Projet de Recherche

Doctorat Informatique Cognitive, DIC9410

Intitulé :

Un Modèle Intelligent d'Estimation des Coûts de Développement de Logiciels

Préparé par :

Ali Idri

Directeurs de recherche :

Professeur Alain Abran, Ecole de Technologie Supérieure
Professeur Serge Robert, Université du Québec À Montréal

Université du Québec À Montréal Année universitaire 2001-2002

Résumé

L'objectif principal de notre projet de recherche est le développement et la validation d'une nouvelle technique dites *intelligente* pour l'estimation des coûts de développement de logiciels. Cette technique a l'avantage, en plus de fournir des estimations proches des valeurs réelles des coûts requis, la prise en considération des imprécisions et de l'incertitude au niveau de la description des projets logiciels, la gestion de l'incertitude au niveau des estimations fournies par la technique, et l'apprentissage afin d'ajuster ses estimations dépendamment des besoins et des changements qui se manifestent dans l'environnement concerné. Pour atteindre ces objectifs, nous utiliserons un ensemble de techniques du domaine de l'intelligence artificielle. Ces techniques essayent d'imiter les caractéristiques du cerveau humain en reproduisant certaines de ses fonctions et ses structures

1. Problématiques de recherche

Dans cette section, nous présentons les techniques d'estimation des coûts de développement de logiciels, en particulier, l'estimation du coût par analogie. Ensuite, nous identifions trois limitations de l'approche d'estimation par analogie. Ces trois limitations constituent les problématiques que nous aborderons dans ce projet de recherche

1.1. Techniques d'estimation des coûts de logiciel

Bien que des chiffres précis soient difficiles à établir, il est de plus en plus admis que les coûts de développement de logiciels représentent une large part des coûts informatiques. La croissance de ces coûts ainsi que les difficultés que l'on rencontre pour les prévoir et les contrôler constituent encore, aujourd'hui, une préoccupation des chercheurs et des responsables informatiques. C'est donc pour rationaliser le processus d'estimation du coût que plusieurs modèles ont été proposés. Ces modèles peuvent être classifiés en deux catégories:

- les modèles algorithmiques, et
- les modèles non-algorithmiques.

Les modèles algorithmiques sont les plus cités dans la littérature. Ils sont développés en appliquant des méthodes statistiques (la régression simple ou multiple, l'analyse en composantes principales, l'analyse bayésienne, etc.) ou des méthodes de l'analyse numérique telle que l'interpolation polynomiale sur des données historiques. Des exemples de tels modèles sont COCOMO'81 [10], COCOMO II [11], PUTNAM-SLIM [78], et ceux se basant sur les points de fonction [2,5,61]. Les modèles non-algorithmiques, quant à eux, sont basés sur des approches de l'intelligence artificielle telles que les réseaux de neurones [33,85,90,94,97,104], le raisonnement par analogie [9,55,91,92,103] et les arbres de régression [75,76,89]. Chaque technique a des avantages et des inconvénients et aucune n'a pas pu prouver qu'elle performe mieux que les autres

dans toutes les situations [12,65,70,93]. En effet, la performance d'une technique dépend, en plus de l'approche adoptée par la technique, de plusieurs caractéristiques de la base des projets historiques telles que sa taille, le nombre d'attributs décrivant les projets logiciels et la présence des points extrêmes (outliers) [93].

Le raisonnement par analogie, ou le raisonnement à base de cas (Case-Based Reasoning –CBR) est une technique prometteuse qui s'adapte bien aux cas où on a peu de connaissances et d'informations sur le problème à résoudre et pour lequel une solution optimale est à priori inconnue. Il consiste à utiliser un historique de données afin de fournir une solution au problème étudié [1,3,56,59,86]. Il a été utilisé avec succès dans plusieurs sous disciplines du génie logiciel telles que la prédiction des erreurs dans un logiciel, la prédiction de la qualité logicielle, la conception et la réutilisation des composants logiciels [13,14,22,51,51,52,81]. En estimation du coût, le processus de raisonnement par analogie est composé de trois étapes : 1) Identification et description des projets logiciels, y compris le nouveau projet pour lequel on veut estimer le coût, par un nombre de variables significatives et indépendantes ; 2) Evaluation de la similarité entre le nouveau projet logiciel et tous les projets logiciels historiques; 3) Utilisation des valeurs des coûts réels des projets logiciels 'similaires' au nouveau projet pour en déduire une estimation de son coût futur. Cependant, jusqu'à présent, l'utilisation de la technique CBR en estimation du coût présentait plusieurs limitations. Premièrement, elle ne traitait pas convenablement le cas des projets logiciels décrits par des valeurs linguistiques telles que 'bas' et 'élevé', souvent définies d'une manière imprécise et incertain. Deuxièment, elle ne permettait de générer qu'une seule valeur estimée du coût sans la possibilité d'avoir d'autres valeurs possibles du coût. Troisièment, elle ne permettait pas un apprentissage automatique au niveau de ses sorties, c.à.d., des coûts estimés.

1.2. Problématiques de l'estimation des coûts par analogie

Le raisonnement par analogie trouve son originalité dans le fait que souvent les humains l'utilisent pour résoudre et manipuler des situations très complexes dans leur vie

quotidienne [24,25,26,32]. Cependant, le raisonnement par analogie chez les humains est souvent approximatif plutôt qu'exact. En effet, le cerveau humain est capable de manipuler des informations imprécises, vagues, incertaines et partielles. Aussi, le cerveau humain est capable d'apprendre et d'opérer dans un contexte où la gestion de l'incertitude est indispensable. Dans ce projet de recherche, la technique d'estimation du coût de logiciels, que nous proposerons, imite le raisonnement par analogie des humains ; elle doit donc tolérer les imprécisions, permettre l'apprentissage et gérer l'incertitude afin qu'elle puisse traiter les situations complexes rencontrées dans la prédiction du coût en fonction d'autres facteurs de logiciels.

Dans la littérature, la technique d'estimation du coût par analogie est souvent évaluée en terme d'un seul critère, à savoir, le degré d'exactitude des estimations qu'elle fournit. Cependant, d'autres critères d'évaluation, relevant de la nature de la problématique d'estimation du coût, sont indispensables pour une large acceptation de la technique d'estimation par analogie au sein de la communauté. Ainsi, nous évaluons la performance de la technique d'estimation par analogie, que nous proposerons, au moyen de quatre critères: (0) l'exactitude de ses estimations, (1) la technique permet-elle la tolérance des imprécisions et de l'incertain au niveau de la description des projets logiciels? (2) La technique permet-elle la gestion de l'incertain au niveau de ses estimations? (3) La technique permet-elle l'apprentissage ? Présentement, aucune technique d'estimation du coût, y compris l'estimation par analogie, ne satisfait les trois derniers critères bien que, en estimation du coût, l'imprécision dans la description des projets logiciels, l'incertitude dans les estimations et l'apprentissage sont souvent souhaités.

1.3. Problématique des valeurs linguistiques et la tolérance des imprécisions dans l'estimation par analogie

En génie logiciel, les projets logiciels sont souvent décrits par des variables linguistiques plutôt que des variables numériques. Ceci est dû au fait que :

1- Le génie logiciel est une science encore ‘jeune’ qui concerne un produit , le logiciel, dont les caractéristiques et les attributs différent beaucoup de ceux des autres produits dits ‘classiques’ tels que les ‘voitures’, les ‘télévisions’ et les ‘caméras’. Par exemple, des attributs du logiciel tels que la complexité, la taille, et la fiabilité ne sont pas encore clairement définis, et par conséquent on retrouve dans la littérature une multitude de définitions qui dépendent des besoins de l’environnement impliqué.

2- Les humains sont directement impliqués dans le processus de mesurage de presque tous les attributs d’un logiciel. Ainsi, la plupart des attributs d’un logiciel sont mesurés sur une échelle au mieux ordinale composée de valeurs linguistiques telles que ‘très bas’, ‘bas’ et ‘excellent’. Par exemple, on retrouve dans le modèle COCOMO’81, 15 par 17 facteurs qui sont mesurés sur une échelle ordinale composée de six qualifications : ‘très faible’, ‘faible’, ‘moyen’, ‘élevé’, ‘très élevé’ et ‘extra-élevé’ [10]. De même, dans le COCOMO II, on trouve 22 parmi 24 facteurs qui sont de type ordinale [11,15].

La problématique d’utilisation des valeurs linguistiques dans la description des projets logiciels nécessite donc la prise en considération des imprécisions dans le processus d’estimation du coût de logiciels. Toutes les techniques d’estimation du coût, y compris l’estimation par analogie, utilisent encore une représentation classique, basée sur les intervalles ou les nombres, pour traiter ces valeurs linguistiques. Cette représentation classique ne permet cependant pas de gérer les imprécisions car elle ne ressemble pas, à priori, à celle utilisée par les humains pour interpréter les valeurs linguistiques. Par exemple, dans le modèle COCOMO, le facteur DATA, expliquant l’influence de la taille de la base de données sur le coût du logiciel, est mesuré par le ratio :

$$\frac{D}{P} ? \frac{Database\ \ size\ in\ bytes\ or\ characters}{\mathrm{Pr}\,ogram\ \ size\ \ in\ DSI}$$

Ensuite, une valeur linguistique est attribuée à DATA en fonction du ratio D/P, tel qu’illustré au Tableau 1.

	Low	Nominal	High	Very High
D/P	D/P < 10	10? D/P < 100	100 ?D/P < 1000	D/P ? 1000

Tableau 1 : Valeurs linguistiques du facteur DATA

Par conséquent chaque projet logiciel appartient à une et une seule classe. Par exemple, si D/P est égal à 9,99 alors le DATA du projet logiciel sera déclaré 'low'; si D/P est égal à 10,01 alors il sera 'nominal'. Ceci pose de sérieux problèmes pour les projets ayant des ratios D/P au voisinage des limites (10, 100 et 1000). Notre projet de recherche développera et utilisera donc des nouvelles approches, à priori similaires à celles utilisées par les humains, pour la représentation et le traitement des valeurs linguistiques

1.4. Problématique de la gestion des incertitudes dans l'estimation par analogie

Une estimation est une évaluation d'une situation future d'une entité particulière. Ainsi, les modèles d'estimation du coût doivent gérer l'incertitude au niveau des estimations qu'ils fournissent [53]. Cependant, les modèles existant d'estimation du coût, y compris l'estimation par analogie, génèrent seulement une seule valeur numérique du coût. Par conséquent, si cette valeur estimé est largement différente de la valeur réelle, les gestionnaires pourront commettre des erreurs fatales au niveau de la gestion du projet logiciel. Il est donc préférable qu'un modèle d'estimation génère un ensemble de valeurs possibles, plutôt qu'une seule valeur, du coût du projet logiciel.

La gestion de l'incertitude dans l'estimation par analogie implique que l'estimation du coût est fournie sous forme d'un ensemble de valeurs avec une possibilité de distribution. Cette fonction de distribution indique les degrés de certitude associés aux valeurs possibles du coût du projet logiciel. La nécessité de tenir compte de l'incertitude dans la technique d'estimation par analogie découle du fait que le raisonnement par

analogie est basé sur l'hypothèse suivante : *'similar software projects have similar costs'*. Il y a deux sources d'incertitude dans cette hypothèse:

1- La conséquence de cette hypothèse est imprécise.
2- L'hypothèse peut être non deterministique, c.à.d., on peut avoir deux projets logiciels similaires mais leurs coûts sont totalement différents.

Notre projet de recherche s'intéresse donc au développement d'une technique pour gérer l'incertitude dans le processus d'estimation par analogie.

1.5. Problématique de la nécessité de l'apprentissage dans l'estimation par analogie

Un modèle d'estimation intelligent doit être capable d'adapter, d'apprendre et de modifier ses estimations dépendamment des changements survenus dans l'environnement concerné. En industrie de logiciels, les technologies, ainsi que les besoins, sont très volatiles:

✍ On ne développe plus le même type d'application
✍ La complexité des applications est de plus en plus élevée
✍ Le personnel est de plus en plus expérimenté
✍ On utilise de nouvelles méthodes de développement et/ou de maintenance
✍ Etc.

Par conséquent, l'apprentissage permettra à notre modèle d'estimation de se reconfigurer afin d'ajuster ses estimations. Autrement, on devrait intervenir périodiquement afin de modifier les différents paramètres du modèle pour qu'il puisse répondre à nos nouveaux besoins. Il y a deux autres avantages de la qualité d'apprentissage en estimation du coût :

1- L'apprentissage est un moyen efficace pour améliorer la précision du modèle au niveau des estimations du coût qu'il fournit

2- L'apprentissage peut être utilisé pour calibrer et adapter un modèle d'estimation du coût, développé dans un environnement, à un autre environnement différent.

La technique d'estimation du coût par analogie n'intègre pas, pour le moment, des mécanismes d'apprentissage dans son processus. Notre projet de recherche s'intéresse donc au développement d'une approche d'apprentissage appropriée à l'estimation du coût par analogie.

2. Méthodologie adoptée pour le développement de notre modèle

Dans cette section, nous décrivons la méthodologie, que nous adoptons, pour le développement de notre modèle intelligent d'estimation du coût. Cette méthodologie est composée de trois étapes principales.

2.1. Première étape : *C'est quoi un modèle intelligent d'estimation du coût ?*

Notre projet de recherche vise à développer un modèle *intelligent* d'estimation du coût de logiciels. L'intelligence est une caractéristique des humains et elle est considérée comme la principale qui nous distingue des autres existants. Les philosophes ont essayé, il y a longtemps, de comprendre et répondre à deux questions relatives au cerveau humain (organe responsable de notre intelligence) :

1- Comment le cerveau humain fonctionne t-il ?
2- Comment peut-on faire des cerveaux aux non-humains ?

Le concept 'intelligence' n'est pas communément bien défini. Comme autres concepts tels que la mort, l'existence et l'unicité, il est de nature philosophique. Il n'est pas donc surprenant de trouver dans la littérature plusieurs définitions de ce concept. Voici deux définitions données dans le dictionnaire (Essential English Dictionary, Collins, London, 1990) :

"Someone's intelligence is their ability to understand and learn things"

"Intelligence is the ability to think and understand instead of doing things by instinct or automatically"

En général, nous pouvons dire que le concept 'intelligence' regroupe plusieurs caractéristiques parmi lesquelles on peut citer l'apprentissage, la créativité, la facilité de comprendre, etc. Ces caractéristiques ne sont pas présentes avec les mêmes degrés chez les humains. On retrouve, par exemple, que certains sont considérés très 'intelligents' en Mathématique contrairement à ce qu'ils sont en Géographie ; d'autres sont très 'intelligents' en investissement d'argents à l'opposé de certains qui sont très 'intelligents' en gaspillage d'argents. Cependant, tous les humains sont capables d'apprendre et de comprendre. La différence est que cette compétence dépend des domaines.

Dans notre projet de recherche, nous identifions, au moins, quatre critères qu'un modèle d'estimation du coût doit satisfaire pour être considéré comme un modèle intelligent: (0) il doit fournir des estimations proches des valeurs réelles du coût, (1) il doit tolérer les imprécisions au niveau de la description des projets logiciels, (2) il doit gérer l'incertitude au niveau des estimations fournies, et (3) il doit être capable d'apprendre.

2.2. Deuxième étape : *Quelle(s) est (sont) l'(es) approche(s) de modélisation de l'IA que nous devrons utiliser pour modéliser la relation entre le coût et les facteurs l'affectant?*

Du fait que le raisonnement est couramment présent dans notre vie quotidienne et constitue la base de la plupart de nos activités mentales, l'intelligence artificielle lui a accordé, tout comme la représentation de la connaissance, un intérêt particulier. D'ailleurs, il y a une étroite imbrication entre le raisonnement et la connaissance : aucune mécanisme ou modèle de raisonnement ne peut faire abstraction des connaissance sur lesquelles il opère. Il y a une grande diversité de modes de raisonnement en IA à l'image

de la richesse des mécanismes mentaux des humains [58]. Cependant, on peut les classifier en trois types de raisonnement: (a) le raisonnement formel fondé sur la manipulation syntaxique des symboles à l'aide de règles (symbolisme), (b) le raisonnement par généralisation lié à l'apprentissage par induction (connexionnisme) et le raisonnement par analogie.

