Estimate Programmatic Effort using the Traditional COCOMO Model and Neural Networks

Jamal Salah Al-Din Sayed Majeed **Isra Zuhair Majeed Qabaa**

College of Computer Science and Mathematics
University of Mosul, Mosul, Iraq

Received on: 16/10/2012 **Accepted on: 30/01/2013**

ABSTRACT

Estimation models in software engineering are used to predict some important and future features for software project such as effort estimation for developing software projects. Failures of software are mainly due to the faulty project management practices. software project effort estimation is an important step in the process of software management of large projects. Continuous changing in software project makes effort estimation more challenging. The main objective of this paper is find a model to get a more accurate estimation. In this paper we used the Intermediate COCOMO model which is categorized as the best of traditional Techniques in Algorithmic effort estimation methods. also we used an Artificial approaches which is presented in (FFNN,CNN,ENN,RBFN) because of the Ability of ANN(Artificial Neural Network) to model a complex set of relationship between the dependent variable (effort) and the independent variables (cost drivers)which makes it as a potential tool for estimation. This paper presents a performance analysis of ANNs used in effort estimation. We create and simulate this networks by MATLAB11 NNTool depending on NASA aerospace dataset which contains a features of 60 software project and its actual effort. the result of estimation in this paper shows that the neural networks in general enhance the performance of traditional COCOMO and we proved that the ENN was the best network between neural networks and the CNN was the next best network and the COCOMO have the worst between the used methods.

Keywords: Estimation models, COCOMO model, Artificial Neural Network.

تخمين الجهد البرمجي باستخدام نموذج الـ COCOMO التقليدي والشبكات العصبية

جمال صلاح الدين سيد مجيد **إسراء زهير مجيد قبع**

كلية علوم الحاسوب والرياضيات

جامعة الموصل، الموصل، العراق

تاريخ استلام البحث: 2012/10/16 **تاريخ قبول البحث: 2013/01/30**

الملخص

إن نماذج التخمين في هندسة البرمجيات تستخدم لتخمين بعض الخصائص المهمة والمستقبلية للمشروع البرمجي مثل تخمين جهد المشروع المطور، وان الفشل في البرنامج يكون بشكل أساسي بسبب ممارسات إدارة المشروع الخاطئة. فتخمين الجهد البرمجي هو خطوة مهمة جداً في عملية إدارة البرمجيات للمشاريع الكبيرة. ولكن التغييرات المستمرة في المشروع البرمجي جعلت من عملية تخمين الجهد عملية ذات تحدي كبير. إن الهدف الرئيسي من هذا البحث هو تحديد طريقة للحصول على تخمين جهد أكثر دقة حيث تم استخدام نموذج الـ *COCOMO* الوسطي والذي يصنف بكونه من أفضل الطرائق التقليدية بين نماذج تخمين الجهد الحسابية. وكذلك تم استخدام طرق ذكائية والمتمثلة بالشبكات العصبية (*FFNN, CNN, ENN, RBFN*) وذلك لقدرة الشبكات

العصبية الاصطناعية على نمذجة المجاميع المعقدة من العلاقات بين المتغيرات الاعتمادية (الجهد) والمتغيرات غير الاعتمادية (عوامل الكلفة) والتي جعلت منها أداة مرتقبة للتخمين وبهذا فان هذا البحث قدم تحليل لأداء الشبكات العصبية المستخدمة في تخمين الجهد حيث تم تكوين واختبار هذه الشبكات باستخدام أدوات الشبكات العصبية الخاصة بلغة *MATLAB11*. وتم الاعتماد على مجموعة بيانات *NASA* الفضائية والتي تحوي على خواص 60 مشروع برمجي إضافة إلى الجهد الحقيقي لهذه المشاريع. وقد بينت نتائج التخمين في هذا البحث إن الشبكات العصبية بصورة عامة قد حسنت من أداء الطريقة التقليدية *COCOMO* وقد تم برهنة شبكة *ENN* على أنها أفضل شبكة بين الشبكات العصبية وتليها شبكة *CNN* وكانت نتائج طريقة الـ *COCOMO* الوسطي هي الأسوأ بين الطرق المستخدمة.

الكلمات المفتاحية: نماذج التخمين، نموذج الـ COCOMO، والشبكات العصبية.

