UNIVERSITÉ 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

# LES TECHNIQUES D'ADAPTATION DANS L'ESTIMATION PAR ANALOGIE : ÉTAT DE L'ART

LIEU DE STAGE : LABORATOIRE SYSTÈMES INTELLIGENTS ET APPLICATIONS

RÉALISÉ PAR : MOULAY TAJ RACHIDA

SOUTENU LE : 25/06/2015

ENCADRÉ PAR :

PR. LOUBNA LAMRINI
PR. AZEDDINE ZAHI

DEVANT LE JURY COMPOSÉ DE :

PR.LOUBNA LAMRINI
PR. AZEDDINE ZAHI
PR.ABDERRAHIM BENABBOU
PR. MED CHAOUWKI ABOUNAIMA

ANNÉE UNIVERSITAIRE 2014-2015

# DEDICACE

Je dédie ce mémoire à :

Ma mère, qui a œuvré pour ma réussite, de par son amour, son soutien, tous les sacrifices consentis et ses précieux conseils, pour toute son assistance et sa présence dans ma vie, reçois à travers ce travail aussi modeste soit-il, l'expression de mes sentiments et de mon éternelle gratitude.

Mon père, qui peut être fier et trouver ici le résultat de longues années de sacrifices et de privations pour m'aider à avancer dans la vie. Puisse Dieu faire en sorte que ce travail porte son fruit ; Merci pour les valeurs nobles, l'éducation et le soutient permanent venu de toi.

Mes frères et sœurs qui n'ont cessé d'être pour moi des exemples de persévérance, de courage et de générosité.

A tous ceux qui m'ont soutenue.

# REMERCIEMENTS

**S**i ce mémoire a pu voir le jour, c'est essentiellement grâce à l'aide précieuse qui ma portée mes professeurs Madame **Loubna Lamrini et** Monsieur **Azeddine Zahi.**

**J**e tiens à lui exprimer ma reconnaissance pour l'effort et la patience qu'ils ont déployés pour m'initier à la recherche et pour l'intérêt qu'ils ont constamment manifesté pour la réalisation de ce travail. Je les prie de croire à l'expression de ma respectueuse reconnaissance.

**J**'exprime aussi ma gratitude et reconnaissance à Monsieur **Soufiane Ezghari** pour les conseils qu'il m'a prodigués et pour tout le temps et l'énergie qu'il a consacré à la réalisation de ce travail.

**Il** m'est agréable à cette occasion de remercier vivement les membres de jurys Mousieur **Mohammed Chaouki Abounaima** et Mousieur **Benabbou Abderrahim** pour leur présence très appréciée.

**S**ans oublié mes amis et ceux de prêt ou de loin qui m'ont apporté leur contribution à la réalisation de mon projet de soutenance.

**M**erci à tous.

# RESUME

L'estimation des coûts de développement de logiciels constitue toujours une tâche complexe à accomplir et une préoccupation majeure pour les gestionnaires de projets logiciels. Améliorer la précision des estimations, va leur permettra un contrôle effectif du temps et du budget durant tout le cycle de développement du logiciel.

Pour ce faire, plusieurs modèles d'estimation ont été développés et utilisés, le raisonnement par analogie (CBR) est l'une des modèles les plus étudiés dans l'estimation des coûts de logiciels. Le processus d'estimation des coûts par CBR est composé de trois étapes : La première étape consiste à la description des projets logiciels par un ensemble d'attributs considérés comme étant les principaux conducteurs du coût. La deuxième étape porte sur l'évaluation de la similarité entre le nouveau projet, pour lequel on veut estimer le coût, et tous les projets logiciels historiques. La troisième étape du processus d'estimation de CBR est nommé étape d'adaptation consiste à utiliser les coûts réels des projets historiques les plus similaires au nouveau projet pour en déduire une estimation à son coût.

La mémoire se divise en deux objectifs principaux ; le premier consiste à étudier les différentes techniques d'adaptation du modèle CBR. Le deuxième objectif est la comparaison des différentes techniques afin d'étudier leurs évolution et leurs adéquation avec différents type de projets par la réalisation d'une Framework d'expérimentation utilise la base de projets historiques COCOMO'81 .

**Mots-clés** : Estimation des coûts de logiciels, Raisonnement par analogie, CBR, Techniques d'adaptation, COCOMO'81.

# TABLE DES MATIERES

# LISTE DES FIGURES

# LISTE DES TABLEAUX

# LISTE DES ACRONYMES

LSIA : **L**aboratoire **S**ystèmes **I**ntelligents et **A**pplications

SCTC : **S**ystèmes de **C**ommunication et **T**raitement de **C**onnaissances

VIA : en**v**ironnement **I**ntelligents & **A**pplications

VASE : **V**ision **A**rtificielle & **S**ystèmes **E**mbarqués

CBR: **C**ase **B**ased **R**easonning

RA: **R**aisonnement par **A**nalogie

SLCM: **S**oftware **L**ife **C**ycle **M**anagement

RIM: **R**egular **I**ncreasing **M**onotone **Q**uantifier

OWA: **O**rdered **W**eighted **A**veraging

BK: **B**isecting **k**-medoids

AG: **A**lgorithmes **G**énétiques

RTM: **R**egression **T**o the **M**ean

ANN: **A**rtificial **N**eural **N**etwork

MSE : **E**RREUR **Q**uadratique **M**oyenne

BP : **R**etro-**P**ropagation

NABE : **N**on linière **A** **B**ase **A**nalogie

KDSI: **K**ilo **D**elivered **S**ource **I**nstructions

MRE: **M**agnitude **R**elative **E**rror

# INTRODUCTION

# 1 Présentation du lieu de stage

Ce travail a été réalisé dans le cadre d'un stage au *Laboratoire Systèmes Intelligents et Applications* (LSIA), crée en 2011. C'est une unité de Recherche du Centre d'Etudes Doctorales en Sciences et Techniques de l'Ingénieur domicilié à la Faculté des Sciences et Techniques de Fès et regroupant 17 laboratoires de recherche tous accrédités par l'Université Sidi Mohamed Ben Abdellah de Fès, et domiciliés à la Facultés des Sciences et Techniques, l'Ecole Supérieure de Technologie, la Faculté Polydisciplinaire de Taza et la Faculté de Médecine et de Pharmacie.

Le LSIA est composé de 13 enseignants-chercheurs du département d'Informatique de la FST de Fès et de 8 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 LSIA.

Le LSIA est organisé en trois équipes :

- Systèmes de Communication et Traitement de Connaissances (SCTC),
- enVironnement Intelligents & Applications (VIA),
- Vision Artificielle & Systèmes Embarqués (VASE).

# 2 Objectifs du travail

La prise de décision dans la conduite de projet informatique est fortement affectée par les aspects économiques associés au développement d'un logiciel à savoir l'effort, le coût et le temps. Ainsi, une prédiction de ces attributs à une étape précoce du cycle de développement permettra de contrôler les différents aspects de la gestion de projet : l'allocation des ressources, la planification des tâches, le déploiement de budgets, etc. L'estimation de l'effort de développement de logiciels semble susciter l'intérêt des chercheurs en génie logiciel plus que les autres attributs. Cette importance provient, d'une part, de la relation qui relie l'effort d'un projet à son délai, à son coût et à la qualité de son produit. D'autres parts à l'importance qu'occupe le coût de l'effort requis pour le développement dans le coût total du projet; le coût de l'effort représente plus de 80% du coût total de développement.

De la sorte, plusieurs travaux de recherche et d'investigations se sont intéressés à l'application de la théorie de mesure en génie logiciel [1]dans l'objectif de développer des

techniques d'estimation fiables et précises. Comme résultats, plusieurs techniques et modèles d'estimation des coûts de développement de logiciels ont été développés [1]. Ces techniques peuvent être regroupées en deux catégories :

- les techniques informelles sont celles basées sur l'avis des experts. Dans cette catégorie, nous trouvons l'estimation par jugement de l'expert [2][3], l'estimation par analogie, l'estimation ascendante et descendante [4] .
- Les techniques formelles sont celles basées sur la modélisation où il s'agit de modéliser la relation qui existe entre les facteurs conducteurs de coûts et l'effort de développement.

L'émergence de la modélisation comme technique d'estimation a induit un challenge chez les chercheurs en génie logiciel à développer des modèles d'estimation précis et fiables avec des avantages tels que : la capacité à gérer correctement l'imprécision et l'incertitude ; l'habilité à apprendre à partir de projets historiques et aussi la possibilité d'interpréter et d'expliquer la procédure d'estimation incarné par le modèle. Dans ce contexte, plusieurs modèles ont été proposés et qu'on peut classer en deux catégories :

- Les modèles paramétriques modélisent la relation entre les facteurs conducteurs de coût et l'effort par une formule mathématique prédéfinie. Les plus populaires de cette catégorie sont COCOMO I , COCOMO  et SLCM (Software LIfecycle Management)[5].
- Les modèles non-paramétriques modélisent la relation exprimant l'effort en fonction des conducteurs du coût en recourant aux techniques de l'intelligence artificielle telles que les réseaux de neurones(ANNs) [6][7], le raisonnement par analogie (CBR) [8] [9],les algorithmes génétiques [10][11] ,les arbres de décisions [12] et les systèmes à base de règles floues [13].

L'estimation basée sur le raisonnement par analogie se compte actuellement parmi les techniques d'estimation des coûts les plus étudiées par les chercheurs en génie logiciel. En effet, le raisonnement par analogie ou raisonnement à base de cas (Case Based reasonning – CBR) [14] s'avère une technique prometteuse et bien adaptée au problème de l'estimation des coûts de développement de logiciels. C'est une forme systématique du raisonnement de l'expert humain qui élabore ses décisions concernant une nouvelle prédiction en cherchant des projets semblables déjà réalisés. En conséquence, l'estimation basée sur le raisonnement par analogie permet de fournir une estimation à un nouveau projet en se basant sur l'historique des projets déjà réalisés. Ainsi, le processus de raisonnement par analogie le plus adopté en estimation des coûts est composé de trois étapes [15]:

- Identification et description des projets logiciels, y compris le nouveau projet pour lequel on veut estimer le coût, par un nombre de facteurs conducteurs significatifs et indépendants ;
- Evaluation de la similarité entre le nouveau projet logiciel et tous les projets logiciels historiques;
- Adaptation des valeurs des coûts réels des projets logiciels 'similaires' au nouveau projet afin de dériver l'effort du nouveau projet.

L'un des problèmes majeur que rencontre la procédure d'estimation basée sur le raisonnement par analogie est l'habilité à capturer la non-corrélation qui peut survenir entre les facteurs conducteurs des coûts d'un projet et son effort. Cette non-corrélation provient généralement de la variation aléatoire, de l'erreur de mesurage des attributs du projet, de la non-linéarité ou des changements de caractéristiques au fil du temps. Ainsi, les performances de l'estimation basée sur le raisonnement par analogie dépendent de sa capacité à traiter le cas des projets similaires et qui ont une valeur différente de l'effort. L'adaptation se veut une solution à cette problématique. Elle vise à :

- Capturer la structure des analogies avec le projet cible afin de réduire au minimum l'écart entre l'estimation de l'effort du projet cible et celui de ses analogies les plus proches;
- Produire des estimations les plus raisonnables possibles par rapport à l'estimation sans adaptation;
- Tolérer les imprécisions et gérer les incertitudes liées à la mesure des attributs et à l'estimation de la valeur de l'effort.