L'analogie est une forme de raisonnement très courante et efficace chez les humains. Elle guide implicitement des problèmes cognitifs importants et encore mal connus, par exemple l'accès associatif à des mémoires ou la pertinence de ressemblances détectées entre des objets ou des situations. Le paradigme de base du raisonnement par analogie consiste en la mise en correspondante de deux situations, parfois appartenant à deux domaines différents, afin de fournir un comportement devant une situation nouvelle en fonction d'informations mémorisées sur des situations déjà rencontrées [25,26,32]. Holyoak et Thagard (1995) reportent que, avant d'être utilisé dans le développement des théories scientifiques, le raisonnement par analogie a été utilisé dans la plupart des religions pour expliquer certains concepts de base tels la création, la vie, la mort, et la relation entre Dieu et les humains. Souvent, ces concepts ne sont pas faciles à expliquer directement. Par contre, ils deviennent compréhensibles quand on les explique en faisant une analogie avec d'autres concepts facilement explicables. Par exemple, l'analogie suivante 'God is related to people as a father is to his children: as progenitor, protector, and disciplinarian' met en évidence la relation entre 'God' et 'people' avec la relation entre 'father' et 'children' (R[God, people]=R[father, children]).

Dans le développement des théories scientifiques, Holyoak et Taghard reportent que l'utilisation de l'analogie dans les sciences date, au moins, de vingt siècles; la première utilisation reconnue de l'analogie était celle de Vitrivius qui avait décrit la propagation du sons dans la construction des théâtres Romains par la forme des vagues [32]. Bien que Holyoak et Thagard reconnaissent que le raisonnement par analogie ne soit pas le seul mécanisme responsable de la créativité chez les humains, ils affirment, par contraste, que

c'est le plus important. En effet, l'analogie peut être un moyen efficace pour le développement de nouvelles idées; d'ailleurs, l'approche symboliste en sciences cognitives était née de l'analogie entre le traitement de l'information dans l'ordinateur avec celui dans les cerveaux humains.

Le raisonnement par analogie en IA n'a pas eu le même intérêt que celui accordé au raisonnement formel (logique) ou au raisonnement par généralisation (apprentissage). En effet, il n'a pas été formalisé rigoureusement et par conséquent il est resté marginal. Ce n'est que récemment qu'une version simplifiée, du raisonnement par analogie, a été mise au point sous forme d'un raisonnement à base de cas (case-based Reasoning -CBR). La technique CBR se base sur l'hypothèse que la résolution d'un problème consiste en l'accès à des informations mémorisées lors d'expériences précédentes en vue d'une exploitation ultérieure. Elle retrouve ses origines dans les travaux de Schank (1982) sur la mémoire dynamique et ceux de Gentner (1983) sur le raisonnement par analogie [24,86]. Depuis, la technique CBR a été développée simultanément par plusieurs groupes de recherche en Amérique et en Europe [1,3,56,59]. En général, le processus de la technique CBR est composé de quatre étapes (Aamodt et Plaza 1994):

1- Retrouver les cas similaires au cas étudié
2- Utiliser les cas similaires retrouvés pour résoudre le problème
3- Réviser la solution proposée.
4- Retenir les éléments de cette expérience qui serviront à la résolution des problèmes futurs

Dans notre projet de recherche, nous nous sommes basés sur la technique CBR pour modéliser la relation entre le coût et les facteurs l'affectant. Ainsi, notre modèle d'estimation adopte le raisonnement à base de cas pour estimer le coût d'un projet logiciel à partir des attributs décrivant ce projet logiciel. En effet, la technique CBR présente, en plus de sa plausibilité psychologique car elle est une forme simplifiée du

raisonnement par analogie, plusieurs avantages sur les autres techniques couramment utilisées en estimation du coût :

1- Elle est facile à expliquer contrairement au cas des réseaux de neurones.
2- Elle peut modéliser les relations complexes existantes entre le coût et les facteurs affectant le coût contrairement au cas des méthodes statistiques.
3- Elle peut être utilisée dans le cas où on a peu de connaissances sur le problème étudié contrairement au cas des systèmes à base de règles.
4- Elle utilise la connaissance dans sa forme 'brute' contrairement au cas des systèmes à base de règles où la connaissance doit être encodée dans des règles (ce processus d'encodage n'est pas souvent facile).
5- Elle peut être utilisée dans le cas où une solution optimale du problème n'est pas évidente
6- Elle généralise plusieurs types de modélisation en IA :
    a. Cas d'un arbre de régression
    b. Cas de certains réseaux de neurones (Perceptron simple, Réseau de Kohonen, Radial Basis Function Network, Réseau de Hopfield)

Cependant, le raisonnement à base de cas, tel qu'il a été défini dans la littérature, utilise la logique classique (une proposition ne peut être que 'vraie' ou 'fausse'). Par conséquent, il ne peut traiter les situations où les cas sont définis par des caractéristiques floues et incertaines. Pour remédier à cet inconvénient, quelques chercheurs ont développé des versions floues de la technique CBR classique [18,71]. En estimation du coût, en plus que les projets logiciels sont souvent décrits par des valeurs imprécises et incertaines, le raisonnement à base de cas, que nous proposons pour notre modèle d'estimation, doit gérer l'incertitude au niveau des estimations fournies et permettre l'apprentissage. Nous intégrons donc plusieurs techniques dans le processus classique du CBR afin de satisfaire les trois exigences relevant du domaine de l'estimation du coût. La principale technique utilisée est la Logique Floue (LF) pour le traitement des imprécisions et des incertitudes dans tout le cycle du CBR . En effet, aucune technique de

l'IA ne permet la tolérance des imprécisions au niveau de la description des projets logiciels. Les attributs d'un projet logiciel sont souvent mesurés par des valeurs linguistiques, telles que 'bas', 'très bas' et 'élevé', plutôt que par des valeurs numériques [21,123]. Les techniques de l'intelligence artificielle utilisent la logique classique pour représenter et manipuler ces valeurs linguistiques. Selon Zadeh [108,118], cette représentation ne ressemble pas, à priori, à celle adoptée par les humains quand ils traitent des valeurs linguistiques. Dans la suite, nous explicitons les principaux thèmes de débat entre la logique floue et l'intelligence artificielle classique. Ceci nous permettra de justifier notre choix d'utiliser la logique floue, pour le traitement des imprécisions et des incertitudes, dans la technique d'estimation par analogie.

2.3. Troisième étape : *Quelles sont les techniques que l'estimation par analogie doit utiliser pour satisfaire les quatre critères d'intelligence cités dans l'étape 1?*

Bien que l'expression 'Intelligence Artificielle' ne soit apparue officiellement que lors de la conférence du Dartmouth College en 1956, ses origines datent de 1900 quand David Hilbert présenta une liste de trente-trois problèmes théoriques qui, d'après lui, attireraient les mathématiciens du siècle à venir. Le trente-troisième de ces problèmes était celui de la décidabilité, c.à.d, l'existence d'une procédure générale permettant d'évaluer une proposition logique donnée à l'une des deux valeurs de vérité : 'vraie' ou 'fausse'. La première preuve de l'impossibilité de construire une telle procédure est due à Kurt Gödel en 1931 qui démontra que, dans tout système axiomatique, il existe des propositions pour lesquelles le système est incapable d'assigner la valeur 'vraie' ou 'fausse'. La deuxième réponse, due à Alain Turing en 1936, concerne la parenté entre calculabilité et décidabilité. Turing utilise la calculabilité pour reproduire le résultat de Gödel concernant l'indécidabilité de certaines propositions logiques. En effet, Turing démontra l'existence de fonctions non calculables et par conséquent, leurs résultats, qui sont les propositions logiques, sont indécidables [100].

En 1950, Turing esquisse une théorie de l'esprit fondé sur les concepts de Test de Turing et de la Machine Universelle de Turing [101]. Elle inspire aux pionniers de l'intelligence artificielle deux programmes :

1- L'intelligence artificielle 'faible' localisée dans certains groupes de MIT (Minsky et al.) dont l'objectif est le développement de programmes informatiques simulant des comportements humains admis et reconnus par tous comme étant 'intelligents'. Cette position lui évite de se lancer dans une recherche difficile et hasardeuse sur les processus cognitifs en jeu.

2- L'intelligence artificielle 'forte' localisée dans Carnegie-Mellon (Newell et Simon) dont l'objectif est de modéliser l'esprit humain en construisant un programme qui, conjointement avec l'ordinateur, reproduira toute la cognition humaine. C'est ici qu'on retrouve les fameuses thèses comme 'cognition = représentation + computation', 'esprit =programme', et 'cerveau=ordinateur'.

.

Issue des travaux de Newell et Simon, l'approche symboliste assume, comme hypothèse de travail, l'analogie de fonctionnement entre le cerveau humain et l'ordinateur [67-69,79,80]. Elle postule qu'un système intelligent est un système physique de traitement de symboles, lequel opère à partir de règles. Cependant, la nature symbolique et sérielle de l'approche symboliste ne peut traiter des problèmes complexes vu l'explosion combinatoire qu'elle peut manifester. Pour pallier à ce type de problèmes, l'approche connexionniste propose une méthode massivement parallèle du traitement de l'information, basée sur les opérations d'un réseau neuronal composé de plusieurs unités simples et interconnectées [83,84,95]. Depuis, les deux approches (symboliste et connexionniste) s'affrontent et débattent des questions philosophiques et scientifiques. A ce jour, les questions provenant du 'dark ages' de l'intelligence artificielle telles que 'les machines peuvent-elles penser ?' et 'les machines peuvent-elle reproduire toute l'intelligence humaine ?' sont encore sans réponses. Aussi, puisque l'intelligence artificielle est aussi une discipline d'ingénierie, l'IA n'a pas encore réussi à résoudre

plusieurs problèmes relevant de notre vie quotidienne tels que le stationnement d'un véhicule dans un parking et le déplacement d'un robot comme les humains.

Introduites en 1965 par Zadeh, la logique floue est restée marginale pendant longtemps, n'intéressant qu'un cercle restreint d'adeptes parmi les enseignants et les chercheurs. Cette traversée du désert que connaissent la plupart des nouvelles disciplines (du fait que très souvent elles basculent l'ordre établi) a duré au moins une dizaine d'année. Mais la logique floue va connaître au terme de cette période de gestation, un essor prodigieux grâce à l'évolution de l'intelligence artificielle et aux applications pratiques dans le domaine de la commande et des contrôles des automatismes des processus.

La logique floue propose de modes de raisonnement approximatifs plutôt que exacts; c'est principalement le mode de raisonnement utilisé par les humains dans la plupart des cas. Par conséquent, il semblait au départ que cette nouvelle science (logique floue) serait la bienvenue au sein de la communauté de l'intelligence artificielle. La réalité était que la plupart l'ont reçu avec un scepticisme et parfois même avec une hostilité. Selon Zadeh, il y a deux raisons majeures derrière cette réticence [118]:

- Les fondateurs de l'intelligence artificielle ne manipulaient avant que deux valeurs de vérité : 'vraie' et fausse'.
- Une mauvaise compréhension, de la part de la communauté de l'intelligence artificielle, de ce que la logique floue et ce qu'elle peut offrir comme techniques pour résoudre des problèmes très complexes.

Les deux approches classiques l'intelligence artificielle (symbolisme et connexionnisme) se sont initialement basées sur une logique ne manipulant que les deux valeurs de vérité: 'vraie' et 'fausse'. C'est le principe de "excluded middle" de la logique d'Aristote: une proposition ne peut être que vraie ou fausse. La prédominance de ces deux approche en IA n'a pas permis l'application de l'intelligence artificielle à plusieurs

domaines où la connaissance était imprécise, vague et incertaine, autrement dit *partielle*. Cependant, la *partialité* plutôt que la *totalité* est omniprésente dans notre quotidien. Généralement, notre connaissance est *partielle*, notre croyance est *partielle*, notre compréhension est *partielle*, et la vérité elle aussi est *partielle*. Ce dernier, le concept de partialité dans la vérité, était à l'origine de toute la logique floue. Le débat sur les valeurs de vérité possibles dépasse le cadre de l'intelligence artificielle et on le retrouve aussi dans les religions, les philosophies, la politique et autres domaines où l'humain joue un rôle primordial. Il est peut être impossible de le dater ou de le relier à une référence bien précise.

Bien que les théories scientifiques classiques, y compris l'intelligence artificielle, qui se basent sur la précision et la non- ambiguïté (deux valeurs de vérité 'vraie' et 'fausse'), ont pu achever des succès considérables comme l'arrivée des humains à la lune, la construction d'ordinateurs très puissants et l'invention de télescopes explorant des régions lointaines dans notre univers, elles n'ont pas encore réussi à construire des robots qui se déplacent de la même façon que les animaux ou les humains, à automatiser la conduite de voitures dans le trafic routier et à développer des outils de traduction d'un langage vers un autre sans l'intervention des humains. Selon Zadeh, ces limites des théories classiques viennent du fait qu'elles exigent et manipulent seulement des informations précises.

En intelligence artificielle, du fait qu'on ne manipule que deux valeurs de vérité, les systèmes dits intelligents étaient tels que ceux de compréhension d'un texte, ceux de vision, ceux de preuve de théorème, et ceux de jeu. En général, les domaines où la connaissance était précise et non-ambigüe. Aujourd'hui, un des grands défis pour l'IA se trouve dans les domaines où les connaissances sont souvent imprécises et incertaines. Dans nos sociétés modernes, on a besoin, de plus en plus, à des produits intelligents facilitant notre vie quotidienne: machines à laver, caméras, magnétoscope, téléviseurs, etc. Dans ces produits, on exige de plus en plus un haut niveau de ce qu'on appelle le

MIQ (Machine Intelligent Quotient). Un exemple typique serait les caméras modernes dotées d'un mécanisme intelligent permettant d'ajuster et d'adapter la photo dépendamment de votre position; par conséquent, le rôle du photographe serait facultatif contrairement au cas des caméras classiques.

L'idée de Zadeh consiste à utiliser le modèle de l'esprit humain pour formaliser le processus que les nouveaux produits industriels doivent adopter afin d'augmenter leur MIQ. En effet, l'être humain dispose d'une très forte capacité à appréhender la complexité et à manier des notations vagues et imprécises. Un exemple typique est le stationnement d'un véhicule dans un parking. Les humains sont capables de stationner un véhicule facilement car souvent la position finale ainsi que l'orientation du véhicule sont spécifiées d'une manière imprécise plutôt qu'exacte; sinon on pourrait imaginer la difficulté d'un stationnement si on exige un degré de précision très élevé. Cette compétence est due à l'habilité des humains à manipuler des perceptions en plus des mesures. La différence entre une perception et une mesure est qu'en général une mesure est 'crisp' (précise) tandis que une perception est 'fuzzy' (floue). Nos perceptions sont souvent décrites au moyen de ce puissant outil qu'est le langage naturel. En effet, le langage naturel supporte l'imprécision et l'incertain. Dans ses travaux récents, Zadeh a initié le développement d'une nouvelle méthodologie de computation par des mots (Computing with words - CW) dont l'objectif est d'imiter le langage naturel utilisé par les humains. Le computing with words fournit une methodologie pour ce que Zadeh appelle le Computational Theory of Perceptions (CTP). En effet, dans le CTP, les perceptions sont exprimées par des propositions du langage naturel. Ainsi, ces perceptions sont traitées par des méthodes basées sur le computing with words pour répondre à une question exprimée elle aussi en langage naturel. La Figure 1 illustre le processus du CW.

