UNIVERSITE SIDI MOHAMED BEN ABDELLAH
FACULTÉ DES SCIENCES ET TECHNIQUES FÈS
DÉPARTEMENT D'INFORMATIQUE

PROJET DE FIN D'ETUDES

MASTER SCIENCES ET TECHNIQUES
SYSTÈMES INTELLIGENTS & RÉSEAUX

ESTIMATION PROBABILISTE DE L'EFFORT DE DÉVELOPPEMENT LOGICIEL

LIEU DE STAGE : LABORATOIRE DES SYSTÈMES INTELLIGENTS ET APPLICATIONS (LSIA) À FST - FÈS

RÉALISÉ PAR : MONCIF EL-KASSIMI

SOUTENU LE 14/06/2017

ENCADRÉ PAR :

PR. L. LAMRINI

DEVANT LE JURY COMPOSÉ DE :

PR. L. LAMRINI
PR. A. ZAHI
PR. A. MAJDA
PR. A. BOUSHABA

ANNÉE UNIVERSITAIRE 2016-2017

Résumé

L'estimation de l'effort de développement logiciel est l'une des tâches les plus importantes dans le management de projets logiciels. Elle constitue la base pour la planification, le contrôle et la prise de décision. De nombreux modèles et techniques d'estimation existent dans la littérature, mais il est difficile d'identifier un modèle performant pour tous les types de projets,. Les projets de développement logiciel, en particulier pour ce qui concerne l'activité d'estimation, se caractérisent par la présence d'importantes incertitudes, surtout en phases amont. Généralement, ces modèles ne prennent pas toutes les sources éventuelles d'incertitudes. Le travail de ce PFE s'intéresse à améliorer un framework d'estimation probabiliste. Ce framework se base sur des métriques issues de la modélisation UML. Notre contribution se focalise alors, à ajouter des nouveaux composants prenant en compte d'autres sources d'incertitudes, la version améliorée du framework estime l'effort sous forme d'un intervalle de prédiction avec à un degré de confiance. Cette solution assiste les chefs de projets dans l'aide à l'estimation de l'effort de leurs projets logiciels.

MOTS-CLÉS : Sélection de modèle d'estimation, Incertitude, Validation croisée, Réseaux de neurones, Forêts d'arbres décisionnels, Bootstrap.

Abstract

Software effort estimation is one of the most important tasks in the management of software projects. It is the basis for planning, control and decision making. Many estimation models exist, but it is difficult to identify a successful model for all types of projects and that is applicable to all companies (different levels of experience, mastered technologies and project management practices). Generally, these models do not take all possible sources of uncertainty. The work of this project is interested in improving a probabilistic estimation framework. This framework is based on metrics derived from UML modeling. Our contribution is to add new components taking into account other sources of uncertainty, the improved version of the framework estimates the effort in the form of a prediction interval with a degree of confidence. This solution assists project managers in helping to estimate the effort of their software projects.

KEYWORDS : Estimation model selection, Uncertitudy, Cross validation, Neural networks, Random forests, Bootstrap.

Remerciements

Premièrement je tiens à remercier mon encadrante Madame Loubna LAMRINI, pour le privilège qu'elle m'a fait en acceptant de diriger ce travail. Sa gentillesse, sa modestie, sa riche expérience et l'accueil cordial qu'elle m'a toujours réservé.

Je tiens à remercier également, toute l'équipe pédagogique de la Faculté des Sciences et Techniques de Fès, et les intervenants professionnels responsables de la formation Master Systèmes Intelligents et Réseaux (SIR). Je remercie ainsi, Monsieur Jamal KHARROUBI le coordinateur du Master pour les efforts déployés pendant cette formation.

Finalement, Je tiens à remercier sincèrement les membres du jury qui me font le grand honneur d'évaluer ce travail.

Qu'ils puissent trouver dans ce travail le témoignage de ma sincère gratitude et de mon profond respect.

Merci.

Dédicaces

Je dédie travail,

À ma chère mère qui a toujours étée avec moi pour moi tout au long de mes études et qui m'a donnée un magnifique modèle de labeur et de persévérance. J'espère qu'elle trouvera dans ce travail toute mes reconnaissances et tout mon amour. Ce travail est également un hommage à mon père qui nous a quitté.

Liste des figures

Liste des Tableaux

Glossaire

Biais : L'erreur systématique, il représente la distance entre la valeur réelle et la valeur estimée. Dans le contexte de l'estimation de l'effort, le biais représente la distance entre l'effort réel et l'effort estimé. Le biais d'une méthode de validation est la différence entre la performance réel et la performance estimée/évaluée du modèle d'estimation par cette méthode de validation.

Hyper-paramètres : Les hyper-paramètres d'un modèle non paramétrique sont les variables qui contrôlent la configuration de ce modèle.

Inconstance : Disposition à se changer d'une étude à une autre.

Système d'estimation : Un modèle ou ensemble de modèles d'estimation calibrés sur la base de données étudiée. Il est prêt à être utiliser pour l'estimation d'efforts de nouveaux projets.

Performance prédictive : La capacité d'un modèle d'estimation à produire des estimations d'efforts proches des efforts réels.

Modèles d'estimation paramétriques : Sont les modèles ayant une forme bien définie et un nombre fixe de paramètres. Les paramètres des modèles d'estimation paramétriques sont appelés les coefficients.

Liste des Acronymes

COCOMO	Constructive Cost Model
FDP	Fonction de Densité de Probabilité
k-PPV	k-Plus Proches Voisins
MMRE	Mean Magnitude of Relative Error
MRMD	Mean Relative Mean Difference
PCU	Points de Cas d'Utilisation
Pred	Prediction accuracy at level
SLIM	Software Lifecycle Management
Und	Under estimation proneness at level
FAD	Forêts d'Arbres Décisionnels
IvP	Intervalle de Prédiction
MdMRE	Median Magnitude of Relative Error
MRE	Magnitude of Relative Error
Ovr	Over estimation proneness at level
PF	Points de Fonction
RLM	Régression Linéaire Multiple
RNA	Réseaux de Neurones Artificiels
VCK	Validation Croisée "k-fold »
RB	Régression bayésienne
MC	Monte Carlo

Table des matières

Introduction générale

Actuellement beaucoup des entreprises dans le monde s'intéresse à développer des projets logiciels, et pour qu'une entreprise spécialiste dans ce domaine répond à un appel d'offre d'un projet logiciel et négocier les contrats d'une manière satisfaisante elle doit savoir tout d'abord le coût qu'il doit fournir pour réaliser le projet.

Chaque organisation spécialiste dans le développement logiciel, possède un certain nombre de développeurs et de concepteurs avec des chefs de projets qui travaillent tous en collaboration pour implémenter toutes les spécifications écrites dans le cahier de charges d'un projet logiciel. Suivant ce contexte **le coût** ou **l'effort** signifie la quantité des personnes (développeurs, concepteurs,…) nécessaire pour réaliser un projet logiciel, son unité de mesure la plus répandue est hommes-mois. Cette mesure nous permet de répondre à la question : combien l'entreprise a besoin des personnes pour réaliser un tel projet dans la période d'un mois ?

Au début des années 80 les informaticiens ont essayé de répondre à cette question en proposant un ensemble de méthodes comme par exemple la méthode de points de fonction proposée par Albrecht en 1979. Au fur et à mesure de l'évolution de développement informatique, ces méthodes viennent de devenir informelles car elles sont non adaptées pour estimer l'effort en prenant en considération les avantages et inconvénients concernant les nouveaux langages de programmation apparus (Java, python, R, …) et également les nouvelles méthodes de conception (orientée objets, méthodes agiles …).

Le but ultime des entreprises en général est d'avoir des gains matériels au bout de développement de chaque projet, et en particulier dans le domaine de développement informatique lorsque l'entreprise estime d'une façon réaliste l'effort, ça lui permettra d'une part de bien négocier le contrat avec le client et d'autre part elle va éviter toute sorte éventuelle de l'échec et l'abandon de projets, et parmi les facteurs d'échec des projets logiciel on retrouve la mauvaise estimation du coût (effort), ce qui veut dire que le coût est un paramètre essentiel et même vital pour la réussite du projet logiciel.

Lorsqu'on on veut estimer une valeur en général on se retrouve toujours face au problème d'incertitude, et plus particulièrement, l'activité de l'estimation de l'effort du développement logiciel se caractérise par une importante incertitude. La présence de l'incertitude peut entrainer :

— Une surestimation : l'entreprise pourra perdre l'opportunité de réaliser d'autres projets ;

— Une sous-estimation : peut avoir des conséquences négatives au niveau de la qualité de logiciel produit. La mauvaise qualité due à l'empressement des développements pour terminer le projet à la date livraison.

Les questions qui se posent alors, sont :

— **Q.1** Comment estimer l'effort de développement logiciel d'une façon probabiliste ?
— **Q.2** Comment prendre en considération les incertitudes relatives aux informations sur le nouveau projet logiciel ?

Il existe beaucoup des travaux qui permettent de répondre à ces deux questions [1]. Dans le cadre de projet de fin d'études Master, nous répondons aux questions posées par l'amélioration un framework qui existe déjà et répond seulement à **Q.1**. Notre contribution se focalise alors à mettre à niveau ce framework pour qu'il soit capable de répondre à la fois à **Q.1** et **Q.2**.

Organisation du document

Après l'introduction générale, la suite de ce document est composée de 4 chapitres :

— Chapitre I fait l'état de l'art autour de la problématique du PFE. Dans un premier temps, nous classifions les différentes techniques existantes sur l'estimation de l'effort de développement logiciel. Dans un deuxième temps, nous présentons les incertitudes inhérentes au processus d'estimation. Finalement nous posons le problème de la gestion des incertitudes dans le processus d'estimation.

— Chapitre II explique tout d'abord le fonctionnement des composants d'un framework déjà proposé, puis nous présentons ses limitations. Ensuite nous proposons une architecture générale de la version améliorée du framework qui répond à Q.1 et Q.2. Finalement nous donnons une description brève de différentes étapes ajoutées au framework.

— Chapitre III contient une description détaillée des nouvelles étapes ajoutées pour améliorer le framework. Il se focalise dans un premier lieu sur la procédure de sélection de modèle d'estimation par à une base de données historique, puis nous présentons la démarche de construction du système d'estimation d'effort. Il présente après la procédure d'utilisation du système d'estimation d'effort construit dans l'estimation d'un nouveau projet caractérisé par des incertitudes.

— Chapitre IV expérimente et valide l'approche utilisée pour l'amélioration du framework sur la base de données étudiée. Nous exposons ensuite, dans le même

chapitre, l'application implémentée pour automatiser toutes les étapes du framework.

Chapitre I : Cadre de travail

Introduction

L'estimation du logiciel est l'une des plus importantes activités de la planification du projet de développement de logiciel, et est définie comme un processus de prédiction de la durée et du coût du projet logiciel [2]. Ce processus est appliqué avant de commencer la phase de développement du projet, afin de prédéterminer le temps à prévoir, le budget à obtenir et les fonctions du logiciel à réaliser afin de mener à bien le développement du logiciel.

En général on trouve deux axes différentes dans la recherche concernant l'estimation de l'effort logiciel, la (Fig. 1) illustre les deux axes.

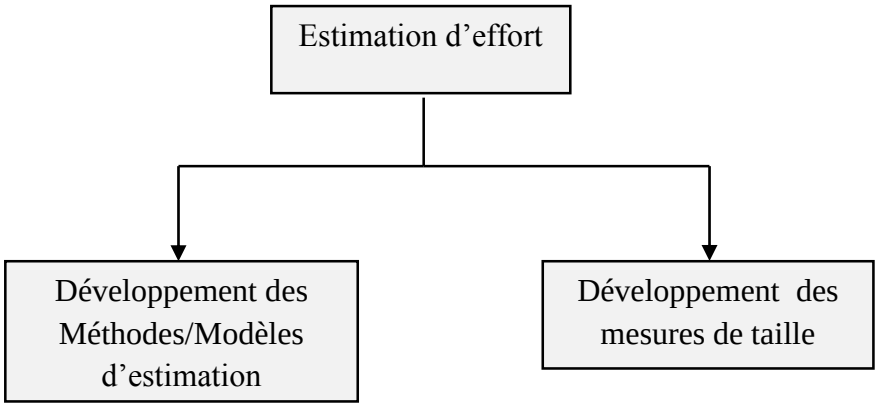


Figure 1 - Les différents domaines d'estimation d'effort [3]

Le développement des mesures de taille peut être utilisé comme une couche intergiciel pour estimer l'effort, sachant que l'objectif de l'ingénierie logicielle est de développer des modèles utiles qui peuvent expliquer le cycle de vie du logiciel et d'estimer avec précision le coût/l'effort de développement.

Après avoir identifié, dans l'introduction, l'objectif de ce travail, on va présenter tout d'abord dans ce chapitre, le lieu du stage où nous avons réalisé ce travail, ensuite nous entamons l'état de l'art relatif aux techniques d'estimation, les mesures de taille logiciel, les incertitudes et leurs sources, et les différents critères d'aide à la sélection de modèle d'estimation.

I.1 Lieu de stage

Ce stage de fin d'études a été effectué dans le Laboratoire des Systèmes Intelligents et Applications (LSIA) au sein de la Faculté des Sciences et Techniques de Fès.

I.1.1 Présentation du Laboratoire SIA

Le laboratoire SIA, crée en 2011, est une unité de Recherche du Centre d'Études Doctorales en Sciences et Techniques de l'Ingénieur domicilié à la Faculté des Sciences et Techniques de Fès et regroupant des laboratoires de recherche tous accrédités par l'Université Sidi Mohamed Ben Abdellah de Fès, et domiciliés à la Facultés des Sciences et Techniques, l'École Supérieure de Technologie, la Faculté Polydisciplinaire de Taza, l'École Nationale des Sciences Appliquées de Fès et l'ENS de Fès.

Le LSIA est composé de 15 enseignants-chercheurs du département d'Informatique de la FST de Fès et de 34 doctorants. Cette imbrication étroite entre enseignement et recherche, est un élément essentiel de la dynamique du laboratoire.

Les thématiques de recherche se situent au cœur des Sciences et Technologies de l'Information et de la Communication et s'articulent essentiellement autour des thématiques de recherche des enseignants chercheurs du laboratoire et assure une large couverture thématique présentant un atout très important pour le laboratoire.

I.1.2 Equipes de recherche

Le laboratoire est composé de 3 équipes de recherche :

- Environnement Intelligent et Applications ;
- Systèmes de Communication et Traitement de Connaissances ;
- Vision Artificielle et Systèmes Embarqués.

Figure 2 - logo du laboratoire LSIA

I.2 Mesures de taille d'un projet logiciel

Le premier axe de recherche consiste à trouver des mesures pour estimer la taille du logiciel à développer. Il existe un certain nombre de mesures de taille dans la littérature, et chacune de ces mesures diffère par rapport au type d'artefact logiciel qu'elles utilisent pour l'estimation comme (le code source, les cas d'utilisation, les diagrammes de classe,…).

Beaucoup de ces mesures ont été utilisées pour estimer l'effort de développement logiciel. En outre, différents paramètres peuvent être utilisés dans différentes phases du

développement de logiciel. Ces paramètres ont des diverses capacités d'estimation. La (Fig. 3) représente l'estimation de l'effort effectuée à différentes phases du développement de logiciel.

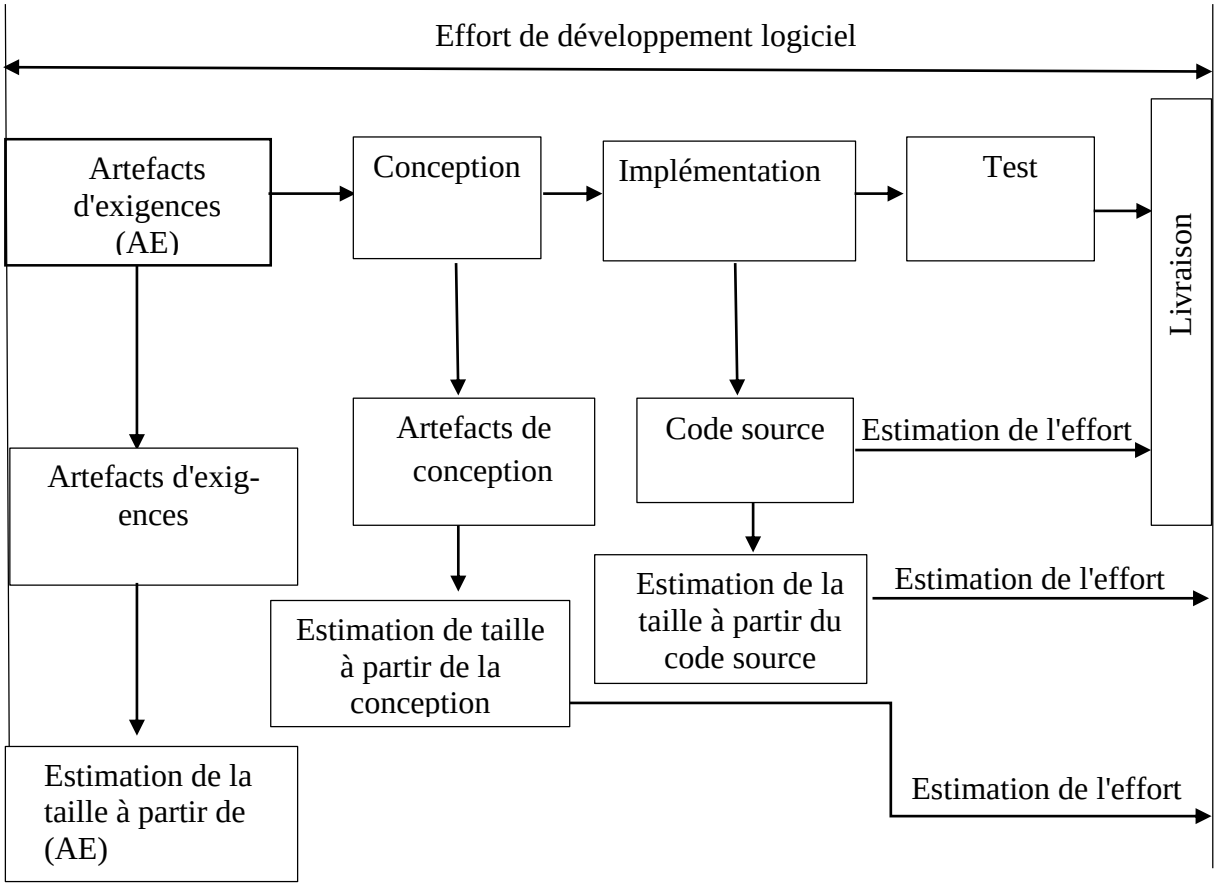


Figure 3 - Estimation de l'effort durant les différentes phases de développement logiciel [4]

I.2.1 Ligne de code

Le nombre de milliers de ligne de code représente la "taille" du projet. C'est principalement en fonction de cette "taille" que l'on estimera le temps nécessaire à la réalisation du projet.

