documents licence et le Readme.md à la racine du projet. Ce dernier est la porte d'accès du projet puisqu'il est exposé directement aux personnes se rendant sur la page du projet. Il est en effet exposé en page d'accueil, de manière standardisée, dans la plupart des services web d'hébergement de code comme github ou gitlab.
Là où cela peut être plus coûteux, c'est que le code doit également être mis en forme pour rendre accessible pour un utilisateur à l'international. On a donc accès à un code aux objets métiers bien définis et à la documentation directe (signature des fonctions, tests, commentaires). Cela a donc de très bonnes propriétés pour la maintenabilité d'un projet, même en interne. La réduction des coûts d'entrée étant un sujet très important avec la flexibilisation des ressources par groupe projet comme on peut le voir à différents niveaux des Services Nationaux de développement de l'INSEE.
Un autre point important est une petite feuille de route sur comment installer son poste de travail pour être opérationnel sur le projet et comment éventuellement tester certaines fonctionnalités (lancement de certains batchs, architecture du projet, environnements de tests). Pour cela, j'ai donc présenté une paramétrisation pour les batchs, l'architecture et un jeu de tests anonymisé.

4.2 Versionning et organisation du projet avec git et gitflow

Tout d'abord, en ce qui concerne le projet, il est préférable qu'il soit soumis à du versionnement, cela a beaucoup de propriétés pour une vision long-termiste et la mise en place d'une organisation de travail. A l'INSEE, comme dans beaucoup d'entreprises, nous ne partageons pas le code par mail, mais bien par l'utilisation d'un gestionnaire de versions : git. Créé a l'origine par Linus Torvalds, ce projet opensource regroupe beaucoup de bonnes commandes issues du monde Unix.

Afin de permettre un contrôle sur les ajouts de code sur le dépôt, il est pertinent de définir des règles strictes d'intégration. Une pull-request consiste en la demande d'un utilisateur externe à l'intégration d'un code qu'il a développé. La mise en place de règles identiques pour tous les utilisateurs est également importante, puisque marquantes de la différence entre le monde fermé et ouvert.

4.2.1 Gitflow

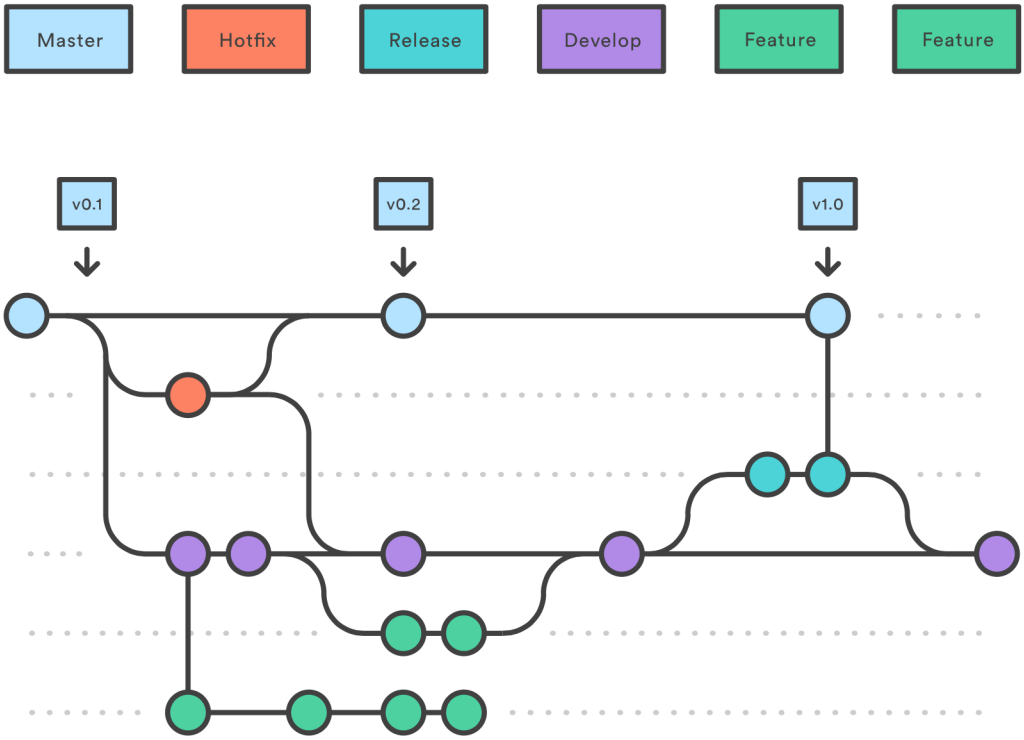

FIGURE 4.1 – Gitflow

Le gitflow consiste a désigner un workflow d'intégration du code, en désignant différentes branches (espaces de travail sur des versions du code) plus ou moins sensibles. Un exemple est la définition d'une version stable et d'une version preview ou de plusieurs versions preview dans le cadre de grands projets. La validation par les pairs en est un exemple. Mais également le regroupement du travail par fonctionnalités (ou features) pour permettre un bon découpage des tâches, précisées par le libre accès aux fonctionnalités de reporting : les issues. Tous les utilisateurs et développeurs peuvent en effet déclarer des problèmes rencontrés, bugs ou simplement fonctionnalités attendues par ce biais, et ainsi contribuer

dans les spécifications fonctionnelles de l'application.

4.2.2 Cas particulier de l'intégration continue

L'intégration continue, en génie logiciel, consiste en la vérification de la non régression des fonctionnalités existantes à l'ajout d'un code nouveau. Cela s'inscrit dans le gitflow par la création d'une infrastructure de scoring et de validation de l'intégration d'un code. L'utilisation d'outils de qualimétrie, permettant d'analyser la couverture de test, de la duplication de code ou de la génération de failles de sécurité permet d'établir des règles objectives de non intégration d'un code proposé par pull request. Cela permet également de valoriser le développement par les tests ou Test Driven Developpement, puisque les propriétés de vérification de la non régression sur les ajouts unitaires des développeurs assurent une intégration facilitée et une évaluation plus rapide du code.