Il faut noter qu'en intelligence artificielle classique, on s'est intéressé aussi, comme dans le CW de Zadeh, à la manipulation des propositions décrites en langage naturel (logique des propositions, logique des prédicats, traitement du langage naturel, etc.). La

différence entre l'approche de Zadeh et les techniques traditionnelles de l'IA est que le CW est basé sur la logique floue plutôt que sur la logique classique. La logique floue a donc le mérite d'être la seule discipline qui fournit les outils de base pour évaluer et raisonner selon des approches approximatives plutôt que exactes. Zadeh illustre ceci par deux exemples [121,122] :

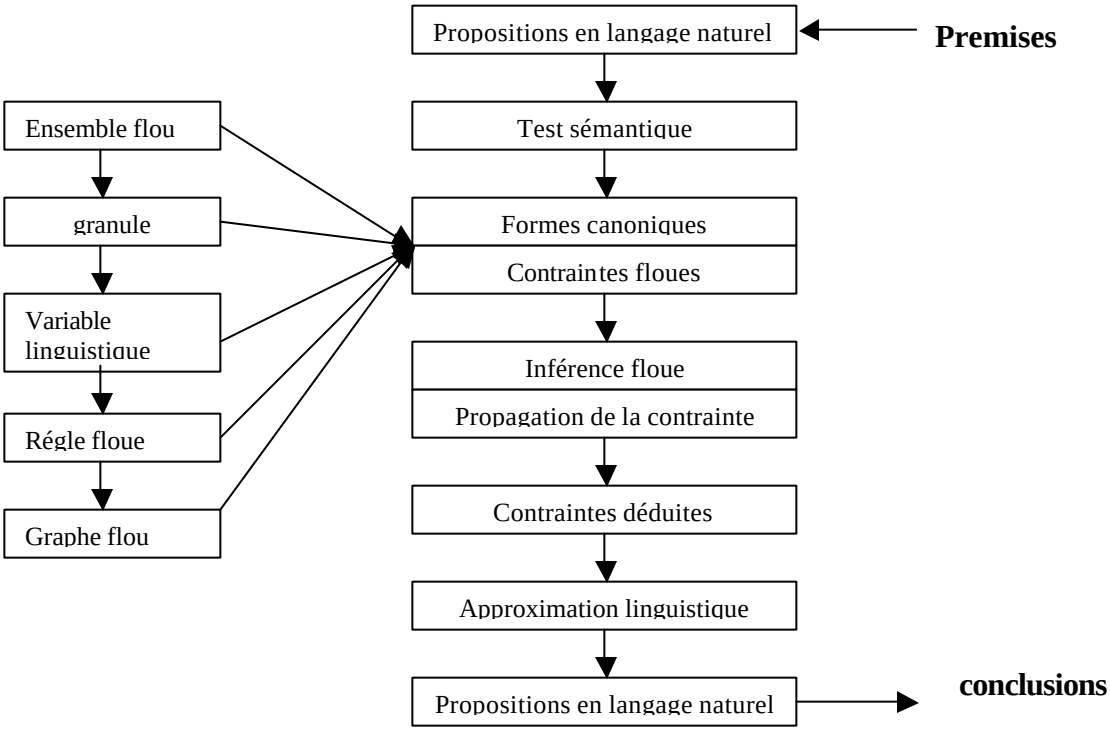

Figure 1 : La structure conceptuelle du CW [120,121]

1- Considérons les deux propositions suivantes: 'most young men are healthy' et 'Robert is a young men'. Que peut-on dire de la santé de Robert? Ni la logique de prédicats ni la théorie de probabilité ne sont capables de répondre à cette question.

2- Considérons un sac contenant des balles de tailles différentes : la plupart des balles sont de grande taille et peu d'entre elles sont de petite taille. Quelle est la probabilité pour tirer du sac au hasard une balle de taille ni grande ni petite? Dans

ce cas, pour répondre à cette question, nous devons faire appel à la computation par des perceptions pour définir la signification de 'la plupart des balles sont de grande taille', 'peu d'entre elles sont de petite taille' et 'ni grande ni petite'.

Nous pouvons donc conclure que sans l'utilisation de la logique floue, qui fournit des outils puissants tels que les ensembles flous, le soft computing, la 'computational theory of perceptions' et le 'computing with words', beaucoup de problèmes du monde réel, que l'intelligence artificielle est sensé de résoudre, resteront sans solutions satisfaisantes.

3. Contributions de notre projet de recherche

En abordant les problématiques de recherches cités dans la Section 1, notre projet de recherche visera à apporter des contributions méthodologiques à plusieurs domaines de recherche.

Estimation du coût de logiciels : Récemment, quelques chercheurs ont commencé à s'intéresser d'avantage à l'estimation du coût en utilisant la technique CBR. Les recherches menées ne traitent pas convenablement le cas des valeurs linguistiques bien qu'en génie logiciel, les projets sont souvent décrits par ce type de valeurs. Notre projet propose donc l'utilisation de la logique floue pour la représentation et le traitement de ces valeurs linguistiques. Aussi, notre projet traite la gestion de l'incertitude au niveau des estimations fournies et l'apprentissage à l'aide de la théorie des possibilités et les réseaux de neurones flous. Notre projet est donc le premier, en son genre, à traiter ces trois caractéristiques importantes qu'un modèle d'estimation devraient satisfaire. Il aboutira à un modèle intelligent, c.à.d., satisfaisant les quatre critères d'intelligence cités auparavant. Par conséquent notre modèle devrait être facilement interprétable, flexible et facilement adaptable aux besoins des utilisateurs.

Raisonnement à base de cas : La technique CBR a été appliquée dans plusieurs domaines de recherche, notamment en estimation du coût. Par conséquent, chaque domaine a contribué, à l'évolution et à la popularité de la technique CBR, par le développement de nouveaux algorithmes qui peuvent être utilisés dans les différentes étapes du processus CBR. La contribution de notre projet est le développement de nouvelles mesures de similarités qui traitent le cas des projets logiciels décrits par des valeurs linguistiques. Ces mesures peuvent être utilisées dans d'autres domaines de recherche. Ainsi, nous proposons une nouvelle méthode à utiliser au niveau de l'étape d'adaptation des projets similaires afin de générer une estimation du coût.

Logique floue et intelligence artificielle : De plus en plus la contribution de la logique floue dans le développement des systèmes intelligents est évidente. En effet, elle est la seule approche, jusqu'à présent, qui permet la tolérance des imprécisions. Une telle caractéristique est nécessaire dans tout système intelligent qui opère dans des domaines où la connaissance est imprécise. En traitant le cas du domaine d'estimation du coût, notre projet illustre le rôle important de la logique floue au sein de l'intelligence artificielle. De plus , dans notre projet, nous utilisons une panoplie de techniques relevant de l'approche symboliste et connexionniste de l'IA afin de développer notre modèle; ceci est conforme et soutient les nouvelles tendances en IA encourageant une approche hybride plutôt qu'une approche purement symboliste ou connexionniste pour la résolution de problèmes complexes [47,62,98,99].

Mesure de logiciels : La mesure de logiciels est l'application directe de la théorie de mesures développée par Krantz et al. [57]. Cette théorie considère que, souvent, les résultats des mesures sont des valeurs numériques. En génie logiciel, l'utilisation des valeurs numériques, pour l'évaluation des attributs d'un logiciel, ne nous semble pas toujours la plus appropriée. En effet, souvent les évaluations d'un attribut d'un logiciel sont subjectives. Par conséquent, une solution possible à ce problème est d'utiliser des ensembles flous plutôt que des valeurs numériques pour quantifier ces évaluations. Notre

projet explore cette issue et montre les avantages de l'utilisation des valeurs linguistiques comme résultats du processus de mesure d'un attribut en estimation du coût.

Computation intelligente en génie logiciel : L'utilisation de la computation intelligente en génie logiciel (GL) a pour objectif l'amélioration des paradigmes classiques adoptés en GL; ceci est pour répondre aux nouveaux besoins provenant de domaines dans lesquelles on exige un degré d'intelligence élevé au logiciel à développer (robotique, contrôle de processus industriels, etc.). La computation intelligente englobe, principalement, la logique floue [108-121], les réseaux de neurones [23,30,31,66,74] et les algorithmes génétiques [18,27]. Peu de travaux de recherche se sont intéressés à l'application de la computation intelligente en génie logiciel [20,29,50,51,72,73]. Notre projet est parmi ces premiers travaux visant principalement l'estimation du coût de logiciels. Il montrera ainsi l'utilité de l'utilisation de la computation intelligente en génie logiciel.

4. Etat d'avancement des travaux

Les travaux de recherches pour aborder les problématiques et développer les méthodologies requises ont progressé de façon significatives. Plusieurs de ces résultats de notre projet de recherche ont été déjà fait l'objet de publications [35-44]. Dans cette section, nous résumons les contributions les plus pertinentes de ces publications par rapport aux problématiques de recherche.

Incorporation de la logique floue dans le modèle COCOMO'81: Dans cette publication, 12 facteurs parmi les 17 conducteurs du coût utilisés dans le modèle COCOMO'81 ont été représentés par des ensembles flous plutôt que par des intervalles classiques. Les résultats obtenus, après la validation de la version 'fuzzy' du modèle, ont montré que la logique floue permet au modèle une meilleure stabilité (à partir d'entrées

approximativement égales, on produit des sorties aussi approximativement égales) contrairement au cas de la version initiale du COCOMO'81 [34-35]

Proposition d'un ensemble de mesures de similarité entre les projets logiciels : Dans cette publication, nous avons proposé un ensemble de nouvelles mesures de similarité entre les projets logiciels. Ces mesures sont à utiliser dans le cas des projets logiciels décrits par des valeurs linguistiques. Elles ont plusieurs avantages sur celles qu'on utilisait avant pour mesurer la similarité entre les projets logiciels (distance euclidienne, distance d'égalité). Ces mesures ont été, ensuite, améliorées en utilisant le concept de quantificateur linguistique [36,37,38].

Proposition d'une nouvelle approche d'estimation du coût utilisant le CBR et la logique floue : Dans cette publication, nous avons développé une nouvelle technique, baptisée Fuzzy Analogy, pour l'estimation du coût de logiciels. Fuzzy Analogy peut être utilisée dans le cas des projets logiciels décrits soit par des valeurs numériques ou des valeurs linguistiques. Notre approche a été validée sur la base de données COCOMO'81 et a prouvé qu'elle performe mieux que trois autres techniques [39,40].

Propositions d'une technique pour la gestion de l'incertitude dans l'estimation par analogie : Dans ce travail, nous avons développé une technique pour la gestion de l'incertitude au niveaux des estimations fournies par Fuzzy Analogy. Cette technique utilise la théorie de possibilité de Zadeh, une sous discipline de la logique floue, pour modéliser l'affirmation *'similar software projects have possibly similar costs'* adoptée par notre approche d'estimation du coût. Autre que de fournir aux gestionnaires plusieurs valeurs estimées du coût, la gestion de l'incertain permet aussi l'évaluation des risques au niveau de la prise de décision [44].

5. Dernière étape à entreprendre avant d'entamer la rédaction de la thèse

Notre approche doit satisfaire aussi le dernier critère d'intelligence : l'apprentissage. Les réseaux de neurones est l'outil préféré actuellement pour implanter l'apprentissage dans les systèmes intelligents [23,30,31,66,74]. Ils se distinguent de toutes les autres approches de l'intelligence artificielle par le fait qu'ils intègrent des mécanismes d'apprentissage automatiques dans les solutions qu'ils proposent.

Certaines fonctionnalités d'apprentissage sont déjà introduites dans notre modèle d'estimation du coût [40]. En effet, l'outil F_ANGEL, supportant notre approche Fuzzy Analogy, permet à l'utilisateur de

1- Supprimer les attributs qui ne sont plus significatifs à l'estimation du coût.
2- Ajouter de nouveaux attributs significatifs pour l'estimation du coût.
3- Modifier les informations concernant un attribut particulier :
   - ✍ *Ajout de nouvelles valeurs linguistiques*. Par exemple, un attribut logiciel tel que l'expérience des programmeurs pourra être évalué, durant une première période, par seulement trois valeurs linguistiques: 'bas', 'moyen' et 'élevé'. Avec l'ancienneté cumulée par les programmeurs, d'autres valeurs linguistiques telles que 'très élevé' et 'extra-élevé' seront nécessaires pour une bonne évaluation de l'expérience des programmeurs.
   - ✍ *Suppression des anciennes valeurs linguistiques*. Par exemple, puisque de plus en plus on demande des logiciels très fiables, la valeur linguistique 'très bas' ne serait plus significative pour l'évaluation de la fiabilité d'un logiciel. Il est donc préférable de la supprimer de l'échelle d'évaluation de la fiabilité logicielle.
   - ✍ *Changement de la représentation des valeurs linguistiques*. Par exemple, la fiabilité d'un logiciel serait évaluée, dans un organisme, par 'élevé' si son nombre de défaillances est de 6 à 10 par mois. Pour répondre à des exigences externes, l'organisme pourra redéfinir la valeur linguistique 'élevé' par 3 à 7 défaillances par mois.

✍ *Changement du poids associé à l'attribut*: Par exemple, un organisme qui développe des applications de gestion mono-poste affectera un poids faible à l'attribut 'sécurité du logiciel'. Si l'organisme s'investit dans le développement des applications de gestion distribuées, il doit affecter un poids supérieur, que celui du cas des applications mon-poste, à l'attribut 'sécurité du logiciel'.

4- Modifier la définition du quantificateur linguistique utilisé dans l'évaluation de la similarité entre les projets logiciels.
5- Modifier la définition de la valeur linguistique 'closely similar' utilisée dans l'étape d'adaptation

Pour l'apprentissage au niveau des sorties de notre approche, c.à.d., des coûts estimés, nous avons opté pour l'utilisation des réseaux de neurones. L'apprentissage est de type supervisé puisque nous disposons des coûts réels des projets logiciels. Cependant, l'utilisation des réseaux de neurones en estimation du coût a été beaucoup contestée à cause de leur caractère opaque. Par conséquent, dans un premier travail, nous avons étudié l'interprétation d'un réseau de neurones en estimation du coût de développement de logiciels. Pour ceci, nous avons développé un modèle basé sur un réseau de neurones 'feedforward' de trois couches utilisant la fonction sigmoïde pour les neurones de la couche cachée et la fonction Identité pour ceux de la couche de sortie. Nous avons utilisé les projets de la base de données COCOMO'81 pour l'apprentissage et le test de notre modèle d'estimation. Pour interpréter le réseau, nous avons utilisé une méthode pour transformer notre réseau de neurones en un système à base de règles floues (si-alors) [8]. Les règles floues obtenues expriment la connaissance encodée dans le réseau par les poids synaptiques. L'interprétation de chaque règle floue est donnée par l'interprétation de sa prémisse et de sa conséquence. Notre étude a montré que nous pouvons fournir une interprétation compréhensible aux propositions floues composant la prémisse ainsi qu'à la conséquence de chaque règle floue. Cependant, la combinaison des différentes

propositions floues par l'opérateur 'i-or' défini dans [8] semble inapproprié en estimation du coût [42,43].

Dans un deuxième travail, nous utiliserons un réseau de neurones du type Radial Basis Function Network développé par Moody et Darken [64]. Ce genre de réseaux de neurones est, à priori, facilement interprétable. En plus, ils utilisent les deux types d'apprentissage : supervisé et non-supervisé. L'architecture d'un RBNF est composé de trois couches. Dans la couche cachée, on utilise la fonction Gaussienne comme fonction d'activation. Dans la couche de sortie, on utilise une fonction dérivable ; dans notre cas, nous utilisons la fonction Identité. Des travaux de recherche pour la validation de ce genre de réseaux et son intégration dans notre approche sont actuellement en cours d'exploration.

6. Méthodologie de validation

Dans cette section, nous présentons la méthodologie de validation que nous adopterons pour évaluer notre modèle d'estimation du coût. Dans la littérature, un modèle d'estimation du coût est souvent évalué en comparant les estimations qu'il fournit avec les valeurs réelles des coûts de logiciels. En plus de l'exactitude des estimations, notre méthodologie de validation considère aussi les trois autres critères d'intelligence que nous avons définis dans la Section 1, à savoir :

1- La tolérances des imprécisions au niveau de la description des projets logiciels,
2- la gestion de l'incertitude au niveau des estimations fournies, et
3- l'apprentissage

La validation d'un modèle d'estimation du coût nécessite l'utilisation d'une base de données contenant des projets logiciels déjà complétés. En estimation du coût, il n'y a pas un grand nombre de bases de données disponibles et ceci pour deux raisons principales :

✍ Le processus de collecte de données en industrie de logiciels est très dispendieux.