On considère la définition de la ligne de code donnée par [5] : « Une ligne de code est toute ligne du texte d'un programme qui n'est pas une ligne de commentaire, ou une ligne blanche, sans considération du nombre d'instructions ou de fragments d'instructions dans la ligne. Sont incluses toutes les lignes contenant des en-têtes de programmes, des déclarations, et des instructions exécutables et non exécutables. »

I.2.2 Points de fonction (PF)

La méthode des points de fonction a été introduite par Albrecht en 1979, puis, après la formation de l'organisation IFPUG[1], cette méthode a été standardisée afin de clarifier les règles

[1] IFPUG est l'organisation « International Function Point User Group»

de la méthode et de promouvoir son utilisation [6]. Brièvement, la méthode tente d'estimer la taille d'un système logiciel selon des fonctions prédéfinies; ces dernières sont obtenues à partir des documents générés dans les phases préliminaires comme la phase d'analyse et la phase de conception. La méthode des points de fonction comporte les étapes. Tout d'abord, on identifie les composants du système en fonction des cinq types suivant :

- Le type d'entrée identifie les données entrées par l'usager dans l'application ;
- Le type de sortie indique les données qui sortent de l'application et qui sont souvent le résultat du traitement d'une application ;
- Le type GDI (Groupe logique de Données Internes) représente les groupes de données logiquement liées ou des paramètres de contrôle qui sont mis à jour et utilisés à l'intérieur d'une application du système ;
- Le type GDE (Groupe de Données Externes) identifie les fichiers d'interfaces, qui sont mis à jour par une autre application ;
- le type d'interrogation concerne les requêtes interactives qui exigent une réponse.

Pour chaque composant du système, on détermine son niveau de complexité, classifié comme Faible, Moyen et Élevé. Chaque niveau de complexité d'un composant correspond à un poids prédéfini, qui varie de 3 à 15.

Le processus de comptage donne le nombre de points de fonction brut. Après avoir compté le nombre de points de fonction de chaque type, le calcul des points de fonction du système se fait en ajoutant les poids des différents types de composants, il est décrit par la formule suivante (Eq. 1) :

$$NPF = \sum NF \times Poinds \quad (1)$$

Où NPF est le nombre de points de fonction brut, NF est le nombre des fonctions dans chaque composant, Poids représente le poids associé à chaque type.

I.2.3 Points de cas d'utilisation

Le point de cas d'utilisation est une mesure dérivée des exigences du client exprimées sous la forme d'un diagramme de cas d'utilisation. Cette mesure est calculée en s'appuyant sur les éléments des cas d'utilisation et prend en compte des considérations techniques et environnementales.

I.3 Techniques et modèles d'estimation de l'effort

Les Techniques d'estimation d'effort sont divisées à deux catégories :

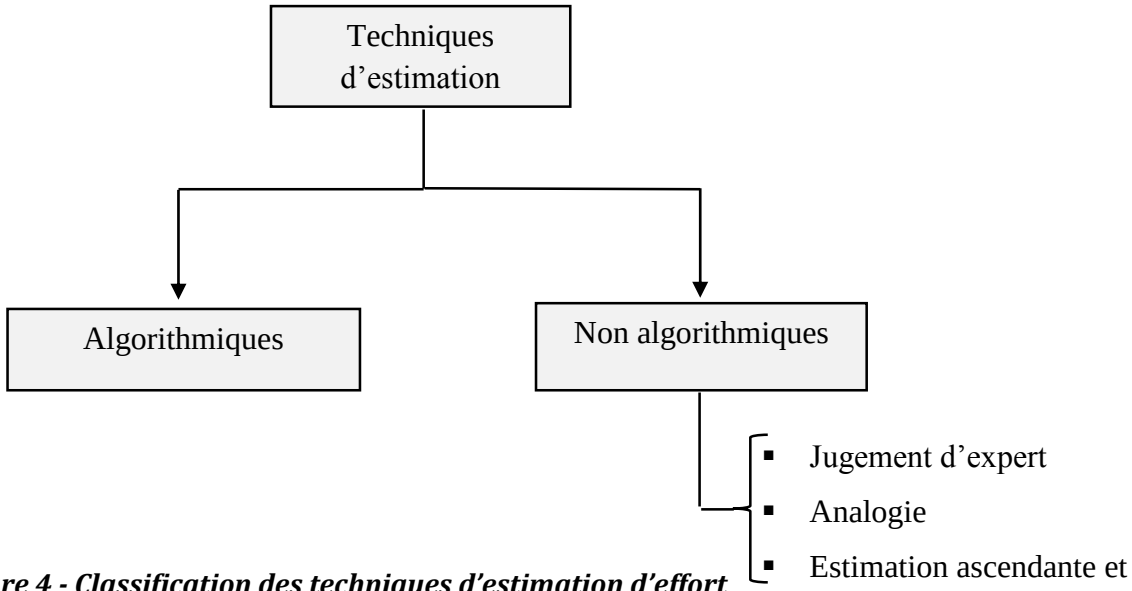


Figure 4 - Classification des techniques d'estimation d'effort

I.3.1 Technique non algorithmique

Les techniques non algorithmique sont informelles reposent sur l'expérience d'une ou plusieurs personnes dites experts. Parmi ces méthodes nous en citons : le jugement d'expert, l'estimation par analogie, etc.

I.3.1.1 Jugement de l'expert (méthode Delphi)

Cette technique consiste à consulter un ou plusieurs experts qui utilisent leurs expériences ainsi que leur compréhension du projet afin de fournir une estimation à son coût. Elle était parmi les premières techniques utilisées en estimation des coûts. Il est préférable d'avoir plusieurs valeurs estimées du coût émanant de plusieurs experts pour alléger les problèmes de subjectivité, de pessimisme et d'optimisme. Il y a plusieurs façons de combiner les différentes valeurs estimées du coût d'un projet logiciel. La plus simple est d'utiliser une des méthodes statistiques telles que la moyenne, la médiane ou le mode de toutes les valeurs pour déterminer une estimation du coût. Cette approche présente l'inconvénient d'être sujette aux préjugés des estimations extrêmes. Une autre méthode référée souvent dans la littérature par la méthode Delphi, consiste à réitérer ce processus d'estimation plusieurs fois jusqu'à ce qu'un consensus soit établi. L'idée principale de la méthode Delphi est qu'un coordinateur distribue une description du projet logiciel (souvent le dossier d'analyse des besoins du logiciel) ainsi qu'un formulaire d'estimation aux experts; ensuite, les experts remplissent les formulaires et les renvoient au coordinateur. Ce dernier pourra, dépendamment de la version de la méthode

Delphi, convoquer une réunion des experts pour discuter des différentes estimations. Ce processus est réitéré jusqu'à l'adoption d'une estimation par tout le groupe

I.3.1.2 Estimation par analogie

Dans cette technique d'estimation, l'évaluateur compare le projet logiciel dont on veut estimer le coût avec des projets logiciels déjà achevés. Il doit identifier les similarités et les différences entre le nouveau projet et tous les anciens projets. Pour cela, il doit auparavant décrire les projets logiciels par un ensemble de caractéristiques. Les ressemblances entre les caractéristiques du nouveau projet avec celles des projets achevés déterminent les degrés de similarité entre les projets. Ensuite, il utilise ces degrés de similarité pour déterminer une estimation au coût du nouveau projet. Cela est illustré par la formule suivante (Eq. 2):

$$S = \sum F \times Taille \qquad (2)$$

Où « S » signifie la taille estimée du système entier, «F» est un paramètre déterminé par l'expérience ou la politique d'estimation, et le paramètre « Taille» représente la taille du module similaire du projet achevé.

L'estimation par analogie est basée sur les données historiques de chaque organisation de développement, l'avantage qu'elle offre est sa simplicité d'utilisation; mais à cause de sa dépendance aux données historiques, le modèle présente certains inconvénients comme la restriction du type de langage employé et l'environnement d'application, ainsi que la difficulté de l'appliquer dans différentes organisations [7].

I.3.1.3 Estimation ascendante et descendante

L'approche ascendante consiste à décomposer le système en plusieurs modules selon la classification des tâches à accomplir. Un arbre sera construit avec les tâches représentées par les nœuds de l'arbre: on commence à estimer les efforts nécessaires des sous-tâches au niveau le plus bas dans l'arbre et on en rassemble les résultats afin d'obtenir les efforts des tâches au niveau supérieur; on atteint ainsi progressivement le résultat estimé du système global.

L'approche descendante entreprend de livrer une estimation d'effort à partir du système global, en se basant sur le jugement d'expert ou l'estimation par analogie des données. Par la suite, l'effort estimé total sera réparti et assigné aux composants du système à développer.

I.3.2 Techniques algorithmique

Les techniques algorithmiques sont basées sur l'apprentissage automatique. C'est un domaine de recherche relativement plus récent. Ceux-ci comprennent les réseaux bayésiennes, la logique floue, les réseaux neuronaux artificiels, etc.

Les techniques d'estimation algorithmiques sont classifiées selon la (Fig. 5) :

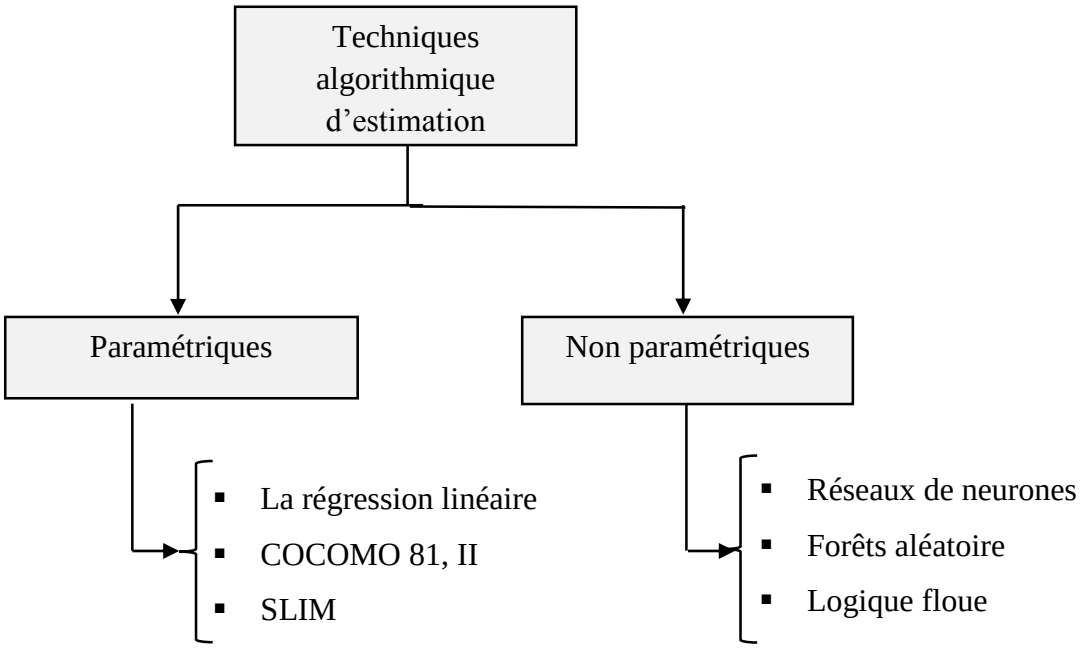


Figure 5 - Classification des techniques d'estimation algorithmiques

— Techniques paramétriques : Ce genre de modèle est toujours basé sur une équation mathématique ; une fonction f représente la relation entre l'effort de développement et les facteurs impliqués dans le développement tels que la taille du logiciel, l'expérience du personnel, la complexité du système à implémenter, le degré de réutilisation, etc. Ces facteurs, qui affectent le coût du développement, seront estimés au préalable et considérés comme des variables d'entrée principales dans la mise au point du modèle. Les techniques les plus adoptées du modèle sont listées dans ce qui suit [8].

I.3.2.1 Régression linéaire

Le modèle linéaire dans le domaine d'estimation d'effort peut s'écrire sous la forme (Eq. 3)

$$Effort = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \cdots + \beta_n X_n \qquad (3)$$

Où les X_i ont les facteurs affectant l'effort et les β_i sont des coefficients choisis pour fournir le meilleur ajustement à l'ensemble des données historiques observées.

I.3.2.2 COCOMO 81

COCOMO est l'acronyme de **C**onstructive **C**ost **M**odel. C'est une méthode pour estimer le coût d'un projet logiciel dans le but d'éviter les erreurs de budget et les retards de livraison, qui sont malheureusement habituels dans l'industrie de développement logiciel.

Il a été développé par Dr. Barry Boehm en 1981 pour estimer le coût, en s'appuyant sur une donnée de base : le nombre de ligne de code source prévisible, exprimé en Kilo Instruction

Source Livrées (KISL). A l'origine il a été construit sur une étude de 63 projets logiciels de 2000 à 100.000 lignes de code.

Il permet de formuler des estimations de l'effort et du temps de développement ainsi que ceux de maintenance d'un logiciel. En plus, il fournit la répartition de cet effort sur les quatre phases du cycle de développement (Analyse, Conception, Codage et Tests unitaires, Intégration et Tests) et sur les huit activités de chaque phase (Analyse de besoins, Conception, Programmation, Plan de test, Vérification et Validation, Gestion du projet, Gestion des configurations et Assurance qualité, Documentation).

I.3.2.3 COCOMO II

Le modèle COCOMO II il basé sur l'idée d'améliorer et de réactualiser le modèle COCOMO'81 en considérant les besoins suivants :

— Développer un modèle d'estimation fondé sur une architecture de processus mieux adapté que le cycle en V aux nouvelles pratiques de la programmation, en particulier en se positionnant sur la problématique du maquettage / prototypage, de l'utilisation des progiciels et des perspectives offertes par l'approche composants réutilisables comme les logiciels « libre/gratuit » ;

— Adapter les paramètres qualitatifs du modèle intermédiaire de COCOMO pour prendre en compte les progrès en ingénierie du logiciel, ou tout simplement la plus grande diversité des projets, tant en ce qui concerne les coûts que les durées de réalisation des logiciel.

Le modèle COCOMO II prend en compte les pratiques prévisibles de développement du logiciel aux états unis, à l'horizon 2005, sur la base d'une répartition sociologique des activités relevant de l'informatique en cinq secteurs, selon le niveau d'expertise informatique requis.

I.3.2.4 Réseaux de neurones artificiels

La modélisation par les réseaux de neurones est inspirée de la structure biologique du cerveau humain. Un réseau de neurones est caractérisé par son architecture, son algorithme d'apprentissage et ses fonctions d'activation. Dans la littérature, plusieurs modèles de réseaux de neurones ont été développés. Ils peuvent être classifiés en deux catégories principales : les réseaux de neurones « feedforward » où il n'y a aucune boucle dans l'architecture du réseau, et les réseaux récurrents où une ou plusieurs boucles récursives apparaissent dans l'architecture du réseau. Le Perceptron multicouches utilisant l'algorithme d'apprentissage de rétro-propagation est souvent le plus adopté en estimation des coûts de logiciels.