Dans ce contexte, plusieurs techniques d'adaptation ont été proposées et validées sur différentes base de projets historiques. Ainsi, le travail présenté dans ce rapport vise les objectifs suivants :

- Un état de l'art sur les techniques d'adaptation utilisées par les modèles d'estimation basée sur le raisonnement par analogie. Nous proposons une revue taxonomie
- Réalisation d'un Framework d'expérimentation mettant en œuvre les différentes techniques d'adaptation.
- Etude comparative entre les différentes techniques afin d'étudier leurs évolution et leurs adéquation avec différents type de projets.

# 3 Organisation du rapport

Le mémoire est organisé en 3 chapitres :

- le chapitre 1 présente le contexte général de notre travail. Nous présentons des généralités sur le raisonnement par analogie et la problématique de l'estimation des coûts. L'accent sera mis par la suite sur l'estimation des coûts basée sur le raisonnement par analogie.
- Le chapitre 2 sera consacré à l'état de l'art sur les différentes techniques d'adaptation.
- Le chapitre 3 présentera les résultats de l'étude expérimentale. Il décrit l'ensemble des caractéristiques des bases de données et discuté les résultats obtenu.
- Ce rapport est clôturé par une conclusion.

# CHAPITRE 1 :

# CONTEXTE DU TRAVAIL

# 1 Introduction

Vu l'importance d'une estimation fiable du coût dans la réussite d'un projet logiciel, le domaine d'estimation des coûts de logiciels a été l'objet de plusieurs travaux de recherche ainsi que d'une étroite collaboration entre les chercheurs et les industriels afin de développer et de valider des approches d'estimation propres au domaine du génie logiciel.

Ce chapitre présente un ensemble de techniques d'estimation des coûts, spécifiquement celles se basant sur la modélisation. Ainsi, nous avons décrit des exemples de modèles paramétriques et non-paramétriques, en particulier, le modèle d'estimation des coûts basé sur le raisonnement par analogie.

# 2 Raisonnement par analogie

Le raisonnement par analogie (RA) est une composante centrale de l'intelligence humaine. Il fait partie de la pensée inductive qui constitue un mécanisme important dans l'apprentissage et la résolution de problèmes chez l'être humain. En effet, face à une nouvelle situation, l'être humain fait appel à son expérience. Il se rappelle des situations semblables déjà rencontrées. Puis, il les compare à la situation courante pour trouver une nouvelle solution qui, à son tour, deviendra une expérience.

En intelligence artificielle, le raisonnement par analogie se présente sous une forme simplifiée appelée raisonnement à base de cas (Case Based Reasoning— CBR). Elle est considérée comme un paradigme de résolution de problèmes fondé sur la réutilisation des situations concrètes du problème résolues lors des expériences passées. La technique CBR diffère des autres paradigmes de résolution en IA par les caractéristiques suivantes :

- Elle modélise les expériences sous forme d'une base de cas où chaque cas correspond à une situation (expérience) résolue du problème qui est définie par (figure1):
  - **Une partie description** qui contient les valeurs des descripteurs identifiés lors de la définition du problème.
  - **Une partie solution** qui décrit la solution pour le problème spécifié.

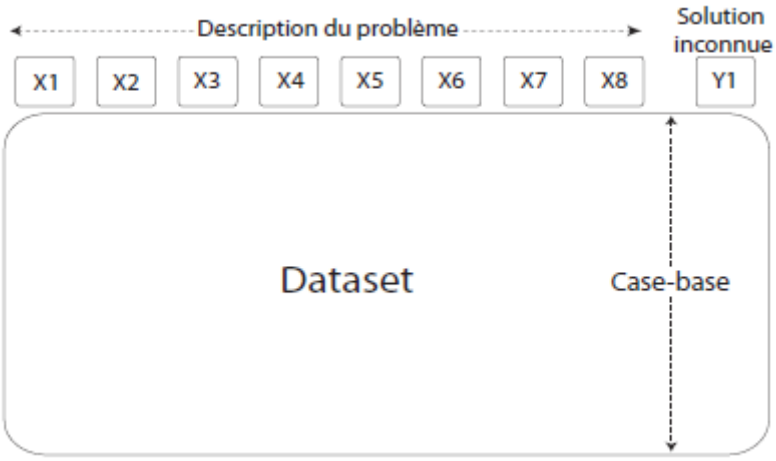


**Figure 1**: **Structure de la base de cas de la technique CBR**.

- Elle utilise directement les connaissances spécifiques de la base des cas, représentant l'expérience passée, plutôt que des connaissances générales et abstraites du domaine du problème. En effet, au lieu de modéliser les relations existantes entre les descriptions du problème et la solution par des modèles abstraits, la technique CBR cherche un cas concret, similaire à la nouvelle situation du problème, déjà résolu dans la base de cas et le réutilise pour construire la solution correspondante à la situation spécifiée.
- Elle peut être considérée comme technique d'apprentissage qui consiste à mettre à jour la base de cas après qu'un problème a été résolu. Quand un problème est résolu avec succès, l'expérience est retenue afin de résoudre des problèmes similaires à l'avenir. Lorsqu'une tentative de résolution échoue, la raison de l'échec est identifiée et retenue afin d'éviter la même erreur à l'avenir.

Le processus de la technique CBR dans la figure 2 est composé de quatre étapes [16] :

1. Retrouver les cas similaires au cas étudié : Cette phase consiste à évaluer la similarité entre le nouveau cas et tous les cas déjà résolus (base de cas) en utilisant des mesures de similarité prédéfinies. Ensuite, on choisit le(s) cas le(s) plus similaire(s) au nouveau cas.

2. Proposer une solution au problème en utilisant les cas similaires retrouvés.

3. Réviser la solution proposée; cela consiste à évaluer la solution et à lui apporter des modifications afin qu'elle soit adaptée au problème traité.

4. Retenir les éléments de cette expérience (cas + solution) qui serviront à la résolution des situations futures. En effet, son rôle est de décider des modifications qu'il faut apporter à la base de cas afin d'améliorer les performances du processus CBR.

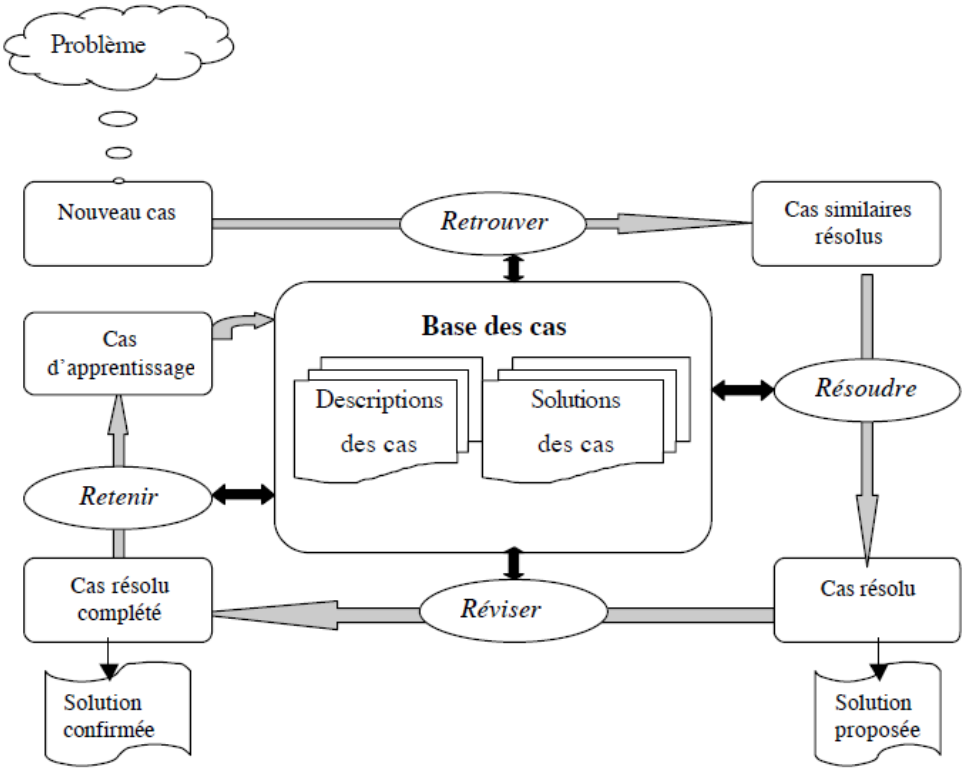


**Figure 2:Processus de résolution basé sur la technique CBR.**

# 3 Estimation des coûts de développement de logiciels

## 3.1 Techniques d'estimation des coûts de développement

L'estimation de l'effort de développement de logiciels est une tâche qui consiste à fournir, *automatiquement* ou *manuellement*, une estimation de l'effort pour un projet à partir de sa description. Le processus d'estimation se compose de deux éléments essentiels:

- les *facteurs conducteurs* des coûts: ce sont des *attributs* logiciels caractérisant un projet et influençant le coût de développement. Les facteurs sélectionnés sont ceux jugés significatifs tels que la *taille*, *l'environnement développement*, l'*expérience du personnel et les méthodes de développement.*

- la *technique* d'estimation : c'est une procédure formelle ou informelle qui permet de fournir une estimation pour un nouveau projet décrit par une assignation des facteurs conducteurs.

Plusieurs techniques d'estimation ont été proposées et validées ; elles peuvent être classées en deux catégories [8][32] ( figure 3):

- Techniques informelles: ce sont des techniques qui se basent essentiellement sur *l'expertise* d'un ou plusieurs experts pour fournir une estimation. Les techniques les plus connues sont : le *jugement de l'expert,* l'estimation par *analogie, l'estimation ascendante et descendante.* Quoiqu'elles soient largement utilisées en pratique, elles présentent plusieurs inconvénients majeurs tels que la subjectivité et la non-disponibilité des experts, notamment dans les nouvelles technologies de développement.

- Techniques basées sur la *modélisation :* ces techniques modélisent la relation entre les facteurs conducteurs des coûts et l'effort par un modèle formel, rationnel, facilement automatisable et capable de s'adapter à de nouvelles situations. Les modèles résultants sont basés généralement sur :
  - des formules mathématiques : plusieurs modèles célèbres ont été développé et validé dans ce contexte, tels que, COCOMO'81, COCOMOII.
  - des techniques issues de l'intelligence artificielle comme les arbres de régression, la logique floue, les réseaux de neurones, et le raisonnement par analogie.