✍ La plupart des organismes dans l'industrie de logiciels ne sont pas encore convaincus de la nécessité des études empiriques dans l'évolution de cette industrie.

6.1 Validation de notre modèle selon les deux critères : exactitude des estimations et tolérance des imprécisions

Nous utiliserons donc la base de données COCOMO'81 pour la validation de notre modèle. Cette base de données a été publiée dans [10]. 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. Cette base de données contient 63 projets logiciels. Chaque projet est décrit par 17 attributs : la taille mesurée en terme de KDSI (Kilo Delivered Source Instructions), le mode du projet défini par trois catégories (organic, semi-detached, embedded), et 15 autres facteurs représentant l'environnement de développement tels que l'expérience des programmeurs, la complexité logicielle et la volatilité de la machine virtuelle. Ces 15 facteurs sont mesurés sur une échelle ordinale composée de six valeurs linguistiques : 'very low', 'low', 'nominal', 'high', 'very high', et 'extra-high'. Ces valeurs linguistique ont été représentées dans le modèle COCOMO'81 par des intervalles classiques (voir Tableau 1 pour le cas du facteur DATA). Afin de valider notre modèle, la première étape consistera à représenter ces valeurs linguistiques par des ensembles flous. Par exemple, pour le cas du facteur DATA, nous avons défini les ensembles flous suivants (Figure 2). Nous avons aussi défini les ensembles flous correspondant aux autres facteurs du modèle COCOMO'81. Parmi ces 15 facteurs, les quatre attributs RELY, CPLX, MODP, et TOOL n'ont pas été 'fuzzifiés' car leurs descriptions dans [10] sont insuffisantes :

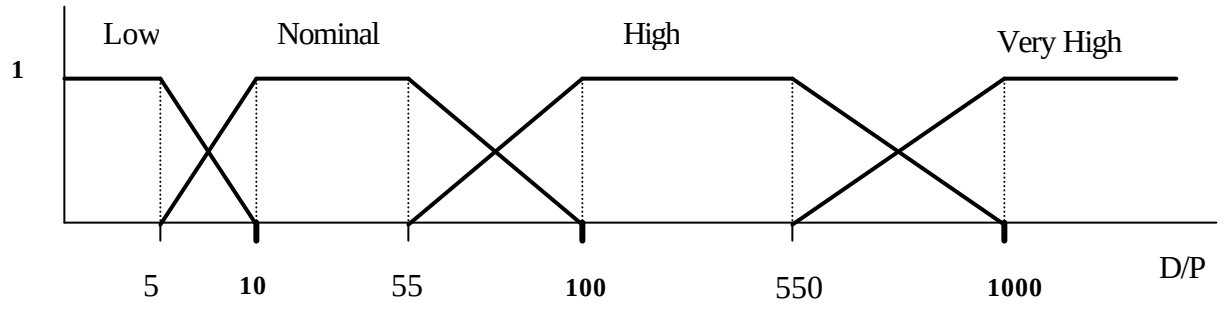

Figure 2 : Ensembles flous pour le facteur DATA du modèle COCOMO

.

La validation de notre modèle se fera sur quatre bases de données déduites, en utilisant une procédure aléatoire, à partir de la base de données du modèle COCOMO'81. En effet, cette dernière contient seulement les multiplicateurs d'effort associés aux 63 projets du modèle; or pour calculer les degrés d'appartenance relatifs aux ensembles flous, que nous avons définis pour les facteurs, nous aurons besoin des valeurs réelles des différents facteurs des 63 projets. Ainsi, chacune des quatre bases contient 63 projets décrits par des valeurs générées aléatoirement selon les qualifications données de leurs facteurs dans la base initiale du modèle. Par exemple, le facteur DATA du cinquième projet de la base de données du modèle est déclaré 'low'. Par conséquent, selon le tableau 1, les quatre valeurs générées pour les quatre bases sont entre 0 et 10.

La performance de notre modèle d'estimation sera comparée à celle de trois autres modèles : (1) COCOMO'81, (2) 'fuzzy' COCOMO'81 [34,35], et (3) la procédure classique d'estimation par analogie. Au niveau de l'exactitude des estimations, nous utilisons l'erreur moyenne relative (MRE) définie par :

$$MRE \; ? \left| \frac{coût_{réel} \; ? \; coût_{estimé}}{coût_{réel}} \right|$$

L'erreur moyenne relative est calculée pour chaque projet des bases de données utilisées dans la validation. Nous utilisons aussi un autre critère d'évaluation de l'exactitude d'un modèle d'estimation du coût, à savoir, le niveau de prédiction (Pred) défini par :

$$Pred(p) \; ? \; \frac{k}{N}$$

où N est le nombre total des projets logiciels; k est le nombre de projets logiciels ayant un MRE inférieure ou égale à p. Souvent, dans la littérature, on fixe p à 0,25. En plus des deux indicateurs (MRE et Pred), nous utilisons dans notre validation quatre autres

quantités : minimum des MREs, maximum des MREs, moyen des MREs et l'écart-type des MREs (pour plus de détails sur les résultats de cette validation, voir notre article dans l'Annexe A) .

Notre modèle d'estimation par analogie engage beaucoup de calculs; nous avons donc développé un prototype d'un logiciel supportant notre modèle. Ce prototype, Fuzzy AnaloGy softwaE tool-F_ANGEL, représente la version floue de celui qui a été développé par Shepperd et al [91,92].

6.2 Validation de notre modèle selon le critère : gestion de l'incertitude

L'estimation par analogie est basée sur l'hypothèse suivante : 'similar projets have similar costs'. Cette hypothèse n'est pas toujours déterministique. En effet, on peut avoir deux projets qui sont, à priori, similaires mais leurs coûts sont considérablement différents. Afin de vérifier cette hypothèse en estimation du coût, nous avons mené une expérimentation en utilisant les données du modèle COCOMO'81. L'hypothèse 'similar projects have similar costs' est formulée par l'implication suivante :

$$S(P_1, P_2) \text{ -> } C(C_1,C_2)$$

où S est une mesure de similarité entre les deux projets logiciels P_1 et P_2 ; C est une mesure de similarité entre les deux coûts (C_1,C_2) des projets P_1 et P_2. Si l'hypothèse est déterministique alors nous devrons avoir, pour tous couples de projets (P_1, P_2), $S(P_1, P_2)$? $C(C_1,C_2)$; sinon l'hypothèse est non-déterministique. En effet $S(P_1, P_2)$ (respectivement $C(C_1,C_2)$) représente le degré de vérité de l'hypothèse 'P_1 et P_2 sont similaires' (respectivement 'C_1et C_2 sont similaires'. Au niveau des projets logiciels, nous utilisons les mesures de similarités que nous avons développées et validées [36,37,38]. Pour les coûts, nous utilisons les deux mesures de similarités suivantes C_R et C_A (Figure 3). Cette expérimentation nous permettra d'identifier la nature de l'hypothèse 'similar projects have similar costs' qui est à la base de notre modèle d'estimation par analogie.

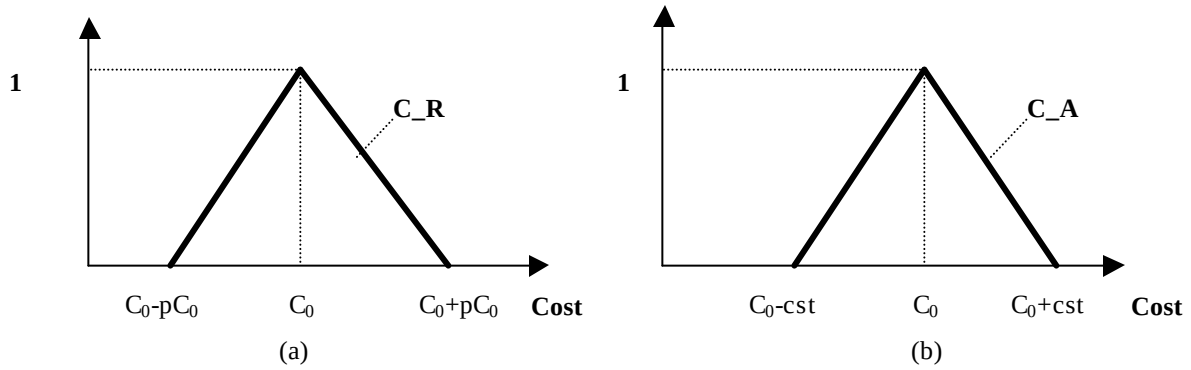

Figure 3. Deux mesures de similarité entre les coûts de logiciels : C_R prend en considération le coût C0 pour définir l'intervalle des coûts similaires à C0; C_A ne prend pas en considération le coût C0 pour définir l'intervalle des coûts similaires à C0

.

6.3 Validation de notre modèle selon le critère : apprentissage

Cette phase de validation est composée de deux étapes. Dans la première étape, on validera quelques méthodes de transformation d'un réseau de neurones en un système `base de règles floues' afin d'éviter la critique de 'boîte noire' envers notre modèle d'estimation. Le réseau de neurones sera considéré facilement interprétable si les règles floues 'si-alors', qui en découlent, sont aussi de même. Dans une deuxième étape, nous adopterons le réseau de neurones, qui a été considéré comme facilement interprétable dans la première étape, pour intégrer l'apprentissage dans notre modèle d'estimation par analogie. La validation de notre modèle d'estimation par analogie, utilisant un réseau de neurones pour l'apprentissage, serait faite sur la base de données COCOMO'81. Nous divisons cette base de données en deux : une pour l'apprentissage et l'autre pour les tests de l'exactitude des estimations de notre modèle. L'exactitude des estimations de notre modèle serait évaluée par les deux indicateurs MRE et Pred..

7. Plan de la thèse

Nous proposons que la thèse soit composée de huit chapitres :

Chapitre 1 : Introduction

Nous présenterons la problématique des limites actuelles des modèles existants d'estimation du coût de logiciels, en particulier nous traitons le cas d'estimation du coût par Analogie

Chapitre 2 : Intelligence Artificielle

Dans ce chapitre, nous présenterons l'état de l'art de l'intelligence Artificielle. Ainsi, nous présentons les éléments de l'histoire de l'IA, relatifs aux problématiques que nous traitons, l'IA du point de vue des sciences cognitives, et l'IA du point de vue ingénierie. Nous mettrons l'accent sur deux éléments principaux au cours de cette discussion :

1- Le rôle de la logique floue dans le projet de l'IA, et
2- L'utilité d'une approche hybride (symboliste et connexionniste) dans l'IA

Chapitre 3 : Eléments de base de la Logique Floue

Dans ce chapitre, nous présenterons les concepts de base de la logique floue tels que les ensembles flous, les propositions floues, le raisonnement flou, le soft computing et le computing with words.

Chapitre 4 : Estimation du coût de logiciels et la Logique Floue

Ce chapitre présentera un état de l'art sur les modèles d'estimation du coût. En particulier, nous discuterons, dans ce chapitre, de l'utilité et de l'intérêt de l'utilisation de la logique floue dans le processus d'estimation du coût de logiciels. Comme démonstration, nous présentons notre application de la logique floue au modèle COCOMO.

Chapitre 5 : Un modèle d'estimation du coût utilisant le raisonnement par Analogie et la Logique Floue

Dans ce chapitre, nous proposerons, l'adéquation du raisonnement par analogie à l'estimation du coût de logiciels. Aussi, nous présenterons, quelques travaux de recherches traitant l'utilisation de l'analogie en estimation du coût. Nous présenterons, ensuite, notre nouvelle méthode intégrant la logique floue dans le processus d'estimation par analogie (Fuzzy Analogy). Le chapitre décrit, enfin, une validation de notre approche ainsi qu'une comparaison de cette approche avec celles utilisant l'analogie classique.

Chapitre 6 : La gestion de l'incertitude dans l'estimation du coût par l'analogie floue

Ce chapitre rappellera l'utilité et l'intérêt de la gestion de l'incertain dans le processus d'estimation du coût; en particulier, nous traitons le cas de notre approche : Fuzzy Analogy. Nous présenterons, après, une technique que nous aurons développée pour la gestion de l'incertain dans notre approche. Le chapitre décrira, enfin, l'utilisation de la gestion de l'incertain dans l'évaluation des risques (Risk Assessment) dans notre approche.

.

Chapitre 7 : L'apprentissage dans l'estimation du coût par l'analogie floue

Ce chapitre présentera, dans une première partie, une étude sur les applications des réseaux de neurones en estimation du coût; en particulier, nous discutons leur caractère 'black boxes'. Dans une deuxième partie, nous présentons une technique pour l'apprentissage dans notre approche Fuzzy Analogy. Cette technique utilise les Radial Basis Function Networks.

Chapitre 8 : Conclusions et Perspectives

Ce chapitre présentera une synthèse du travail réalisé dans ce projet de recherche. Aussi, il présente les éventuelles critiques et limites identifiés de la solution que nous avons apportée à la problématique de recherche. Nous terminerons, ce chapitre, par la présentation des perspectives de cette recherche en informatique ainsi qu'en sciences cognitives.

7. Extrait des références utilisées

[1] Aamodt and E. Plaza. "Case-Based Reasoning: Foundational Issues, Methodological Variations. and System Approaches", AI Communications, IOS Press, vol. 7:1, pp. 39-59, 1994.

[2] A. Abran and P.N. Robillard. "Functions Points Analysis: An Empirical Study of its Measurement Processes", IEEE Transactions on Software Engineering, 22(12): pp. 895-909, 1996.

[3] W. D. Aha, "Case-Based Learning Algorithms", DARPA Case-Based Reasoning Workshop, Morgan Kaufmann, LoaAtlos, CA, 1991

[4] Amrane, B., Slimani, Y., "ALCOMO Model: A statistical reformulation of the COCOMO'81 model", Information science and Technology, Apr, 1993.

[5] Albrecht, A.J., Gaffney, J.E., "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation", IEEE Transactions on Soft Eng, Vol. SE-9, No. 6, Nov, 1983, pp. 639-647

[6] L. Angelis and I. Stamelos. "A Simulation Tool for Efficient Analogy Based Cost Estimation", Empirical Software Engineering, vol. 5, no. 1, pp. 35-68, 2000.

[7] L. Angelis, I. Stamelos, and M. Morisio. "Building a Software Cost Estimation Model Based on Categorical Data", In Proceedings of the 7th International Software Metrics Symposium, April 2001, pp. 4-15, London, UK, IEEE Computer Society.

[8] J. M. Benitez, J. L. Castro, I. Requena, "Are Artificial Neural Networks Black Boxes", IEEE Transactions on Neural Networks, Vol. 8, no. 5, September, 1997, pp. 1156-1164

[9] Bisio R., Malabocchia F., "Cost estimation of software project through case base reasoning", First International conf. on Case-Based reasoning, Sesimbra, 1995, Springer pp.11-22

[10] B.W. Boehm. "Software Engineering Economics", Prentice-Hall, 1981.

[11] B.W. Boehm et. al. "Cost Models for Future Software Life Cycle Processes: COCOMO II". Annals of Software Engineering: Software Process and Product Measurement, Amsterdam, 1995.

[12] L. Briand, T. Langley, and I. Wieczorek.. "Using the European Space Agency Data Set: A Replicated Assessment and Comparison of Common Software Cost Modeling". In Proceedings of the 22nd International Conference on Software Engineering, Limerik, Ireland, 2000, pp. 377-386

[13] B. Bartsch-Spoerl, "Towards the integration of case-based, schema-based, and model-based resoning for supporting complex design tasks", Proc. 1st International Conference on case-based reasoning, 1995, pp. 145-156

[14] B. Bell, S. Keddar, R. Bareiss, "Interactive model-driven case adaptation for instructional software design", Proc. 16th annal conference of the Cognitive Science Society, Lawrence Erlbaum, 1994, pp. 33-38

[15] D.S. Chulani. "Incorporating Bayesian Analysis to Improve the Accuracy of COCOMO II and Its Quality Model Extension", Ph.D. Qualifying Exam Report, University of Southern California, February 1998.