Dans ce modèle, les neurones sont organisés en couches et chaque couche ne peut avoir des connexions que vers la couche suivante. La (Fig. 6) illustre un exemple d'un tel réseau pour l'estimation de l'effort de développement de logiciels. Le réseau produit un résultat (effort) en propageant ses entrées initiales (facteurs du coût ou attributs du projet logiciel) à travers les différents neurones du réseau jusqu'à la sortie. Chaque neurone d'une couche calcule sa sortie en appliquant sa fonction d'activation à ses entrées. Souvent, la fonction d'activation d'un neurone est la fonction Sigmoïde définie par :

$$f(x) = \frac{1}{1 + e^{-x}} \qquad (2)$$

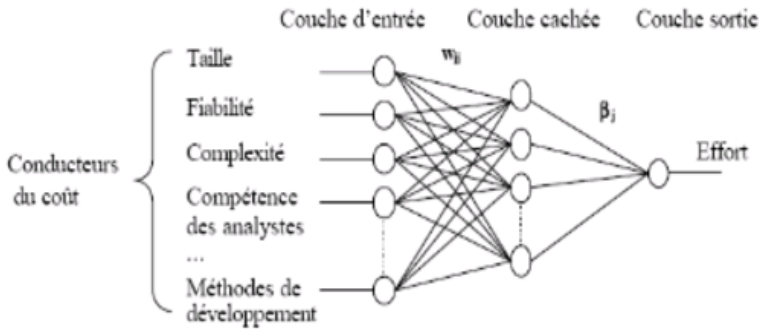


Figure 6 - Architecture d'un réseau de neurones (Multicouches) pour l'estimation de l'effort [3]

I.3.2.5 Arbres de régression

Les arbres de régression, en particulier binaires, ont été utilisés avec succès pour la mise au point de plusieurs modèles d'estimation en génie logiciel. En estimation des coûts, ils ont été utilisés pour estimer l'effort de développement d'un logiciel. Premièrement, chaque projet logiciel est décrit par un ensemble de variables qui serviront à la prédiction de son effort de développement. Ensuite et progressivement, à partir de la racine de l'arbre, le projet logiciel auquel on veut estimer le coût emprunte le chemin approprié, selon les valeurs de ses variables, en descendant dans l'arborescence jusqu'à l'une des feuilles de l'arbre. Chaque feuille contient une valeur numérique représentant l'effort de développement des projets logiciels associés.

I.4 Critères de validation d'un modèle d'estimation

Plusieurs critères de validation ont été proposés dans la littérature pour valider un modèle d'estimation d'effort, nous citons dans ce chapitre trois indicateurs de performance les plus répandus.

I.4.1 Magnitude de l'erreur relative

La magnitude moyenne de l'erreur relative MMRE (Magnitude of the relative Error) est l'indicateur de performance le plus communément utilisé pour ce propos. Un bon modèle d'estimation correspond à un MMRE inférieur ou égale à 0.25 (Eq. 4) :

$$MRE = \left|\frac{E_{actuel} - E_{estimé}}{E_{actuel}}\right| \quad (4)$$

Avec E_{actuel} et $E_{estimé}$ sont respectivement l'effort actuel et estimé.

MMRE est la moyenne de MRE pour tous les projets dans l'ensemble de test.:

- — ***MMRE*** ≤ 0.25 est considérée comme acceptable.
- — ***MMRE*** = 0.2 est considérée comme prédictive
- — *0.2*< ***MMRE*** <*0.5* est bien acceptable
- — ***MMRE*** > *0.5* est inacceptable à des fins de planification.

MRE pénalise la surestimation plus que la sous-estimation. Une solution alternative permettant d'éviter l'influence des MRE extrêmes ou aberrants est l'utilisation de la médiane des *MRE* (*MdMRE*). (Eq. 5) plutôt que la moyenne. Le *MdMRE* est considéré plus représentatif des estimations typiques que la moyenne *MRE*.

$$MdMRE = Médiane_{i=1}^{n}(MRE_i) \quad (5)$$

I.4.2 Niveau de signification

Pred est un autre indicateur de performance assez fréquemment utilisé dans la littérature. Il représente la probabilité qu'un projet ait une erreur inférieure ou égale à l. Il est défini dans (Eq. 6) avec l est le niveau de tolérance et p est le nombre de projets ayant une MRE inférieure ou égale à l. Généralement, l prend la valeur 0.25. un bon système d'estimation doit avoir un Pred(0.25) supérieur ou égal à 0.75.

$$Pred(l) = \frac{P}{n} \quad (6)$$

I.5 Problème d'incertitude dans l'estimation d'effort

L'incertitude d'une estimation d'effort de développement logiciel signifie : le doute sur la validité de cette estimation. Ce doute est généralement la conséquence, entre autres, d'une connaissance incomplète, imprécise et/ou limitée des projets de développement logiciel à estimer et de l'instabilité de l'environnement de développement [9]

Comme le développement du projet logiciel est un processus complexe, les facteurs d'incertitude entraînent des inexactitudes dans l'estimation logicielle. Selon Sommerville [10], il est difficile de procurer une estimation rigoureuse de l'effort de développement dans les phases initiales de développement de logiciel, parce que l'estimation logicielle se base sur la spécification des besoins de la clientèle à un haut niveau; celle-ci comprend des ambiguïtés, des confusions et l'estimation est souvent modifiée au cours de la construction à cause des facteurs humains et des contraintes en ressources matérielles.

McConnell a conclu que l'estimation logicielle est un processus de raffinement progressif [11] cela veut dire que la taille de l'incertitude en estimation logicielle sera diminuée et minimisée progressivement au fur et à mesure de l'avancement du développement de logiciel. Dans la (Fig. 7), McConnell représente ce fait et il constate que, théoriquement, il est impossible d'avoir une estimation précise dans les phases initiales du développement de logiciel, comme la phase d'analyse de faisabilité, la phase d'analyse des besoins, et la phase de conception.

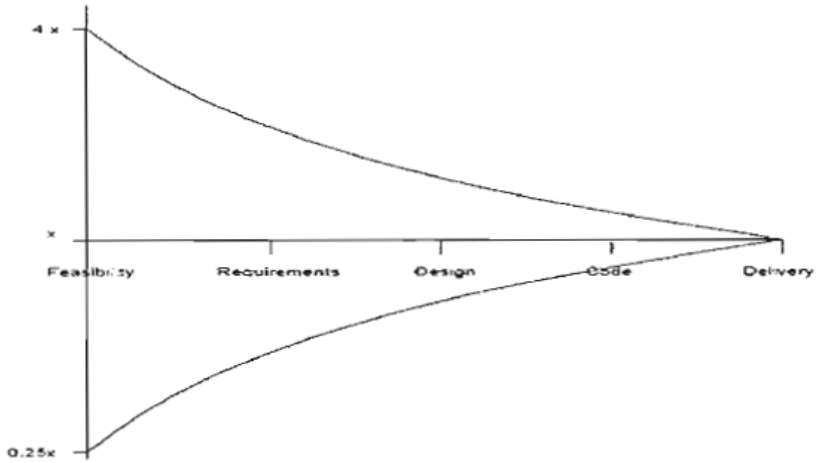


Figure 7 - Les incertitudes dans l'estimation logicielle par McConneli [3]

Dans cette partie nous cherchons tout d'abord à identifier les différentes sources d'incertitudes éventuelle d'incertitude dans le processus d'estimation, puis nous mettons en évidence la nécessité de modéliser les incertitudes, et la fin nous présentons une manière répandue pour exprimer les incertitudes dans le processus d'estimation.

I.5.1 Sources d'incertitudes

Plusieurs composants contribuent à l'estimation de l'effort de développement logiciel, chacun de ces composants peut avoir un aspect important d'incertitude qui ne doit être négligé dans la valeur finale de l'effort estimé. En bref, les sources d'incertitudes dans le processus d'estimation ont étaient identifiées dans [12], ce travail détermine 3 sources importantes d'incertitudes :

- La collection de la base de données historique contenant les informations sur des projets logiciel achevés ;
- Le modèle d'estimation utilisé pour estimer l'effort ;
- Les informations disponibles sur le nouveau projet logiciel dont l'effort doit être estimé.

I.5.2 Modélisation des incertitudes

La théorie des probabilités et des statistiques est l'outil mathématique pour étudier les incertitudes. Le but ultime est d'effectuer une inférence sur la véracité d'une hypothèse, sur la valeur d'un paramètre inconnu ou sur la distribution de ce paramètre, au vu d'observations (directes ou indirecte) liées à ce paramètre ou à cette hypothèse.

I.5.2.1 Construction des intervalles de prédiction

Le bootstrap est une technique de ré-échantillonnage basée sur des tirages aléatoires avec remise dans les données constituant un échantillon. Utilisées pour approcher la distribution inconnue d'une statistique par sa distribution empirique, les méthodes de bootstrap sont mises en œuvre afin d'améliorer la précision des estimations statistiques. Des présentations détaillées de cette approche sont proposées, notamment par Hall (1992) ainsi que Efron et Tibshyrani (1993).

Conclusion

D'après ce que nous avons présenté, nous déduisons que le domaine d'estimation de l'effort de développement logiciel est très délicat et ce n'est pas vraiment évident d'estimer l'effort d'un nouveau projet en phase amont à cause de manque d'informations, et on ne peut pas juger qu'une méthode est plus performante par rapport à l'autre car chaque méthode sert dans un contexte différent.

Les techniques d'estimation classiques sont devenues inadaptées pour refléter l'état actuel de nouveaux langages de programmation, par exemple la méthode COCOMO qui se base

sur le nombre de lignes de code pour estimer l'effort ne pourra pas suivre les nouveaux langages qui utilisent déjà des frameworks implémentant toutes les tâches répétitives.

Il est le temps de chercher une alternative permettant de modéliser l'insuffisance des anciennes méthodes en prenant en compte les incertitudes dans le processus d'estimation. L'incertitude nous permet d'avoir une idée sur la valeur estimée. Il y a plusieurs façons de modéliser les incertitudes dans notre étude on va se contenter par l'intervalle de prédiction.

Chapitre II : Framework d'estimation probabiliste de l'effort

---

Introduction

Estimer la taille d'un projet logiciel en points de fonctions aux premiers stades du développement est une tâche difficile à atteindre, car on ne connaît pas beaucoup sur le projet au début du cycle de vie. La taille mesurée est souvent incertaine et moins précise .L'incertitude est due à l'incapacité de considérer tous les facteurs qui contribueraient à la mesure de taille, c.à.d. dans certains cas on est obligé de négliger quelques facteurs pertinents en raison de leur absence.

Par exemple, un modèle d'estimation se base sur la conception UML d'un projet comme une couche intergiciel pour estimer l'effort, et ce modèle suppose que l'effort est uniquement corrélé au nombre de cas d'utilisation et au nombre de classes. Il est bien évident que cette hypothèse ne pourrait pas être valable pour une estimation précise d'effort, car il existe d'autres prédicteurs [2] (non calculés dans la mesure de taille) pouvant affecter l'effort estimé. Par exemple, le nombre et les types de relations entre les différentes classes du projet et les contraintes sur chaque cas d'utilisation [4].

L'estimation d'effort basée uniquement sur le nombre de cas d'utilisation et le nombre de classes devrait entraîner une erreur considérable. La valeur de l'erreur reflète la contribution des prédicteurs négligés. Par exemple, supposons que l'aspect contrainte d'un cas d'utilisation est ignoré comme prédicteur et supposons également que 15% des projets ont des contraintes sur les cas d'utilisation et que ces contraintes augmentent l'effort global de 5%. Dans ce cas, en négligeant les contraintes, la valeur d'erreur sera probabiliste entre ± 5% avec une probabilité de 15%. Ce qui implique que l'erreur commise lors d'estimation est de son tour probabiliste.

II.1 Framework probabiliste d'estimation de l'effort

L'estimation de l'effort calculée au début du cycle de vie du développement logiciel est généralement incertaine. Et aucune des techniques d'estimation d'effort ou de mesure de taille ne répond d'une manière satisfaisante au problème de l'incertitude. De surcroit, un framework a été proposé pour évaluer l'incertitude d'estimation [4]. Ce dernier développe un système d'estimation probabiliste, en s'appuyant sur des informations obtenues à partir des modèles conceptuels orienté objets (par exemple UML), créés au début du cycle de vie logiciel. Le système d'estimation probabiliste développé à l'aide du framework estime l'effort comme étant une fonction de densité de probabilité (FDP), plutôt qu'une seule valeur déterministe.

[2] Le terme «prédicteur» est utilisé plutôt que le terme facteur pour mentionner qu'il contribue à la prédiction d'effort.

Il est important de mentionner ici que ce framework décrit une approche pour construire un système estimation probabiliste dont l'objectif est d'estimer l'effort de développement logiciel, mais ne dicte pas les prédicteurs impliqués. En revanche, Les prédicteurs sont généralement définis en fonction des données disponibles, en tenant compte les données historiques disponibles. De toute évidence, plus les données historiques sont riches, plus la confiance résultante dans le système d'estimation est élevée.

Pour tenir compte à l'incertitude, le framework se base sur deux composants principaux pour générer le système d'estimation probabiliste : (1) une estimation de l'effort moyen et (2) une erreur résiduelle d'estimation représentée par l'écart-type. Il est bien évident que, plus l'écart type est élevé, plus la confiance dans l'estimation est basse. En effet, l'effort estimé est représenté par la relation suivante (Eq. 7) :

$$E_{estimé} = Effort(\mu_{effort}) \pm \sigma_{erreur} \quad (7)$$

Où : $E_{estimé}$ est l'effort estimé et σ_{erreur} est l'écart type de l'erreur.

La distribution de probabilité de l'erreur est supposée normale, car la distribution normale est la plus couramment utilisée dans l'analyse des données. Cette hypothèse est supportée par le théorème de limite centrale (Eq. 8) :

$$E_{estimé}(N(\mu, \sigma)) = N(\mu_{effort}, \sigma_{erreur}) \quad (8)$$

Comme expliqué dans l'introduction de ce chapitre, l'erreur est une variable aléatoire avec un écart type non nul. Par conséquent, la combinaison de la mesure moyenne de l'effort et de l'écart type de l'erreur représente une fonction de densité de probabilité (FDP).

II.1.1 Exemple pour démontrer le concept du système d'estimation probabiliste

Pour illustrer le concept du système d'estimation probabiliste, nous proposons un cas simple utilise les métriques d'un seul prédicteur, Le (Tab. 1) montre un ensemble de données hypothétiques pour huit projets ayant des mesures du nombre de cas d'utilisation (NUC) et les valeurs de l'effort réel. A partir du tableau on peut bien apercevoir des incertitudes. Nous désirons alors d'estimer l'effort d'un nouveau projet dont la valeur de NUC est connue. En appliquant la méthode des moindres carrés sur les données historiques on obtient la valeur moyenne de l'effort estimé : $E(\mu) = -18.51 + 26.30 * NUC$, et l'écart type qui représente l'incertitude est de son tour va être en fonction de NUC. L'erreur de l'estimation (σ_{erreur}) est développée en appliquant la méthode des moindres carrés sur les données erronées dans l'estimation de l'ensemble de formation. Finalement, on obtient le système d'estimation

probabiliste entrainé comme (FDP), c.à.d. $N(\mu, \sigma)$ dans cet exemple $N(-18.51 + 26.30 * NUC, 1.01 + 1.71 * NUC)$.

Tableau 1 - Un ensemble de données hypothétiques pour expliquer le système d'estimation

Numéro de projet	NUC	Effort (E)
1	10	240
2	10	265
3	10	230
4	20	480
5	20	530
6	25	590
7	25	670
8	25	660

II.1.1.1 Estimation de l'effort d'un nouveau projet

Après la construction de système d'estimation probabiliste, nous désirons maintenant effectuer une estimation de l'effort d'un nouveau projet avec NUC=30, on aura alors (Eq. 8) :

$$E_{estimé} = N(\mu, \sigma) = N(-18.51 + 26.3 * NUC, 1.01 + 1.71 * NUC) \qquad (8)$$

$$E_{estimé} = N(770.41, 52.26)$$

Sachant que 95% de confiance correspond à 1.96 dans le tableau de niveau de confiance [16]. On aura alors :

$$E_{estimé}\ avec\ 95\%\ de\ confiance = 770.41\ \pm 1.96 * (52.26) = \ = 770.41\ \pm 102.42$$

II.1.2 Description détaillée du framework

Le framework est décrit ci-dessous (FIG. 8), il illustre l'intersection entre ses étapes pour arriver finalement à construire le système d'estimation probabiliste.