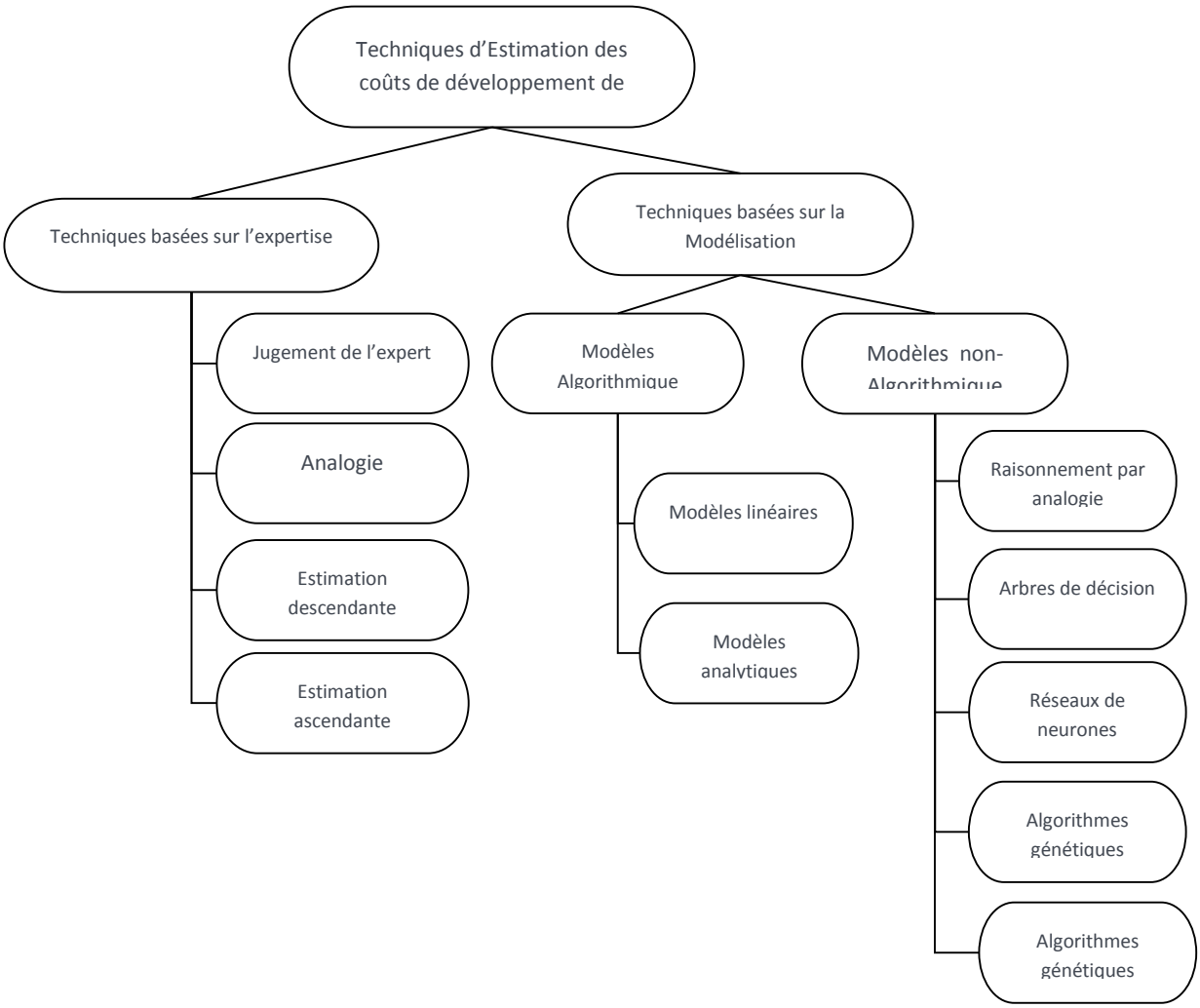


**Figure 3:Les techniques d'estimation des coûts de développement.**

## 3.2 Estimation basée sur la modélisation

La modélisation s'avère une approche judicieuse, plus scientifique et aussi plus prometteuse pour les chercheurs en génie logiciel pour remédier aux inconvénients des techniques basées sur l'expertise à savoir la subjectivité, la disponibilité des experts etc. L'estimation de l'effort logiciel par la modélisation consiste donc à construire un modèle formel qui capture la *relation* exprimant l'effort en fonction des facteurs conducteurs des coûts et dérive, automatiquement, une estimation de l'effort pour un projet à partir de sa description. La détermination de cette relation est souvent accomplie à partir d'une base de projets historiques qui contient un certain nombre de projets logiciels déjà réalisés. Chaque projet est décrit par une assignation des facteurs conducteurs et l'effort réel.

Selon la nature de la fonction d'estimation, deux catégories de modèles sont distingués [21]:

- les modèles *algorithmiques* : ils présument au préalable que la fonction d'estimation a une forme bien déterminée (linéaire ou analytique), qu'on peut décrire par nombre de paramètres connu d'avance. Le processus d'apprentissage consiste dans ce cas à déterminer les paramètres de la fonction *f* et s'appuie sur des méthodes comme la méthode des moindres carrées, l'interpolation polynomiale, etc. Parmi les modèles paramétriques les plus connus, on peut citer la régression linéaire simple ou multiple, la formule de Bayes, etc.
- Les modèles *non-algorithmiques* : ils n'exigent aucune forme de la fonction d'estimation *f*. Ils modélisent la relation reliant l'effort aux facteurs conducteurs par les *machines d'apprentissage* issues de l'intelligence artificielle telles que les réseaux de neurones, le raisonnement à base de cas et les arbres de décisions.

Ces dernières années, les modèles non–algorithmiques ont gagné une très popularité de la part des chercheurs en génie logiciel. Ceci est dû à l'habilité de ces modèles à fournir des estimations plus précise et à s'adapter à différents environnement. La problématique de l'estimation des coûts peut être formulée comme un problème d'**apprentissage supervisé.**

Où :

- les **variables prédictives** sont les facteurs conducteurs des coûts, identifiés auparavant, soient$X_1,\ldots, X_n,$ ;
- la **variable cible** est l'effort de développement, noté *effort* ;
- **la base d'apprentissage** est la base de projets historiques ;
- La **fonction** d'apprentissage, $f$, est la relation reliant l'effort aux facteurs conducteurs des coûts exprimée par:

$$\boldsymbol{Effort = f(X1, X2, \ldots, Xn)} \quad (1)$$

La figure 4 présente le processus de construction et d'estimation de l'effort par un modèle d'apprentissage.

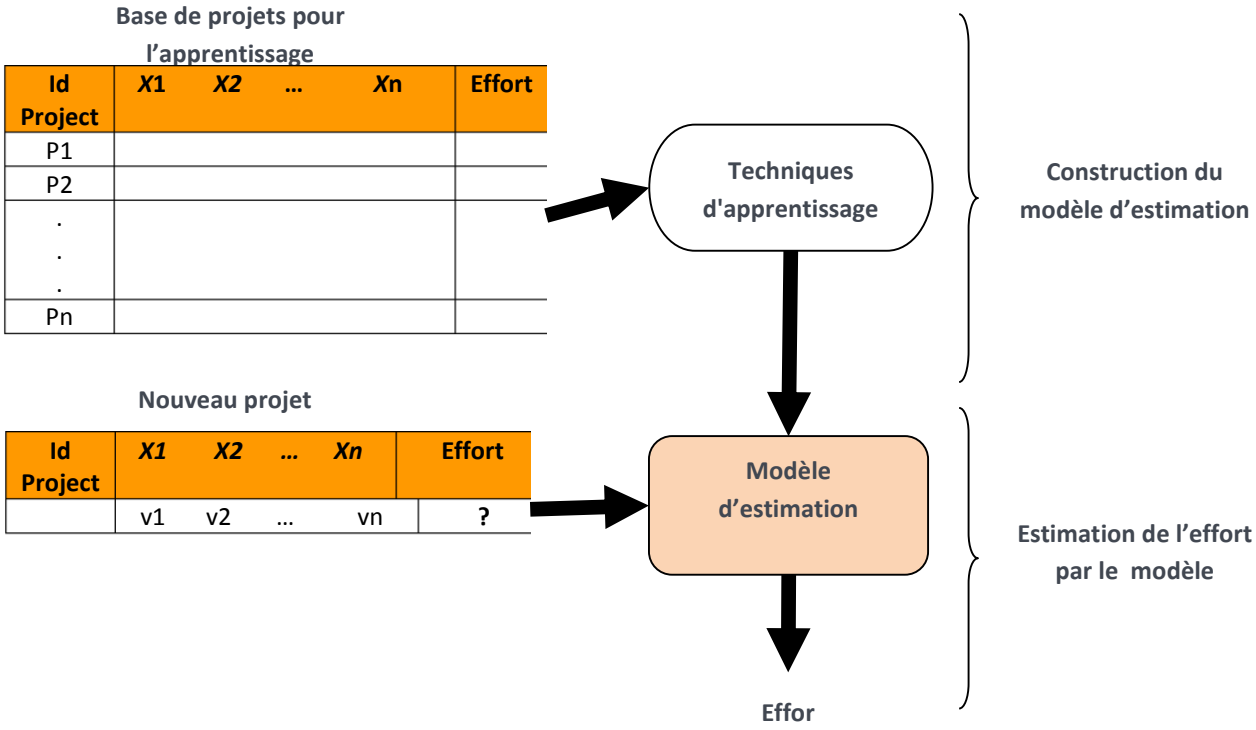


**Figure 4: Processus de construction d'un modèle d'estimation.**

# 4 Estimation des coûts basée sur le raisonnement par analogie

## 4.1 Motivations

Comme nous l'avons déjà mentionné, la technique CBR constitue un paradigme de résolution de problèmes qui se base sur la réutilisation des expériences passées. Ce paradigme est particulièrement adapté aux domaines où :

- l'expérience de résolution des problèmes repose principalement sur l'expertise humaine.
- la relation entre les descripteurs du problème et la solution est très complexe, n'est pas toujours compréhensible et ne peut être modélisée et formalisée par des outils théoriques.
- les connaissances disponibles sur le problème étudié sont insuffisantes pour construire des abstractions modélisant le problème ;
- le processus de résolution du problème doit être facile à expliquer,

Le domaine de l'estimation des coûts de développement de logiciels s'inscrit parfaitement dans cette situation. En effet, l'être humain intervient de manière très forte que ça soit dans la sélection des facteurs conducteurs des coûts, le mesurage de ces attributs, ou même l'estimation de l'effort pour un nouveau projet. Ainsi, la technique CBR s'avère parfaitement adéquate et présente plusieurs avantages sur les autres techniques couramment utilisées en estimation des coûts.

## 4.2 Modèle classique d'estimation par analogie

La mise au point d'un modèle d'estimation des coûts de développement de logiciels basé sur la technique CBR a été proposé par Vicinanza, Prietula et Mukhopadhyay [14] vers 1990. Les auteurs ont reformulé le processus de la technique CBR afin d'être appliqué en estimation des coûts. L'idée repose sur l'affirmation « *les projets similaires ont des coûts similaires* », qui a été déployée comme suit :

- **Identification et description** des projets logiciels, y compris le nouveau projet pour lequel on veut estimer le coût. L'objectif de cette étape est de caractériser les projets logiciels par un ensemble d'attributs indépendants et significatifs pour l'estimation des coûts. Cette étape s'appuie un ensemble de facteurs capturant la qualité du logiciel et influençant le coût tels que la fiabilité et la complexité logicielle, la compétence des analystes intervenant dans le projet et les méthodes de développement utilisées.

- **Recherche des projets similaires** au nouveau projet logiciel. Cette étape consiste à rechercher les projets les plus similaires au projet cible en évaluant la similarité entre le nouveau projet et les projets historiques déjà réalisés. Le choix la mesure de similarité influence fortement la précision des estimations, impactée à son tour par les analogies trouvées (projets similaires). Pour cela plusieurs mesures de similarité ont été proposées et validées dans le contexte de l'estimation basée sur la technique CBR [17]. Les plus populaires sont basées sur des fonctions de distance. Ainsi, la similarité entre deux projets logiciels est évaluée en fonction de la distance entre les attributs de ces projets par la fonction définie par :

$$\boldsymbol{d(P1, P2, V)} = \frac{\mathbf{1}}{(\sum_{\boldsymbol{Vj}} \boldsymbol{dVj(P1,P2)})^{\mathbf{1/2}}} \qquad (1)$$

Où

- P1et P2 sont des projets logiciels,
- V est l'ensemble des attributs
- $dVj(P1, P2)$ est la distance entre les deux projets P1 et P2 selon l'attribut Vj

La distance entre deux projets selon un attribut donné est définie selon sa nature:

- dans le cas d'un attribut numérique, on utilise la distance Euclidienne définie comme suit :

$$\boldsymbol{dVj(P1, P2) = (Vj(P1) - Vj(P2))^2} \quad (2)$$

- dans le cas d'un attribut qualitatif, on utilise la distance d'Égalité définie par :

$$\boldsymbol{dVj(P1, P2)} \begin{cases} \boldsymbol{1\ si} & \boldsymbol{\left(Vj(P1) = Vj(P2)\right)} \\ \boldsymbol{0\ sinon} & \end{cases}$$

- **Adaptation** L'objectif de cette étape est de dériver une estimation pour le nouveau projet en utilisant les valeurs de l'effort de projets similaires. Dans cette étape, deux questions doivent être abordées.
  - Combien de projets similaires doivent être utilisés dans l'adaptation ?
    Dans la littérature, il n'y a pas de règles prédéfinies pour le choix du nombre de projets les plus similaires, k. En général, le nombre k est fixé à une constante (souvent entre 1 et 10) ou est évalué par le nombre de projets logiciels ayant un degré de similarité, La plupart des expérimentations de validation de ce stratégies de choix du nombre k ont avantagé l'utilisation d'une valeur de k inférieure ou égale à 3 [18][19].
  - Comment adapter les analogies choisis pour générer une estimation pour le nouveau projet ?
    Il s'agit de choisir une technique pour combiner les efforts des k projets similaires au projet cible.