1- مقدمة

تخمين الكلفة البرمجية الدقيق هو مهمة حرجة للمطورين والزبائن على حد سواء فبمعرفة الكلفة المتوقعة للمشروع فان فريق إدارة البرمجيات سيتمكن من السيطرة على عملية التطوير البرمجي باستخدام أسلوب كفوء. حيث التخمين الدقيق لحجم، كلفة، جهد، والجدول الزمني للبرمجيات هو في الغالب التحدي الأكبر الذي يواجه مطوري البرمجيات هذه الأيام. حيث له تأثير أساسي على إدارة تطوير البرمجيات[1]. والفكرة الأساسية للتخمين البرمجي تشير إلى أن التطوير البرمجي هو عملية تصفية تدريجية تبدأ بصورة مضببة عن ما نريد بناءه وبعدها يتم قضاء باقي المشروع في محاولة نقل هذه الصورة إلى تركيز أوضح [2].

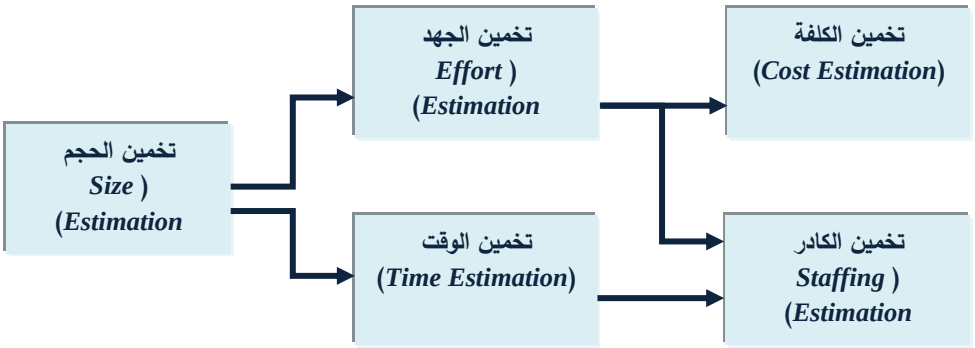
يمكن تصنيف التخمين البرمجي إلى ثلاث مراحل:

المرحلة الأولى: تشمل تخمين الحجم.

المرحلة الثانية: تتمثل في تخمين الجهد وتخمين الوقت.

المرحلة الثالثة: تتمثل في تخمين الكلفة وتخمين عدد الموظفين اللازمين.

والشكل(1) يبين التداخل بين المراحل الثلاثة في عملية التخمين البرمجي النموذجية في دورة حياة تطوير البرمجيات [3].



الشكل (1). يبين تسلسل عمليات التخمين في دورة حياة تطوير البرمجيات

يعتبر الجهد (*Effort*) المصدر الاستهلاكي الرئيسي في المشروع البرمجي، ففي مشروع تطوير البرمجيات كل تخمينات الجهد والجدولة تعتبر كشروط مسبقة لتخطيط المشروع، فتخمين الجهد هو أمر حيوي لمشاريع البرامج الناجحة [4] [9].

وعليه يمكن تحديد ثلاث خطوات رئيسية في عملية التخمين للبرنامج الذي يكون تحت التطوير [2].

- تخمين الحجم.
- تخمين الجهد.
- تخمين الجدولة.

2- الأعمال السابقة

في العام 2002 قام الباحثون *Taghi M. Khoshgoftaar* و*Alain Abran* و*Ali Idri* بتصميم أسلوب جديد بالاعتماد على المنطق المضبب والمحددات (*fuzzy logic and quantifiers*) والاستدلال اللغوي باستخدام أسلوب المقارنة (*linguistic reasoning by analogy*) لتخمين جهد المشروع البرمجي سواء تم وصفه بالقيم العددية أو القيم اللغوية وأطلقوا على هذا الأسلوب اسم المقارنة المضببة (*Fuzzy Analogy*) وهذا البحث يتضمن أيضا تجارب فعالة لتطبيق الأنموذج المقترح على الـ *COCOMO'81 dataset* إذ لوحظ أن الطريقة المتقرحة قد حسنت عملية التخمين وحصلت على نتائج أفضل من الطرائق التقليدية وأكثر دقة [5].