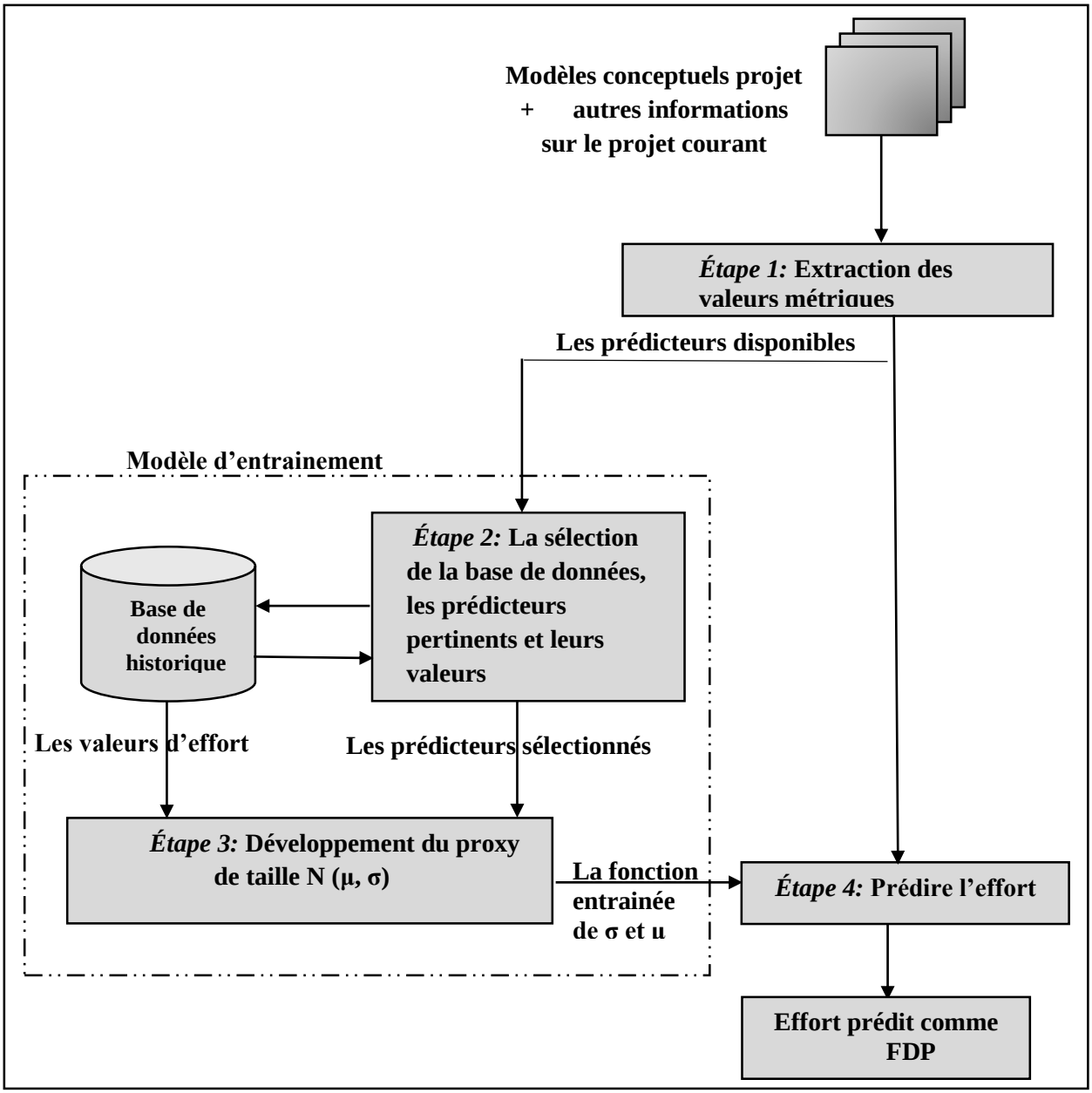


Figure 8 – L'architecture générale du framework

Étape 1: *Extraction des métriques*

Dans cette étape on extrait les prédicteurs et leurs valeurs pour un nouveau projet logiciel dont l'effort doit être estimé.

L'extraction des mesures se fait à partir de la conception UML (le diagramme de classes, le diagramme cas d'utilisation, etc.), en utilisant des outils (comme SDMetrics et Together) pour collecter une base de données historique et également pour calculer des métriques relatives au nouveau projet. L'extraction des métriques tient en compte seulement les informations disponibles dans la conception, avec autres informations spécifiques au projet : Comme les facteurs techniques et les exigences non fonctionnelles qui affectent également l'effort.

Les outils, comme SDMetrics, appliquent des règles de calcul pour obtenir les métriques dont on a besoin. Par exemple: la Complexité des attributs (CA) et la complexité de l'héritage (CI). Il est bien évident que l'indisponibilité des métriques relatives aux prédicteurs importants augmentera certainement l'incertitude de l'effort estimé.

Étape 2: Sélection des prédicteurs pertinents

Les mesures préparées par l'étape précédente sont fournies à ce composant. Ce dernier sélectionne selon la disponibilité des prédicteurs un sous-ensemble des attributs (les prédicteurs) les plus pertinents en tant que paramètres du système d'estimation probabiliste, et en prenant en compte aussi la corrélation mutuelle entre les prédicteurs sélectionnés. La sélection des prédicteurs à partir de la base de données se fait en respectant certaines conditions : il faut sélectionner un nombre suffisant de prédicteurs pour entrainer le système d'estimation. D'une part les prédicteurs ne doivent pas être nombreux cela va rendre le système trop complexe, d'autre part, peu de prédicteurs sélectionnés auront un impact négatif sur la précision de l'estimation.

Par conséquent, il est nécessaire de calculer la corrélation de chaque prédicteur avec l'effort et de sélectionner seulement ceux qui sont fortement corrélés. On calcule le coefficient de corrélation de pearson puisque toutes les données sont de type numérique.

Les prédicteurs devraient également être sélectionnés en gardant à l'esprit s'ils sont disponibles à partir d'un nombre suffisant de projets passés. En règle générale, le nombre de projets sélectionnés pour la formation du système d'estimation probabiliste doit être au moins cinq fois le nombre de prédicteurs sélectionnés (les variables indépendantes) .

Étape 3 : Développement d'un système d'estimation probabiliste

Cette étape utilise tous les prédicteurs sélectionnés à l'étape (2) ,ainsi que les valeurs d'effort réelles des projets terminés pour développer le système d'estimation. La régression linéaire multiple est utilisée comme étant un modèle d'apprentissage automatique pour entrainer finalement la forme générale du système d'estimation, sachant qu'on a déjà supposé que les valeurs d'effort sont normalement distribuées, on aura alors la forme générale du système construit: $N(\mu_{effort}, \sigma_{erreur})$.

Étape 4 : Estimation de l'effort

Une fois que le système est finalement entrainé, il peut être utilisé pour effectuer des estimations de l'effort, car on suppose dans ce stade qu'on possède les valeurs des prédicteurs du nouveau projet dont on veut estimer L'effort.

II.1.3 Génération des données artificielles

Malheureusement, à cause de confidentialité, c'est difficile d'avoir une grande plage de données sur des projets logiciels achevés. On a alors utilisé une approche qui permet de générer des données artificielles en introduisant des perturbations sur les données originales [4].

L'idée derrière le processus de perturbation est que pour un projet réel donné, on produit un certain nombre de données en perturbant (ajoutant un bruit) les données originales plusieurs fois.

Le nombre de fois de perturbation des données dépend du nombre de points de données dont nous avons besoin et de la quantité de bruit ajoutée aux données originales. Nous créons alors, pour chaque donnée d'origine un ensemble des données "N" appelé « cluster » en ajoutant un écart aléatoire normal $\pm x$ % qui représente le bruit. L'objectif est d'avoir pour chaque projet de la base de données un ensemble de projets regroupés dans un « cluster » ayant les mêmes valeurs des prédicteurs, et des valeurs d'effort différentes.

Le degré de perturbation x est la quantité moyenne de variation entre les valeurs réelles et estimées de l'effort dans tous les projets (Eq. 9) :

$$Degrée\ de\ perturbation\ x\% = \frac{1}{n} \times \sum_{i=1}^{n} \left| \frac{E^i_{réel} - E^i_{estimé}}{E^i_{réel}} \right| \qquad (9)$$

Avec n est le nombre total de projets dans la base de données, $E^i_{réel}, E^i_{estimé}$ sont respectivement l'effort réel et estimé du projet i.

II.1.4 Limites du framework

La régression linéaire a été utilisée lors d'entrainement du système d'estimation probabiliste. Alors que, il se peut qu'il existe une autre technique d'apprentissage automatique qui nous mène à une estimation d'effort plus réaliste.

Le framework ne prend pas en compte les facteurs environnementaux qui peuvent affecter l'effort estimé. Ainsi que, il ne traite pas l'imprécision des mesures de taille, surtout lorsqu'on utilise des prédicteurs ayant des valeurs incertaines. De plus l'effort estimé se base sur l'hypothèse que l'erreur commise lors d'estimation est normalement distribuée, alors que, aucun test logique n'a était effectuer pour prouver cette loi de probabilité.

II.2 Amélioration du framework

Afin d'améliorer le fonctionnement du framework au niveau de :

— Evaluation des incertitudes, c.à.d. prendre en compte d'autres sources d'incertitudes ;

— Sélectionner le modèle d'estimation le plus performant pour chaque ensemble de données.

Pour ce fait, nous proposons une amélioration comme il est décrit dans la FIG. 9. Le principe de l'étape (1) et (2) reste le même comme il est déjà indiqué dans le framework original. Nous présentons maintenant une description générale concernant les nouvelles étapes introduites.

Étape 3 : Sélection de modèle d'estimation le plus performant

On a ajouté une nouvelle étape qui s'occupe à mettre en place une approche structurée permettant d'assister les organisations dans la sélection du modèle d'estimation le plus adapté à leurs besoins, contexte et environnement [12]. Cette approche permettra la comparaison fiable de plusieurs modèles candidats et la sélection d'un modèle d'estimation selon plusieurs critères.

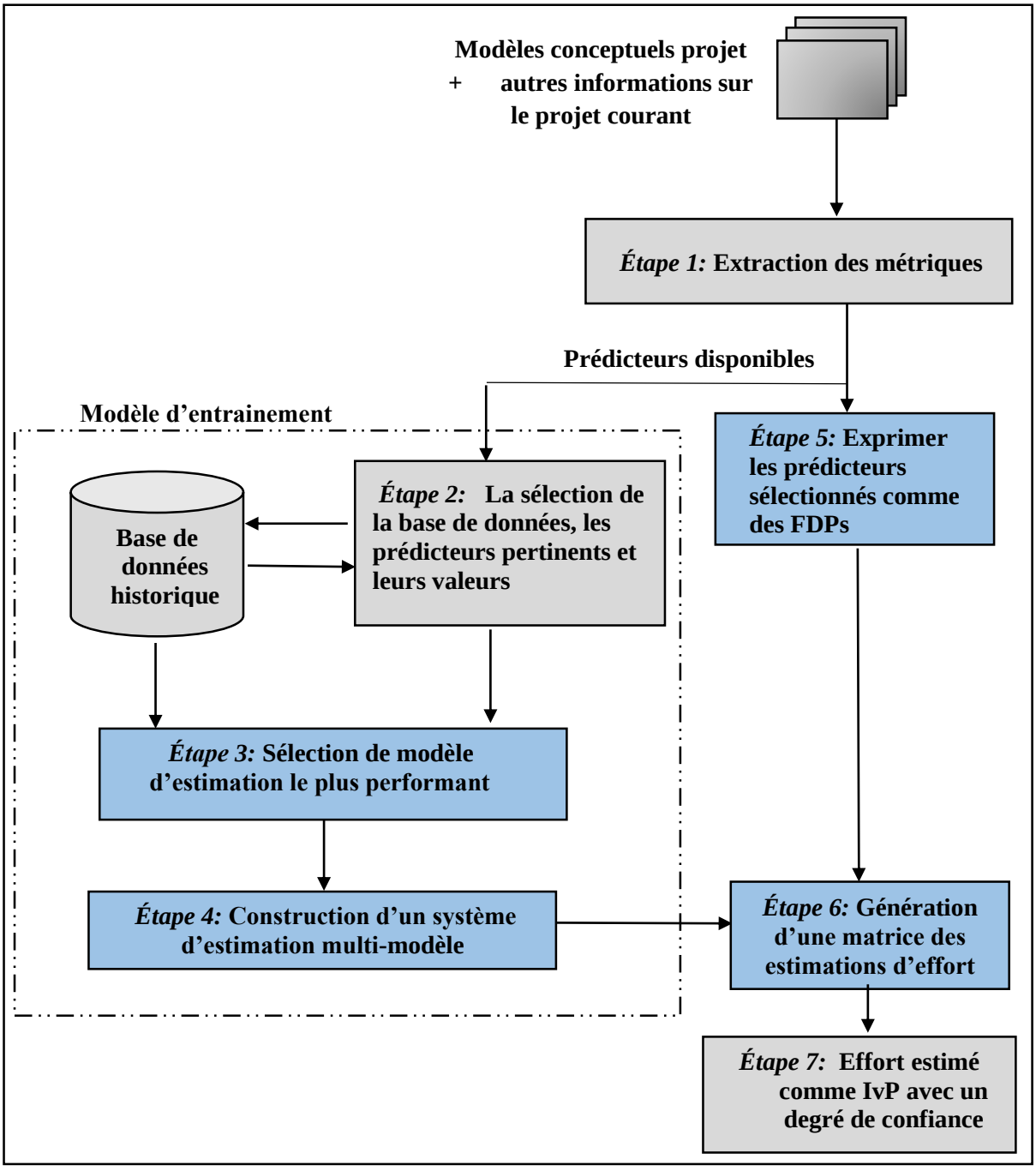


Figure 9 – Architecture améliorée du framework

Étape 4 : Développement d'un système d'estimation (multi-modèles)

Le développement du système d'estimation se base sur le modèle d'estimation sélectionné. Elle consiste à calibrer ce modèle sur la base de données de l'organisation, en d'autres termes, déterminer les paramètres du modèle d'estimation (coefficients et aussi hyper-paramètres dans le cas des modèles d'estimation non paramétriques). Dans l'approche classique, les paramètres sont déterminés d'une façon déterministe, c.à.d. chaque paramètre

consiste en une valeur unique. Ainsi, le système d'estimation construit consiste en un modèle d'estimation unique calibré sur la base de données de l'entreprise.

La phase de construction de système d'estimation se caractérise par la présence des incertitudes qui proviennent de la base de données (σ_s) et du modèle d'estimation sélectionné (σ_M). Ces incertitudes affectent le système d'estimation et plus précisément ses paramètres. Ces derniers sont entachés d'incertitudes. Une façon de représenter ces incertitudes est d'exprimer chacun des paramètres d'une façon probabiliste sous la forme d'un ensemble de valeurs possibles. Pour cela on va utiliser une approche qui conduit à construire un système d'estimation multi-modèles, c'est à dire composé de plusieurs modèles d'estimation calibrés sur la base de données de l'organisation.

Étape 5 : Exprimer les prédicteurs disponibles comme des FDPs

Le processus de spécification d'un nouveau projet de développement logiciel est naturellement sujet aux incertitudes relatives au manque de la connaissance sur ce projet σ_p. Cela par exemple peut être illustré par la difficulté à donner une mesure exacte de la taille du projet. Ces incertitudes peuvent être représentées dans les prédicteurs d'effort sous la forme de Fonction de Densité de Probabilité (FDP) [12].

Étape 6 : Génération d'une matrice des estimations d'effort

Cette étape représente l'intersection de l'étape (3) et (5) :

- L'étape (3) fourni un système d'estimation multi-modèles calibré sur la base de données prêt à estimer des nouvelles valeurs d'effort si les valeurs d'entrées sont connues (les valeurs des prédicteurs) ;
- L'étape (4) fait l'échantillonnage des fonctions de densité de probabilité des prédicteurs disponibles pour avoir finalement un ensemble de valeurs pour chaque prédicteur. Ces valeurs vont être l'entrée du système d'estimation multi-modèles qui va nous générer à la fin une matrice des valeurs d'effort estimées

Étape 7 : Estimation de l'effort comme intervalle de prédiction

La dernière étape exploite l'ensemble des résultats issus des étapes précédentes et par l'application de quelques lois statistique (ex : test sur la normalité de données), on obtiendra l'effort final sous forme d'un intervalle de prédiction avec un degré de confiance.

Conclusion

Dans ce chapitre on a expliqué l'imprécision de mesure de taille au début de cycle de vie d'un projet logiciel, ce qui implique que la taille mesurée à partir de la modélisation UML est souvent incertaine. Cela revient au manque des informations sur le projet.

Le travail présenté traite le problème d'incertitude par un framework qui développe un système d'estimation probabiliste, c.à.d. il utilise la taille du projet comme une couche intergiciel afin d'estimer l'effort : (1) en se basant sur les prédicteurs disponibles par rapport au nouveau projet et aussi par rapport à la base de données historique contenant des projets terminés, (2) en supposant que l'erreur résiduelle associée à l'estimation de l'effort est normalement distribuée. La régression linéaire multiple est utilisée pour entrainer le système d'estimation probabiliste sous la forme $N(\mu_{effort}, \sigma_{erreur})$.

Certainement ce framework apporte des points faibles qui peuvent affecter l'effort estimé. En effet, on a proposé de l'améliorer en introduisant d'autres nouvelles étapes : (1) Inclure une étape importante qui vise à sélectionner le modèle d'estimation le plus performant selon le contexte de l'organisation, (2) Construire un système d'estimation multi-modèle qui prend en compte l'incertitude issue de la base de données, (3) Les prédicteurs mesurés sont incertains, alors nous les exprimons sous forme d'une fonction de densité de probabilité.

Il y a plusieurs approches de sélection de modèle d'estimation, nous utilisons dans le chapitre suivant une approche simple qui compare et sélectionne le modèle d'estimation selon les besoins de l'utilisateur.

Chapitre III : Estimation de l'effort logiciel en tenant en compte les incertitudes

---

Introduction

Comme il est bien connu il n'existe aucun modèle d'apprentissage automatique performant dans tous les contextes d'utilisation, c.à.d. chaque modèle repose sur certaines contraintes pour qu'il puisse montrer sa performance. Sur ce principe on a pensé d'introduire une étape dans le framework qui vise à sélectionner le modèle d'estimation le plus performant par rapport à la base données utilisées.