## 4.3 Estimation par analogie floue

En estimation des coûts la plupart des facteurs conducteurs des coûts (attributs) sont mesurés sur une échelle ordinale par des valeurs qualitatives en utilisant des termes linguistiques tels que *très bas, bas et excellent*. Par exemple dans le modèle COCOMO'81, 15 parmi 17 facteurs sont mesurés sur une échelle composée de six valeurs linguistiques *très faible, faible, moyen, élevé, très élevé et extra élevé* [14]. La technique CBR en estimation des coûts, gère ces valeurs linguistiques le long du processus comme suit :

- Dans l'étape de description des projets logiciels, les valeurs linguistiques associées aux différents attributs affectant le coût sont représentées par des intervalles classiques ou des nombres réels.

- Dans l'étape de recherche des projets similaires considèrent les valeurs linguistiques comme des variables binaires ; la similarité entre les projets logiciels est évaluée en utilisant la distance d'égalité.
- Dans l'étape d'Adaptation, le choix des projets logiciels les plus similaires au nouveau projet utilise la technique du seuil.

Cette gestion ne permet pas à la technique CBR de capturer *l'imprécision* induite, lors du processus de mesurage des attributs et *l'incertitude* des estimations générées. Dans ce contexte, le modèle d'estimation « *Fuzzy Analogy* » a été développé. C'est un modèle basé sur la logique floue qui est considéré comme une «fuzzification» de la technique CBR classique. Son objectif est de permettre la tolérance des imprécisions et des incertitudes tout au long du processus d'estimation par analogie. Ainsi, le processus d'estimation par analogie floue devient :

- **Identification et description projets** : dans cette étape chaque projet de logiciel est décrit par un ensemble d'attributs où chaque attribut est mesuré par un ensemble de valeurs linguistiques. Les termes linguistiques utilisés pour mesurer les attributs sont représentées par des *ensembles flous* au lieu des ensembles classiques. En effet, chaque terme est représenté par un ensemble flou associé à une fonction *d'appartenance* qui représente le degré d'appartenance à l'ensemble flou. La fonction d'appartenance d'un ensemble flou permet de modéliser le passage graduel d'une valeur linguistique à une autre, i.e., plus la valeur de la fonction d'appartenance est grande plus le facteur s'identifie bien dans la valeur linguistique.
- **Recherche des projets similaires**: Cette étape est basée sur une mesure de similarité basée sur la représentation floue des attributs et les opérateurs d'agrégation OWA. Elle permet d'évaluer la similarité floue en deux étapes [14] :
  - La première consiste à évaluer les similarités individuelles selon chaque variable linguistique (attribut). La similarité entre deux projets $P_1$ et $P_2$, et la variable linguistique $V$, est évaluée par l'une des formules :

$$d_{V_j}(C_1, C_2) = \begin{cases} \max_k \min\left(\mu_{V_j^k}(C_1), \mu_{V_j^k}(C_2)\right) \\ max - min\ aggregation \\ \sum_k \mu_{V_j^k}(C_1) \times \mu_{V_j^k}(C_2) \\ sum - product\ aggregation \\ \min_k \max\left(1 - \mu_{V_j^k}(C_1), \mu_{V_j^k}(C_2)\right) \\ min - Kleene - Dienes\ aggregation \end{cases} \quad (3)$$

Où

- $d_{V_j}(C_1, C_2) = 1$ signifie que les deux cas$C_1 et\ C_2$sont parfaitement similaires selon la variable $V_j$,
- $d_{V_j}(C_1, C_2) = 0$ Signifie que $C_1 et\ C_2$ ne sont pas similaires,
- et $0 < d_{V_j}(C_1, C_2) < 1$ signifie que $C_1 et\ C_2$sont partiellement similaires.

– Dans la deuxième les similarités individuelles obtenues sont combinées pour calculer la similarité globale. La similarité globale entre deux cas est évaluée en combinant les similarités individuels à l'aide des quantificateurs RIM (Regular Increasing Monotone Quantifier).

Tels que $all, most, many,\ at - most\alpha, et\ there\ exists.$

$$d(\mathbf{C_1}, \mathbf{C_2}) = \begin{cases} \textbf{all of}\left(\mathbf{d_{A_j}(C_1, C_2)}\right), \\ \textbf{most of}\left(\mathbf{d_{A_j}(C_1, C_2)}\right), \\ \textbf{many of}\left(\mathbf{d_{A_j}(C_1, C_2)}\right), \\ \dots \\ \textbf{there exists of}\left(\mathbf{d_{A_j}(C_1, C_2)}\right) \end{cases} \quad (3)$$

Les quantificateurs RIM permet d'indiquer la proportion des similarités individuelle nécessaires. Ainsi la similarité globale est donnée. Implémentation d'un quantificateur RIM se fait par un opérateur OWA [20].Il est défini par :

– Un ensemble flou associé à une fonction d'appartenance défini sur l'intervalle [0,1] et vérifier($Q(0) = 0\ et\ Q(1) = 1$). Dans la pratique ont choisi

– $Q(x) = x^{\alpha}\ \alpha > 0$.

– Un vecteur poids $W = (w_j)$ de dimension $c$, avec $c$ , le nombre de variable linguistique, tels que :

- $w_i \in [0,1]$,

- $\sum_i w_i = 1, et$

Ce vecteur permet de définir l'imprécision et l'importance des variables dans l'évaluation de la similarité globale.

Pour deux cas$C_1 et\ C_2$, le poids $u_j(C_1, C_2)$ selon la variable $V_j$est donnée par :

$$\boldsymbol{w_j(C_1, C_2) = Q\left(\frac{\sum_{k=1}^{j} u_k}{T}\right) - Q\left(\frac{\sum_{k=1}^{j-1} u_k}{T}\right)} \quad (4)$$

Où :

- $u_k$Est le poids associé au $k^{éme}$ attribut décrivant les cas.
- $T$ Est la somme des $\mu_k$. Les poids $w_j(C_1, C_2)$dépendent de$C_1 et\ C_2$.

La similarité globale est évaluer par :

$$\boldsymbol{d(C_1, C_2) = \sum_{j=1}^{M} w_j(C_1, C_2) d_{A_j}(C_1, C_2)} \quad (5)$$

- L'adaptation de cas : Cette étape permet de déduire une estimation du coût du nouveau projet P, en utilisant les coûts réels des projets logiciels les plus similaires à P. Fuzzy analogie propose la solution pour le choix du nombre analogie et la formule d'adaptation.

  – **Le nombre des analogies :** la stratégie se base sur les degrés de similarités, $dv\ (P, Pi\ )$ par la représentation de la qualification « étroitement similaire » par ensemble flou défini dans l'intervalle[0,1]. La figure 5 montre un exemple de fonction d'appartenance pour cette qualification. Dans cet exemple, tout cas ayant un degré de similarité supérieur ou égal à 0,5, contribuera à l'adaptation de la solution. la contribution de chaque projet *Pi* est pondérée par µ*voisinage* de1 (*d(P,Pi)*).

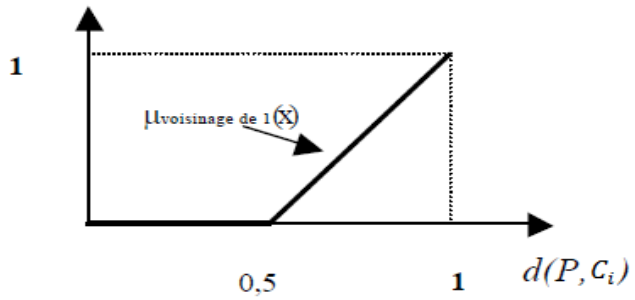


**Figure 5:Exemple d'une représentation de la qualification étroitement similaire par un ensemble flou.**

- **La formule d'adaptation :** la formule d'Adaptation la plus utilisée pour déduire une estimation du coût d'un nouveau projet P est celle utilisant la moyenne arithmétique (pondérée). La formule d'adaptation utilisée est :

$$\boldsymbol{Effort} = \frac{\sum_{i=1}^{n} \boldsymbol{\mu_{voisinage\ de\ P}(d(P,P_i))} \times \boldsymbol{Effort(,P_i)}}{\sum_{i=1}^{n} \boldsymbol{\mu_{voisinage\ de\ P}(d(P,P_i))}} \quad (6)$$

# CHAPITRE 2 :

# ETAT DE L'ART SUR LES TECHNIQUES D'ADAPTATION

# 1 Introduction

L'*adaptation* est une étape importante dans la procédure d'estimation de l'effort logiciel basée sur le raisonnement par analogie. Elle vise à refléter la structure des analogies avec le projet cible et capturer les différences entre la description du projet cible et les descriptions des analogies afin de générer une solution plus raisonnable. Une technique d'adaptation se caractérise par comment déterminer deux paramètres interdépendants :

- Le nombre d'analogies à utiliser dans l'estimation de l'effort.
- La stratégie d'adaptation qui détermine comment combiner les projets similaires pour dériver une estimation. .

Ces dernières années, plusieurs techniques d'adaptation ont été proposées dans le contexte de l'estimation des coûts de développement. Ces stratégies souffrent de problèmes communs tels qu'ils ne sont pas en mesure de produire les mêmes résultats lorsqu'elles sont appliquées dans des contextes différents. Dans ce contexte, nous allons présenter un état de l'art sur les techniques d'adaptation afin de guider leur contexte d'utilisation. Nous allons classifier ces techniques selon qu'elles interviennent au niveau de la sélection des projets similaires ou au niveau de la stratégie d'adaptation.

# 2 Les stratégies de sélection des projets similaires

Au niveau de la sélection des projets similaires à utiliser dans l'adaptation de l'effort, dans la littérature plusieurs techniques existes, on peut les classées en trois catégories[21] :

- Les techniques qui sélectionnent un nombre fixe de projets similaires de manière empirique ou en variant ce nombre dans un intervalle.[22][23] [24] ;
- Les techniques qui sélectionnent le nombre de projets similaires dynamiquement en faisant recours aux techniques de regroupement[25] ;
- Les techniques qui sélectionnent le nombre de projets similaires en se basant un seuil de similarité avec le projet cible [9][26].

## 2.1 Sélection d'un nombre fixe de projets similaires

Cette stratégie basée sur l'utilisation du nombre K fixe de projets similaires.

Tous les projets de la base de données utilisent le même nombre K pour choisir ses projets les plus similaires.

- Le nombre de projets est fixé dans un intervalle, souvent entre 1 et 10 ;
- La plupart des expérimentations de validation de cette stratégie de choix du nombre de projets similaires ont avantagé l'utilisation d'une valeur inférieure ou égale à 3.