[16] Eric Davalo, Patrick Naïm, "Des Réseaux de neurones", Eyrolles, 1990

[17] L. Davis, "Handbook of Genetic Algorithms", Van Nostrand Reinhold, New York, 1991.

[18] D. Dubois, F. Esteva, P. Garcia, L. Godo, R.l. deMantaras, H. Prade, "Case-based Reasoning: A Fuzzy Approach", IJCAI'97, Workshop on Fuzzy Logic in Artificial Intelligence, Lecture Notes in Atificial Intelligence, Vol. 1566, Springer, Berlin, 1999, pp. 79-90

[19] D. Dubois, H. Prade, "Fuzzy set Teory and Systems", NY Academic, 1980

[20] C. Ebert, "An Approach to fuzzy data analysis for software quality models", Proc. 1st European Congress on Fuzzy and Intelligent Technologies, Aachen, September, 1993, pp.1156-1161

[21] N. Fenton and S.L. Pfleeger, "Software metrics: A Rigorous and Practical Approach". International Computer, Thomson Press, 1997.

[22] K. Ganesan, T. M. Khoshgoftaar, and E. Allen. "Case-Based Software Quality Prediction", International Journal of Software Engineering and Knowledge Engineering, 10(2):139-152, 2000.

[23] S. I Gallant, "Neural Network Learning and Expert Systems", MIT Press, Cambridge, 1993

[24] D. Gentner, "Structure Mapping: A Theorotical Framework of Analogy", Cognitive Science, Vol. 7, 1983, pp. 155-170

[25] D. Gentner, "Analogy", In W. Bechtel & G. Graham (Eds), A companion to cognitive science, Oxford: Blackwell, 1998, pp. 107-113

[26] D. Gentner, K. J. Holyoak, B. Kokinov. "The Analogical Mind: Perspectives from Cognitive Science", Bradford Books, March, 2001

[27] D. E. Goldberg, " Genetic Algorithms in search, optimisation and learning", Addison Wesley, Reading, MA, 1989

[28] R. Gulezian. "Reformulating and Calibrating COCOMO", Journal Systems Software, vol. 16, pp.235-242, 1991.

[29] A. R. Gray, S. G. NacDonell, "Fuzzy Logic Techniques For Software Metric Models of Development", Advances in Fuzzy Systems-Applications and Theory, Vol. 16, Computational Intelligence in Software Engineering, Editors : W. Pedrycz and J.F. Peters, 1997, pp. 321-338

[30] S. Hayken, "Neural Networks: A Guide to Intelligents Systems", Wiley, 1999

[31] John Hertz, Anders Krogh & Richard G. Palmer, "Introduction to the theory of neural computation", Addison Wesley, 1991

[32] J. K. Holyoak, P. Taghard, "Mental Leaps: Analogy in Creative Thought', MIT Press, 1995.

[33] R.T Hughes, "An Evaluation of Machine Learning Techniques for Software Effort Estimation", University of Brighton, 1996

[34] A. Idri, L. Kjiri, and A. Abran. "COCOMO Cost Model Using Fuzzy Logic", In Proceedings of the 7th International Conference on Fuzzy Theory and Technology, Atlantic City, NJ, February, 2000, pp. 219-223.

[35] A. Idri, A. Abran, "La mise en valeur d'une approche floue pour l'évaluation du coût de développement de logiciels", CARI, Madagacscar, 16-19 Octobre, 2000 , pp.247-254

[36] A. Idri and A. Abran. "Towards A Fuzzy Logic Based Measures for Software Project Similarity", In Proceedings of the 6th Maghrebian Conference on Computer Sciences, November, Fes, Morroco, 2000, pp. 9-18,

[37] A. Idri and A. Abran. "A Fuzzy Logic Based Measures For Software Project Similarity: Validation and Possible Improvements", In Proceedings of the 7th International Symposium on Software Metrics, April, London, IEEE Computer Society, 2001, pp. 85-96

[38] A. Idri and A. Abran. "Evaluating Software Projects Similarity by Using Linguistic Quantifier Guided Aggregations", In Proceedings of the 9th IFSA World Congress and 20th NAFIPS International Conference, July, Vancouver, 2001, pp. 416-421,

[39] A. Idri, A. Abran, and T. M. Khoshgoftaar. "Fuzzy Analogy: A new Approach for Software Cost Estimation", In Proceedings of the 11th International Workshop on Software Measurements, August, Montreal, 2001, pp. 93-101,

[40] A. Idri, A. Abran, T. Khoshgoftaar, S. Robert, "Estimating Software Project Effort by Analogy based on Linguistic Values", 8th International Symposium on Software Metrics, IEEE computer Society, June, Ottawa, 2002, pp.21-30

[41] A. Idri, T. Khoshgoftaar, A. Abran, S. Robert, "Can Neural Networks be easily Interpreted in Software Cost Estimation?", Fuzz-IEEE, 12-17 May, Hawai, 2002, pp. 1162-1166

[42] A. Idri, S. Mbarki, A. Abran, S. Robert, "Interprétation des réseaux de neurones en estimation du coût de logiciels ", to be presented at CARI, Octobre, Cameroun, 2002

[43] A. Idri, A. Abran, T. Khoshgoftaar, "Fuzzy Case-Based Reasoning Models for Software Cost Estimation", to appear in the book Soft Computing in Software Engineering: Theory and Applications, published by Springer-Verlag, 2002

[44] A.Idri, A. Abran, T. Khoshgoftaar, S. Robert, "Investigating Soft Computing in Case-Based Reasoning for Software Cost Estimation", Submitted to International Journal of Engineering Intelligent Systems, 2002

[45] J.P. Jacquet and A. Abran. "Metrics Validation Proposals: A Structured Analysis", In Proceedings of the 8th International Workshop on Software Measurement, September, Magdeburg, Germany,1998,

[46] R. Jager. "Fuzzy Logic in Control", Ph.D. Thesis, Technic University Delft, 1995, Holland.

[47] J. S. R. Jang, C. T. Sun, E. Mizutani, "Neuro-Fuzzy and Soft Computing", Prentice-Hall, 1997

[48] C. F. Kemerer, "An empirical Validation of Software Cost Estimation Models", Communications of the ACM, 30(5), 1987, pp. 416-429

[49] M Jorgersen, "Experience with Accuracy of Software Maintenance Task Effort Prediction Models", IEEE Transaction on Software Engineering, Vol. 21(8), 1995, pp. 674-681

[50] T. M. Khoshgoftaar, E. B. Allen, "Neural Networks for Software Quality Models", Advances in Fuzzy Systems-Applications and Theory, Vol. 16, Computational Intelligence in Software Engineering, Editors : W. Pedrycz and J.F. Peters, 1997, pp. 65-96

[51] T. M. Khoshgoftaar, M. P. Evett, E. B. Allen, P.D. Chien "An Application of Genetic Programming to Software Quality Prediction", Advances in Fuzzy Systems-Applications and Theory, Vol. 16, Computational Intelligence in Software Engineering, Editors : W. Pedrycz and J.F. Peters, 1997, pp. 175-196

[52] T. M. Khoshgoftaar, B. Cukic, and N. Seliya. "Predicting Fault-Prone Modules in Embedded Systems Using Analogy-Based Classification Models". International Journal of Software Engineering and Knowledge Engineering : Special Volume on Embedded Software Engineering, 12(1):1-22, 2002.

[53] B. Kitchenham and S. Linkman. "Estimates, Uncertainty and Risks", IEEE Software, 14(3): pp. 69-74, 1997.

[54] B. Kitchenham, S.L. Pfleeger and N. Fenton. "Towards a Framework for Software Measurement Validation", IEEE Transaction on Software Engineering, vol. 21, December, 1995.

[55] G. Kadoda, M. Cartwright, L. Chen, and M. Shepperd. "Experiences Using Case-Based Reasoning to Predict Software Project Effort", In Proceedings of EASE, p.23-28, 2000, Keele, UK.

[56] J.L. Kolodner, "Case-Based Reasoning", Morgan Kaufmann, 1993.

[57] D. H. Krantz, R. D. Luce, P. Suppes, and A. Tversky. "Foundations of Measurement: Additive and Polynomial Representations ", Academic Press, vol. 1, 1971.

[58] K. J. Kurtz, D. Gentner, V. Gunn, "Reasoning", Cognitive Science: Handbook of Perceptions and Cognition, 2nd ed. 1999, pp145-200

[59] D. Leake, "CBR in context : the present and the future", In Case-Based Rasoning experiences, lessons and future directions, AAAI Press, Menlo Park, 1996.

[60] E. R. Mac Cormac, "A Cognitive Theory of Metaphor", MIT Press, 1985

[61] J. Matson, E. B. E. Barrett, J. M. Mellichamp. "Software Development Cost Estimation Using Function Points", Transactions on Software Engineering, vol. 20, no. 4, pp. 275-287, April 1994, IEEE Computer Society.

[62] L. R. Medsker, "Hyprid Neural Networs and Expert Systems", Kluwer Academic Publishers, 1994

[63]] Miyazaki, Y., Mori, K., "COCOMO'81 evaluation and tailoring", In Proc. Eighth Int. Conf. Soft. Eng., London, UK, Aug, 1985, pp.292-299

[64] J. Moody, C. J. Darken, "Fast Learning in Networks of Locally-Tuned Processing Units", Neural Computation, 1, 1989, pp. 281-294

[65] I. Myrtveit and E. Stensrud. "A Controlled Experiment to Assess the Benefits of Estimating with Analogy and Regression Models", Transactions on Software Engineering, vol. 25, no. 4, IEEE Computer Society, July, 1999, pp. 510-525,

[66] M. Negnevitsky, "Artificial Intelligence: A guide to intelligent systems", Addison-Wesley, 2001

[67] A. Newell, H. A. Simon, "Human Problem Solving", Englewood Cliffs, NJ, Prentice-Hall, 1972

[68] A. Newell, “Physical Symbol Systems”, Cognitive Science, Vol. 4, 1980, 115-183

[69] A. Newell, “Unified Theory of Cognition”, Harvard University Press, 1990

[70] F. Niessink and H. Van Vliet. “Predicting Maintenance Effort with Function Points”, In Proceedings of the International Conference on Software Maintenance, 1997, Bari, Italy, IEEE Computer Society.

[71] S. K. Pal, T. S. Dillon, D. S. Yeung, “Soft computing in case-based reasoning”, Springer-Verlag, 2001

[72] W. Pedrycz, J.F. Peters, “Computational Intelligence in Software Engineering”, IEEE Canadien Conf. On Electrical & Computer Engineering, May, 1997, pp. 253-257

[73] W. Pedrycz, Z. A. Sosnowski, “The Design of decision trees in the framework of granular data and their application to software quality models”, Fuzzy sets and Systms Journal, Vol. 123, 2001, pp. 271-290

[74] L. I. Perlovsky, “Neural Networks and Intellect using Model-Based Concepts”, Oxford University Press, 2001

[75] A. Porter, R. Selby, “Empiracally-Guided Software Development using Metric-based Classification Trees” , IEEE Software, Vol. 7, March, 1990, pp. 46-54

[76] A. Porter, R. Selby, “Evaluating Techniques for Generating Metric-Based Classification Trees”, J. System Software, Vol. 12, July, 1990, pp. 345-361

[77] R. Proulx, “Le traitement symbolique et non symbolique dans les réseaux neuronaux : Un problème de catégorisation », Lekton, Vol. 4, no. 2, 1994, pp. 21-35

[78] L. H. Putnam “A General Empirical Solution to the Macro Software Sizing and Estimation Problem”, Transactions on Software Engineering, vol. 4, no. 4, July, 1978.

[79] Zenon W. Pylyshyn, “Computation and Cognition: Towards a Foundation for Cognitive Science”, MIT Press, 1984

[80] Zenon W. Pylyshyn, “Computing in cognitive Science, Foundations of Cognitive Sciences”, Ed. By M. I. Posner, MIT press, 1989, pp. 51-91

[81] C. V. Ramamoorthy, C. Chandra, S. Ishihara, Y. Ng, "Knowledge based tools risk assessment in software development and reuse", Porc. 6th International conference on Tools with Artificial Intelligence, IEEE computer society, 1992, pp. 364-371

[82] S. Robert, "Intelligence Artificielle et Sciences Cognitives", Intentionnalité en Question, Textes réunis et présentés par Dominique Janicaud, Librairie Philosophique J. Vrin, 1995, 231-250

[83] David E. Rumelhart, J. McClelland, "Parralel Distributed Processing", Explorations in the Microstructure of Cognition, Vol. 1: Foundations, Cambridge, MIT Press, 1986

[84] David E. Rumelhart, "The Architecture of Mind : A connectionist Approach, Foundations of cognitive Sciences", 1989, pp.133-159

[85] B. Samson, Ellison D., Dugard P, "Software Cost Estimation using an Albus Perceptron", 8th International COCOMO Estimation meeting, Pittsburgh, 1993

[86] R. Schank, "Dynamic Memory: A Theory of Reminding and Learning in Computer and People", Cambridge University Press, 1982

[87] C. Schofield, "Non-Algorithmic Effort Estimation Techniques", Tech. Report TR98-01, March, 1998

[88] John R. Searle, "Minds, Brains and Programs", Behavioral and Brain Sciences, Vol. 3, pp. 417-57

[89] R. Selby, A. Porter, "Learning from exemples: Generation and evaluation of Decion Trees for Software Resource Analysis", IEEE Transactions on Software Engineering, Vol. 14, 1988. pp. 1743-1757

[90] C. Serluca, "An Investigation into Software Effort Estimation using a Backpropagation Neural Network", M.Sc. Thesis, Bournemouth University, 1995

[91] M. Shepperd, C. Schofield, and B. Kitchenham. "Effort Estimation using Analogy", In Proceedings of the 18th International Conference on Software Engineering, 1996, pp. 170-178, Berlin.

[92] M. Shepperd and C. Schofield. "Estimating Software Project Effort Using Analogies", Transactions on Software Engineering, vol. 23, no. 12, pp. 736-743, November 1997. IEEE Computer Society.

[93] M. Shepperd and G. Kadoda. "Using Simulation to Evaluate Predictions Systems", In Proceedings of the 7th International Symposium on Software Metrics, pp. 349-358, April 2001, England, UK, IEEE Computer Society.

[94] K. K. Shukla, "Neuro-Genetic Prediction of Software Development Effort", Information Software Technology, Vol. 42, 2000, pp. 701-713

[95] P. Smolensky, "On the Proper Treatment of Connectionnism", Behavioral and Brain Sciences, Vol. 11, 1988, pp. 1-23

[96] S. S. Stevens. "On the Theory of Scales and Measurement", Science Journal, vol. 103, pp. 677-680, 1946.

[97] K. Srinivasan, Fisher D, "Machine Learning Approaches to Estimating Software Development Effort", IEEE Transaction on Software Engineering, Vol. 21, No. 2, February, 1995, pp. 126-136

[98] H. Tirri, "Implementing Expert System Rule Conditions by Neural Networks", New Generation Computing, Vol. 10, 1991, pp. 55-71

[99] G. G. Towell, "Symbolic Knowledge and Neural Networks: Insertion, Refinement, and Extraction", Ph.D dissertation, University of Wisconsin, 1992

[100] A. M. Turing (1936), "Théorie des nombres calculables, avec une application au problème de la décision », Traduit et annoté par Julien Bash., Voir Girard et Turing, 1995, pp. 49-104

[101] A. M. Turing (1950), "Computing Machinery and Intelligence", Traduit par Patrice Blanchard, Voir Girard et Turing, 1995, pp. 135-175

[102] A. Tversky**,** "Features of Similarity", Psycological Review, 84:327-352

[103] S. Vicinanza and M.J. Prietulla. "Case-Based Reasoning in Software Effort Estimation", In Proceedings of the 11th International Conference on Information Systems, 1990.

[104] G. Wittig, G. Finnie, 'Estimating Software Development Effort with connectionist Models', Information and Software Technology, vol. 39, 1997, pp. 469-476

[105] R. R. Yager and J. Kacprzyk. "The Ordered Weighted Averaging Operators: Theory and Applications", Kluwer Academic Publishing, Norwell, MA, 1997.