Le modèle d'estimation sélectionné est utilisé pour faire entraîner le système d'estimation probabiliste, c.à.d. il tient en compte les incertitudes qui proviennent de la base de données. Une fois le système est entraîné il peut estimer l'effort pour un nouveau projet par un intervalle de prédiction.

Dans ce chapitre, on va utiliser une approche qui permettra l'évaluation et comparaison de plusieurs modèles candidats et la sélection de celui le plus performant selon plusieurs critères, et on présente dan suite le principe de l'approche qui manipule les incertitudes relatives aux informations disponibles sur le nouveau projet dont l'effort doit être estimé.

III.1 Forme de la base de données historique

L'approche de développement d'un système probabiliste pour estimer d'effort, s'appuie sur une base de données des projets déjà réalisés. La base de données peut être représentée par une matrice S (Eq. 17) de dimension $N \times L$, avec N représente le nombre des projets et L le nombre de variables ou attributs décrivant la taille des projets que l'on appelle «les prédicteurs».

Chaque projet i de la base de données est représenté par un vecteur R_i composé de valeurs de prédicteurs p_{ij} et de l'effort réel E_i avec $1 \leq i \leq N$.

$$R_i = (p_{i1}, p_{i2}, p_{i3}, \dots, p_{iL}, E_i), 1 \leq i \leq N \tag{10}$$

L'ensemble des vecteurs de prédicteurs caractérisant ces projets est représenté par P_j $(1 \leq j \leq N)$ et noté aussi P.

L'ensemble des efforts réels des projets de la base de données est représenté par (Eq. 11) :

$E = \{E_i, 1 \leq i \leq N\}$, où E_i représente l'effort du projet i .

$$S = \begin{pmatrix} Projets \\ R1 \\ R2 \\ R3 \\ \vdots \\ RN \end{pmatrix} = \begin{pmatrix} P_1 & P_2 & \cdots & P_L & E \\ p_{11} & p_{12} & \cdots & p_{1L} & E_1 \\ p_{21} & p_{22} & \ddots & p_{2L} & E_2 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ p_{N1} & p_{N2} & \cdots & p_{NL} & E_N \end{pmatrix} \tag{11}$$

III.1.1 Préparation de la base de données

Les bases de données historiques sont des représentations imparfaites du monde réel. En effet, elles présentent naturellement des erreurs de mesure, des données erronées, des valeurs aberrantes et des bruits. L'utilisation de la base de données dans son état brut peut conduire à des résultats incohérents.

Jusqu'à présent, il n'existe pas de méthode standardisée et d'outil automatique pour effectuer cette tâche. De nombreuses techniques statistiques peuvent être utilisées pour effectuer différents types de traitement. Dans notre travail nous utilisons une procédure générale de préparation de la base de données (voir FIG. 10).

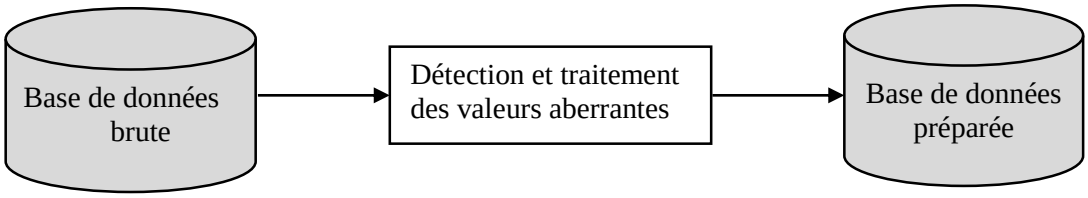


Figure 10 - Procédure de préparation de la base de données

III.1.1.1 Détection et traitement des projets aberrants

L'identification des observations aberrantes est une tâche difficile et critique surtout quand il s'agit de données multidimensionnelles ce qui est le cas dans notre étude. En effet, contrairement au cas uni-varié où les valeurs aberrantes sont toujours des valeurs extrêmes de l'échantillon, dans le cas des données multidimensionnelles, les valeurs aberrantes sont moins apparentes intuitivement. Elles sont effectivement cachées dans la masse de données et se trouve en périphérie du nuage de points. Les techniques de détection ne sont pas assez développées pour ce cas, nous en citons deux : la distance de Cook et la distance Mahalanobis.

Pour détecter les projets aberrants de la base de données en prenant en considération les corrélations des données, nous choisissons d'utiliser la distance Mahalanobis D^2 . Elle mesure la distance d'un projet $1 \leq i \leq N$ de la moyenne multidimensionnelle de l'échantillon de projets logiciels (base de données).

Il faut noter que l'utilisation de la distance Mahalanobis pour l'identification des projets aberrants se base sur l'hypothèse que les données suivent une loi normale. Il faut donc effectuer une vérification de normalité avant de procéder à l'analyse de projets aberrants.

III.2 Technique de sélection de modèle d'estimation

L'idée est de faire sélectionner le modèle de « machine learning », le plus performant par rapport à la base de données étudiée.

Supposons qu'on a un ensemble de modèles candidats, la procédure de sélection sur la base de :

— Evaluation de la performance prédictive de tous les modèles candidats en calculant la valeur numérique des indicateurs de performance ;

— Comparaison des modèles évalués et la sélection d'un modèle selon le besoin de l'utilisateur.

III.2.1 Évaluation de la performance prédictive des modèles d'estimation sur une base de données

Nous utilisons la technique de la validation croisée « VCK », pour évaluer chaque modèle d'estimation candidat sur la base de données (Fig.10).

VCK est une méthode d'estimation de fiabilité d'un modèle fondé sur une technique d'échantillonnage. En fait, il y a au moins trois techniques de validation croisée : « holdout

Nous utilisons dans notre travail la seconde, on divise l'échantillon original en k échantillons, puis on sélectionne un des k échantillons comme ensemble de validation et les (k-1) autres échantillons constitueront l'ensemble d'apprentissage. On calcule comme dans la première méthode le score de performance. Puis on répète l'opération en sélectionnant un autre échantillon de validation parmi les (k-1) échantillons qui n'ont pas encore été utilisés pour la validation du modèle. L'opération se répète ainsi k fois pour qu'en fin de compte chaque sous-échantillon ait été utilisé exactement une fois comme ensemble de validation (Fig. 12).

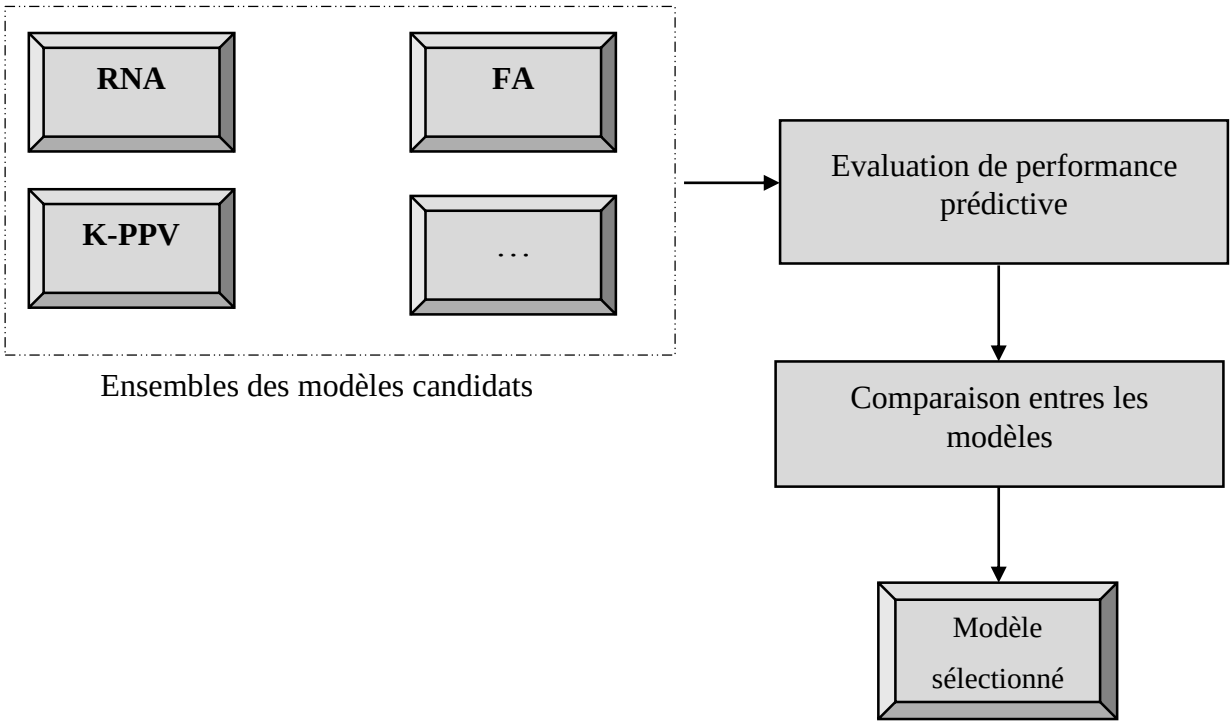


Figure 11 - Procédure de sélection de modèle d'estimation

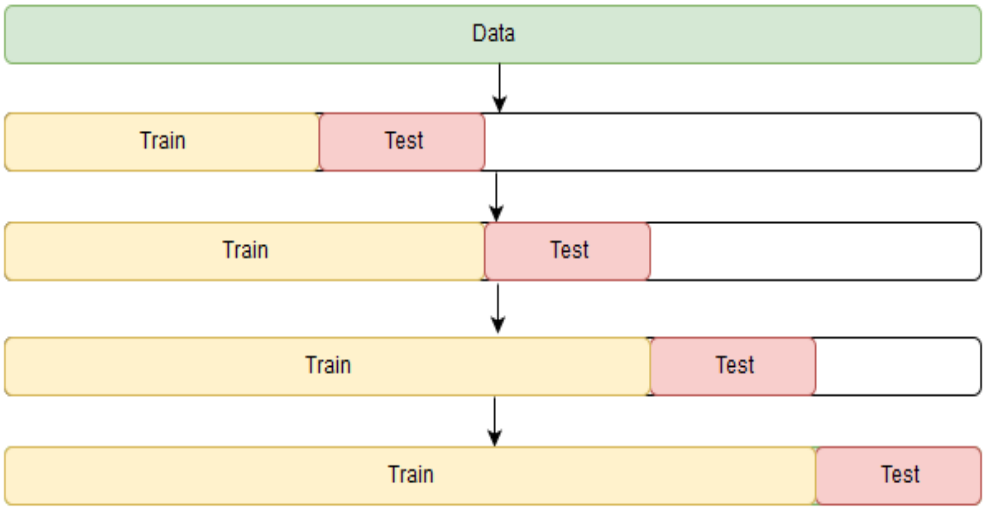


Figure 12 - La procédure de (VCK) K-fold

III.2.2 Indicateurs de performance prédictive utilisés

Nous utilisons différents indicateurs de performance dans l'objectif de représenter et évaluer les aspects importants de la performance prédictive des modèles d'estimation. Ces indicateurs sont : *MMRE*, *MdMRE*, et *Pred(l)*. On a utilisé deux autres indicateurs de performance complémentaires [12] *Und(l)* and *Ovr(l)*. Ces indicateurs indiquent, respectivement, la tendance du modèle M à sous-estimer ou surestimer l'effort dans un niveau de tolérance l. Ils sont représentés dans (Eq. 12) où u et o sont les nombres de projets ayant une MRE inférieure à l et qui, respectivement, sous-estiment et surestiment l'effort.

$$Und = \frac{u}{p} \; et \; Ovr = \frac{o}{p} \qquad (12)$$

III.2.3 Comparaison et sélection de modèle d'estimation

Ce n'est pas vraiment évident de comparer les modèles d'estimation candidats, car il existent des indicateurs à minimiser comme (MMRE et MdMRE) et d'autre à maximiser comme (Und , Ovr, Pred). De surcroit, on a utilisé une approche qui permet de sélectionner le modèle selon le besoin de l'utilisateur.

— Le plus petit MMRE favorise des estimations généralement stables, c'est à dire des estimations de qualité similaire [13].

— Le plus petit MdMRE favorise des estimations stables particulièrement aux projets typiques MdMRE , est moins sensible aux valeurs extrêmes de MRE.

— Le plus grand Pred (0.25) : aversion pour le risque, le modèle d'estimation choisi fourni un plus grand nombre d'estimations performantes.

Face à deux modèles d'estimation ayant des performances similaires (*MMRE*, *MdMRE* ou *Pred(0,25)*), l'utilisateur peut souhaiter favoriser le modèle d'estimation qui a moins de risque de sous-estimation ou au contraire qui a moins de risque de surestimation. La sélection du modèle d'estimation, dans ce cas, doit s'appuyer sur *Und* ou *Ovr* selon le niveau d'acceptation du dépassement éventuel d'effort. Par exemple, dans le cas des start-up en développement informatique, il est très important de ne pas dépasser les budgets prévus étant donné que les ressources sont souvent limitées, ainsi l'estimateur peut favoriser, parmi plusieurs modèles d'estimation ayant les mêmes *MMRE*, *MdMRE* ou *Pred* (0,25), le modèle minimisant le *Und* [12].

Après la sélection du modèle d'estimation le plus adéquat, l'étape qui suit est l'utilisation de ce modèle pour construire un système d'estimation d'effort propre à l'organisation. Il s'agit d'un système qui est "prêt à l'emploi" et qui permet d'estimer d'une façon l'effort des nouveaux projets de développement logiciel..

III.3 Construction d'un système d'estimation probabiliste

Nous présentons tout d'abord dans cette partie la forme de la base de données historique exigée pour le fonctionnement du framework, ainsi la procédure du prétraitement avant de mettre en œuvre la base de données.

Nous détaillons dans la suite le principe de deux étapes ajoutées au framework qui gèrent les incertitudes inhérentes à :

— La collection de la base de données historique ;

— Les informations disponibles sur le nouveau projet à développer.

III.3.1 Construction d'un système d'estimation multi-modèles

La base de données historique qui contient des projets logiciel achevés se caractérise par l'existence d'incertitude importante au niveau de la collection. Pour gérer pour cette source d'incertitude dans le processus d'estimation d'effort nous avons utilisé une approche structurée qui vise à construire un système d'estimation multi-modèles. L'idée derrière la construction un

système d'estimation multi-modèles est que l'ensemble des échantillons tirés de la base de données disponible fournit une meilleure estimation et représentation de la distribution de la population d'origine.explique bien les différentes étapes de bootstrap (1) L'algorithme .

Algorithme 1 : Algorithme de bootstrap pour la construction d'un système d'estimation d'effort à partir d'un modèle d'estimation [12]

Entrée : Modèles d'estimation sélectionné M

Sortie : Système d'estimation d'effort

Génération aléatoire avec remise de B échantillons S_b^*, $1 \leq b \leq$ B de S (même dimension) ;

Pour $b = 1, .., B$ **faire**

Apprentissage M sur S_b^* : $M^{*b} = M(S_b^*)$

E = { $M^{*b}; b \in 1, \ldots, B$}

III.3.1.1 Mise en œuvre du système multi-modèles pour estimer l'effort

Après avoir entraîné le système par une technique d'apprentissage automatique, il va être tout prêt pour estimer des nouvelles valeurs d'effort si les valeurs d'entrée sont connues. Et puisque les entrées sont des prédicteurs exprimés comme des FDPs alors la sortie aussi sera aussi une FDP. La (Fig. 13) illustre le principe d'estimation de l'effort par le système d'estimation multi-modèles.

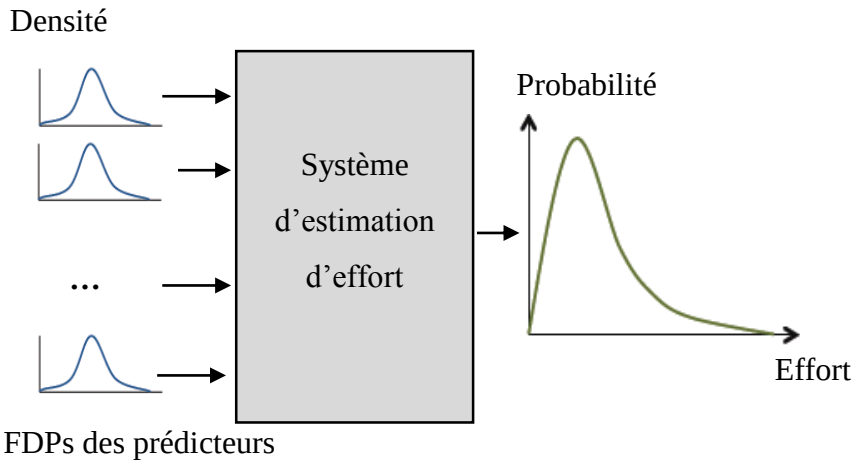


Figure 13- Estimation de l'effort par le système [12]

Dans notre étude on suppose que tous les prédicteurs suivent **la loi normale**, c.à.d. toutes les fonctions de densité de probabilité sont représentées par la loi normale.