- **Exemple :**

Supposons que nous ayons fixé le nombre k à 2 et que les trois premiers projets similaires à un nouveau projet P (P1, P2 et P3) aient respectivement les degrés de similarité suivants 3,30, 4,00, et 4,01.

Dans ce cas, nous utilisons seulement les deux premiers projets (P1 et P2) pour déduire une estimation du coût du nouveau projet P.

## 2.2 Sélection des projets similaires basée sur le regroupement

La prédiction des coûts des projets logiciels basés sur les techniques de regroupement comme stratégie de sélection des projets similaires passe par les étapes suivantes :

### 2.2.1 Le regroupement des projets historiques dans des clusters

Le regroupement des projets par l'algorithme k-medoids permet de regrouper les projets similaires au sien des clusters. Cela nous permet de découvrir la structure de la base de donnée efficacement et donné automatiquement le meilleur nombre K de cas plus similaire.

Pour deviner le nombre du cluster il faut employer l'algorithme k-medoids dichotomie (Bisecting k-medoids) (BK) pour le regroupement. Les clusters ainsi obtenues sont structurés comme un arbre binaire hiérarchique. La décision s'il faut continuer la classification ou arrêt dépend de la comparaison du degré de *compacité* entre les fils et leur parent directe dans l'arborescence (figure 6) :

- Si le maximum de compacité des clusters du fils est plus petit que la compacité de leur parent direct la classification est continuée.
- Sinon la classification arrêté et le cluster parent est considéré comme un nœud sans descendant.

La compacité moyenne de chaque cluster est définie comme :

$$\boldsymbol{Compacité} = \frac{\mathbf{1}}{\boldsymbol{n}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{k}} \sum_{\boldsymbol{j=1xj\in Cj}}^{\boldsymbol{n}} ||\boldsymbol{xj} - \boldsymbol{vi}||^{\mathbf{2}} \qquad (7)$$

Où :

|| • || est la norme euclidienne habituelle, $\mathbf{x}_j$ est le donnée objet, $\mathbf{v}_i$ est le centre de cluster $\mathbf{(C_i)}$ et **k** est le nombre de cluster.

Une valeur faible de cette mesure indique une haute homogénéité (moins de dispersion).

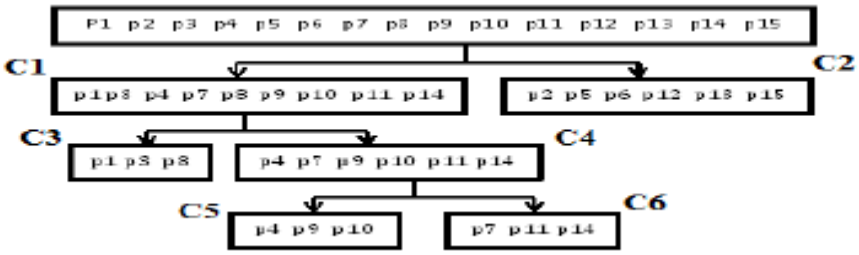


**Figure 6: Illustration de l'algorithme k-medoids dichotomie**

La figure 6 est un arbre très simple de BK qui peut se former sur une base de données simple de 15 projets. Il montre comment le cluster principale est partitionné récursivement en quatre feuilles, les clusters sont : C2, C3, C5 et C6.

### 2.2.2 Evaluation de la similarité

Contrairement aux méthodes de sélection de K fixe, BK n'a pas besoin d'une intervention d'expert pour découvrez les caractéristiques de base donnée afin de définir le nombre des arbres à construire ou le nombre de cas à utiliser dans l'estimation. Le nombre de projets similaire est le nombre des instances de cluster dont la médoïde est plus près du projet cible.

- Un médoïde peut être définie comme une instance du cluster, dont la dissimilarité moyenne à tous les cas dans le cluster est minimale c-à-dire qu'elle est la point la plus centrale dans le cluster.

### 2.2.3 L'estimation de l'effort à partir des projets similaires

$$\boldsymbol{Effort(pi) = \frac{1}{K}\sum_{j=1}^{K} Effort(pj)} \quad (8)$$

Où : K est le nombre d'instances d'entrainement qui sont dans le cluster sélectionné.

## 2.3 Sélection des projets basée sur le seuil de similarité

Cette technique permet l'apprentissage de nombre K par la mesure d'un seuil à partir des similarités entre le nouveau projet et la base de cas [27], dans ce sens le seuil va nous permet de sélectionné seulement les projets le plus similaire. Pour cela le seuil est mesuré en se basant sur une famille d'opérateur nommé le moyen quasi-arithmétique, ces opérateurs permet d'augmenter la valeur de seuil par rapport au nombre des similarités élevés. Dans la suite on présente les opérateurs moyens quasi-arithmétiques et puis nous expliquerons la méthode l'apprentissage de seuil.

### 2.3.1 Les opérateurs quasi-arithmétiques

Les opérateurs moyens quasi-arithmétiques, sont un ensemble d'opérateurs d'agrégation, généralisant le moyen arithmétique. Ces opérateurs sont caractérisés par deux ensembles de propriétés:

- les propriétés comportementales telles qu'interopérabilité, l'importance et comportement décisionnel.
- les propriétés mathématiques telles que la continuité, la monotonie, la neutralité, la stabilité et l'associativité:

$$\boldsymbol{G^{(n)}(g) = f^{-1}(\frac{f(g1)+f(gn)+\cdots+f(gn)}{n})} \quad (9)$$

Où :

- **g**=(g1,g2,……,gn) est le vecteur des éléments de l'agrégation.
- **n** est le nombre d'éléments agrégés.
- **f** est une fonction continue et strictement monotone.

En définissant la fonction, nous pouvons concevoir plusieurs moyennes. Ainsi, avec :

- ✓ f (x) = x, on obtient la moyenne arithmétique classique
- ✓ f (x) = log (x), nous obtenons la moyenne géométrique
- ✓ f (x) =$\frac{1}{x}$ , nous obtenons la moyenne harmonique.

Parmi les opérateurs quasi-arithmétiques le moyen de Minkowski est le plus adéquat, il est définie par la fonction $f(x) = x^P$, c'est un opérateur compensatoire; il compense des scores plus faibles par autre plus élevées en fonction de la valeur de le paramètre p qui définit le degré de compensation. Alors le moyen de Minkowski défini par :

$$\boldsymbol{M^{(n)}(g1, g2, \dots, gn) = (\frac{g1^p+g2^p+\cdots+gn^p}{n})^{\frac{1}{p}}} \quad (10)$$

### 2.3.1 Apprentissage du seuil à l'aide du moyen quasi-arithmétique

L'objectif est d'apprendre, à partir des similarités entre un nouveau projet et la base de cas, le seuil adéquat qui assure la sélection des cas d'adaptation appropriées pour générer l'estimation de l'effort optimale du projet cible.

Supposant que nous vouloir estimer l'effort d'une nouveau projet logiciel Pi et le vecteur de similarité entre Pi et les projets historiques est donne par S= {S1, S2,…,Sn} ou Si est le degré de la similarité entre Pi et le projet Pj et n le nombre des projets historiques. L'apprentissage du seuil est réalisé en deux étapes [27] :

1. Calculer le seuil Th on utilise la moyenne de Minkowski par :

$$Th = \left(\frac{S1^p + S2^p + \cdots + Sn^p}{n}\right)^{1/p} \quad (1)$$

2. Nous sélectionnons les cas ayant un degré de similarité supérieur ou égale au seuil calculé.

- **Exemple**

Considérant que nous avons cinq projets P1, P2, P3, P4 et P5 associé à des vecteurs de similitude suivants:

S1 = {0.5; 0.7; 0.75; 0.8; 0.9; 0.95}

S2 = {0.2; 0.7; 0.75; 0.8; 0.9; 0.95}

S3 = {0.2; 0.2; 0.5; 0.55; 0.8; 0.95}s

S 4 = {0.2; 0.2; 0.2; 0.5; 0.6; 0.7}

S5= {0.2; 0.2; 0.2; 0.3; 0.4; 0.5}

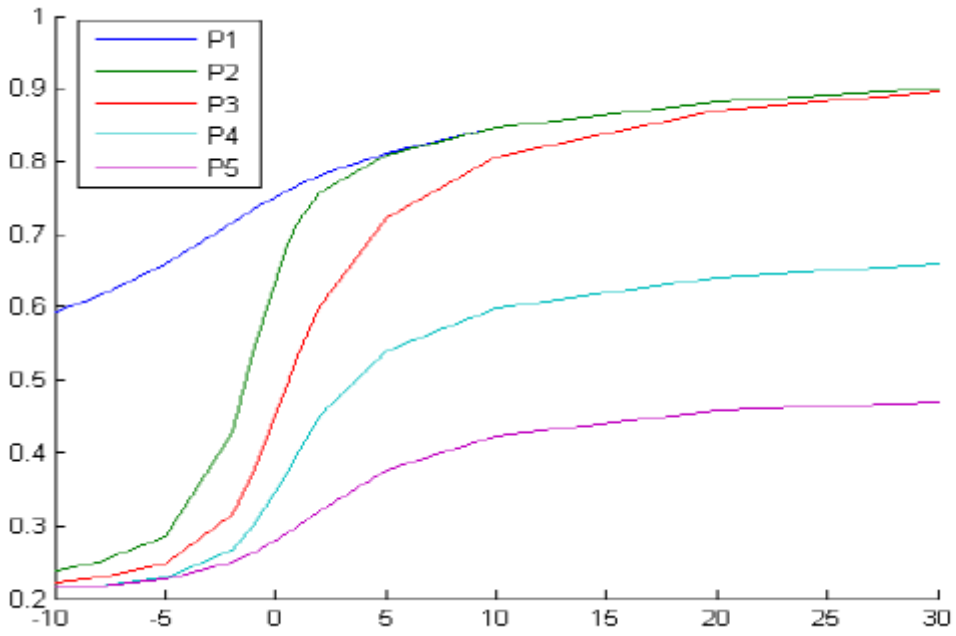


**Figure 7 : les seuils obtenus en utilisant le moyen de Minkowski avec différents valeurs de *p* pour P1, P2, P3, P4 et P5.**

La figure montre les seuils obtenus à l'aide de Minkowski pour chaque projet. Nous remarquons que P1 atteindre rapidement le degré de seuil élevé car elle contient plusieurs similitudes élevées et les autres projets atteignent moins rapidement causés par les degrés des similitudes inférieures.

Par conséquent, la méthode proposée est souple dû au fait qu'elle assure souvent la sélection des projets optimales même nous pouvons avoir des similitudes plus faibles surtout lorsque le nombre de similitudes plus élevés est très petit. Par exemple, si p = 5, puis le seuil obtenu est: thp1 = 0.81, thp2=0.81, thp3= 0.72, .thp4= 0.54, thp5= 0.37.

# 3 La stratégie d'adaptation

Les stratégies d'adaptation sont les techniques d'ajustement utilisée pour combiner les coûts réels des projets les plus similaires au projet, il n'y a pas de schéma général pour la classification de ces stratégies par conséquent, nous les avons divisés en deux catégories selon la procédure qu'ils suivent. Ces catégories sont:

- Ajustement linéaire ;
- Ajustement non linéaire.