4.3 Versionning de base de données

Dans le cas particulier de certains projets de l'INSEE pour lesquels les structures sont adhérentes à des millésimes de données (c'est le cas par exemple pour le projet Nautile, la structure des données fiscales pouvant évoluer) mais également dans le cas des projets liés aux formulaires d'enquête, il peut être intéressant de structurer le versionning de base de données et ainsi de permettre de choisir sur quel millésime se situe le déploiement. Cela permet notamment de s'isoler une fois plus de l'environnement en inversant le contrôle de la gestion de base de données du côté du code source et donc d'isoler d'autant plus les dynamiques de build et d'hébergement du projet en tant que paquet.
Liquibase et Flyway sont deux outils open-source apportant une solution à cette problématique, il a été fait le choix d'ajouter une dépendance à Liquibase au niveau du projet Nautile, pour des raisons de gestions par différences et d'une cohérence en interne avec des projets utilisant déjà cette technologie.

4.4 Processus de packaging avec Docker

A des fins de clarification de l'architecture et de création d'un processus de packaging pour le déploiement, il est maintenant standard d'utiliser la solution Docker containers. Celle ci permet une abstraction au niveau de l'OS, fonctionnant avec un docker docker daemon et est très utile pour la mise en place d'un pipeline d'intégration continue mais également la création de paquet prêt a être orchestrés par la suite pour un déploiement sur un environnement de test. Pour Nautile, le travail a donc été de séparer les couches de l'application en différentes parties, avec la partie tomcat, la partie batch ,la partie ihm et la partie base de données, le broker de message n'étant pas pour l'instant implémenté de manière conteneurisable, le fonctionnement n'est pour l'instant pas forcément acceptable pour l'applicatif Nautile. (schéma archi nautile) Par contre, dans le cadre d'applications de

tests, par exemple pour l'anonymisation de données ou l'intégration du tirage équilibré, cela est tout a fait applicable.

4.5 Environnements de test avec Kubernetes

Kubernetes est une solution open source, leader dans le domaine de l'orchestration de containers. Son utilisation au sein du développement est encore très nouvelle au sein de l'infrastructure de l'INSEE. Le principe global est l'association entre gestion des ressources d'un regroupement de machines, mise a disposition de services préalablement conteneurisés, load-balancing. Dans le cadre de la mise en place d'un service de déploiement en continu d'un service ou même plus simplement pour des fins d'amélioration de la phase de tests dans les projets, que ce soit pour la recette d'une fonctionnalité ou pour la compréhension du code il paraît utile d'industrialiser la mise en recette d'une application. Actuellement à l'INSEE, le processus s'effectue le plus souvent avec un autre orchestrateur qui est Marathon mais de récentes orientations tendraient à l'utilisation de Kubernetes sur le long terme. Afin de simplifier le déploiement sur les clusters kubernetes, il convient de penser au déploiement d'un paquet ou chart Helm. Cela consisterait en effet en une bonne pratique pour une version étendue de la mise en place de l'open-source dans un projet INSEE.

4.5.1 Déploiement en continu

Le déploiement en continu consiste en l'automatisation des processus menant un projet de son code à son hébergement sur un environnement. Dans le cadre des environnements de tests, cela a plusieurs vertus. Tout d'abord, cela permet de définir un fonctionnement global à tous les environnements à partir des livrables construits en amont, et de relier d'autant plus le code à sa finalité, un service. Ensuite cela permet de contractualiser et donc de normaliser les processus qui auparavant relevaient d'une action manuelle répétitive et empirique. Enfin cela répond à la problématique de l'intégration de nouvelles fonctionnalités unitaires et modulaires, en associant l'organisation par pull requests avec l'hébergement d'une version revue de l'application sur un environnement de test. A l'échelle de Nautile, un travail a été effectué pour dockeriser les parties web et ihm, permettant de recetter rapidement les ajouts de ce côté. Le fonctionnement par messages au Service de soumission des tâches n'étant pas directement reproductible, l'infrastructure de batch n'est pas encore reproduite pour les tests toutefois.

5 - Méthode du cube du tirage équilibré

Différentes méthodes existent pour la réalisation d'un échantillonage équilibré ([5]), la méthode du cube est celle qui a été étudiée au cours du stage.

5.1 Rappels de théorie des sondages

Dans cette section seront effectués des rappels sur les concepts abordés au cours du stage pour l'élaboration d'un algorithme de tirage équilibré. L'objectif d'un sondage est l'élaboration d'un échantillon.

5.1.1 Notations

Nous utiliserons les conventions de nommage suivantes : N pour le cardinal de l'ensemble de la population totale U , S pour le sous ensemble des échantillons possibles. Un échantillon tiré est noté s et l'on notera n sa taille. Les variables d'équilibrage seront notées x. Les probabilités d'inclusion des unités seront notées π_i, c'est à dire la probabilité d'inclusion d'une unité dans l'échantillon final .

Selon cette convention de notation, les estimateurs de Horvitz-Thompson pour estimer des caractéristiques comme le total et la moyenne d'une variable à l'aide d'un échantillon représentatif s seront sous la forme :

Pour le total :

$$\boxed{\hat{T}_{xHT} = \sum_{i \in s} \frac{x_i}{\pi_i}} \tag{5.1}$$

et donc pour la moyenne :

$$\boxed{\hat{\bar{x}}_{HT} = \frac{1}{\sum_{i \in s} \frac{1}{\pi_i}} \sum_{i \in s} \frac{x_i}{\pi_i}} \tag{5.2}$$

5.1.2 Plan de sondage

Le plan de sondage est un descriptif méthodologique d'un processus menant à l'élaboration d'un échantillon. La recherche d'un tel processus permettant de maximiser la représentation d'un échantillon par rapport aux caractéristiques étudiées par champ d'enquêtes par exemple, peuvent être des critères importants d'arbitrage interne d'une méthode de sondage ou plan de sondage. Pour l'application Nautile, l'accent s'est retrouvé dans le choix de l'implémentation de la méthode de tirage systématique pour le tirage des échantillons d'individus logement pour les enquêtes ménages. Cette méthode est à la fois plus performante[3] que

l'échantillonnage équilibré sur différentes enquêtes pour le cas particulier de la non réponse dans les grandes villes, et particulièrement testable de par son aspect déterministe post paramétrage. Pour les petits domaines toutefois, comme la détermination d'un sous échantillon d'unités primaires dans lesquelles effectuer le premier degré d'un tirage d'échantillon à deux degrés, l'échantillonnage équilibré est plus préconisé[12]. Au plan fonctionnel, le plan de sondage se définit comme toute fonction de probabilité qui à chaque sous-ensemble s de S, associe une probabilité p(s) telle que s soit l'échantillon tiré, vérifiant :

$$\boxed{\sum_{s \in S} p(s) = 1 \text{ avec } 0 \leq p(s) \leq 1} \tag{5.3}$$

5.1.3 Plan de sondage équilibré

Le plan de sondage équilibré permet, par la sélection d'un vecteur de variables d'équilibrage, d'élaborer des échantillons équilibrés selon ces variables. Le plan de sondage équilibré peut à l'inverse être interprété comme la recherche d'un échantillon parmi un sous espace d'échantillons possibles Q, cet espace d'échantillon étant contraint par les variables d'équilibrage x_i.