III.3.1.2 Principe de simulation Monte Carlo (MC)

Les techniques de simulation Monte Carlo sont utilisées pour simuler des systèmes déterministes avec des paramètres ou des entrées stochastiques. Le nom a été proposé par les scientifiques du projet Manhattan lors de la deuxième guerre mondiale et fait allusion aux jeux de hasard pratiqués à Monaco.

La technique de simulation Monte Carlo s'appuie sur l'échantillonnage des distributions des quantités incertaines. Il peut être visualisé comme une boite noire où entre un flux de nombres pseudo-aléatoires (c.-à-d. générés par l'ordinateur) et un flux de nombres sort; l'estimation de la quantité d'intérêt est obtenue en analysant la sortie. La (Fig. 14) nous approche du principe de simulation MC.

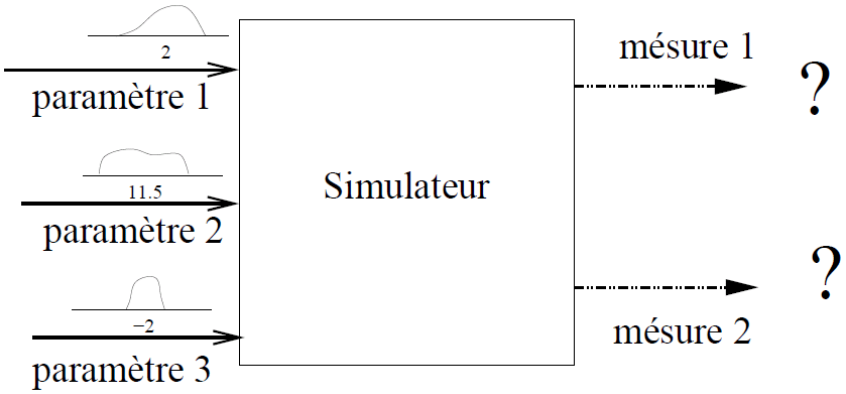


Figure 14 - Principe de simulation Monte Carlo

L'échantillonnage Monte Carlo consiste à passer de la forme continue d'une FDP à la forme discontinue en prenant un certain nombre de points à partir de la fonction continue, il est bien évident que plus le nombre de points pris est grand plus la fonction échantillonnée représente bien la fonction initiale.

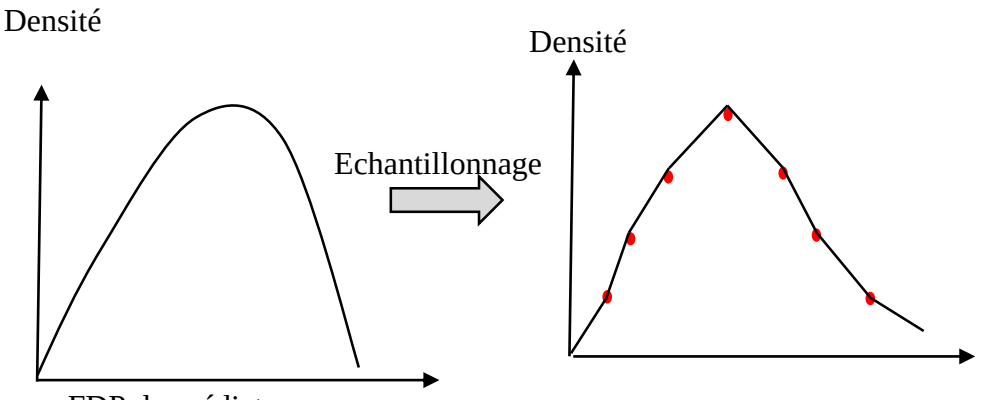


Figure 15 - Principe d'échantillonnage Monte Carlo

III.3.1.3 Application de la méthode Monte Carlo au système d'estimation

Lorsque on applique la méthode monte Carlo sur le système d'estimation construit on obtient un système d'estimation probabiliste qui estime l'effort d'une manière probabiliste (intervalle de prédiction + degré de confiance).

Notons qu'on a e prédicteurs sélectionnés à dans la base de données historiques contenant N projets achevés à leurs valeurs d'effort correspondants, et soit le système d'estimation construit contient B modèles calibrés sur la base de données historique. A partir d'un m points pris de chaque FDP, on obtient une matrice où chaque colonne représente un vecteur d'entrée. Ce vecteur va être l'entrée de tous les modèles constituant du système d'estimation construit. Le même processus se répète pour tous les autres vecteurs de la matrice d'entrée comme il est indiqué dans la (Fig. 16).

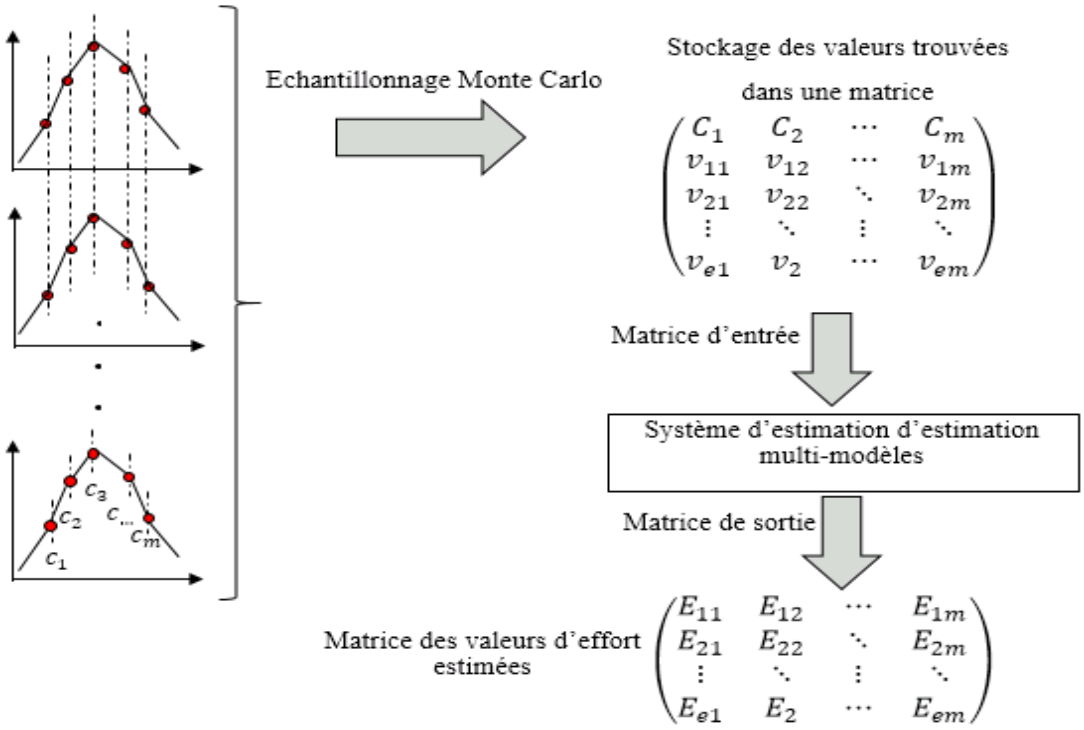


Figure 16 - Application de MC au système d'estimation

III.3.1.4 Génération des intervalles de prédiction (IvPs)

Après avoir obtenu la matrice des estimations d'effort composée de m colonnes et B lignes, il est le temps qu'on se rappel de notre objectif qui est « estimer l'effort final sous forme d'intervalle de prédiction associé à un degré de confiance ». Pour arriver à ce but on va générer des intervalles de prédiction à partir de la matrice contenant les estimations de l'effort. Pour

chaque ligne de la matrice un intervalle de prédiction généré comme il montre la (Fig. 17). Ivp est généré selon (Eq. 13) :

$$IvP(1-\alpha) = [\tilde{E}_m - t_{1-\alpha/2}s\sqrt{1+\frac{1}{m\times B}}\ ,\ \ \tilde{E}_m + t_{1-\alpha/2}s\sqrt{1+\frac{1}{m\times B}}\ ] \quad (13)$$

Avec $\tilde{E}_m$ et s sont respectivement la moyenne et l'écart type des valeurs de l'effort estimé $(\tilde{E}_r^{*b}), r \in \{1,\dots,m\}, b \in \{1,..,B\}$ et α la probabilité que $l'IvP$ ne couvre pas l'effort réel, dans le cas où $= 0.05$, $t_{1-\alpha/2} = 1.96$.

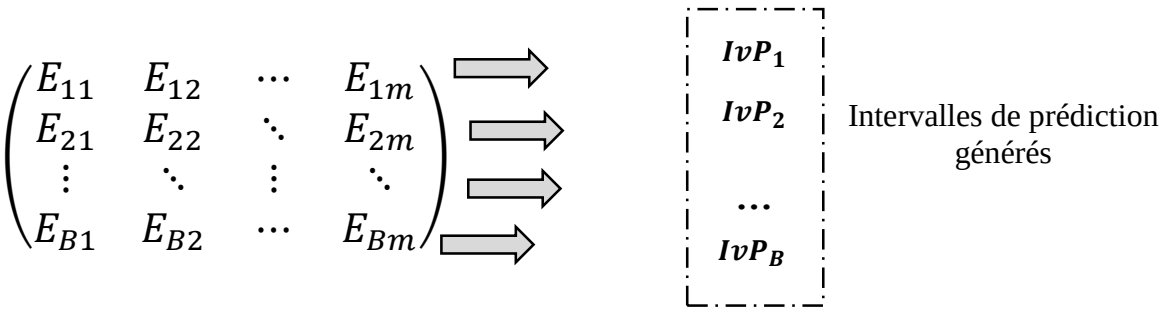


Figure 17 - La méthodologie de génération des intervalles de prédiction

L'étape finale est de trouver *IvP* en se basant sur l'ensemble des IvPs trouvés. En effet, la méthode suivie est simple, il suffit de trouver le minimum et le maximum de tous les intervalles calculés

$$IvP_{final} = \left[Min_{IvP}^{b}, Max_{IvP}^{b}\right], b \in \{1,\dots,B\} \quad (14)$$

Conclusion

On a utilisé VCK pour évaluer la performance de chaque modèle d'estimation candidat. Divers critères de sélection ont été proposés. Ils permettent d'assurer une grande flexibilité de sélection du modèle d'estimation selon les objectifs.

Afin de prétraiter la base de données, on a utilisé la distance de Mahalonbis pour trouver les projets aberrants qui fait partie de la base de données. L'existence des projets aberrants peut biaiser entrainement du système d'estimation. Le bootstrap a été utilisé pour construire le système d'estimation multi-modèles, quant à Monte Carlo a été utilisé pour prendre en considération les incertitudes relatives au nouveau dont on possède peu d'informations.

L'application de la méthode Monte Carlo sur le système d'estimation, nous a aidé à gérer une matrice des estimations d'effort. Cette matrice a été utilisée pour calculer la valeur d'effort final sous forme d'un intervalle de prédiction avec un degré de confiance.

Chapitre IV : Expérimentation

Introduction

Nous présentons dans ce chapitre les détails de l'expérience que nous avons menée pour valider le système d'estimation du framework. Cinq modèles d'estimation différents sont évalués et comparés entre eux sur la même base de données d'une organisation.

Nous expérimentons également l'utilisation du système d'estimation construit pour l'estimation d'effort d'un nouveau projet.

IV.1 Expérimentation et interprétation des résultats

Notre expérimentation vise à :

— Évaluer et comparer un ensemble de modèles d'estimation candidats sur la base de données ;

— Estimer l'effort d'un nouveau projet en se basant sur le système d'estimation construit.

IV.1.1 Description de la base de données

La base de données de l'expérimentation qu'on a utilisée, contient des mesures collectées auprès de 91 projets achevés, développés par C++, VC++ et Java (Tab. 4). Nous avons pu avoir cette base auprès d'une équipe de recherche à l'université de « le Roi FAHD » à l'Arabie Saoudite.

Tableau 2 - Description des attributs de la base de données

Attributs	Description	Types
E (Effort)	Effort en mois-hommes	Numérique
Nombre de classes (NC)	Le nombre total de classes dans le système	Numérique
Nombre de méthodes publiques (NPM)	Le nombre total de méthodes publiques dans toutes les classes	Numérique
Nombre d'attributs (NA)	Le nombre total d'attributs dans toutes les classes	Numérique
Complexité des attributs (AC)	La valeur de la complexité des attributs totaux. Où AC prend 1 à chaque type primitif et 3 à un type de classe. Tout tableau ou collection la complexité est 3 fois supérieure à son type de base. (ex : un tableau de classe a type égale 3 alors sa complexité est 9	Numérique

Profondeur d'arbre d'héritage (DIT)	La profondeur maximale d'arbre d'héritage	Numérique
Complexité de l'héritage (IC)	Le nombre de relations d'héritage dans toutes les classes du système	Numérique
Nombre de relations (NR)	Le nombre total de relations (ex : l'association et la dépendance) dans toutes les classes	Numérique

IV.1.2 Modèles d'estimation candidats

Nous choisissons cinq modèles d'estimation à comparer sur la base de données, ils sont décrits dans (Tab. 5).

Tableau 3 - Propriétés des modèles d'estimation candidats

Code	Modèle d'estimation	Type	Hyper-paramètres
RLM	Régression linéaire multiple	Paramétrique	Aucun
RNA	Réseaux de neurones artificiels	Non Paramétrique	Nombre de neurones dans la couche cachée.
FAD	Forêt d'arbres décisionnels	Non Paramétrique	Nombre de prédicteurs d'effort utilisés dans chaque nœud.
K-PPV	K-Plus Proches Voisins	Non Paramétrique	Nombre k des plus proches projets voisins à considérer dans l'estimation.
BR	Régression bayésienne	Non paramétrique	Loi a priori de la fonction densité des données

IV.1.2.1 Sélection des prédicteurs pertinents

Afin de sélectionner les prédicteurs d'effort parmi les 7 prédicteurs décrits dans (Tab. 4), nous effectuons un test de corrélation entre les prédicteurs et l'effort. Le (Tab. 6) présente les résultats du test de corrélation de pearson.

Tableau 4 - La corrélation des variables avec l'effort

	NC	*NR*	*IC*	*DIT*	*NPM*	*NA*	*AC*
Effort en homme-heures	0.92	0.74	0.37	0.10	0.89	0.95	0.96

Nous remarquons d'une part, que tous les attributs (les prédicteurs) sont bien corrélés avec l'effort sauf l'attribut *DIT* qui est relativement mois corrélé. En effet nous fixons un seuil *S (S = 0.10)* et nous prenons juste les attributs ayant une corrélation supérieure. D'autre part, on a étudié la corrélation mutuelle entre les différents attributs (Tab. 7), nous avons trouvé qu'il y a une forte corrélation entre *AC* et *NA* (coefficient de corrélation = 0.96). Alors, nous gardons seulement le prédicteur *AC* et on néglige *NA* car son existence pourrait entrainer le sur-apprentissage au système d'estimation.

Tableau 5 - La corrélation mutuelle entre les différents attributs

	NC	*NR*	IC	*DIT*	*NPM*	*NA*	*AC*
NC	___	0.80	0.54	0.23	0.88	0.93	0.94
NR	___	___	0.48	0.17	0.75	0.74	0.79
IC	___	___	___	0.70	0.40	0.38	0.33
DIT	___	___	___	___	0.17	0.14	0.05
NPM	___	___	___	___	___	0.86	0.85
NA	___	___	___	___	___	___	**0.96**
AC	___	___	___	___	___	___	___

IV.1.2.2 Détection et traitement des projets aberrants

Pour la détection et la suppression des projets aberrants, nous utilisons le tracé de chi deux [3] qui se base sur la distance de Mahalanobis D^2. C'est un outil graphique permettant de visualiser les projets aberrants (voir FIG. 18). Il trace les distances robustes Mahalanobis des projets de La base de données, ordonnées de la plus petite à la plus grande par rapport aux quantiles de la distribution de chi deux. Les projets aberrants peuvent être détectés visuellement. Ceux sont les projets avec les plus grandes distances D^2. On supprime les 3 projets qui

[3] Chi deux : est un test statistique permettant de tester l'adéquation d'une série de données à une famille de lois de probabilités.

maximisent la distance D^2. Après la suppression de ces projets aberrants, la base de données résultante contient 88 projets.

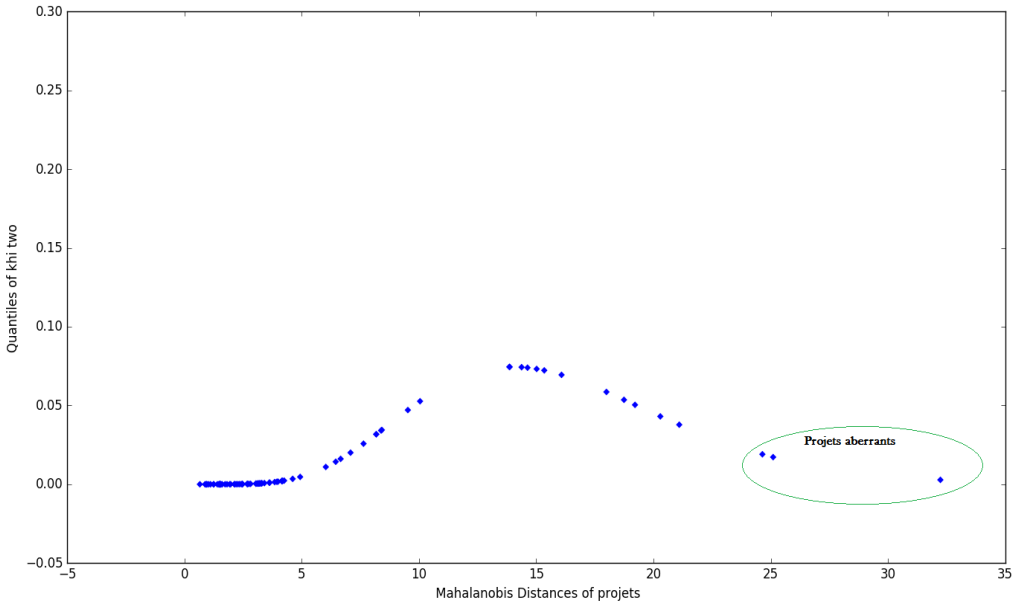


Figure 18 - Distances Mahalanobis des projets ordonnées

IV.1.2.3 Évaluation des modèles d'estimation candidats et analyse des résultats

La plupart des praticiens préconise de choisir un nombre k de validation entre : 3, 5 et 10 [14]. Le (Tab. 6) résume les résultats d'évaluation.