## 3.1 Ajustement linéaire

### 3.1.1 Ajustement linéaire de l'effort

Ce genre de fonctions d'adaptation tente d'ajuster les valeurs de l'effort des projets similaires en utilisant une certaine mesure de tendance centrale comme :

- La moyenne non pondérée :

$$\mathbf{Effort(pi)} = \frac{\boldsymbol{Effort(p1)+Effort(p2)+\cdots.+Effort(pN)}}{\boldsymbol{N}} \quad (11)$$

- La moyenne pondérée par mes poids :

$$\boldsymbol{Effort(pi)} = \frac{\boldsymbol{w1*Effort(p1)+w2*Effort(p2)+\cdots.+wN*Effort(pN)}}{\boldsymbol{\sum_{i=1}^{N} wi}} \quad (12)$$

➢ Où : $w_1$ à $w_N$ sont les poids associées à chaque effort selon sa proximité du projet cible.

Ces fonctions sont simples et plus abstrait mais rarement appliqué pour prédire l'effort car :

- Ils ne tiennent pas comprennent les différences des caractéristiques entre le projet cible et les principaux projets similaires ;
- Ils ignorent l'influence de la caractéristique taille et les autres caractéristiques utiles sur les estimations définitives ;

Toutefois, pour une meilleure utilisation de ces caractéristiques, il est recommandé de les appliquer après avoir réglé chaque effort consulté individuellement.

### 3.1.2 Ajustement linière de la taille

La fonction de la taille est considérée comme la fonction la plus prévisible pour l'estimation de l'effort car :

- La taille a des informations utiles pour l'estimation du coût du logiciel tel que la complexité du projet, langueur du projet, …
- Les études ont confirmé qu'il y a une forte corrélation significative entre la fonction de la taille et l'effort prévisible et, par conséquent, ils pourraient former un facteur d'ajustement utile dans CBR.

Ajustement par la taille est effectuée en fonction de l'extrapolation linéaire entre la taille du projet cible et les projets récupérées. Une fois les projets les plus similaires sont identifier leurs valeurs d'effort sont ajusté et adapté pour tenir en compte des différences. La première fonction de cette catégorie a été développée par Walkerden & Jeffery [28] comme représenté dans l'équation 16. :

$$\boldsymbol{Effort(pt) = \frac{Effort(Pa)}{\mathbf{FP(Pa)}} * FP(Pt)} \quad (13)$$

La principale limitation de cette approche est que la taille et l'effort sont censés être fortement corrélés.

En outre, le mécanisme est peut-être pas applicable lorsque :

- La caractéristique taille n'est pas en points de fonction ;
- Lorsque les projets sont décrits avec un nombre arbitraire de caractéristique relatives à la taille.

Pour remédier à ces limitations Mendes et al[29] proposé une fonction d'adaptation de la taille linéaire amélioré qui s'étend la fonction Walkerden et Jeffery et en tenant compte du nombre arbitraire de taille et le nombre des plus proches analogies.

La forme finale de cette fonction est représentée dans l'équation 17 :

$$\boldsymbol{Effort(pt) = \frac{1}{K}\sum_{i=1}^{K}\frac{1}{M}\left(\sum_{j=1}^{M}\frac{fjt}{fji}Effort(pi)\right)} \quad (14)$$

Où:

- **K** est le nombre d'analogies les plus proches.
- **M** est le nombre de caractéristiques de taille concernés

- $\mathbf{f_{jt}}$ est la valeur de la caractéristique $j_{th}$ du projet cible
- $\mathbf{f_{ji}}$ est la valeur de la caractéristique j de projet i.

### 3.1.3 Ajustement linière de similarité :

Les mesures de similarité entre les projets des logiciels jouent un rôle crucial dans les modèles de CBR. Le degré de similitude peut refléter le degré de différence qui doit être transformé dans la valeur de l'effort.

Ce genre d'adaptation vise à régler les valeurs d'effort récupérée basés sur leurs degrés de similitude soit locale ou global avec un projet cible. La forme générale de cette technique implique la somme du produit des degrés de similitudes normalisée avec la valeur d'effort récupérée comme indiqué dans l'équation :

$$\boldsymbol{Effort(pt) = \sum_{i=1}^{K}\left[\frac{SM(pt,pi)\times Effort(pi)}{\sum_{i=1}^{K} SM(pt,pi)}\right]} \qquad (15)$$

- Où **SM** est le degré de similarité globale entre les deux projets et **K** est le nombre d'analogies.

Pour de meilleurs résultats, il est recommandé de l'utiliser lorsque on a suffisamment de nombre d'analogies est utilisé.

### 3.1.4 Ajustement linéaire de la productivité

La méthode «régression vers la moyenne» (**RTM**) pour ajuster l'estimation des couts utilisant la productivité réajustée du nouveau projet et la productivité des analogies les plus proches comme suivant :

$$\boldsymbol{Effort(pi) = FPca \times \widehat{Pi}} \qquad (16)$$

$$\boldsymbol{\widehat{Pi} = Pca + (M - Pca) \times (1 - r)} \qquad (17)$$

- Où

- **Pi** est la productivité ajusté (Productivité=cout/point de fonction) du nouveau projet i.
- **Pca** la productivité de l'analogie la plus proche de x,
- **M** est la productivité moyenne des projets similaires,
- **r** la corrélation historique ente la productivité de base non-ajuste et la productivité actuelle.

L'ensemble des données doivent être ajustées pour les rapprocher de la valeur moyenne (c'est à dire de tenir des valeurs moins extrêmes). En outre, il est recommandé à partitionner l'ensemble de données en sous-ensembles plus homogènes afin que la procédure régresse vers une moyenne locale de la productivité.

## **3.2** Ajustement non linéaire

### 3.2.1 Réseau de neurone artificiel

Artificial Neural Network (ANN) est une technique d'apprentissage artificielle qui a joué un rôle important dans le rapprochement des relations complexes. Grâce à son excellente capacité d'approximation, ANN a été largement appliquée pour la recherche sur l'estimation des coûts de logiciels

Dans l'architecture ANN il existe généralement trois couches: la couche d'entrée, les couches cachées, et la couche de sortie. Toutes les couches sont composées de neurones Les connexions entre les neurones à travers les couches représentent la transmission d'informations entre les neurones. ANN a la forme mathématique suivante:

$$\boldsymbol{y = y(x) = \sum_{j=1}^{J} wjf\left(\sum_{i=1}^{I} vijf(xi) + \alpha j\right) + \beta + \varepsilon} \qquad (18)$$

Où

- ***x*** est un vecteur d'I-dimensionnelle avec {x1, x2, ..., xI} comme ses éléments ;
- ***f (·)*** est la fonction de transfert définie par l'utilisateur ;
- $\boldsymbol{\varepsilon}$ est une erreur aléatoire avec 0 comme moyen ;
- ***J*** est le nombre total de neurones cachés ;
- ***vij*** est le poids de la connexion entre le neurone d'entrée i et le j$^{ème}$ neurone caché ;
- ***a***j est le biais du j$^{ième}$ neurone caché ;
- ***wj*** est le poids de la connexion entre le neurone caché j$^{ième}$ et le neurone de sortie ;
- $\boldsymbol{\beta}$ est le biais dans le neurone de sortie.

Les poids et les biais sont déterminés par la procédure d'apprentissage qui minimise l'erreur d'entraînement. La fonction d'erreur d'entraînement couramment utilise l'erreur quadratique moyenne (MSE) qui présenté comme suit:

$$\boldsymbol{E = \frac{1}{I}\sum_{i=1}^{I}(ts - ys)^2} \qquad (19)$$

Où

- ***ys*** est la sortie du réseau lorsque l'échantillon s correspond à l'entrée de l'ANN
- ***ts*** est la sortie d'entrainement de s.

L'algorithme classique de Retro-propagation (BP) est souvent utilisé pour mettre à jour les poids et les biais pour minimiser l'erreur d'entrainement.

Comme indiqué par l'équation (21), ANN a trois paramètres définis par l'utilisateur:

- Le nombre de couches d'entrées ;

- Le nombre de nœuds cachés ;
- Le type de fonction de transfert.

Ces paramètres ont un impact important sur les performances de prédiction ANN. ANN est utilisée comme composante d'ajustement non linéaire dans le système d'estimation des coûts non linière à base analogie (NABE).

### 3.2.2 Ajustement non linière par ANN

Basé sur le modèle d'ajustement linière propose par Chiu and Huang [30] on extrait le modèle d'ajustement à la forme additive suivant :

$$\boldsymbol{Effort(pi) = Cw + f(Spi, Sk)} \quad (20)$$

Où

- ***f(.)*** est une fonction arbitraire d'approximation de la mise à jour nécessaire pour changer la solution retrouver et la solution cible. f(.) est le modèle ANN ;
- $Sp_i$ est le vecteur des caractéristiques du projet $p_i$ ;
- *Sx* est la matrice des caractéristiques des k projets les plus similaires ;
- *Cw* est la valeur du cout obtenir par CBR sans ajustement (ou solution retrouver).

Le système NABE se compose de deux étapes[31] : Dans un premier temps le système NABE obtient la solution récupérée (non ajustée) et les composants d'entrainement. Dans la seconde stade, la composante non linéaire est utilisée pour produire la mise à jour, puis la mise à jour est ajoutée avec la solution récupérée pour générer la prédiction finale.

**Etape I-entraînement**

Les procédures de la phase I sont présentées dans la figure. 2. L'approche jackknife [32]est utilisée pour l'entrainement de l'ajustement non linéaire (ANN). Pour chaque projet dans l'ensemble des données d'entraînement, les étapes suivantes sont effectuées:

**Etape 1:** le $i^{ème}$ projet est extrait de l'ensemble de données d'entraînement c'est le nouveau projet a estimé, et les projets de repos sont traités comme les projets historiques dans le système ABE.

**Étape 2:** le système ABE trouve les K analogies les plus proches des projets historiques par la mesure de similarité, on utilise la distance Euclidienne comme fonction de similarité Sim(i,j) :

$$\boldsymbol{Sim(i,j) = 1 \Big/ \delta + \sqrt{\sum_{q=1}^{Q} Dist(Siq, Sjq)}} \quad (21)$$

$$\text{Dist}=\begin{cases} (siq - Sjq)^2 & \text{si Siq et Sjq sont numérique} \\ 1 & \text{si Siq et Sjq sont catégorique et Siq} = \text{Sjq} \\ 0 & \text{si Siq et Sjq sont catégorique et Siq} \neq \text{Sjq} \end{cases}$$

Où :

- ***i*** représente le projet à estimer,
- ***j*** dénote un projet dans l'historique
- ***Siq*** est la valeur de la qth caractéristique du projet i,
- ***Sjq*** est la valeur de la qth caractéristique du projet j,
- Q est le nombre total des caractéristiques pour chaque projet
- $\delta$=0.0001 est une petite constante pour éviter la situation où $\sqrt{\sum_{q=1}^{Q} Dist(Siq, Sjq)} = 0$

Dans ce mémoire, la distance euclidienne non pondéré est utilisée comme la fonction de similarité afin d'éliminer les impacts des poids.

Après avoir obtenu les K analogies, la solution récupérée (la valeur du coût) au i$^{ème}$ projet est généré. La moyenne non pondérée des couts des k projets analogies est la solution récupérée.

**Étape 3**: Après avoir obtenu la solution Récupérée, les entrées et la sortie d'entrainement sont prêts à former le modèle ANN.

Les entrées de l'ANN sont les différences entre les caractéristiques du projet j et les caractéristiques de ses analogies K. La sortie d'entrainement d'ANN est la différence entre Le cout réel du projet i et la solution récupérée de ses analogies K.

$$\boldsymbol{Ci - \sum_{k=1}^{K} \frac{Cx}{K} = \sum_{j=1}^{J} wjf \left( \sum_{k=i}^{K} \sum_{q=1}^{Q} vkqjf(xi) + \alpha j \right) + \beta + \varepsilon} \qquad (22)$$

**Étape 4**. Étant donné les entrées et la sortie alors l'algorithme de BP est exécuté pour mettre à jour les paramètres afin de minimiser l'erreur d'entrainement.