Il peut ainsi s'écrire :

$$\boxed{Q = \left\{ s \in S \middle| \sum_{i \in s} \frac{x_i}{\pi_i} = \sum_{i \in U} x_i \right\}} \tag{5.4}$$

L'intérêt d'un tel plan est qu'il permet une estimation personnalisée selon les variables d'intérêt d'un champ d'étude pour la réalisation d'un échantillon. Il s'agit en quelque sorte d'un calage au niveau du plan de sondage. L'idée sous jacente au sondage équilibré est donc l'utilisation des informations connues corrélées avec les variables d'étude pour une restitution au plus proche de l'échantillon complet des caractéristiques retrouvées dans l'échantillon. Plus l'information sur les variables d'intérêt et leur corrélation avec l'information seront grandes, meilleure l'élaboration d'un échantillon sera en terme de précision.

5.1.4 Viabilité du plan de sondage équilibré

Dans cette section, l'on s'intéresse aux propriétés d'un tel plan. Par exemple, dans le cas d'un modèle classique de type :

$$y_i = \beta x_i + \epsilon_i$$

avec y_i variable d'intérêt, x_i variable de contrôle et ϵ_i résidus

En divisant le tout par π_i et par passage par l'estimateur des totaux on obtient :

$$\hat{T_{yHT}} = \beta \hat{T_{xHT}} + \hat{T_{\epsilon HT}}$$

Et comme l'on a x variable d'équilibrage, alors son total est bien estimé et donc on a bien :

$$\hat{T_{yHT}} = \beta T_{xHT} + \hat{T_{\epsilon HT}}$$

Si l'on ajoute à cela le respect des probabilités d'inclusion ainsi que la connaissance de la variable d'équilibrage par l'isolation de sa variabilité par rapport au plan de sondage, l'on à :

$$\boxed{E(\hat{T_{yHT}}) = T_{yHT} \text{ et } V(\hat{T_{yHT}}) = V(\hat{T_{\epsilon HT}})} \tag{5.5}$$

Soit encore que l'estimation du total sur la variable d'intérêt se fait sans biais et avec une variance égale à la variance de l'estimateur des totaux des résidus, donc au regard de l'information manquante. La même se retrouve, par la même méthode, dans l'estimateur de la moyenne d'Horvitz-Thompson.

5.1.5 Problématiques du plan de sondage équilibré

Bien que le concept du plan de sondage équilibré permette de résoudre les questions de la représentativité de l'échantillon par rapport à la population, toute la difficulté réside à la mise en oeuvre d'algorithme de tirage respectueux des probabilités d'inclusion, sans remise, rapides, et généralisables à tout plan de sondage. (Deville et Tillé, 2004 [11]) La mise en place d'un échantillonage équilibré nécessite donc l'étude préalable du processus de génération d'aléa puisque tout l'équilibrage réside en partie sur une sélection aléatoire parmi une liste d'échantillons. De plus, la mise en place d'un équilibrage par contraintes d'équilibrage entraîne naturellement un déséquilibre dans l'attribution des poids de sondage ou probabilités d'inclusion des unités qu'il ne faut pas négliger. D'autant plus que cela modifie la variance des estimateurs.

5.2 Méthode du cube

La méthode du cube est en cela, un algorithme permettant de répondre à ces problématiques dans une majorité de cas. Il permet le tirage d'un échantillon équilibré satisfaisant les contraintes d'équilibrage et les probabilités d'inclusions données en entrée.

5.2.1 Présentation de l'algorithme

L'algorithme se présente sous un fonctionnement en 2 phases. Une phase de vol, menant a la recherche d'un vecteur unitaire échantillon respectant "le plus possible" les contraintes d'équilibrages données en entrée. Et une phase d'atterissage permettant d'arbitrer sur la

qualité de l'échantillon obtenu durant la phase de vol, au regard des contraintes indiquées en entrée, si aucune solution n'est clairement atteinte, une nouvelle phase de vol est enclenchée, mais après une phase d'atterissage qui consiste par exemple en la réduction du sous espace des contraintes (afin d'augmenter la taille du sous espace des solutions). Ainsi vient la notion d'ordre sur le choix des contraintes d'équilibrage, permettant à l'algorithme, s'il n'arrive pas a aboutir, de réduire le nombre de contraintes "moins importantes" afin de gagner en degrés de liberté dans la détermination de l'échantillon tiré.

5.2.2 Notations spécifiques

Indicatrice d'inclusion dans l'échantillon

Pour une population U de taille N, on pose Z la variable aléatoire indicatrice telle que l'échantillon obtenu s soit égal à :

$$\boxed{s = (Z_1, Z_2, ..., Z_N)'} \tag{5.6}$$

avec Z_i, i $\in [1, N]$ variable aléatoire indicatrice telle que :

$$\boxed{\begin{Bmatrix} Z_i = 1, si\ i \in S \\ Z_i = 0\ sinon \end{Bmatrix}} \tag{5.7}$$

et

$$\boxed{\begin{Bmatrix} P[Z_i = z] = \pi_i, si\ z = 1 \\ P[Z_i = z] = 1 - \pi_i, si\ z = 0 \end{Bmatrix}} \tag{5.8}$$

L'échantillon s'interprète donc comme une réalisation dans l'hypercube $[0, 1]^N$ où chaque unité tirée prend la valeur 1.