[106] R. R. Yager. "Quantifier Guided Aggregation using OWA Operators", International Journal of Intelligent Systems, vol. 11, pp.49-73, 1996.

[107] R. R. Yager, "Fuzzy constraint Satisfaction for E-commerce Agents", 7th International Conference on Fuzzy Theory and Technology, Atlantic City, February, 2000, pp. 111-114

[108] L. A. Zadeh. "Fuzzy Set", Information and Control, vol. 8, pp. 338-353, 1965.

[109] L.A. Zadeh, "Similarity relations and fuzzy ordering", Information Sciences, 1971, pp.177-200

[110] L.A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes", IEEE Transaction, Syst. Man Cybern, Vol. 3, 1973, pp. 28-44

[111] L.A. Zadeh, "Calculus of fuzzy restrictions", In Fuzzy sets and their applications to cognitive and decision Processes, L. A. Zadeh, K. S. Fu, M. Shimura, Eds, Newyorks, 1975, pp. 1-39

[112] L. A. Zadeh, "PRUF- A meaning representation language for natural languages", Int. J. Man-Mach, Studies, Vol. 10, 1978, pp. 395-460

[113] L. A. Zadeh, "A theory of approximate reasoning", Machine Intelligence 9, 1979, pp. 149-194

[114] L.A. Zadeh, "Fuzzy Sets as a basis for a theory of possibility", Fuzzy Sets and Systems 1, 1979, pp. 3-28

[115] L.A. Zadeh, "Test-score semantics for natural languages and meaning representation via PRUF", Empirical Sematics, B. Reiger, Ed, 1981, pp.149-194

[116] L. A. Zadeh, "Outline of a computational approach to meaning and knowledge representation based on a concept of a generalized assignment statement", Proc. International Seminar Artificial Intelligence, Man-Manch, Syst, M. Thoma and A. Wyner, Eds, Springer-Verlag, 1986, pp.198-211

[117] L. A. Zadeh. "Fuzzy Logic. Neural Networks and Soft Computing", Communications of ACM, vol. 37, no. 3, pp.77-84, March 1994.

[118] Zadeh L. A, "Some Reflections on the Relationship Between AI and Fuzzy Logic", A Heretical View', IJCAI, Springer, 1997, pp.1-8

[119] L. A. Zadeh. "A Computational Approach to Fuzzy Quantifiers in Natural Languages", Computing and Mathematics, vol. 9. pp. 149-184, 1983.

[120] L. A. Zadeh, "From computing with numbers to computing with words- from manipulation of measurements to manipulation of perceptions", Computing with words, Ed. Paul P. Wang, John Wiley&Sons, 2001

[121] L.A. Zadeh, "Fuzzy Logic = Computing with Words", IEEE Transactions on Fuzzy Systems , 1996, pp. 103-111

[122] H. J. Zimmerman, "Fuzzy Set Theory and Its Applications", 2nd Edition Boston, MA: Kluwer, 1991

[123] H. Zuse. "A Framework of Software Measurement", de Gruyter, 1998.

[124] H. Zuse. "Foundations of Validation: Prediction and Software Measures", In Proceedings of the AOSW, April 1994, Portland.

[125] H. Zuse. "Validation of Measures and Prediction Models", In Proceedings of the 9th International Workshop on Software Measurement, September 1999, Lac-Supérieur, Canada.

[126] Zuse, "A Framework of Software Measurement", de Gruyter, 1998

Annexe 1

"Estimating Software Project Effort by analogy Based on Linguistic Values"

Presented at

8th IEEE International Symposium on Software Metrics
4-7 June, Ottawa, 2002

Estimating Software Project Effort by Analogy Based on Linguistic Values

Ali Idri
ENSIAS, BP. 713, Agdal
Université Mohamed V
Rabat, Morocco
E-mail : idri@ensias.ma

Alain Abran and Serge Robert
Software Management Research Laboratory
Departmentof computer sciences
Université du Quebéc À Montréal
E-mail : aabran@ele.etsmtl.ca
robert.serge@uqam.ca

Taghi M. Khoshgoftaar
Empirical Software Engineering Laboratory
Florida Atlantic University
E-mail : taghi@cse.fau.edu

Abstract

Estimation models in software engineering are used to predict some important attributes of future entities such as software development effort, software reliability and programmers productivity. Among these models, those estimating software effort have motivated considerable research in recent years. The prediction procedure used by these software-effort models can be based on a mathematical function or other techniques such as analogy based reasoning, neural networks, regression trees, and rule induction models. Estimation by analogy is one of the most attractive techniques in the software effort estimation field. However, the procedure used in estimation by analogy is not yet able to handle correctly linguistic values (categorical data) such as 'very low', 'low' and 'high'. In this paper, we propose a new approach based on reasoning by analogy, fuzzy logic and linguistic quantifiers to estimate software project effort when it is described either by numerical or linguistic values; this approach is referred to as Fuzzy Analogy. This paper also presents an empirical validation of our approach based on the COCOMO'81 dataset.

1. Introduction

Accurate estimation of the effort and the schedule required to develop and/or maintain a software system is one of the most critical activities in managing software projects. This task is known as Software Cost Estimation. In order to make accurate estimates and avoid gross misestimations, several cost estimation techniques have been developed. These techniques may be grouped into two major categories [25]:

- algorithmic models, and
- non-algorithmic models.

The first category holds the most popular techniques (at least in the literature), and is illustrated by estimation models such as COCOMO [5, 6, 8], PUTNAM-SLIM[23], and function points analysis [2, 20]. Algorithmic models are derived from the statistical or numerical analysis of historical projects data (simple/multiple/stepwise regression, Bayesian approach, polynomial interpolation, etc). There are two main disadvantages to these models. First, they make an assumption on the form of the prediction function, represented by: $Effort = ? \times size^{?}$ where ? represents a productivity coefficient and ? an economies (or diseconomies) of scale coefficient. Second, they need to be adjusted or calibrated to local circumstances (an example of calibrating and reformulating COCOMO'81 model is in [10]).

The non-algorithmic models have been developed to avoid the above mentioned shortcomings. Recently, many researchers have begun to turn their attention to this alternative, and in particular to a set of approaches based on neural networks, regression trees, rule induction, and case-based reasoning. This alternative has two significant advantages: First, the capability to model the complex set of relationships between the dependent variable to predict (cost, effort) and the independent variables (cost drivers) collected earlier in the lifecycle. Second, the capability to learn from historical projects data (especially for neural networks).

Experience has shown that there does not exist a 'best' prediction technique outperforming all the others in every situation. Indeed, Shepperd et al., Niessink and Van Vliet found that estimation by analogy generated better results than stepwise regression [22, 24, 25]. However, Briand et al., Stensrud and Myrtveit reported opposite results [7, 21]. Recent research has been initiated to explain the relationship between different properties of historical projects dataset (size, number of attributes, presence of outliers…) and the accuracy of a prediction system [26]. Our work deals with an important limitation of all estimation techniques, which arises when software

projects are described using categorical data (nominal or ordinal scale) such as 'very low', 'low' and 'high'. These qualifications are called linguistic values in fuzzy logic terminology. Building cost estimation models based on linguistic values is a serious challenge for the software cost estimation community. Recently, Angelis et al. [4] were the first to propose the use of categorical regression procedure (CATREG) to build cost estimation models when software projects are described by categorical data. This procedure quantifies categorical attributes by assigning numerical values to their categories in order to produce an optimal linear regression equation for the transformed variables. This approach has the following limitations:

- It replaces each linguistic value by one numerical value. This is based on the assumption that a linguistic value can always be defined without vagueness, imprecision and uncertainty. Unfortunately, this is not often the case. Indeed, linguistic values come from human judgements that are often vague, imprecise and uncertain. For example, let us assume that the experience of programmers is measured by three linguistic values: 'low', 'nominal' and 'high'. Most often the meaning of these values are not defined precisely and consequently we cannot represent them by individual numerical values.
- There is no natural interpretation of the numerical values assigned by this approach.
- It assigns numerical quantities to linguistic values in order to produce an optimal linear regression equation whereas the initial relation between effort and cost drivers may be non-linear.

A more comprehensive approach to deal with linguistic values is by using fuzzy set theory. Consequently, the purpose of this paper is to provide a new approach based on analogy and fuzzy logic to estimate effort when software projects are described either by numerical or linguistic values.

This paper is organized as follows: A discussion why categorical data should be considered as a particular case of linguistic values is presented in Section 2. In Section 3, we present the classical procedure of estimation by analogy;,which procedure cannot handle linguistic values. To overcome this limitation, we propose, in Section 4, a new approach that can be seen as a fuzzification of the classical approach of estimation by analogy. Section 5, presents an empirical validation of our technique by means of the COCOMO'81 dataset. The obtained results are compared to those of three other techniques. Finally, Section 6 discusses our findings and suggests future work in this area.

2. Categorical data and linguistic values

Here we discuss why categorical data, as software measurement researchers define them, are only a particular case of linguistic values. The two terminologies come from two different fields. The first is used in measurement theory whereas the second is used in fuzzy sets theory.

In 2001, measurement theory became over a hundred years old. According to Zuse, it began with Helmholtz pioneering paper: 'Counting and measuring from epistemological point of view' and lead to modern axiomatic representational theory of measurement as presented by Krantz et al. [19, 33].

As in other sciences (physics, medicine, civil, etc.), measurement has been discussed in software engineering for over thirty years. The objective of software measurement is to improve the software development process and consequently the quality of its various deliverables. This can be achieved by evaluating, controlling and predicting some important attributes of software projects such as development effort, software reliability, and programmers productivity. However, measurement in software engineering is challenging due to two main reasons. First, software engineering is a 'young' science and still needs further maturing. Second, many of the software attributes are still qualitative rather than quantitative such as portability, maintainability, and reliability. Their evaluation currently depends on human judgement. The qualitative issue is related to the scale type on which the attributes are measured. In measurement the scale type of a measure have been defined by Stevens in 1946 [27]: Nominal, Ordinal, Interval, Ratio or Absolute.

Categorical attributes are those with nominal or ordinal scale type. The Nominal scale type is the lowest scale level and it only allows the classification of software entities in different classes or categories. Examples of this scale type can be found in literature, such as the language used in the implementation phase (C, C++, Java, Cobol, etc) and the application type (Business, Control, Finance, etc). The Ordinal scale type provides us, in addition to the classification of software entities, information about an ordering of the categories. Examples of ordinal attributes are complexity of software (simple, nominal, complex) and software understandability (very low, low, nominal, high).

In software engineering, many software attributes are measured either on a Nominal or Ordinal scale type. Thus, to evaluate these attributes linguistic values are often used such as 'very low', 'complex', 'important' and 'essential'. When using linguistic values, imprecision, uncertainty and partial truth are unavoidable. However, until now, the software measurement community often use numbers or classical interval to represent these linguistic values. Furthermore, such transformation and representation does not mimic the way in which humans interpret linguistic values and consequently cannot deal with imprecision and uncertainty. To overcome this limitation, we have suggested the use of fuzzy sets rather than classical interval (or numbers) to represent categorical data

[11,12,13,14,15]. The main motivation of fuzzy sets theory, founded by Zadeh in 1965, is apparently the desire to build a formal quantitative framework that captures the vagueness of humans knowledge since it is expressed via natural language. Consequently, in this work we use fuzzy sets theory to deal with linguistic values in the estimation by analogy procedure.

3. Estimation by analogy

Estimation by analogy is essentially a form of Case-Based Reasoning which has four steps [1]:

1- Retrieve the most similar case or cases
2- Reuse the information and knowledge in that case to solve the problem
3- Revise the proposed solution
4- Retain the parts of this experience likely to be useful for future problem solving

For effort estimation, CBR is based on the following affirmation: *similar software projects have similar costs*. It has been deployed as follows. First, each project must be described by a set of attributes that must be relevant and independent. Second, we must determine the similarity between the candidate project and each project in the historical database. Third, we use the known effort values from the historical projects to derive an estimate for the new project. This later step is known as case adaptation.

Vicinanza et al. have suggested that CBR might be usefully adapted to make accurate software effort predictions [28]. Ever since, estimation by analogy has been the subject of studies aimed at evaluating, enhancing, reformulating and adapting the CBR life cycle according to the features of the software effort prediction problem. Shepperd et al. have been involved in the development of CBR techniques and tools to build software effort prediction systems for the past five years [24, 25]. In their recent work, they tried to explain why different research teams have reported widely different results when using CBR technology. In addition to the characteristics of the projects data being used, Shepperd et al. examined the impact of the choice of the number of analogies and adaptation strategies. They used a dataset of software projects collected by a Canadian software house to validate their findings. They found that choosing the number of analogies is important. Specifically, three analogies seemed to be optimal. However, a fixed value for k (number of analogies) was more effective for the large datasets while distance based analogies selection appeared more effective for the smaller datasets. They also found that case adaptation strategies seemed to have little impact on the accuracy of the estimation by analogy [17].

Angelis and Stamelos [3], while studying the estimation by analogy method for Albrecht's software projects, explored the problem of determining the parameters for configuration of the analogy procedure before its application to a new software project. They studied three parameters, which are the choice of distance measure that will be used to evaluate the similarity between software projects, the number of analogies to take into account in the effort estimation, and the statistic that will be used in calculating the unknown effort from the efforts of the similar projects. They suggested that the bootstrap method should be used to configure these three parameters. Bootstrapping consists of drawing news samples of software projects from the original dataset and testing parameter performances on the generated data. This allows the user to identify which parameter values give accurate estimates often. These values can be used to generate prediction for a new software project. This kind of search for optimal parameters is called calibration of the estimation procedure.

However, even though it is well recognized that estimation by analogy is a promising technique for software cost and effort estimation, there are certain limitations that prevent it from being more popular. The most important is that until now it cannot handle linguistic values such as 'very low', 'low', and 'high' whereas many other factors, such as experience of programmers, complexity of modules and software reliability are measured on an ordinal or nominal scale composed of linguistic values. For example, the well-known COCOMO'81 model has 15 attributes out of 17 (22 out of 24 in the COCOMOII) which are measured with six linguistics values: 'very low', 'low', 'nominal', 'high', 'very high', and 'extra-high' [5, 6, 8]. To overcome this limitation, in the next section we present a new method that can be seen as a fuzzification of the classical analogy to deal with linguistic values.

4. Estimation by Fuzzy Analogy

Fuzzy Analogy is a fuzzification of the classical analogy procedure [15]. It is also composed of three steps: identification of cases, retrieval of similar cases and cases adaptation. Each step is a fuzzification of its equivalent in the classical analogy procedure. In the following sub-sections, each step will be further detailed.

4.1 Identification of cases

The goal of this step is the characterization of all software projects by a set of attributes. Selecting attributes describing accurately software projects is a complex task in the analogy procedure. Indeed, the selection of attributes depends on the objective of the CBR system. In our case, the objective is to estimate the software project effort. Consequently, the attributes must be relevant for the effort estimation task. The problem is to detect the attributes exhibiting a significant relationship with the effort in a given environment. The solution adopted by

cost estimation researchers and practitioners is to test the correlation between the effort and all the attributes for which data in the studied environment are available. This solution does not take into account attributes that can affect largely the effort, if they have not yet recorded data. Another interesting criterion is that each relevant attribute must be independent from the other attributes. In the ANGEL tool, Shepperd et al.?24, 25? propose to resolve the attributes selection problem by applying a brute force search of all possible attributes subsets. They acknowledge that this is an NP-hard search problem and consequently is not a feasible solution when the number of the candidate attributes is large. Briand et al. propose to use a t-test procedure to select the set of attributes [7]. Shepperd et al. claimed that this is not a good solution because this procedure is not efficient to model the potential interactions between the software project attributes [17]. They consider that statistical methods cannot solve the selection problem in the software cost estimation field. There are two other criteria every relevant and independent attribute must obey. They are, the attribute must be comprehensive which implies that it must be well defined and the attribute must be operational which implies that it must be easy to measure. These criteria have yet not been the subject of an in-depth study in the cost estimation literature. It is proposed here to solve the attributes selection problem by integrating a learning procedure in the analogy approach. We discuss in Section 6 how Fuzzy Analogy can satisfy the learning criterion. Before learning, during the training phase of our approach, we adopt a variation of Shepperd's solution by allowing the estimators to use the attributes that they believe best characterize their projects and are more appropriate in their environment.