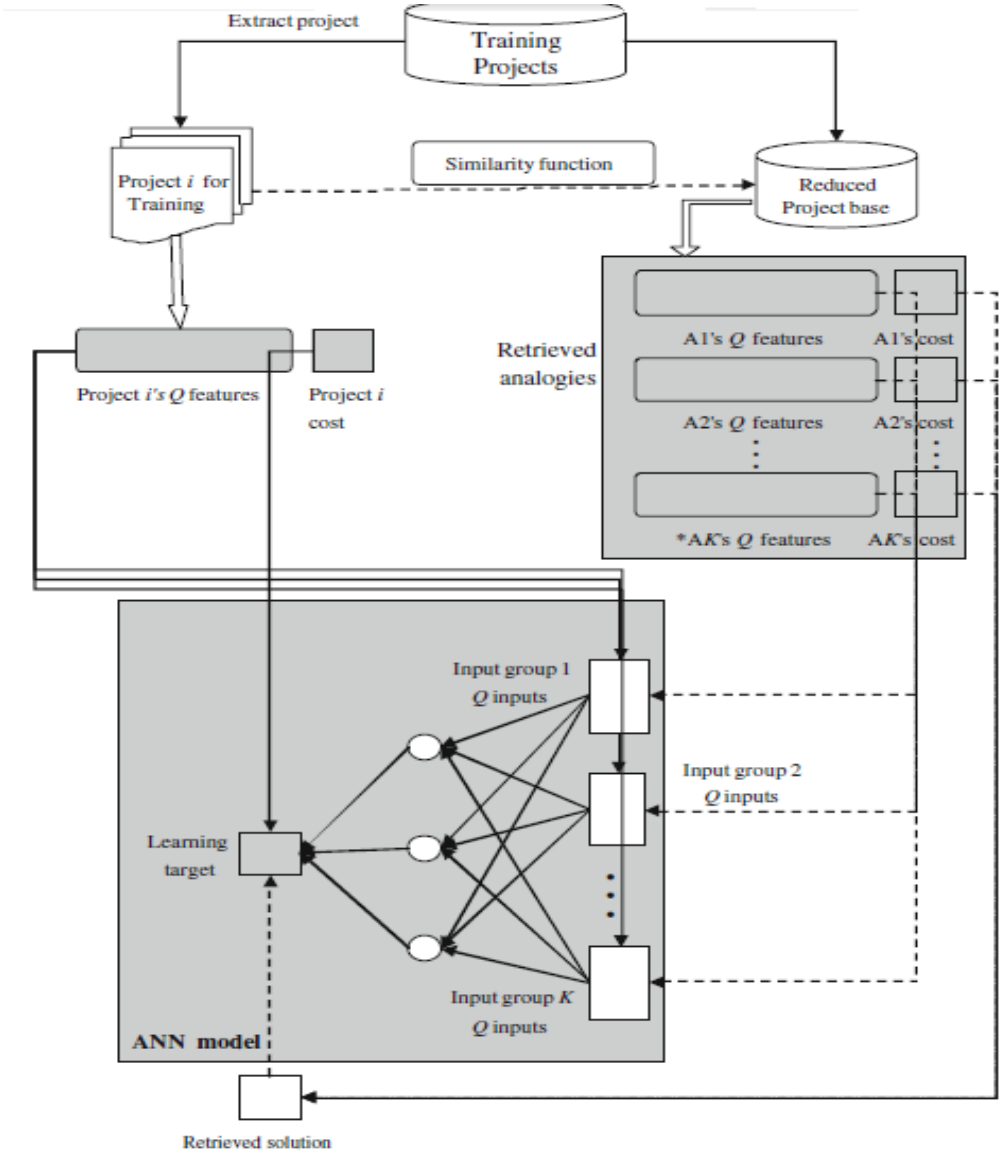


**Figure 8: Etape d'entraînement du système CBR avec K plus proches analogies.**

## Phase II- Prédiction

L'étape de prédiction est illustrée à la Figure 9 :

- Tout d'abord, un nouveau projet x est présenté au système ;
- Ensuite, un ensemble de K d'analogies sont extraites de l'ensemble des données d'entrainement par le calcul des similarités ;
- Après avoir obtenu les analogies de K, la solution récupérée est utilisée pour générer la prédiction non ajustée et les différences entre les caractéristiques du projet x et ses K analogies sont saisis dans le modèle ANN pour générer l'ajustement ;

- Enfin, la prédiction de l'ABE et l'ajustement de l'ANN sont additionnées.

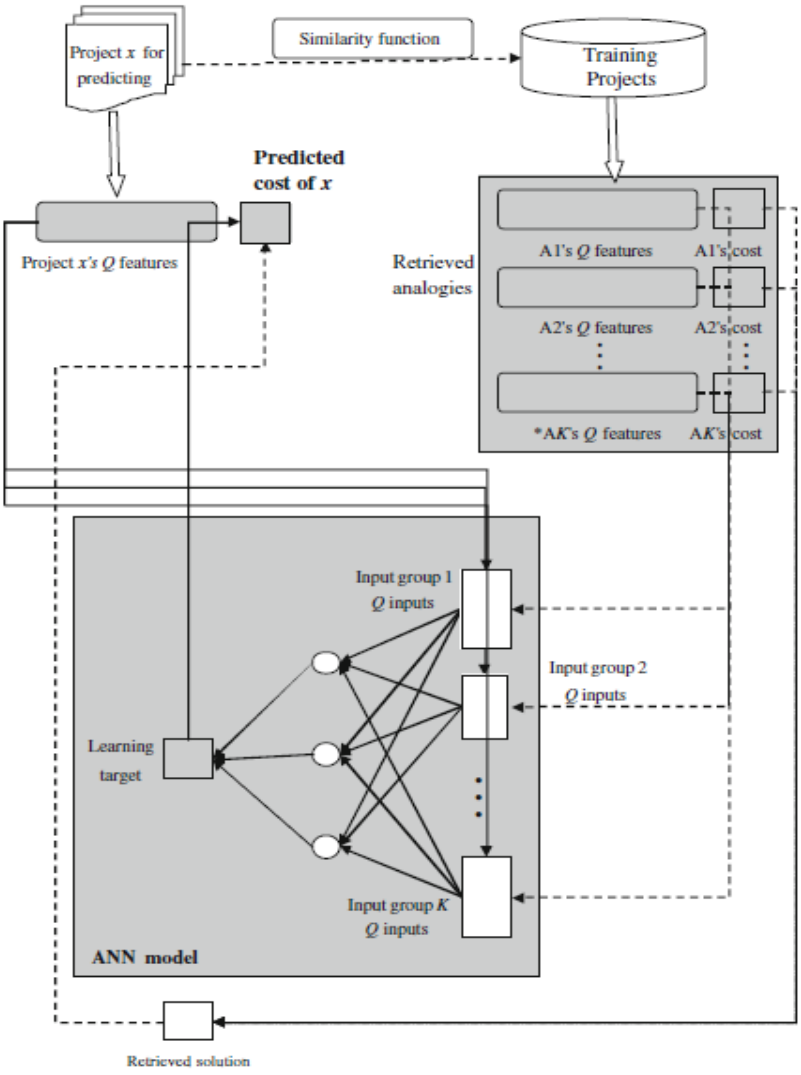


**Figure 9:Etape de prédiction d'ajustement ANN de CBR.**

# CHAPITRE 3:

# EXPÉRIMENTATION

# 1 Introduction

Ce mémoire est une étude d'évaluation et de la comparaison des variantes techniques d'adaptation utilisées dans la prédiction de l'effort de logiciel basé sur l'analogie.

Les techniques d'adaptation qui seront comparés dans cette étude sont des variantes de modèles d'ajustement linéaires et non linéaires. L'étude est une tentative de répondre aux questions suivantes:

- Q1: Quel est la meilleure technique d'adaptation linéaire ou non linéaire pour un ensemble de données spécifique?
- Q2: Quel est le meilleur nombre d'analogies que les techniques d'adaptation favorisent ou chercher?

Dans ce chapitre nous présentons la procédure d'expérimentation c.-à-d. les données, la technique de validation et les métriques de performance et les résultats obtenues.

# 2 Base de projets historiques

Nous utiliserons la base de données COCOMO'81 pour la validation de notre expérimentation. Cette base de données a été publiée dans[33]. Elle répond bien à nos critères de validation car la plupart des attributs, utilisés dans la description de logiciels, sont évalués par des valeurs linguistiques. , il contient 252 projet, chaque projet et décrit par 13 attributs. La taille mesurée en termes de KDSI (Kilo Delivered Source Instructions), et 12 autres facteurs représentant l'environnement de développement tels que l'expérience des programmeurs, la complexité logicielle et la volatilité de la machine virtuelle.

| Attribut | Désignation |
|---|---|
| SIZE | Software Size |
| DATA | Database Size |
| TIME | Execution Time Constraint |
| STOR | Main Storage Constraint |
| VIRTMIN, VIRT MAJ | Virtual Machine Volatility |
| TURN | Computer Turnaround |
| ACAP | Analyst Capability |
| AEXP | Applications Experience |
| PCAP | Programmer Capability |
| VEXP | Virtual Machine Experience |
| LEXP | Programming Language Experience |
| SCED | Required Development |

**Tableau 1: liste des attributs d'un projet logiciel.**

## 3 Procédure de validation

Les expériences sont effectuées à l'aide d'un procédé de validation croisée qui consiste à la répartition de l'ensemble de données en J partitions.

En fait, il y a au moins trois techniques de validation croisée :

- ✓ *testset validation*
- ✓ *k-fold cross-validation*
- ✓ *leave-one-out cross-validation*

Nous avons utilisé la seconde, on divise la partition originale en J partitions, puis on sélectionne un des J partitions comme ensemble de validation et les (J-1) autres échantillons constitueront l'ensemble d'apprentissage. Puis on répète l'opération en sélectionnant un autre échantillon de validation parmi les (J-1) partitions qui n'ont pas encore été utilisés pour la validation du modèle.

L'opération se répète ainsi J fois pour qu'en fin de compte chaque sous-échantillon ait été utilisé exactement une fois comme ensemble de validation.

La moyenne des k erreurs quadratiques moyennes est enfin calculée pour estimer l'erreur de prédiction.

# 4 Métriques de performances

Les indicateurs utilisés pour mesurer la précision d'un modèle sont, généralement, basés sur l'estimation de l'erreur entre les valeurs prédites et les valeurs réelles.

- La précision des estimations est évaluée en utilisant les indices basés sur la valeur absolue de l'erreur relative d'une estimation (Magnitude Relative Error), MRE, est définie, pour un cas par la formule :

$$\boldsymbol{MRE} = \left|\frac{\boldsymbol{Effort_{iactuel}} - \boldsymbol{Effort_{estimer}}}{\boldsymbol{Effort_{iactuel}}}\right| \qquad (23)$$

- Le MRE est calculée pour chaque projet dans l'ensemble de données. En plus, nous utilisons une mesure de prédiction notée Pred. Cette mesure est souvent utilisée dans la littérature. Il est défini par:

$$\boldsymbol{Pred(p)} = \frac{\boldsymbol{K}}{\boldsymbol{n}} \qquad (27)$$

Où, K est le nombre de projet a une MRE inférieure ou égale à p dans l'ensemble de test, n est le nombre total des projets utilisés dans le test. Dans la littérature, un modèle ayant une valeur de Pred(0,25) égale à 70% est dit acceptable [34].