Sous espace de vol

On appelle sous espace de vol le sous espace de construction d'un échantillon équilibré au cours de la phase de vol. Au sein de l'hypercube des échantillons possibles sur toutes les unités de la population, chaque sommet du cube représentant un échantillon d'unités tirées, l'on souhaite représenter le sous ensemble des échantillons possiblement tirés qui respectent les contraintes définies en amont, en notant $x_1, x_2, ..., x_p$ cet ensemble de variables d'équilibrage. D'après (5.4) et (5.6) l'on à, dans le cadre d'un plan de sondage équilibré :

$$\boxed{\sum_{i \in U} Z_i \frac{x_i}{\pi_i} = \sum_{i \in U} x_i} \tag{5.9}$$

Cela permet de déduire le sous espace de vol caractérisant toutes les équations de l'espace de Q par le champ des unités possiblement tirées.

$$\boxed{Q = \pi + Ker(A)} \tag{5.10}$$

avec A la matrice suivante :

$$\boxed{A = \begin{pmatrix} \frac{x_{11}}{\pi_{11}} & ... & \frac{x_{N1}}{\pi_{N1}} \\ ... & ... & ... \\ \frac{x_{1p}}{\pi_{1p}} & ... & \frac{x_{Np}}{\pi_{Np}} \end{pmatrix}} \tag{5.11}$$

où x_{ij}, $i \in [1, p], j \in [1, N]$ représentent les valeurs prises par chaque unité sur les variables d'équilibre. Pour la suite il sera noté :

$$\boxed{K = \{[0,1]^N \cap (\pi + KerA)\}} \tag{5.12}$$

Martingale équilibrante

Un processus stochastique est dit une martingale équilibrante pour un vecteur de probabilités d'inclusion π et un ensemble p de variables $x_1, x_2, ..., x_p$ s'il s'agit d'un processus aléatoire à temps discret π_t défini par :

$$\boxed{\pi(t) \in \mathbb{R}^N \Big| \pi(t) = \begin{pmatrix} ... \\ \pi_i(t) \\ ... \end{pmatrix}, t \in \mathbb{N}} \tag{5.13}$$

Tel que :

$$\boxed{\begin{array}{c} \pi(0) = \pi, \\ \forall t > 0, E(\pi(t)|\pi(t-1), .., \pi(0)) = \pi(t-1), \\ \forall t,\ \pi(t) \in K, \end{array}} \tag{5.14}$$

5.2.3 Phase de vol

La phase de vol de la méthode du cube consiste en une marche aléatoire menant à l'élimination d'échantillons dans le sous espace des échantillons Q répondant aux contraintes d'équilibrage définies en entrée de la phase. Cette marche aléatoire s'effectue par modification des probabilités d'inclusion par une martingale équilibrante, aléatoire (en les passant progressivement de leur valeur initiale à une valeur égale à 0 ou 1). Cette modification du vecteur de probabilités d'inclusion s'effectue selon les contraintes, se rapprochant "le plus possible" du plan formé par les contraintes dans l'hypercube de tous les échantillons possibles S. Par définition, cet hypercube de dimension $N \times n$ regroupe en chacun de ces sommets un échantillon de n unités tirées, l'objectif de l'algorithme du cube étant donc d'aboutir à l'arrivée sur un de ces sommets. Son fonctionnement est le suivant :

Soient $\pi(t)$ une martingale équilibrante (cf 5.13) pour $x_1, ..., x_p$ et π le vecteur de probabilités d'inclusion tel que $\pi(0) = \pi$. A la matrice des valeurs de chaque unité sur leurs variables d'équilibre (5.11) et

Étape 1

On définit un vecteur aléatoire quelconque noté $u(t)$ tel que :

$$\boxed{\begin{array}{c} u(t) \in Ker(A) \\ \forall i, \pi_i(t) \in \{0,1\} => u_i(t) = 0 \end{array}} \tag{5.15}$$

Étape 2

Trouver les plus grandes valeurs ${\lambda_1}^*(t), \lambda_2^*(t)$ telles que :

$$\boxed{\begin{array}{c} 0 \leq \pi(t) + \lambda_1(t)u(t) \leq 1 \\ 0 \leq \pi(t) - \lambda_2(t)u(t) \leq 1 \end{array}} \tag{5.16}$$

Remarque : cela assure qu'à chaque itération, au moins une unité est choisie

Étape 3

Calculer $\pi(t+1)$:

$$\pi(t+1) = \begin{cases} \pi(t) + \lambda_1(t)u(t) & \text{avec la probabilité } q(t) \\ \pi(t) - \lambda_2(t) & \text{avec la probabilité } 1 - q(t) \end{cases} \tag{5.17}$$

avec q(t)= $\frac{\lambda_2(t)}{\lambda_1(t)+\lambda_2(t)}$

5.2.4 Phase d'atterrissage

La phase d'atterissage consiste en l'établissement final d'un échantillon. Si la phase de vol a abouti à un échantillon il est choisi. Sinon, il existe plusieurs possibilités pour cette phase d'algorithme. Le principe étant, pour ne pas rogner sur l'objectif de représentativité en altérant les poids initiaux, la question de la recherche du "plus possible" dans les contraintes définies.

La réduction des contraintes

L'implémentation la plus classique est la réduction du nombre de contraintes si un échantillon n'a pas pu être atteint. En effectuant une nouvelle phase de vol en levant une contrainte, la dernière selon l'ordre défini en amont. L'on augmente donc naturellement le nombre d'échantillons dans K, ce qui permet d'augmenter la probabilité d'aboutir à un échantillon pendant la phase de vol.

Minimisation de l'écart à l'équilibre : programmation linéaire

Si certaines unités ont pu être tirées, et que l'on n'arrive pas à aboutir, une autre méthode peut être de définir un plan de sondage sur les unités restantes en respectant les probabilités d'inclusion de départ et en introduisant un critère de distance à l'équilibre. Par exemple, dans le document sur l'échantillonnage équilibré de Thomas Merly-Alpa & Laurent Costa (2017)[8], l'on introduit une fonction de minimisation de la norme d'écart à l'équilibre des estimateurs d'Horvitz-Thompson comme critère d'équilibrage global sur une population (mais seulement sur de petits échantillons).

Échantillonage par introduction de la probabilité d'inclusion

D'autres méthodes existent en utilisant les propriétés de l'échantillonnage de taille fixe par introduction des probabilités d'inclusion (indicatrices) dans les variables d'équilibrage.

5.3 Exemples en dimension 3

Prenons le cas d'un échantillon de 3 unités avec comme contrainte pour la taille fixe de l'échantillon de taille n=2, En prenant pour $\pi_i = \frac{2}{3}$ pour chaque unité cela donne : en arrête du cube en intersection avec le plan des contraintes de la fonction identité : (1,0,1), (1,1,0), (0,1,1). Ici la phase de vol se termine directement et l'échantillon respecte les contraintes d'équilibre à la fin de la phase de vol.