Tableau 6 - Performances des modèles d'estimation candidats sur la base de données

Modèle d'estimation	**MMRE**	**MdMRE**	**Pred(0.25)**	**Und(0.25)**	**Ovr(0.25)**
RLM	0.27	0.24	0.57	0.24	0.33
K-PPV	0.38	0.31	0.41	0.19	0.23
RNA	**0.22**	**0.19**	**0.65**	0.35	0.30
FAD	0.40	0.2	0.36	**0.15**	0.21
BR	0.25	0.33	0.37	0.15	**0.20**

Nous constatons que le modèle d'estimation *RNA* montre les meilleurs indicateurs *MMRE* (= 0.22), *Pred* (0.25) (=0.65) et *MdMRE* (= 0.19), il est donc le plus adéquat pour une stratégie d'aversion pour le risque. Nous remarquons aussi que le modèle d'estimation *FAD* présente les meilleurs *Und (= 0.*15) et *BR* a le meilleur *Ovr (= 0.20).*

Nous présentons le diagramme en boites à moustaches des *MREs* des modèles d'estimation d'effort sur la base de données étudiée afin de mieux appréhender leurs comportements prédictifs (voir FIG. 19).

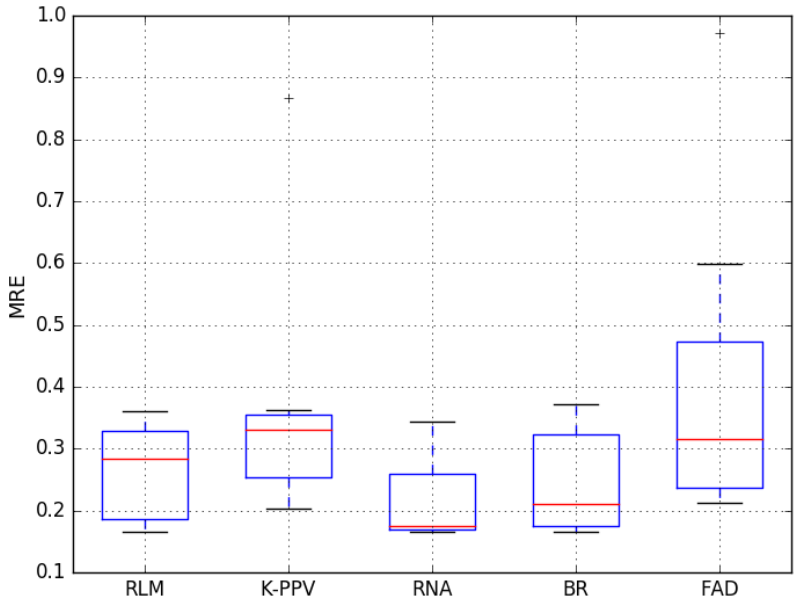
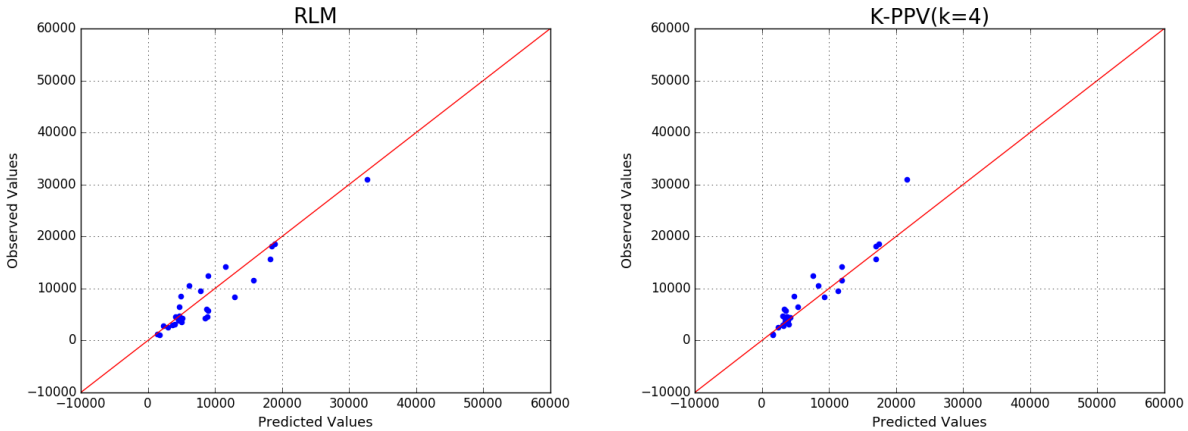


Figure 19 - Boites à moustaches des modèles d'estimation d'effort sur la base de données

En conclusion, sur cette base de données, le modèle d'estimation *RNA* présente les meilleures performances.

Pour avoir une vision complète et claire des comportements des modèles d'estimation, nous présentons dans la (FIG. 20). Chaque graphe de la figure nous montre la performance d'un modèle d'estimation sur la base de données de test.



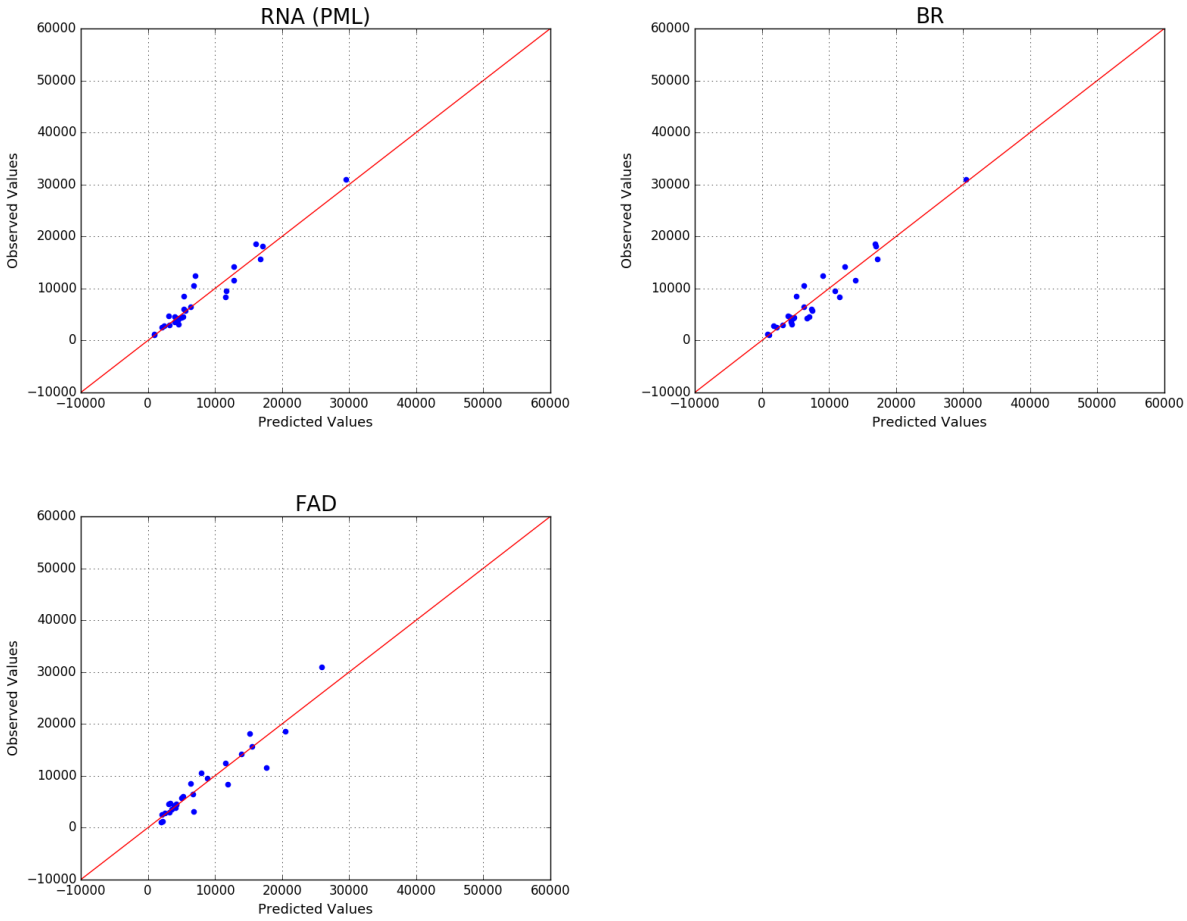


Figure 20 - Graphes des efforts réels et efforts estimés par les modèles d'estimation candidats

IV.2 Estimation de l'effort d'un nouveau projet

Afin d'estimer l'effort d'un nouveau projet nous devons avoir des mesures de conception de ce projet. Pour cette raison nous avons importé un diagramme de classes petit et simple [15], juste pour expliquer la procédure l'estimation de ce nouveau projet (Fig. 21). Le diagramme de classes est le seul diagramme de la modélisation UML que nous disposons sur ce projet. Les métriques en extraites sont décrites dans (Tab. 7)

Tableau 7 - Les métriques extraites du diagramme de classes

Prédicteurs	NC	NPM	AC	DIT	IC	NR
Valeurs	8	8	15	1	2	3

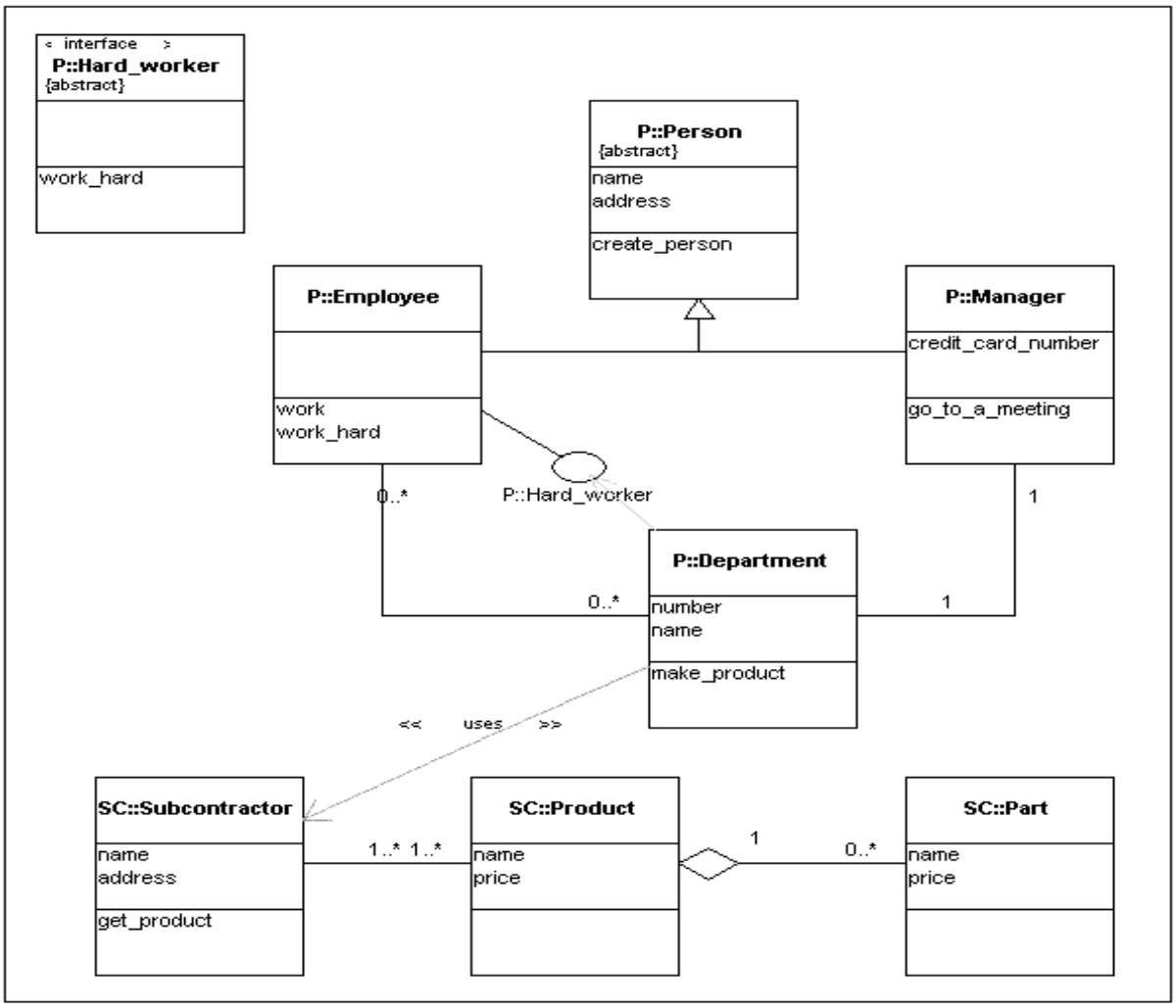


Figure 21- Diagramme de classes de nouveau projet **[15]**

IV.2.1 Construction d'un système d'estimation

L'approche que nous avons utilisée dans le chapitre *IV* pour la construction du système d'estimation et l'utilisation de ce système pour l'estimation d'un nouveau projet.

Pour la base de données utilisée les résultats de comparaison de modèles d'estimation ont montré que le modèle *RNA* présente les meilleures performances.

Malgré la bonne performance du modèle *RNA*, nous choisissons de construire le système d'estimation *RLM* seulement pour rendre l'explication plus facile, mais la même chose pourrait être faite avec *RNA*. Nous fixons le nombre de modèle composant le système d'estimation à 10 $M^{*b}, b \in 1, \dots, 10$. Le système d'estimation peut être décrit comme dans (Eq. 22). Les coefficients des modèles, obtenus après apprentissage sur les échantillons bootstrap de la base de données, sont présentés dans le tableau (Tab. 7).

$$E = \beta_{b0} + \beta_{b1} NC + \beta_{b2} NR + \beta_{b3} IC * \beta_{b4} DIT + \beta_{b5} NPM + \beta_{b6} AC \qquad (32)$$

Tableau 7 - Coefficients des modèles composant le système d'estimation

M^{*b}	β_{b0}	β_{b1}	β_{b2}	β_{b3}	β_{b4}	β_{b5}	β_{b6}
M^{*1}	71.93	-8.71	-19.04	4.03	16.47	3.75	1.91
M^{*2}	786.49	-12.83	-25.91	7.46	-117.31	3.56	2.1
…	…	…	…	…	…	…	…
M^{*10}	378.61	-4.35	-19.96	4.49	-45.93	3.95	1.84

IV.2.1.1 Génération des intervalles de prédictions (IvPs)

Chaque *Ivp* est calculé l'aide d'un ensemble des valeurs d'effort selon (Eq. 21 Chap. *IV*). Pour bien montrer le principe de calcul d'*Ivps* qui est basé sur (Bootsrap et Monte Carlo), nous présentons pour ce fait, deux histogrammes prises de l'ensemble d'histogrammes générés lors du calcul d'*IvPs* (voir Fig. 22).

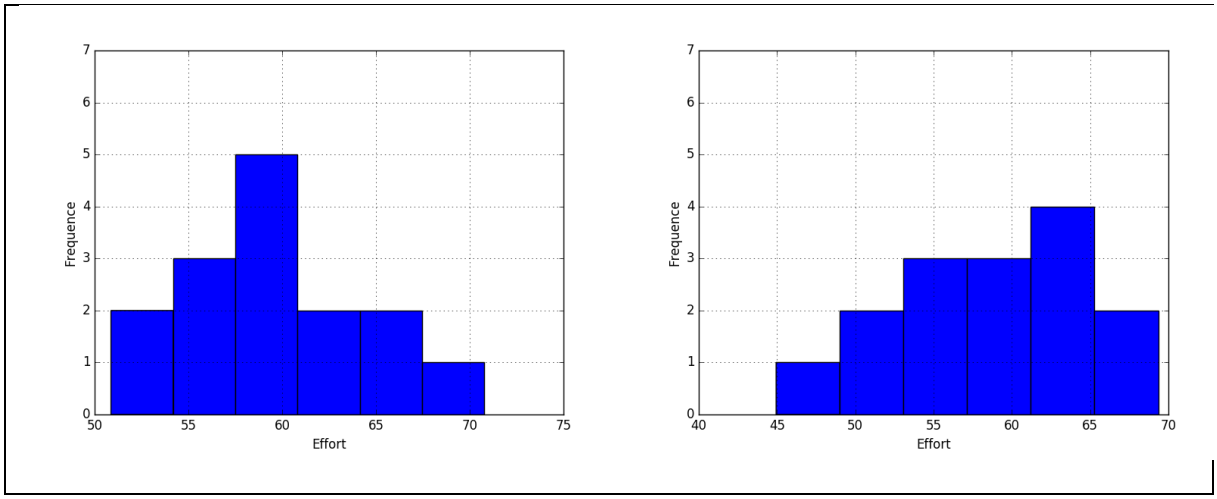


Figure 22 - Histogrammes illustrent la répartition des valeurs d'effort estimées de deux Ivps.