The objective of our Fuzzy Analogy approach is to deal with linguistic values. In the identification step, each software project is described by a set of selected attributes that can be measured by numerical or linguistic values. These values will be represented by fuzzy sets. In the case of a numerical value x_0 (no uncertainty), its fuzzification will be done by the membership function that takes the value of 1 when x is equal to x_0 and 0 otherwise. Let us suppose for linguistic values that we have M attributes and for each attribute V_j, a measure with linguistic values is defined (A_k^j). Each linguistic value, A_k^j, is represented by a fuzzy set with a membership function ($?_{A_k^j}$). It is preferable that these fuzzy sets satisfy the *normal condition*, i.e., they form a fuzzy partition and each of them is convex and normal [13]. The use of fuzzy sets to represent categorical data, such as 'very low' and 'low' mimics the way in which humans interpret these values and consequently it allows us to deal with vagueness, imprecision and uncertainty in the cases identification step. Another advantage of our Fuzzy Analogy approach is that it takes into account the importance of each selected attribute in the cases identification step. Since all selected attributes do not necessarily have the same influence on the software project effort. Hence, we are required to indicate the weights, u_k, associated with all selected attributes in the cases identification step.

4.2 Retrieval of similar cases

This step is based on the choice of a software project similarity measure. This choice is very important since it will influence which analogies are found. In literature, most researchers have used the Euclidean distance when projects are described by numerical data and the equality distance when they are described by linguistic values (categorical data) [24]. These two measures are not suitable when linguistic values are represented by fuzzy sets. Consequently, we have proposed a set of candidate measures for software project similarity to avoid this limitation [12]. These measures evaluate the overall similarity of two projects P_1 and P_2, $d(P_1,P_2)$, by combining the individual similarities of P_1 and P_2 associated with the various linguistic variables (attributes) (V_j) describing P_1 and P_2, $d_{v_j}(P_1,P_2)$. After an axiomatic validation of some proposed candidate measures for the individual distances $d_{v_j}(P_1,P_2)$, we have retained two measures [13]:

$$d_{v_j}(P_1,P_2)\ ?\ \begin{cases} ?\ \max_k \min(\mu_{A_k^j}(P_1),\mu_{A_k^j}(P_2)) & ?\ (1.1) \\ ?\ \max ?\ \min\ aggregation & ? \\ ?\ \sum_k \mu_{A_k^j}(P_1)\ ?\ \mu_{A_k^j}(P_2) & ?\ (1.2) \\ ?\ sum\ ?\ product\ aggregation & ? \end{cases}$$

where V_j are the linguistic variables describing projects P_1 and P_2, A_k^j are the fuzzy sets associated with V_j, and $?_{A_k^j}$ are the membership functions representing fuzzy sets A_k^j.

To evaluate the overall distance of P_1 and P_2, the individual distances $d_{v_j}(P_1,P_2)$ are aggregated using Regular Increasing Monotone (RIM) linguistic quantifiers such as *'all'*, *'most'*, *'many'*, *'at most ?'* or *'there exists'*. The choice of the appropriate RIM linguistic quantifier, Q, depends on the characteristics and the needs of each environment. Q indicates the proportion of individual distances that we feel is necessary for a good evaluation of the overall distance. The overall similarity of P_1 and P_2, $d(P_1, P_2)$ is given by one of the following formulas [14]:

$$d(P_1,P_2) = \begin{cases} all\ of\ (d_{v_j}(P_1,P_2)) \\ most\ of\ (d_{v_j}(P_1,P_2)) \\ many\ of\ (d_{v_j}(P_1,P_2)) \\ \ldots \\ there\ exists\ of\ (d_{v_j}(P_1,P_2)) \end{cases}$$

When choosing the appropriate RIM linguistic quantifier to guide the aggregation of the individual distances, its implementation is realized by an Ordered Weight Averaging operator [14, 29, 30]. The overall distance, $d(P_1, P_2)$, is calculated by means of the following formula:

$$d(P_1,P_2) = \sum_{j=1}^{M} \left(Q\left(\frac{\sum_{k=1}^{j} u_k}{T}\right) - Q\left(\frac{\sum_{k=1}^{j-1} u_k}{T}\right)\right) \cdot d_{v_j}(P_1,P_2) \qquad (2)$$

where $d_{v_j}(P_1,P_2)$ is the jth largest individual distance, u_k is the importance weight associated with the kth variable describing the software project, and T is the total sum of all importance weights u_k which are provided in the cases identification step.

4.3 Case adaptation

The objective of this step is to derive an estimate for the new project by using the known effort values of similar projects. There are two problems here. First, the choice of how many similar projects should be used in the adaptation. Second, how to adapt the chosen analogies in order to generate an estimate for the new project. In the literature, one can notice that there is no clear rule to guide the choice of the number of analogies, k. Shepperd et al. have tested two strategies to calculate the number k, by setting it to a constant value (they explore values between 1 and 5), or by determining it dynamically as the number of projects that fall within distance (d) of the new project [17]. Briand et al. have used a single analogy [7]. Angelis and Stamelos have tested a number of analogies in the range of 1 to 10 when studying the calibration of the analogy procedure for the Albrecht's dataset [3]. The results obtained from these experimentations seemed to favour the case where k is lower than 3. Fixing the number of analogies to be considered in the case adaptation step will not be considered here neither as a requirement nor as a constraint. The principle of this approach is to take only the first k projects that are similar to the new project. Let us suppose that the distances between the first three projects of one dataset (P_1, P_2, P_3) and the new project (P) are respectively: 3.30, 4.00 and 4.01. When we consider k equal to 2, we use only the two projects P_1 and P_2 in the calculation of an estimate of P. Project P_3 is not considered in this case although there is no clear difference between $d(P_2, P)=4.00$, and $d(P_3, P)=4.01$! We believe that the use of the number k relies on the use of the classical logic principle: the transition from one situation (contribution in the estimated cost) to the following (no contribution in the estimated cost) is abrupt rather than gradual. In Fuzzy Analogy, we propose a new strategy to select projects that will be used in the adaptation step. This strategy is based on the distances $d(P, P_i)$ and the definition adopted in the studied environment for the proposition 'P_i is closely similar project to P'. Intuitively, P_i is closely similar to P if $d(P,P_i)$ is in the vicinity of 1 (0 in the case of Euclidean distance). A better way to represent the value 'vicinity of 1' is by using a fuzzy set defined in the unit interval [0, 1]. Indeed, this fuzzy set defines the 'closely similar' qualification adopted in the environment. Figure 1 shows a possible representation for the value 'vicinity of 1'. In this example all projects that have $d(P,P_i)$ higher than 0.5 contribute to the estimated cost of P; the contribution of each P_i is weighted by $\mu_{\text{vicinity of 1}}(d(P,P_i))$.

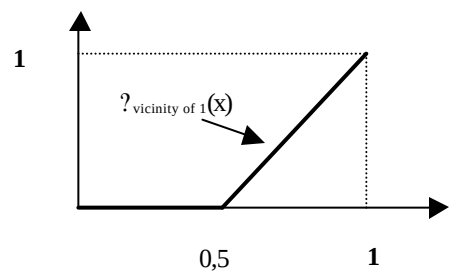

Figure 1: A possible definition of the value 'vicinity of 1'

The second problem in this step is to adapt the chosen analogies in order to generate an estimate for the new project. The most common solutions use the (weighted) mean or the median of the k chosen analogies. In the case of weighted mean, the weights can be the similarity distances or the ranks of the projects. For our Fuzzy Analogy approach, we use the weighted mean of all known effort projects in the dataset. The weights are the values of the membership function defining the fuzzy set 'vicinity of 1'. The formula is then:

$$Effort(P) = \frac{\sum_{i=1}^{N} \mu_{\text{vicinity of 1}}(d(P,P_i)) \cdot Effort(P_i)}{\sum_{i=1}^{N} \mu_{\text{vicinity of 1}}(d(P,P_i))} \qquad (3)$$

The main advantage of our adaptation approach is that it can be easily configured by defining the value 'vicinity of 1' according to the needs of each environment. An

interesting case arises when $?_{vicinityof1}(x) = x$ in Formula 3 since it gives exactly the ordinary weighted average. This property will be used in the validation of our approach on the COCOMO'81 dataset.

5. Empirical results

The following section presents and discusses the results obtained when applying the Fuzzy Analogy approach on the COCOMO'81 dataset. The calculations are made by using the F_ANGEL tool. F_ANGEL is a software prototype that we have developed to automate the Fuzzy Analogy approach. It can be seen as a fuzzification of the software ANGEL developed by Shepperd et al. for the classical analogy procedure. The results were compared with those of three other models: classical analogy, the original intermediate COCOMO'81, and 'fuzzy' in

termediate COCOMO'81 [5, 11]. The accuracy of the estimates is evaluated by using the magnitude of relative error (MRE) defined as:

$$MRE \; ? \left| \frac{Effort_{actual} \; ? \; Effort_{estimated}}{Effort_{actual}} \right|$$

The *MRE* is calculated for each project in the dataset. In addition, we use the measure prediction level *Pred*. This measure is often used in the literature. It is defined by:

$$\Pr ed(p) \; ? \; \frac{k}{N}$$

where N is the total number of observations, k is the number of observations with an *MRE* less than or equal to p. A common value for p is 0.25; in our evaluation, we use p equal to 0.20 as it was used for evaluation of the original version of the intermediate COCOMO'81 model. The *Pred*(0.20) gives the percentage of projects that were predicted with an *MRE* equal or less than 0.20. Other four quantities are used in this evaluation: min of *MRE*, max of *MRE*, standard deviation of *MRE* (SDMRE), and mean *MRE* (MMRE).

The original intermediate COCOMO'81 database was chosen as the basis for this validation [5]. It contains 63 software projects. Each project is described by 17 attributes: the software size measured in KDSI (Kilo Delivered Source Instructions), the project mode is defined as either 'organic', 'semi-detached' or 'embedded', and the remaining 15 cost drivers are generally related to the software environment. Each cost driver is measured on a scale composed of six qualifications: 'very low', 'low', 'nominal', 'high', 'very high' and 'extra high'. It seems that this scale is ordinal, but an analysis indicates that one of the 15 cost drivers (SCED attribute) is only assessed to be nominal. This does not cause any problem for the Fuzzy Analogy technique. Indeed, we are dealing with these six qualifications as linguistic values rather than categorical data. In the original intermediate COCOMO'81, the assignment of linguistic values to the 15 cost drivers uses conventional quantization where the values are classical intervals (see [5], pp. 119). Because of the advantages of representation by fuzzy sets over classical intervals, the 15 cost drivers should be represented by fuzzy sets. Among these, we have retained 12 attributes that we had already fuzzified. The other attributes are not studied because these relative descriptions proved insufficient [11]. In this evaluation, we assume that all the COCOMO'81 software projects are described only by these 12 cost drivers (Figure 2).

Because the original COCOMO'81 database contains only the effort multipliers, our evaluation of the Fuzzy Analogy will be made on four 'fuzzy' datasets deduced from the original COCOMO'81 database. Each one of these four 'fuzzy' datasets contains 63 projects with the real values necessary to determine the 12 linguistic values associated to each project. These 12 linguistic values are used to evaluate the similarity between software projects. One of these four fuzzy datasets is considered as an historical dataset, the other three are the current datasets containing the new projects.

Table 1 shows the results obtained using only the max-min aggregation to evaluate the individual distances (Formula (1.1)). We have not used sum-product aggregation (Formula (1.2)) for two reasons [13]:

- Under what we have called *normal condition*, the max-min and sum-product aggregations give approximately the same results. This is the case for the COCOMO'81 database.
- The sum-product aggregation does not respect all established axioms.

Usually for the overall distances, each environment must define its appropriate quantifier by studying its features and its requirements. Because a lack of knowledge concerning the appropriate quantifier for the environment from which the COCOMO'81 data was collected, we used various quantifiers: 'all', 'there exists', and ?-RIM linguistic quantifiers to combine the individual similarities. An ?-RIM linguistic quantifier is defined by a fuzzy set in the unit interval with membership function Q given by:

$$Q(r) \; ? \; r^{?} \quad ? \; ? \; 0$$

To calculate the weights w_j's (Formula 2), we must determine the importance weights u_k's associated with the 12 variables describing COCOMO'81 software projects. We use here the productivity ratio, which is the project's productivity ratio (expressed in Delivered Source Instructions by Man-Month) for the best possible variable rating to its worst possible variable rating, assuming that

the ratings for all other variables remain constant (Figure 2).

By analyzing the results of the validation of the Fuzzy Analogy technique (Table 1), we noticed that the accuracy of the estimates depends on the linguistic quantifier (*?*) used in the evaluation of the overall similarity between software projects. So, if we consider the accuracy measured by *Pred*(0.20) as a function of ?, we can notice that, in general, it is monotonous increasing according to ?. This is due to the fact that our similarity measures are monotonous decreasing according to ?. Indeed, when ? tends towards zero, this implies that the overall similarity will take into account fewer attributes among all those describing software projects. The minimum number of attributes to consider is one. This is the case when using the 'max' operator where the selected attribute is the one for which the associated individual distance is the maximum of all individual distances.

Table 1. Results of the evaluation of Fuzzy Analogy

		Datasets								
		Dataset #1			Dataset #2			Dataset #3		
		***Pred(0.20)* (%0)**	***MMRE* (%)**	***SDMRE* (%)**	***Pred(0.20)* (%)**	***MMRE* (%)**	***SDMRE* (%)**	***Pred(0.20)* (%)**	***MMRE* (%)**	***SDMRE* (%)**
? **-RIM**	**Max**	4,76	1801,48	2902,94	4,76	2902,49	1807,17	4,76	1789,64	2865,42
	1/100	4,76	1798,85	2897,77	4,76	2894,28	1803,41	4,76	1786,20	2858,34
	1/30	4,76	1792,70	2885,74	4,76	2875,26	1794,69	4,76	1778,21	2841,96
	1/15	4,76	1783,91	2868,69	4,76	2848,44	1782,32	4,76	1766,86	2818,84
	1/10	4,76	1757,13	2851,77	4,76	2822,64	1770,06	4,76	1755,64	2796,06
	1/7	4,76	1763,86	2830,24	4,76	2788,70	1754,48	4,76	1741,29	2767,29
	1/3	6,34	1714,20	2737,66	6,34	2648,90	1687,68	6,34	1679,67	2646,43
	1	6,34	1550,89	2455,21	3,17	2258,36	1485,66	7,93	1491,93	2306,39
	3	6,34	1168,24	1889,23	9,52	1571,48	1063,19	9,52	1102,40	1694,91
	7	9,52	633,99	1215,81	14,28	830,79	526,57	9,52	603,76	968,01
	10	15,87	371.84	802,30	20,63	525,45	305,98	23,80	385,90	662,02
	15	38,09	143,92	337,76	36,50	284,20	140,68	44,44	206,19	393.30
	30	74,60	20,40	42,06	77,77	160,38	51,67	66,66	62,83	118,44
	100	92,06	4,06	9,05	84,12	30,37	10,49	87,63	23,24	76,68
	Min	92,06	4,03	9,17	87,30	29,12	8,53	88,88	21,99	77,13

Table 2. Results of the evaluation of classical analogy, 'fuzzy' and classical intermediate COCOMO'81[11]

	'fuzzy'/classical intermediate COCOMO'81						**Classical Analogy (three datasets)**	
	Dataset #1		**Dataset #2**		**Dataset #3**		**K**	**Pred(0.20) %**
Pred(20) (%)	62.14	68	46.86	68	41.27	68	**2**	31,75
Min MRE (%)	0.11	0.02	0.40	0.02	0.06	0.02	**3**	25,40
Max MRE (%)	88.60	83.58	3233.03	83.58	88.03	83.58	**4**	19,05
Mean MRE_i(%)	22.50	18.52	78.45	18.52	30.80	18.52	**5**	12,70
Standard deviation MRE	19.69	16.97	404.40	16.97	22.95	16.97	**6**	12,70