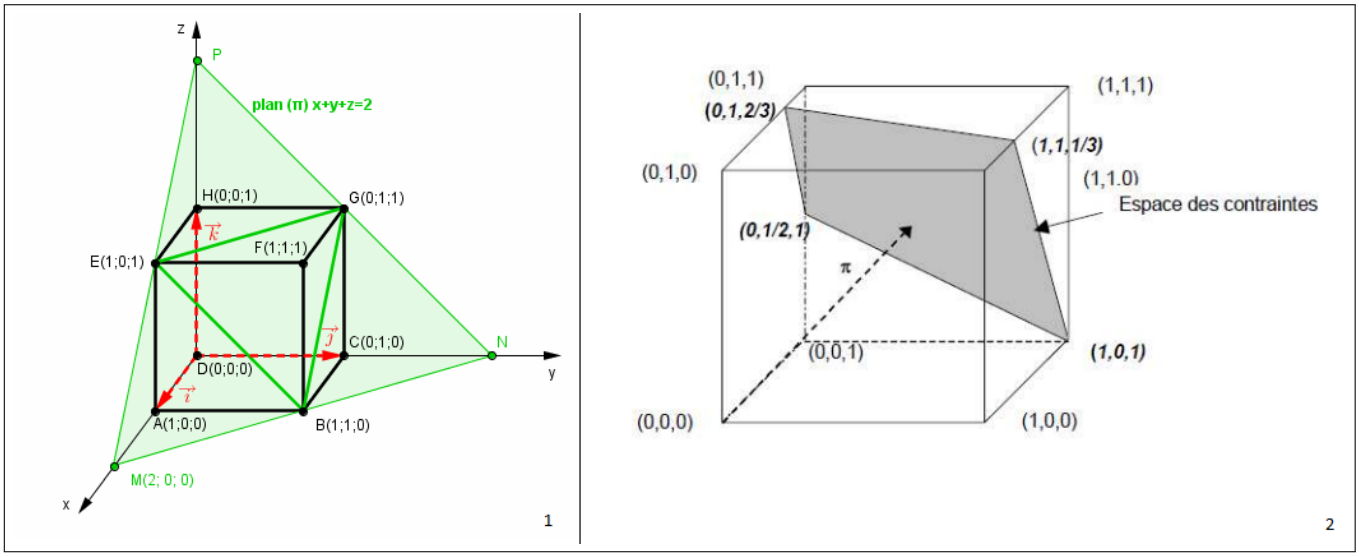

FIGURE 5.1 – Représentation graphique du sous espace de l'intersection du sous espaces des contraintes et de l'hypercube des échantillons
figure 1 : Cas constante 1 — figure 2 : Cas du numéro d'ordre

Autre exemple, pour le cas d'un échantillon de 3 unités avec comme contrainte d'équilibrage le numéro d'ordre de l'unité. On peut prendre $\pi_i = \frac{2}{3}$, et cela donnera comme arrête (1,0,1), (0,1,2/3), (1,1,1/3),(0,1/2,1). Donc ici, la phase de vol aboutira seulement si l'on est dans le cas où l'échantillon sélectionné serait (1,0,1). Si ce n'est pas le cas, la phase d'atterissage, permettrait d'obtenir, par minimisation de l'écart à l'équilibre (cf partie précédentes), on obtient les échantillons (1,1,0), (0,1,1) par exemple.

6 - Réalisation du projet opensource

Cette section fait echo à tous les concepts présentés au cours des parties précédentes. Les projets dont il est fait mention, ont été initialisés selon la politique d'open source par défaut.

6.1 Étude de l'existant

Dans le cadre de la recherche d'une solution technique à l'implémentation de l'algorithme du cube, des solutions existent. Historiquement, au sein de la structure INSEE, la macro SAS cube (2004) est l'implémentation de l'algorithme utilisé dans les enquêtes nécessitant l'utilisation d'une méthodologie d'échantillonage équilibrée. Sa documentation étant riche et vulgarisée, elle offre, en elle-même, une description fonctionnelle détaillée des différentes étapes de l'algorithme, et des contrôles qu'il est intéressant d'intégrer au traitement.

Plus récemment, des solutions ont été développées sur R, avec les packages samplecube, sampling et cube en sont une illustration. Plus récemment, le package BalancedSampling, développé par des chercheurs non INSEE. La licence choisie pour le projet étant GPL, la réutisabilité du code est donc assurée.

6.2 Architecture générale

L'objectif de cette partie est de construire les différents projets selon la logique open source décrite précedemment.

6.2.1 Documentation pour la contribution

Afin de permettre a tous les utilisateurs de contribuer, le projet sera documenté par un fichier nommé contributing.md, qui fait office de description de la politique d'acceptation des contributions, ainsi que de présentations des lignes directrices du projet.

En ce qui concerne la documentation pour l'environnement des développeurs, un Readme.md, page de garde du projet, est mis en place. Il contient toutes les informations permettant de comprendre le cadre du projet, ses objectifs ainsi qu'une notice de prise en main du code source.

6.2.2 Organisation du code source

Le projet sera a minima séparé en plusieurs sous-projets au sein d'un même groupe applicatif, en regroupant les projets sous 2 catégories. Les projets bibliothèques, présentant un aspect méthodologique et des propriétés permettant une utilisation contractualisée (API).

D'autre part, les projets d'intégration, présentant une façon d'intégrer les modules au sein d'une infrastructure s'approchant d'un environnement de production par les moyens de la conteneurisation et la mise en place d'un chart Helm. Ceux ci accompagneront la documentation, pour l'utilisation facilitée pour un nouvel arrivant et la réutisabilité du code dans des environnements similaires. C'est ici que l'injection de dépendance sera intéressante à implémenter puisque permettant d'isoler la bibliothèque de son utilisation. La mise en place d'un pipeline de CI/CD sera également bénéfique, pour d'un côté le bon contrôle sur la qualité et le déploiement et versionning des bibliothèques. Et de l'autre pour la construction d'un environnement de review[1] pour d'éventuels utilisateurs agnostiques de l'implémentation.

6.2.3 Licences

Le choix de la licence s'est porté sur la licence GNU, qui présente un strong copyleft et permet donc de cibler directement le projet comme un projet de partage commun.

6.3 Exploration des solutions

6.3.1 Réutilisation du code R par plumber

Une première exploration au cours de mon stage était à la création d'une api native en R afin de réutiliser directement les développements déjà proposés au niveau CRAN pour un résultat rapide. L'application résultante étant donc une application utilisant le package samplecube, se connectant à une base de données par un driver JDBC et exposant tout cela sous forme de web service documenté par le biais du package Swagger. Ce projet a été ensuite abandonné de par son hétérogénéité avec l'existant sur Nautile et de par la particularité d'une telle solution dans le SI global INSEE, les coûts de maintenances et les potentielles failles de sécurité étant grandement augmentés par cette concomitance des accès en base. La solution a néanmoins pu être dockerisée pour des essais positifs en local.