Nous remarquons alors que, la répartition des valeurs d'effort calculées se rapprochent de la loi normale. La distribution normale de ces valeurs nous aide à bien évaluer *IvP* de l'effort final. Finalement l'intervalle de prédiction finale trouvé est : $[36.52, 74.74]$ (heures-homme) avec 95% de confiance.

IV.3 Implémentation Informatique

On a implémenté l'ensemble des étapes décrites dans le framewrok à fin de les automatiser dans une application réalisée par les technologies suivantes :

— **Python (version 2.7)**

Python est un langage de programmation objet, multi-paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions ; il est ainsi similaire à Perl, Ruby, Smalltalk, etc .

— **Glade**

Glade est un outil interactif de conception d'interface graphique (GTK+). Il prend en charge toute la partie de gestion/génération de l'interface pour permettre au développeur de se concentrer sur le code. Glade enregistre les interfaces graphiques en générant des fichiers XML.

— **Scikit learn**

Scikit learn est une bibliothèque libre Python dédiée à l'apprentissage automatique. Elle est développée par de nombreux contributeurs notamment dans le monde académique par des instituts français d'enseignement supérieur et de recherche comme Inria3 et Télécom ParisTech. Elle comprend notamment des fonctions pour estimer des forêts aléatoires, des régressions, des algorithmes de classification, et les machines à vecteurs de support. Elle est conçue pour s'harmoniser avec des autres bibliothèques libres de Python, notamment NumPy et SciPy.

IV.3.1 Déroulement d'un cas d'utilisation

Afin d'illustrer les fonctionnalités développées dans notre application, nous présentons le déroulement d'un cas d'utilisation étape par étape. La procédure pour comparer différents modèles d'estimation sur une base de données, commence tout d'abord par le chargement du fichier contenant la base de données (le fichier doit être sous le format ".csv"). Une fois chargé, l'application analyse les données et elle permet de visualiser l'histogramme qui représente la corrélation pearson entre l'effort et prédicteurs de la base de données (voir FIG. 25).

La logique de l'application est divisée en trois étapes, chaque étape est représentée par une interface graphique :

— **Interface I** (Fig. 23) :

— Bouton « Show the correlation » : Analyser les corrélations et l'importance de chaque prédicteur, on peut visualiser graphiquement l'importance de chaque

prédicteurs. La sélection définitive des prédicteurs d'effort se fait automatiquement par le système selon un seuil.

— Bouton « Show outliers » : Détecte les projets aberrants résidant dans la base de données et les supprime automatiquement. Il permet aussi de visualiser la répartition des projets aberrants selon la distance Mahalanobis.

— Bouton « Next Step » : Prend la base de données prétraitée contenant : (1) les prédicteurs pertinents sélectionnés, (2) les projets aberrants supprimés. Cette base de données est prête à utiliser pour la sélection de modèle d'estimation.

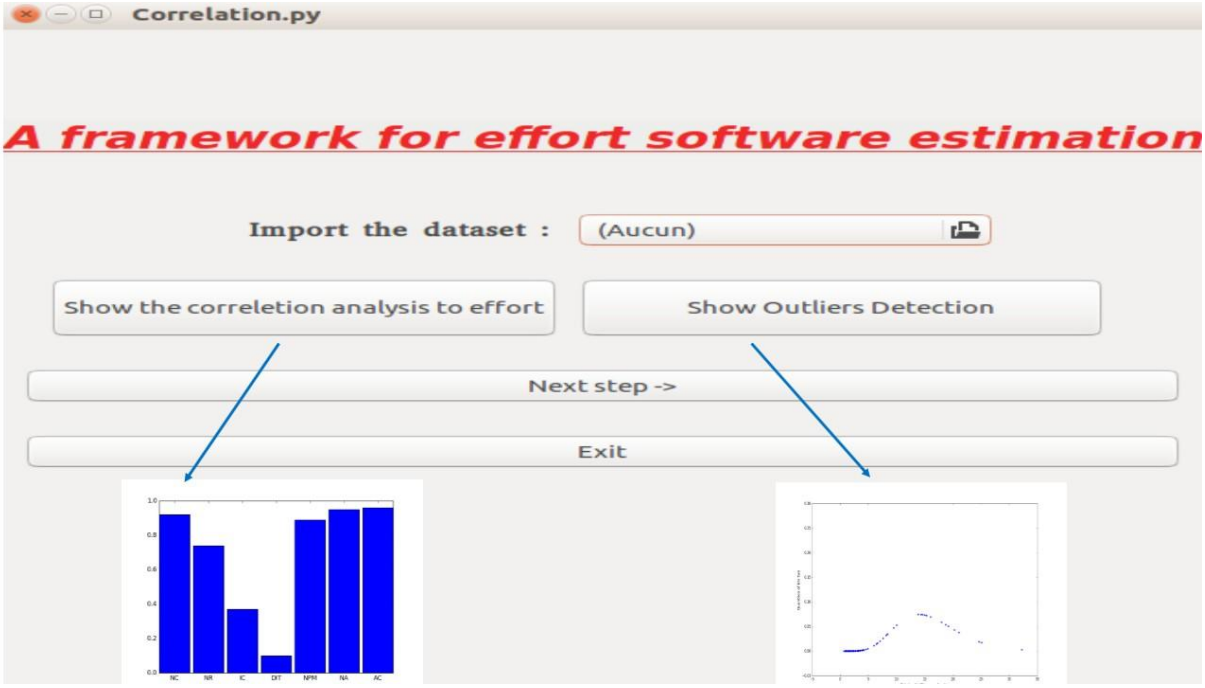


Figure 23 - La première interface de l'application

— **Interface II** (Fig. 24) :

Cette interface permet l'évaluation et comparaison des modèles d'estimation. Il invite l'utilisateur à déterminer le paramètre de l'évaluation qui est le nombre des blocs de la validation croisée k-fold. Grâce à cette interface l'utilisateur peut visualiser : (1) la performance prédictive de chaque modèle d'estimation, (2) une description statistique des MREs relative à chaque modèle d'estimation. Finalement l'utilisateur pourra Sélectionner le modèle d'estimation selon les stratégies décrites dans (Chap. III).

Figure 24 - La deuxième interface de l'application

— **Interface III** (Fig. 25) :

Le modèle d'estimation est bien sélectionné dans l'interface précédente, alors il est prêt à l'entraînement selon les informations disponibles sur le nouveau projet. C.à.d. dans la phase d'entrainement du système d'estimation, on garde dans la base de données seulement les attributs (les prédicteurs) connus sur le nouveau projet dont l'effort doit être estimé.

L'interface requis la saisie manuelle de toutes les métriques des prédicteurs disponibles, l'utilisateur doit saisir alors, la valeur de chaque prédicteur mesuré et aussi l'erreur éventuelle commise lors de mesure (c.à.d. lors de conception), puis l'utilisateur doit cocher obligatoirement la disponibilité du prédicteur pour qu'il soit pris en considération par le système lors de l'entrainement du système d'estimation.

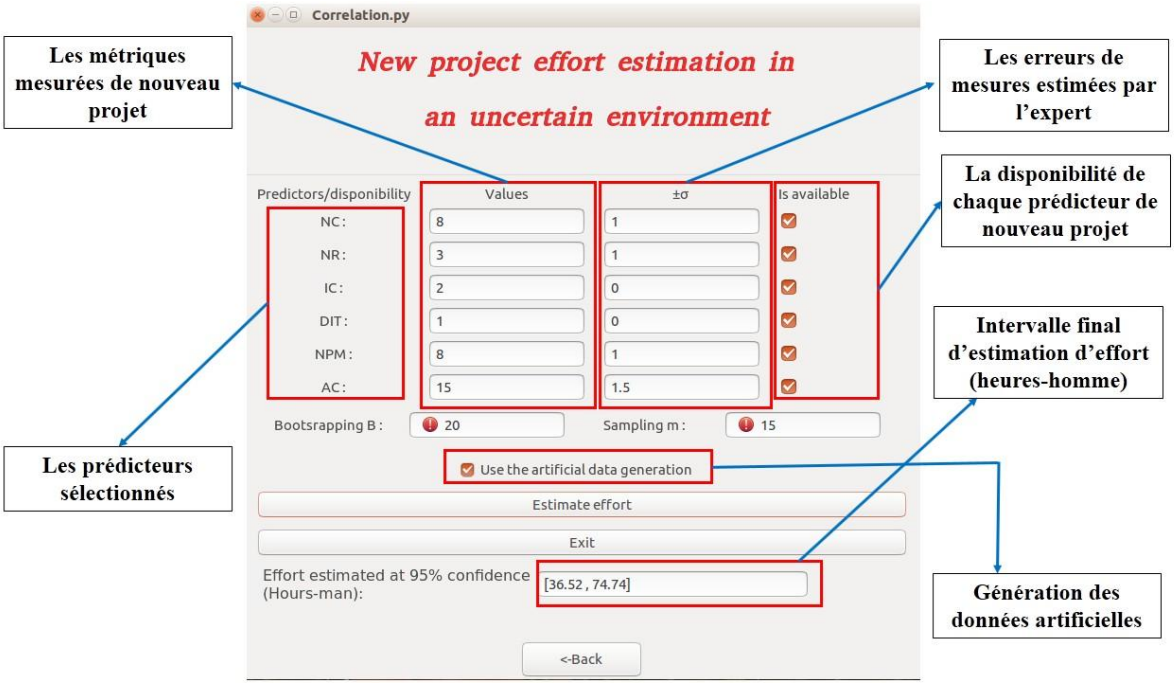


Figure 25 - Troisième interface de l'application

Il reste deux champs de saisie restant : le premier sert pour entrer le nombre de bootstrap *b*, et le deuxième pour le nombre des points dans l'échantillonnage Monte Carlo.

Le système fait ses calculs comme il est indiqué dans (chap. III), et nous affiche à la fin l'effort estimé sous forme d'intervalle de prédiction associé à un degré de confiance.

Conclusion

Nous avons expérimenté l'approche de sélection de modèle d'estimation sur une base de données. Les résultats ont montré, d'une part, que les modèles d'estimation présentent différentes performances prédictives. D'autre part, ils ont montré que la sélection d'un modèle d'estimation n'est pas toujours une tâche évidente. En effet, l'utilisateur doit prioriser les indicateurs en fonction de ses objectifs et besoins.

L'expérimentation montre également que le *RNA* présente les meilleures performances. Nous avons développé dans le cadre de ce PFE, une application permettant d'automatiser le framework d'estimation probabiliste que nous avons améliorée. L'application est développée par (Python 2.7), ce langage est récemment le plus utilisé dans les domaines d'apprentissage automatique (machine learning) et exploration de données (data maning).

L'application développée fournit l'ensemble des outils d'analyse statistique et de modélisation permettant de supporter l'utilisateur tout au long du processus de l'estimation de l'effort.

Conclusion générale

L'estimation réaliste de l'effort de développement logiciel est une tâche très importante pour le succès d'un projet logiciel. La littérature dans ce domaine est riche de méthodes d'estimation alors que ces méthodes ne peuvent pas suivre l'évolution rapide de développement informatique, et pour surmonter aux limitations des anciennes (techniques/méthodes), on a agi sur l'architecture d'un framework existant dont on a proposé une nouvelle architecture.

L'architecture originale permet seulement de répondre à la **Q.1** soulevée dans l'introduction par l'estimation de l'effort comme étant une fonction de densité de probabilité $N(\mu, \sigma)$ avec μ est le moyen de l'effort estimé et σ est l'erreur résiduelle estimée.

La nouvelle architecture permet de répondre à la **Q.2**, l'amélioration est effectuée en introduisant un nouveau composant à l'architecture originale du framework : le composant ajouté s'occupe à exprimer les mesures des prédicteurs d'une manière probabiliste notamment par la loi Gaussienne.

Afin d'améliorer la performance prédictive, une étape dédiée pour ce fait a était introduite dans le framework. Elle nous a permet de sélectionner le modèle d'estimation qui présente la bonne performance.

Finalement au lieu d'exprimer l'effort final estimé par une fonction de densité de probabilité, on l'avait exprimé par un intervalle de prédiction associé à un degré de confiance (95%). Ce dernier veut dire que la valeur réelle de l'effort de nouveau projet se trouve dans l'intervalle avec 0.95 de probabilité.

Pour valider notre contribution sur le framework nous avons mené le protocole expérimental suivant :

— Pour élargir la base de données d'apprentissage on a mis en place une technique qui permet de générer des données artificielles à partir de la base de données originale ;

— Pour améliorer la performance prédictive, on a sélectionné le modèle le plus adéquat par rapport à une base de données étudiée ;

— Grâce au modèle sélectionné on a entraîné le système d'estimation multi-modèles sur la base de données étudiée et également nous avons utilisé un ensemble des critères de validation pour tester la performance du système ;

— Les résultats trouvés étaient encourageant au niveau des critères requis pour valider le système d'estimation construit et également au niveau d'intervalle d'estimation d'effort qui recouvre la valeur réelle d'un nouveau de test.

Pour entrainer le système d'estimation par *RLM* nous devons tester la normalité de données d'apprentissage, alors que aucun test n'a était effectué pour prouver cette hypothèse. Cela peut être considérer comme première limitation de notre travail, la deuxième limitation est qu'on a mis l'hypothèse sur prédicteurs incertains comme si ils suivent tous la loi normale. Cette affirmation est subjective, il se peut que chaque prédicteur suit une loi de probabilité différente.

Perceptives

Dans un premier temps, il était raisonnable d'impliquer un composant spécial dans le framework qui configure les hyper paramètres optimaux pour chaque modèle d'estimation candidat à être sélectionner.

Dans un deuxième temps, les facteurs techniques et environnementaux peuvent influencer sur l'estimation de l'effort, nous proposons de collecter une base de données (ex : chez un partenaire industriel), contenant par exemple l'expertise de l'équipe de développement avec d'autres mesures sur des projets logiciels achevés.

Références

[1] S. Dragicevic, S. Celar, and M. Turic, "Bayesian network model for task effort estimation in agile software development," J. Syst. Softw., vol. 127, pp. 109–119, 2017.

[2] K Kavoussanakis, Terry Sloan. 2001. UKHEC Report on Software Estimation. Technical Report, UKHEC.

[3] Boudiaf, M. (2017, 06 25). Récupéré sur Mémoire: http://www.memoireonline.com

[4] M. A. Ahmed, I. Ahmad, and J. S. Alghamdi, "Probabilistic size proxy for software effort prediction: A framework," Inf. Softw. Technol., vol. 55, no. 2, pp. 241–251, 2013

[5] N.E.Fenton (SOFTWARE METRICS: A RIGOROUS AND PRACTICAL DATA, 1998)

[6] Alain Abran, Pierre N. Robillard. 1996. Function point analysis: an empiricaJ study of its measurement processes. lEEE Transactions on Software Engineering, Vol. 22, No. 12, pp. 895-910, Institute of Electrical and Electronics Engineers, NY, USA

[7] Daniel V. Ferens.1988. Software size estimation techniques. Aerospace and Electronics Conference, Vol. 2, pp. 701-705

[8] R. Agarwal, Manish Kumar, Yogesh, S. Mallick, R.M. Bharadwaj, D. Anantwar. 2001 Estimating software projects. ACM SigSoft Software Engineering Notes. Vol. 26, issue 4, pp. 60-67, ACM Press

[9] JCGM. Évaluation des données de mesure—Guide pour l'expression de l'incertitude de mesure. 2008 (cf. p. 24, 25, 61)].

[10] Sommerville. 2001. Software Engineering, Sixth Edition. Addison-Wesley Publishers Limited.

[11] Steve McConnel1. 1996. Rapid development: taming wild software schedules. Microsoft Press

[12] S. Laqrichi, thèse « Approche pour la construction de modèles d'estimation réaliste de l'effort, cout, de projet dans un environnement incertain : application au domaine du développement logiciel ».In : Hal (2015).

[13] Murali CHEMUTURI et Thomas M. CAGLEY. Mastering Software Project Management : Best Practices, Tools and Techniques. en J. Ross Publishing, juil.2010.

[14] Damjan KRSTAJIC, Ljubomir J. BUTUROVIC, David E. LEAHY et Simon THOMAS. «Cross-validation pitfalls when selecting and assessing regression and classification models ». In : Journal of cheminformatics 6.1 (2014), p. 1–15.

[15] iro.umontreal < https://www.iro.umontreal.ca/~sahraouh/qaoose/papers/Nenonen.html >

[16] (2017, 06 25). é surrRécupdevlevel : http://www.devlevel.com