# 5 Les résultats des expériences

Cette section présente les résultats et les comparaisons sur l'ensemble de données COCOMO'81. Le tableau 2 résume les chiffres des performances des différents modèles d'ajustement utilisant différents valeurs de K à savoir l'ajustement linéaire de l'effort, l'ajustement linéaire de la taille, l'ajustement linéaire de la productivité, l'ajustement linéaire de la similarité et l'ajustement non linéaire par ANN.

| | | Technique d'ajustemennt | | | | | | AVG |
|---|---|---|---|---|---|---|---|---|
| | | Effort | Size | Simularité | RTM | ANN | BK | |
| k=2 | MMRE | 54 | 130 | 75 | 23,4 | 50 | MMRE=674 | 66,48 |
| | pred | 71 | 76 | 7 | 87 | 36 | | 55,4 |
| k=3 | MMRE | 92 | 87 | 171 | 22,7 | 66 | | 87,74 |
| | pred | 54 | 81 | 47 | 86 | 33 | | 60,2 |
| k=4 | MMRE | 111 | 590 | 177 | 22,19 | 78 | | 195,638 |
| | pred | 36 | 3 | 25 | 87 | 26 | | 35,4 |
| k=5 | MMRE | 133 | 470 | 244 | 22,38 | 93 | | 192,476 |
| | pred | 25 | 8,4 | 19 | 86 | 24 | | 32,48 |
| k=6 | MMRE | 149 | 150 | 252 | 22,31 | 103 | | 135,262 |
| | pred | 20 | 8,6 | 20 | 86 | 20 | pred=3 | 30,92 |
| k=7 | MMRE | 167 | 130 | 315 | 22,96 | 114 | | 149,792 |
| | pred | 15 | 86 | 12 | 86 | 22 | | 44,2 |
| k=8 | MMRE | 182 | 122 | 314 | 23,61 | 121 | | 152,522 |
| | pred | 17 | 88 | 20 | 88 | 19 | | 46,4 |
| k=9 | MMRE | 189 | 99 | 283 | 23,21 | 127 | | 144,242 |
| | pred | 15 | 89 | 16 | 87 | 20 | | 45,4 |
| k=10 | MMRE | 205 | 89 | 376 | 23,38 | 140 | | 166,676 |
| | pred | 12 | 89 | 11 | 86 | 18 | | 43,2 |
| AVG(MMRE) | | 142,4444 | 207,4444 | 245,2222 | 22,90444 | 99,11111 | | |
| AVG(pred) | | 29,44444 | 58,77778 | 19,66667 | 86,55556 | 24,22222 | | |

**Tableau 2:résultats obtenues**

L'objectif d'étude expérimentale et de montrer l'impact de choix de K sur les techniques d'adaptation, ainsi la robustesse des techniques d'adaptation dans le cas de changement de nombre K.

Le résultat montre qu'avec un nombre K petit le taux de précision et grand, dans autre coté le MMRE diminuer par ce que dans le cas ou K est petit la probabilité d'erreur à diminuer. C'est le cas d'ajustement linéaire de l'effort et de la similarité.

Pour l'ajustement par la taille il n y a pas une forte corrélation entre le nombre d'analogie K est la valeur de MMRE.

Pour les technique d'adaptation le résultat montre que la méthode RTM et le plus robuste au changement de nombre d'analogie. Le RTM surpasse mieux que d'autres modèles avec des valeurs de MMRE minimales pour tous les différents paramètres (K). Une raison possible est due à la forte relation entre fonction de la taille et l'effort qui a abouti à une forte relation avec la productivité.

L'ANN est très performante par rapport à l'estimation de l'effort ou la taille cela est dû aux caractéristiques de la base donné puisque la plus part de ces attributs sont catégoriques.

# CONCLUSION

L'adaptation est une étape importante dans le modèle d'estimation du coût par raisonnement analogies.

L'objectif de ce mémoire est l'étude et la comparaison des différentes méthodes d'adaptation afin de déduire l'impact des nombre des projets analogie ainsi que les stratégies d'ajustement sur l'estimation du coût cherché.

Les variantes de techniques d'adaptation ont été évaluées à l'aide de la base de données COCOMO'81 avec des valeurs différentes de K .Cependant, après avoir examiné les différentes techniques d'ajustement les remarques suivantes peuvent être conclues:

- Avec un nombre K petit le taux de précision et grand, dans autre coté le MMRE diminuer ;

- Le RTM surpasse mieux que d'autres modèles avec des valeurs de MMRE minimales pour tous les différents paramètres K ;

Comme perspectives, nous espérant améliorer cette l'étude. En, effet nos activités futures est l'application des différents méthodes d'adaptation sur autres base de donnée afin d'analyser l'influence de ses attributs sur les performances des méthodes.

# RÉFÉRENCES


[1] M. Jorgensen and M. Shepperd, "A systematic review of software development cost estimation studies," *Softw. Eng. IEEE Trans.*, vol. 33, no. 1, pp. 33–53, 2007.

[2] R. T. Hughes, "Expert judgement as an estimating method," *Inf. Softw. Technol.*, vol. 38, no. 2, pp. 67–75, 1996.

[3] M. Jørgensen, "Practical guidelines for expert-judgment-based software effort estimation," *Software, IEEE*, vol. 22, no. 3, pp. 57–63, 2005.

[4] T. Yamaura and T. Kikuno, "A framework for top-down cost estimation of software development," in *Computer Software and Applications Conference, 1999. COMPSAC'99. Proceedings. The Twenty-Third Annual International*, 1999, pp. 322–323.

[5] Z. Abdelali, "Estimation des coûts de développement de logiciels par un réseau neuronal RBF flou," 2012.

[6] G. Wittig and G. Finnie, "Estimating software development effort with connectionist models," *Inf. Softw. Technol.*, vol. 39, no. 7, pp. 469–476, 1997.

[7] A. Idri, A. Zahi, M. ElKoutbi, and A. Abran, "Impacts des techniques de construction des ensembles flous sur la précision d'un modele d'estimation des coûts de logiciels par analogie floue," in *4th International Conference: Sciences of Electronic, Technologies of Information and Telecommunications*, 2007, p. 124.

[8] S. S. Vicinanza, T. Mukhopadhyay, and M. J. Prietula, "Software-effort estimation: an exploratory study of expert performance," *Inf. Syst. Res.*, vol. 2, no. 4, pp. 243–262, 1991.

[9] A. Idri, A. Abran, and T. M. Khoshgoftaar, "Estimating software project effort by analogy based on linguistic values," in *Software Metrics, 2002. Proceedings. Eighth IEEE Symposium on*, 2002, pp. 21–30.

[10] J. J. Dolado, "On the problem of the software cost function," *Inf. Softw. Technol.*, vol. 43, no. 1, pp. 61–72, 2001.

[11] M. Lefley and M. J. Shepperd, "Using genetic programming to improve software effort estimation based on general data sets," in *Genetic and Evolutionary Computation—GECCO 2003*, 2003, pp. 2477–2487.

[12] R. W. Selby and A. Porter, "Learning from examples: generation and evaluation of decision trees for software resource analysis," *Softw. Eng. IEEE Trans.*, vol. 14, no. 12, pp. 1743–1757, 1988.

[13] M. A. Ahmed and Z. Muzaffar, "Handling imprecision and uncertainty in software development effort prediction: A type-2 fuzzy logic based framework," *Inf. Softw. Technol.*, vol. 51, no. 3, pp. 640–654, 2009.

[14] A. L. I. IDRI, "UN MODÈLE INTELLIGENT D'ESTIMATION DES COÛTS DE DÉVELOPPEMENT DE LOGICIELS," 2003.

[15] M. Shepperd and C. Schofield, "Estimating software project effort using analogies," *Softw. Eng. IEEE Trans.*, vol. 23, no. 11, pp. 736–743, 1997.

[16] A. Aamodt and E. Plaza, "Case-based reasoning: Foundational issues, methodological variations, and system approaches," *AI Commun.*, vol. 7, no. 1, pp. 39–59, 1994.

[17] J. L. Kolodner, "Educational implications of analogy: A view from case-based reasoning.," *Am. Psychol.*, vol. 52, no. 1, p. 57, 1997.

[18] G. Kadoda, M. Cartwright, L. Chen, and M. Shepperd, "Experiences using case-based reasoning to predict software project effort," in *Proceedings of the EASE 2000 conference, Keele, UK*, 2000.

[19] L. Angelis and I. Stamelos, "A simulation tool for efficient analogy based cost estimation," *Empir. Softw. Eng.*, vol. 5, no. 1, pp. 35–68, 2000.

[20] R. R. Yager, "Prioritized aggregation operators," *Int. J. Approx. Reason.*, vol. 48, no. 1, pp. 263–274, 2008.

[21] M. Azzeh, Y. Elsheikh, and M. Alseid, "An Optimized Analogy-Based Project Effort Estimation."

[22] U. Lipowezky, "Selection of the optimal prototype subset for 1-NN classification," *Pattern Recognit. Lett.*, vol. 19, no. 10, pp. 907–918, 1998.

[23] E. Mendes, N. Mosley, and S. Counsell, "A replicated assessment of the use of adaptation rules to improve Web cost estimation," in *Empirical Software Engineering, 2003. ISESE 2003. Proceedings. 2003 International Symposium on*, 2003, pp. 100–109.

[24] C. Kirsopp, E. Mendes, R. Premraj, and M. Shepperd, "An empirical analysis of linear adaptation techniques for case-based prediction," in *Case-Based Reasoning Research and Development*, Springer, 2003, pp. 231–245.

[25] M. Azzeh, "Adjusted case-based software effort estimation using bees optimization algorithm," in *Knowlege-Based and Intelligent Information and Engineering Systems*, Springer, 2011, pp. 315–324.

[26] J. Li and G. Ruhe, "Decision support analysis for software effort estimation by analogy," in *Predictor Models in Software Engineering, 2007. PROMISE'07: ICSE Workshops 2007. International Workshop on*, 2007, p. 6.

[27] S. Ezghari, A. Zahi, and A. Idri, "A learning adaptation cases technique for Fuzzy Analogy-based software development effort estimation," in *Complex Systems (WCCS), 2014 Second World Conference on*, 2014, pp. 492–497.

[28] F. Walkerden and R. Jeffery, "An empirical study of analogy-based software effort estimation," *Empir. Softw. Eng.*, vol. 4, no. 2, pp. 135–158, 1999.

[29] E. Mendes, I. Watson, C. Triggs, N. Mosley, and S. Counsell, "A comparative study of cost estimation models for web hypermedia applications," *Empir. Softw. Eng.*, vol. 8, no. 2, pp. 163–196, 2003.

[30] N.-H. Chiu and S.-J. Huang, "The adjusted analogy-based software effort estimation based on similarity distances," *J. Syst. Softw.*, vol. 80, no. 4, pp. 628–640, 2007.

[31] Y.-F. Li, M. Xie, and T. N. Goh, "A study of the non-linear adjustment for analogy based software cost estimation," *Empir. Softw. Eng.*, vol. 14, no. 6, pp. 603–643, 2009.

[32] L. Angelis, I. Stamelos, and M. Morisio, "Building a software cost estimation model based on categorical data," in *Software Metrics Symposium, 2001. METRICS 2001. Proceedings. Seventh International*, 2001, pp. 4–15.

[33] A. L. Lederer and J. Prasad, "Causes of inaccurate software development cost estimates," *J. Syst. Softw.*, vol. 31, no. 2, pp. 125–134, 1995.

[34] S. D. Conte, H. E. Dunsmore, and V. Y. Shen, *Software engineering metrics and models*. Benjamin-Cummings Publishing Co., Inc., 1986.