6.3.2 Réutilisation du code R avec Renjin

Ensuite une seconde approche s'est faite par l'utilisation de Renjin pour l'intégration du package BalancedSampling au sein d'une bibliothèque Java/R. Ce package, passerelle entre R et Java permet en effet d'utiliser certains scripts R à l'intérieur d'un projet java et propose également une api de conversion de R à java de par la dimension interpréteur natif à la JVM de Renjin. Cela permettrait d'avoir plus de contrôle et d'harmoniser avec les environnements de production tout en profitant des avancées sur la méthodologie de tirage du cube, codée en C++, au sein d'un projet java. J'ai donc créé une librairie implémentant cette solution.

6.3.3 Création d'un projet de méthode du cube native Java

Ensuite j'ai décidé de créer un projet pour développer une solution native java, avec pour objectif de pouvoir donc apporter une maintenabilité et une testabilité plus grande que pour les solutions précédentes, pour lesquelles, j'avais moins d'affinités. La difficulté a ici été la manipulation de vecteurs et de matrices en Java. J'ai donc également créé une librairie implémentant cette solution pour la comparer à la solution précédente.

6.3.4 Pour aller plus loin

Au cours de mon stage j'ai pu découvrir GraalVM et native image. Il s'agit d'une bibliothèque permettant la compilation dans une machine virtuelle polyglotte cloud native. Cela offre des propriétés d'interopérabilité entre R et Java, permettant d'exploiter des programmes existants en R et des intégrations en Java simultanément. Cela implique également un boost très signifiant dans les potentialités micro-services et FAaS du genre de solutions que j'ai pu trouver déjà développées sur R. Cela a également des propriétés pour l'interopérabilité java/R. Je n'ai malheureusement pas eu le temps d'explorer plus de ce côté là, mais cela pourra être l'objet d'une prochaine investigation. De plus, la priorisation des tâches s'étant fait par anticipation d'un éventuel passage en production d'une implémentation, cette solution n'a pas été considérée comme viable actuellement.

6.4 Algorithmie

Cette partie regroupe les différentes modalités de l'algorithme du cube en précisant les implémentations / contrôles des points des plus complexes de celui ci.

Un diagramme d'activité du traitement est disponible en annexe A.3

6.4.1 Détermination du noyau de la matrice des contraintes A

Afin de générer un vecteur quelconque u appartenant au noyau de la matrice des contraintes A, et s'annulant dans le cas d'un échantillon strictement tiré ou non, j'ai procédé a la recherche d'une méthode algorithmique pour déterminer le noyau. La résolution de systèmes linéaires est un vaste sujet, mais dans le cas précis une méthode algorithmique fonctionnelle pour déterminer le noyau d'une matrice est la méthode de l'élimination de Gauss Jordan. Le principe est la réalisation d'un pivot de gauss sur la matrice A a laquelle l'on ajoute la matrice identité, les vecteurs restants de ce pivot sur la partie de la matrice identité forment donc les sous espaces des vecteurs J'ai donc tout d'abord recherché s'il existait une implémentation existante native de l'algorithme d'élimination de Gauss Jordan, et j'ai trouvé une implémentation issue d'un package sous licence GNU d'un manuel d'algorithmie de Princeton [13]. J'ai donc réalisé l'intégration de cette solution pour cette étape de l'algorithme après son étude.

6.4.2 Nécessité de vérifications sur l'aléa

Afin d'assurer une bonne représentativité de l'échantillon tiré, il a été question d'étudier la qualité de la fonction de génération de nombre sur une loi uniforme dans [0,1]. Pour cela, la méthode adéquate consistait en l'implémentation d'un test de Kolmogorov-Smirnov pour l'étude de la fonction de répartition continue de la loi uniforme U[0,1]. Le principe étant donc de construire la statistique de test de Kolmogorov sur un échantillon N important (1000000) par injection de dépendance de la fonction de génération de l'aléa. On a donc :

$$H_0 : F_N = F^0 \text{ vs } H_1 : F_N \neq F^0$$

en notant F^0 la fonction de repartion de la loi de probabilité testée. Ensuite, l'on génère un million de réalisation selon cet algorithme. Par exemple, pour l'algorithme **Math.random** de *java.util*, pour un risque de première espèce alpha à 15%, le test effectué a été concluant, et donc H_0 n'est donc pas rejetée.

Conclusion

L'objectif de ce stage était l'exploration de la démarche open source interne et de sa mise en contexte pour la réalisation d'un périmètre étendu, avec pour exemple l'implémentation d'un module de tirage équilibré. Il a permis une prise de contact avec différents acteurs de l'informatique et de la donnée au niveau INSEE et a été très intéressant d'un point de vue académique.

La réalisation complète et l'intégration d'une solution de tirage n'ont pas abouti à une intégration dans un environnement de production interne et les solutions n'ont pas été validées par une équipe statistique, toutefois cela sera probablement le cas lors d'échanges pour l'implémentation du tirage équilibré dans le cadre de tirage des unités primaires au niveau des DOM (soit dans un périmètre hors échantillon maître). Cela se fera normalement au courant de cette année scolaire 2020/2021.

Le stage m'a permis de voir que la structure interne était incubatrice de projets dans la direction du partage de code et m'a également permis de voir certains potentiels besoins comme par exemple le besoin d'un générateur de données relationnelles à volumétrie variable. De plus certains développement relatifs a l'ouverture du projet Nautile, projet assez représentatif d'un projet moyen a l'INSEE offrent des pistes intéressantes dans la documentation du processus d'opensourcisation des applications internes.

Enfin ce stage m'a permis d'élargir mes compétences en génie logiciel et dans l'expertise métier, avec le balayage de nombreuses notions importantes en méthodologie des sondages. Cela afin d'améliorer la prise de décision relative a mon poste d'analyste et aux responsabilités aux échéances proches inhérentes au poste de Responsable Applicatif de l'application Nautile. La vision globale des différents corps du monde informatique, innovation, développeurs et managers a également été un point d'attention crucial pour moi. Un objectif annexe du stage étant également d'échanger avec ces acteurs et de contextualiser le besoin du passage a l'open source des briques applicatives de l'administration INSEE à différentes échelles.

Bibliographie

[1] Raquel Campuzano. *Move a Custom Spring Boot Application to Production Using Bitnami Helm Charts*. URL : https://medium.com/@unicorn_dev/control-source-code-quality-using-the-sonarqube-platform-fe8f9904b6d9. 2019.