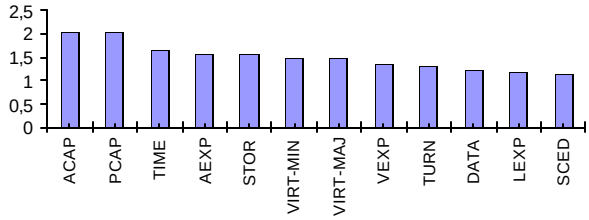

Figure 2: Comparison of the productivity ratios for the 12 variables describing the COCOMO'81 software projects

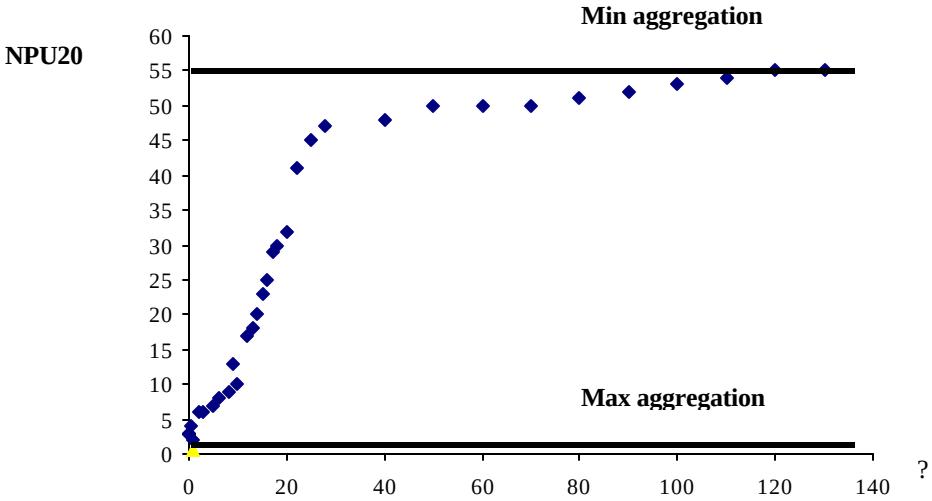

Figure 3: Relationship between ? and the number of dataset #2 projects which have an MRE lower or equal than 0.20 (NPU20)

As a consequence, the overall similarity will be higher because we are more likely to find in the COCOMO'81 dataset at least one attribute for which the associated linguistic values are the same for the two projects. By contrast, when ? tends towards infinity it implies that the overall similarity will take into account many attributes among all the available ones describing the software projects. As a maximum we may consider all attributes. This is the case when combining by the 'min' operator. As consequence the overall similarity will be minor because we are more likely to find in the COCOMO'81 dataset one attribute for which the associated linguistic values are different for the two projects. Here, it is very important to stress the soft aspect of our Fuzzy Analogy approach. First, we can choose the appropriate weights associated with linguistic variables describing a software project (u_k). These weights represent the importance of the variables in the environment. Second, we can choose the appropriate linguistic quantifier to combine the individual distances; this linguistic quantifier is used to generate the weights w_j. These weights represent the importance associated with the individual distances when evaluating the overall distance. They depend upon the weights u_k and the chosen linguistic quantifier. An interesting case arises if u_k is equal to w_k. This is when ? is equal to 1. As a consequence, Formula 2 gives the ordinary weighted average.

Figure 3 shows the relation between ? and the number of projects that have an *MRE* smaller than 0.20 (NPU20) for dataset #2. The two bold lines represent respectively the minimum and the maximum accuracy of Fuzzy Analogy when it uses the min and the max aggregation to combine individual similarities. The 'max' ('min') aggregation gives lower (higher) accuracy because it considers only one (all) attribute(s) in the evaluation of the similarity. For the other ?-RIM linguistic quantifiers

(0<? <?), the accuracy increases with ? because additional attributes will be considered in the evaluation of the overall similarity. For example, a software project P_i which has an overall similarity with P different from zero when ? is equal to 10, may have a null overall similarity when ? is equal to 30. Because of that, it is not used in the estimation of the cost for ?=30. When ? tends towards infinity (this implies that most attributes are considered in the evaluation of the similarity), only projects P_i which are closely similar to P will contribute in the estimate of the cost of P. This is in conformity with common knowledge in the cost estimation field: *evaluation of the similarity between projects is meaningful if they are described by a sufficient number of attributes*. As we can see in Figure 3, the accuracy is a monotonous increasing function of ?. However, because Formula 3 used in the adaptation step is not monotonous increasing, we may observe certain anomalies that can lead to misinterpretations of the results. This is the case when ? is equal to 1 in dataset #2. It seems that the accuracy when ? =1/3 (MMRE=6,34, NPU20=4) is better than that when ?=1 (MMRE=3,17, NPU20=2). Indeed, the two additional projects, which have an *MRE* lower than 0,20 (18,74 and 17,68) for ?=1/3, have respectively an *MRE* equal to 21,68 and 20,24 when ?=1. So, when we have fixed *Pred*(p) at *Pred*(0,20), these two projects are not counted. This should not give the impression that the case for ?=1/3 generates more accurate estimates than the case for ?=1. By analysing the results for all projects, we have found the opposite (see the mean and the standard deviation of the *MRE* for ?=1/3 and ?=1).

We compared the results of the Fuzzy Analogy with the other three techniques in regards to two criteria: the type of the technique and whether or not the technique uses fuzzy logic in its estimation process. Our findings were the following:

✍Fuzzy analogy performs better than the classical analogy in all three datasets when ? is higher than a given value. In the classical analogy, we have used the classical equality distance (equal or not) in the evaluation of similarity between projects. All attributes are considered in this evaluation. The best accuracy was obtained when we consider only the two first projects in the adaptation step (*Pred*(0,20)=31,75). The Fuzzy Analogy when using the 'min' aggregation also took into account all attributes in the evaluation of projects similarity. Its accuracy was much higher than that for classical analogy. Two advantages were found when using fuzzy logic with the estimation by analogy. First, it tolerates imprecision and uncertainty in its inputs (cost drivers) and consequently it generates gradual outputs (cost). This is why Fuzzy Analogy gives closer results for the three datasets while classical analogy generates the same or significantly different outputs when the inputs are different (this is the same case between 'fuzzy' and classical intermediate COCOMO'81, see [11] for more details, Table 2). Second, it improves the accuracy of the estimates because our similarity measures are more appropriate than those used in the literature.

✍Intermediate COCOMO'81 generates more accurate results than classical analogy but when integrating fuzzy logic in the estimation by analogy procedure, the Fuzzy Analogy performs better than intermediate COCOMO'81. This illustrates that fuzzy logic is an appropriate tool to deal with linguistic values rather than the classical logic (Aristote logic) used in the original version of the COCOMO'81.

Taking into account these results, we suggest the following ranking of the four techniques in terms of accuracy and adequacy to deal with linguistic values :

1- Fuzzy Analogy
2- Fuzzy intermediate COCOMO'81
3- Classical intermediate COCOMO'81
4- Classical analogy.

6. Discussion and future improvements

In this paper, we have proposed a new approach to estimate the software project effort. This approach is based on reasoning by analogy, fuzzy logic and linguistic quantifiers. Such an approach can be used when the software projects are described by linguistic and/or numerical values. Thus, our approach improves the classical analogy procedure that does not take into account linguistic values. In the Fuzzy Analogy approach, both linguistic and numerical data are represented by fuzzy sets. The advantage of this is to handle correctly the imprecision and the uncertainty when describing a software project. Also, by using RIM linguistic quantifier to guide the aggregation of the individual similarities between two projects, the Fuzzy Analogy approach can be easily adapted and configured according to the needs specifications of each environment. Also, we have validated the Fuzzy Analogy by using the COCOMO'81 dataset. The results of this validation were compared to those of the classical analogy approach, 'fuzzy' intermediate COCOMO'81 and original intermediate COCOMO'81. We can conclude that fuzzy logic improves the estimation process and consequently generates more accurate estimates.

By using fuzzy logic in its estimation process, this approach satisfies the first criterion of the concept Soft Computing, which is the tolerance of imprecision. As defined by Zadeh [32], Soft Computing is composed of three criteria part of human nature: tolerance of imprecision, learning, and uncertainty.

In this work, we have introduced some learning functionalities in our approach. In the identification step, we can update all information concerning the linguistic

variables describing software projects, specifically, their linguistic values which depend on humans judgement. For example, the linguistic value 'high' for software reliability may mean that the number of software failures is lower than 6 per month, but in the future, we may require less than 3 software failures per month to evaluate it as 'high'. In the case retrieval step, we can update the definition of the linguistic quantifier used in the environment. Here also, the meaning of a linguistic quantifier depends on human judgement. However, other learning characteristics that are not included in our approach remain. For example, the Fuzzy Analogy must be able to provide its user with a subset of linguistic variables that have always led to accurate estimates in the past. We may then use this sub-set in the identification step. Thus, the selection attributes problem can be solved. Also, Fuzzy Analogy must be able to propose the appropriate linguistic quantifier to be used in retrieval step by using those that have often led to accurate estimates.

In order to satisfy the third criteria of Soft Computing, Fuzzy Analogy must be able to handle the uncertainty when estimating the cost of the new project. Estimate uncertainty occurs because an estimate is a probabilistic assessment of a future condition. Kitchenham and Linkman have examined four sources of estimate uncertainty: model error, measurement error, assumption error and scope error [16]. In our case, we are concerned by the first source of uncertainty. Fuzzy Analogy is based on the affirmation: *'similar projects have similar costs'*. There are two sources of uncertainty in this affirmation. First, the consequence of this affirmation is imprecise. Second, the affirmation *'similar projects have similar costs'* is not always deterministic. We can find in some applications of CBR cases that are similar but the outcomes are completely different. It seems that it can be the case in the cost estimation field. Indeed, no cost estimation model can include all the factors that affect the cost required to develop the software. So, the factors that affect cost and are not included explicitly in the evaluation of the similarity between projects contribute to the uncertainty in the predicted cost. In order to take into account the uncertainty of the classical affirmation of CBR, we may replace it by the following *'similar projects have **possibly** similar costs'*. Further research work has been initiated to look at the use of this affirmation as the basis of an improvement of our approach.

7. References

[1] A. Aamodt, E. Plaza, "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches", AI Communications, IOS Press, Vol. 7:1. 1994, pp. 39-59

[2] A. Abran, P.N. Robillard, "Functions points analysis: an empirical study of its measurement processes", IEEE Trans. On Software Enginnering, 22(12): 1996, pp. 895-909

[3] L. Angelis, I. Stamelos, "A Simulation Tool For Efficient Analogy Based Cost Estimation", Empirical Software Engineering, Vol. 5, no. 1, 2000, pp. 35-68

[4] L. Angelis, I. Stamelos, M. Morisio, "Building a Software Cost Estimation Model Based on Categorical Data", 7th Int, Soft. Metrics Symp., IEEE Computer Society, London, April, 2001, pp. 4-15.

[5] B.W. Boehm, *Software Engineering Economics*, Prentice-Hall, 1981.

[6] B.W. Boehm, and *al.*, "Cost Models for Future Software Life Cycle Processes: COCOMO 2.0", *Annals of Software Engineering on Software Process and Product Measurement*, Amsterdam, 1995.

[7] L. Briand, T. Langley, I. Wieczorek., "Using the European Space Agency data set: A replicated assessment and comparison of common software cost modeling", In Proc 22th IEEE International Conference on Software engineering, Limerik, Ireland, 2000, pp. 377-386

[8] D.S. Chulani, "Incorporating Bayesian Analysis to Improve the Accuracy of COCOMO II and Its Quality Model Extension", Ph.D. Qualifying Exam Report, USC, February, 1998.

[9] N. Fenton, and S.L. Pfleeger, *Software metrics: A Rigorous and Practical Approach*, International Computer, Thomson Press, 1997.

[10] Gulezian R., 'Reformulating and Calibrating COCOMO' Journal Systems Software, Vol 16, 1991, pp.235-242

[11] A. Idri, L. Kjiri, and A. Abran, "COCOMO Cost Model Using Fuzzy Logic", *7th International Conference on Fuzzy Theory & Technology*, Atlantic City, NJ, February, 2000. pp. 219-223

[12] A. Idri, and A. Abran, "Towards A Fuzzy Logic Based Measures For Software Project Similarity", *Sixth Maghrebian Conference on Computer Sciences, Fes*, Morroco, November, 2000. pp. 9-18

[13] A. Idri, and A. Abran, "A Fuzzy Logic Based Measures For Software Project Similarity: Validation and Possible Improvements", 7th International Symposium on Software Metrics, *IEEE Computer Society*,4-6 April, England, 2001. pp. 85-96

[14] A. Idri, and A. Abran, "Evaluating Software Projects Similarity by using Linguistic Quantifier Guided Aggregations", 9th IFSA World Congress/20th NAFIPS International Conference, 25-28 July, Vancouver, 2001. pp. 416-421

[15] A. Idri, A. Abran, T. M. Khoshgoftaar, "Fuzzy Analogy: A new Approach for Software Cost Estimation", 11th International Workshop in Software Measurements, 28-29 August, Montreal, 2001, pp. 93-101

[16] B. Kitchenham, S. Linkman, "Estimates, uncertainty and risks", IEEE Software, 14(3), 1997, pp. 69-74

[17] G. Kadoda. M. Cartwright, L. Chen, M. Shepperd, "Experiences Using Case-Based Reasoning to Predict Software Project Effort", EASE, Keele, UK, 2000, p.23

[18] J.L. Kolodner, Case-Based Reasoning, Morgan Kaufmann, 1993

[19] D. H. Krantz, R. D. Luce, P. Suppes, A. Tversky, "Foundations of Measurement: Additive and Polynomial Representations ", Academic Press, Volume 1, 1971

[20] Matson J., E, Barrett B., E, Mellichamp J., M, 'Software Development Cost Estimation Using Function Points', IEEE, Vol. 20, No. 4, Apr., 1994, pp. 275-287

[21] I. Myrtveit, E. Stensrud, "A Controlled Experiment to Assess the Benefits of Estimating with Analogy and Regression Models", IEEE Transaction on Software Engineering, 25, 4, July/August, 1999, pp. 510-525.

[22] F. Niessink, H. Van Vliet "Predicting Maintenance Effort with Function Points", in Proc Inter. Conf. On Soft. Maintenance, Bari, Italy, IEEE Computer Society, 1997.

[23] Putnam L. H, ' A General Empirical Solution to the Macro Software Sizing and Estmation Problem', IEEE Tronsactions on Soft. Eng., Vol. SE-4, No. 4, July, 1978.

[24] M. Shepperd, C. Schofield, and B. Kitchenham, "Effort Estimation using Analogy", ICSE-18, Berlin, 1996, pp. 170-178

[25] M. Shepperd, and C. Schofield, "Estimating Software Project Effort Using Analogies", IEEE Trans. on Software Engineering, Vol. 23, no. 12, November, 1997, pp. 736-743

[26] S. Shepperd, G. Kadoda, "Using simulation to evaluate predictions systems", 7th International Symposium on Software Metrics, IEEE Computer Society, 4-6 April, England, 2001. pp. 349-358

[27] S. S. Stevens, "On the Theory of scales and Measurement", Science 103, 1946, pp. 677-680

[28] S. Vicinanza, and M.J. Prietolla, "Case Based Reasoning in Software Effort Estimation", Proceedings 11th Int. Conf. on Information Systems, 1990

[29] R.R. Yager, and J. Kacprzyk, The Ordered Weighted Averaging Operators: Theory and Applications" Kluwer: Norwell, MA, 1997.

[30] R.R. Yager, "Quantifier Guided Aggregation using OWA Operators", International Journal of Intelligent Systems, 11, 1996, pp.49-73

[31] L.A. Zadeh, "Fuzzy Set", Information and Control, Vol. 8, 1965, pp. 338-353

[32] L.A. Zadeh, "Fuzzy Logic, Neural Networks, and Soft Computing", Comm. ACM, Vol. 37, no. 3, March, 1994, pp.77-84

[33] H. Zuse, "A Framework of Software Measurement", de Gruyter, 1998.