[2] Dalibo. *PostgreSql Anonymizer*. 2020.

[3] Lionel Delta. "Sélection des logements pour les enquêtes ménages dans les grandes villes : tirage équilibré versus systématique en présence de non-réponse." In : *Journées de Statistique : 2019* (2019).

[4] Unicorn Developer. *Control source code quality using the SonarQube platform*. URL : https://medium.com/@unicorn_dev/control-source-code-quality-using-the-sonarqube-platform-fe8f9904b6d9. 16 Novembre 2016.

[5] Ibrahima Ousmane Ida. "L'échantillonnage équilibré par la méthode du cube et la méthode rejective". In : *Université de Laval* (2016).

[6] OpenSource Initiative. URL : https://opensource.org/.

[7] Jérôme Mainaud. *anonymisation des donnees*. Bonnes pratiques d'anonymisation des données - Devoxx 2017.

[8] Laurent Costa Thomas Merly-Alpa. "L'échantillonnage équilibré". In : *Département des méthodes statistiques* (21 juin 2017).

[9] Dmitri Pavlutin. *How to make your open source project successful*. URL : https://dmitripavlutin.com/how-to-make-your-open-source-project-successful/. 6 Août 2019.

[10] PostgreSQL. *Documentation officielle - fonctions de randomisation*. URL : https://www.postgresql.org/docs/11/functions-srf.html. 2020.

[11] J.C. Deville Y. Tillé. "Efficient balanced sampling, the cube method". In : *Biometrika* (2004).

[12] Yves Tillé. "Dix années d'échantillonnage équilibré par la méthode du cube : une évaluation". In : *Statistiques Canada,Techniques d'enquête* (décembre 2011).

[13] Robert Sedgewick Kevin Wayne. *GaussJordanElimination Implementation Java*. URL : https://algs4.cs.princeton.edu/code/edu/princeton/cs/algs4/GaussJordanElimination.java.html.

[14] Ken Wheeler. *Medium : A Bitter Guide To Open Source*. 24 Mai 2018.

A - Annexe

A.1 Spécificités liées a la pandémie de COVID 19

La pandémie du COVID 19 a naturellement entrainé des conséquences sur le déroulement de ce stage. Tout, en ce qui concerne les développeurs de l'équipe projet Nautile, fraîchement remaniée en mars 2020, il a fallu assez vite s'adapter pour organiser le travail au niveau de l'équipe informatique. De plus, les contraintes de matérielles étant fortes de part l'imprévu, il a fallu attendre jusqu'à plusieurs semaines pour récupérer un matériel permettant officiellement de travailler sur le réseau interne de l'entreprise. Ainsi, au début du confinement et jusqu'à mi avril j'étais personnellement le seul équipé au niveau informatique et ai donc oeuvré a l'ouverture du code pour pouvoir permettre aux développeurs de travailler mais également pour permettre un travail plus efficient : Les contraintes réseau étaient également très importantes au début du confinement : Travail partiel, saturation du réseau entraînant des problématiques "d'input lag" lors d'utilisation de machines distantes, pertes de sessions. Donc la question du passage a l'Opensource du projet Nautile a dévié l'orientation d'une partie du stage, surtout qu'un suivi de prestation était à prévoir sur des développements conséquents de pré-mise en production.

De plus, des contraintes sont apparues sur la mise en production du projet (été 2020) ainsi que sur le suivi de prestation et des développements qui ont forcément impacté la place du stage dans l'organisation projet. Je suis personnellement assez content du travail que j'ai pu faire, j'aurais aimé aboutir à une solution bien marquée plutôt qu'a l'aboutissement de plusieurs pocs. Pour la suite, l'ouverture du code de Nautile se fera en début de maintenance en fonction des priorités mais surtout, l'intégration d'une méthode de tirage équilibré sera également une question à laquelle l'exploration menée par ce stage me permettront de répondre.

A.2 Organigramme du SNDIO

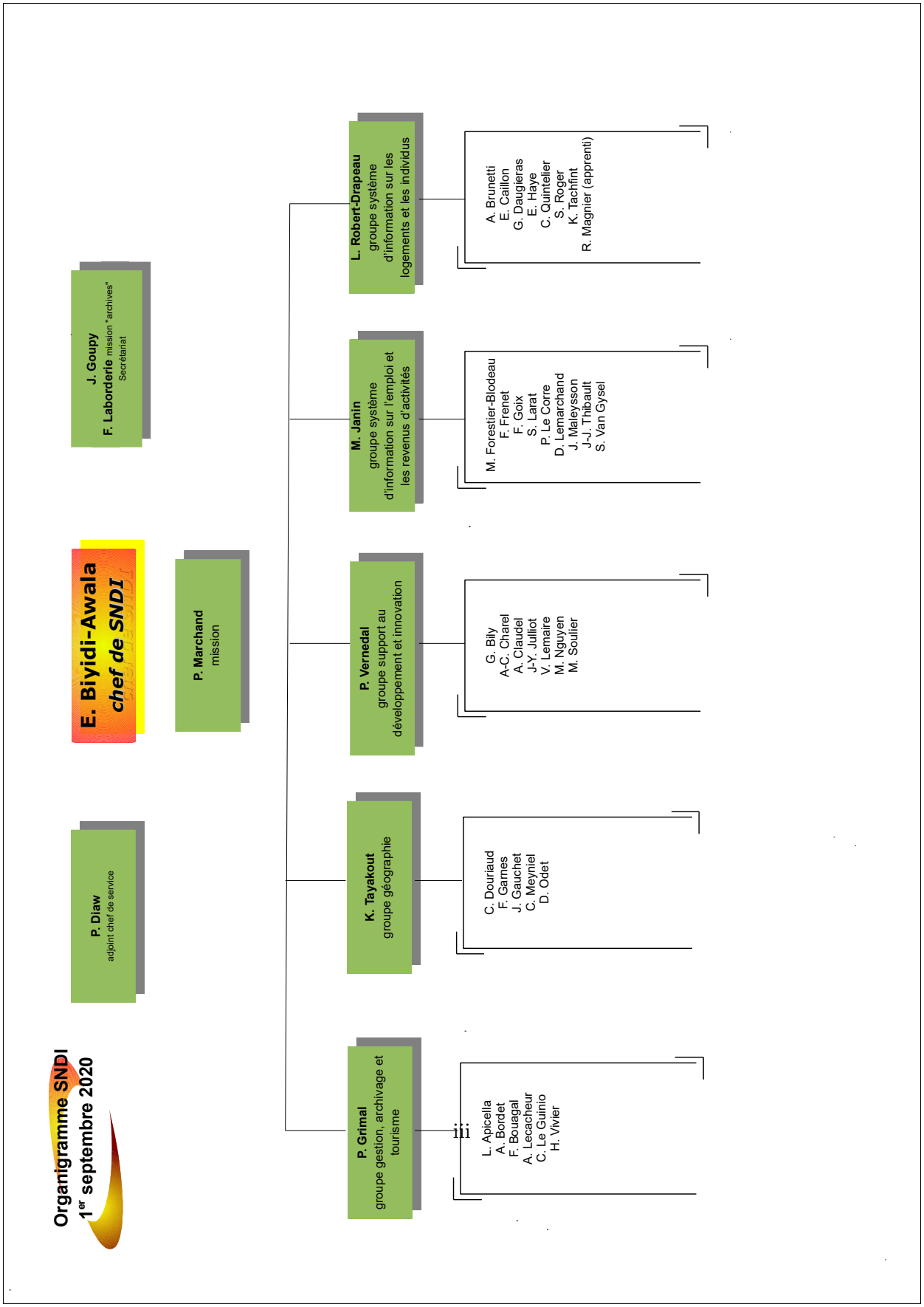

FIGURE A.1 – Organigramme du SNDIO

A.3 Diagramme d'activité de l'algorithme du cube

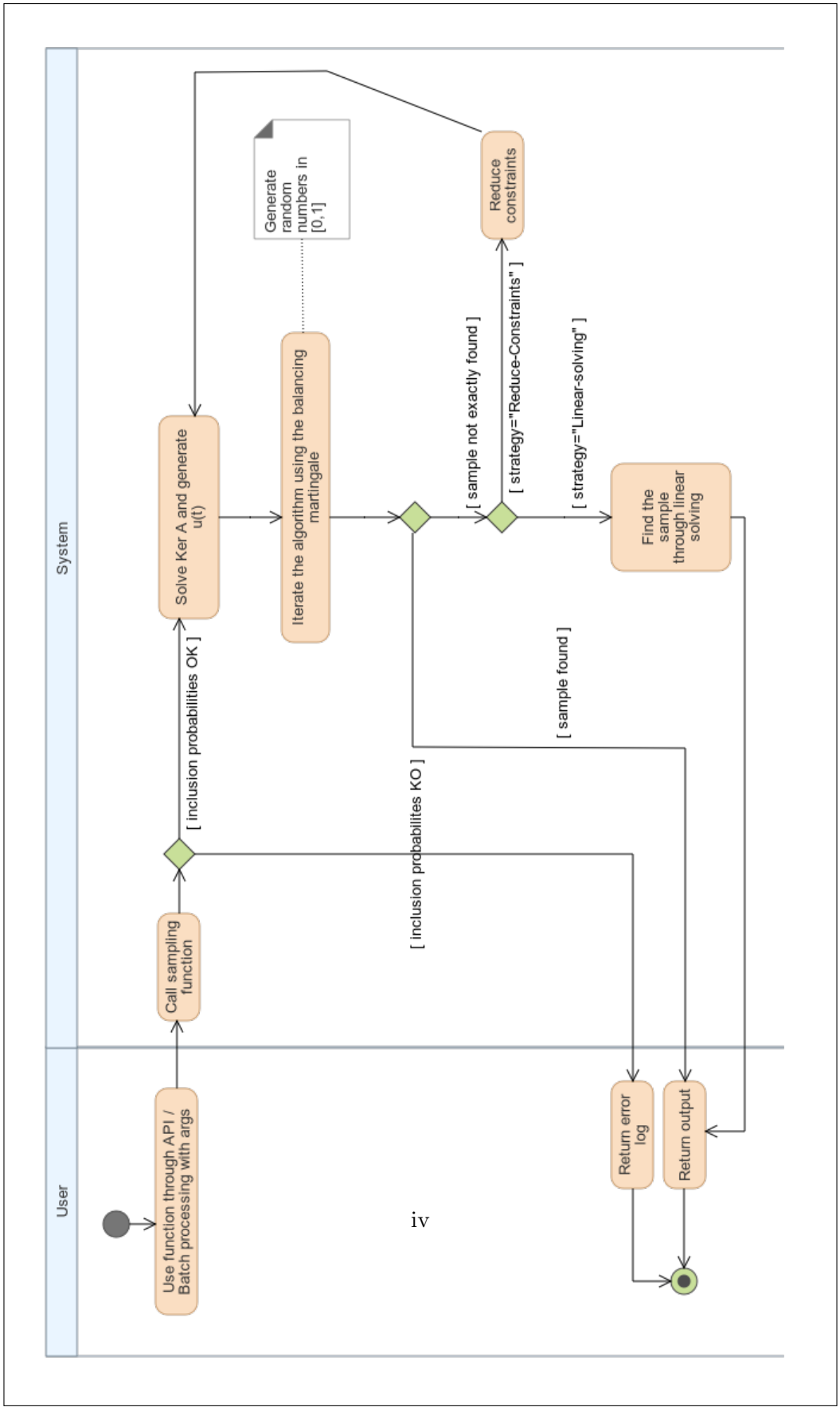

FIGURE A.2 – Diagramme d'activité de l'algorithme du cube

A.3.1 Commentaire du diagramme

Le diagramme d'activité présente le fonctionnement du traitement de tirage par méthode du cube.

Après appel de la fonction, un contrôle sur le vecteurs π est effectué, permettant d'initialiser la martingale équilibrante pour la matrice A construite par le vecteur x des contraintes de l'utilisateur. Ensuite, si ce contrôle aboutit, la phase de vol est initiée, sinon l'algorithme s'arrête et renvoie une erreur à l'utilisateur. La phase de vol est composée de la création de ce vecteur u, par l'implémentation de la méthode de gauss jordan et l'itération sur la martingale pour équilibrer le vecteur et donc aboutir a un potentiel échantillon s. A la fin de la phase de vol, soit un échantillon est trouvé et l'on retourne cet échantillon, soit il est nécessaire d'effectuer une phase d'atterissage. En paramétrant en entrée le paramètre "strategy", l'utilisateur peut définir la méthode a utiliser pendant la phase de vol, soit une reduction de contraintes pour une approche recursive sur l'échantillon restant avec des contraintes réduites, soit par résolution à l'aide d'une fonction de distance en entrée. Le traitement aboutit enfin à un échantillon.