وفي عام 2008 قام الباحثون *Parvinder S. Sandhu* و*Porush Bassi* و *Amanpreet Singh Brar* باستخدام أسلوب الغموض العصبي *Neuro-Fuzzy* وتطبيقه على قاعدة بيانات *NASA* وعمل مقارنة بين أدائه وأداء النماذج *the Halstead, Walston-Felix, Bailey-Basili and Doty* وأظهرت النتائج أن الـ *Neuro-Fuzzy* قد حقق اقل *MMRE* و*RMSSE* ويليها أنموذج *Bailey-Basili*. وبهذا أثبت الباحث بأنه الـ *Neuro-Fuzzy* ممكن أن يستخدم في عملية تخمين الجهد ويناسب كل أنواع المشاريع [6].

وفي عام 2010 قام الباحثون *Jaswinder Kaur* و*Satwinder Singh* و *Dr. Karanjeet Singh Kahlon* و*Pourush Bassi* بعمل مقارنة بين الشبكات العصبية ذات الانتشار الخلفي *Backpropagation Neural Network* والطرق التالية *Halstead, Walston-Felix, Bailey-Basili and Doty* وبالاعتماد على مجموعة بيانات *NASA* لغرض تخمين الجهد والنتائج أظهرت بأنه الشبكة العصبية هي الأفضل في عملية التخمين وذلك بعد حساب قيمتي الـ *MMRE* والـ *RMSSE* والطريقة التي تليها هي الـ *Bailey-Basili* [7].

وفي عام 2011 قام الباحثون *Rama Sree P* و*Sudha K. R* و *Prasad Reddy P.V.G.D* باستخدام المنطق المضبب *Fuzzy Logic (FL)* وبينوا بأنه استخدام دالة العضوية الثلاثية (*triangular membership function*) في المنطق المضبب لغرض عمل تخمين للجهد هو أفضل من استخدام دالة العضوية العامة بيل (*Generalized Bell Membership Function (GBellMF)*) ، وفقط بتعديل القيم للمعاملات المستخدمة في أنظمة الاستدلال المضبب *FIS* ممكن أن نحصل على تخمين جيد للجهد [8].

3- نماذج تخمين الجهد Effort Estimation Methods

إن نماذج الكلفة تصنف إلى نماذج حسابية *Algorithmic Models* ونماذج غير حسابية *Non-Algorithmic Models* إذ تحوي النماذج غير الحسابية على الطرائق الآتية: حكم الخبراء (*Expert Judgment*)، المقارنة (*Analogy*)، نموذج باركنسون (*Parkinson*)، التسعير من أجل كسب المشروع (*Price to Win*)، من أعلى إلى أسفل (*Top-Down*) وأخيراً من أسفل إلى أعلى (*Bottom-Up*) [10].

4- الـ COCOMO الوسطي Intermediate COCOMO(Constructive Cost Model)

إن الـ *COCOMO* الوسطي يقوم بتعديل المعادلة الأساسية لـ *COCOMO* الأساسي وهي:

$$Effort = a * KLOC^{b} \quad \ldots(1)$$

حيث أن: *KLOC (Kilo Lines Of Code)* هي قيمة تمثل عدد الأسطر البرمجية للمشروع مقاساً بالآلاف.

وهذا يتم بواسطة استخدام عوامل الكلفة. وهذه العوامل يجب أن تُحسب لخواص مشروع معين لجعله يحيد عن إنتاجية المشاريع ذات القيمة المتوسطة (*nominal*). فهي مبنية على 15 عامل كلفة (*15 cost-drivers*) وكل عامل له تأثير معين وهذا التأثير سيأخذ قيمة معينة تسمى مضاعف الجهد (*Effort Multiplier(EM)*) والتي بدورها إما تزيد أو تقلل جهد المشروع. المعادلة الأساسية لنموذج الـ *COCOMO* الوسطي تأخذ الصيغة التالية : [11]

$$Effort = a * (KLOC)^{b} * \prod_{i=1}^{15} EM_i \quad \ldots(2)$$

حيث إن الـ *EMi*: القيمة العددية لعامل الكلفة i.

$\prod_{i=1}^{15} EM_i$ هو عبارة عن مضروب قيم عوامل الكلفة الخمسة عشر لمشروع معين والناتج هو عبارة عن قيمة تسمى عامل تعديل الجهد (*Effort Adjustment Factor EAF*) .

وقيم $\{a,b\}$ تتغير حسب نمط المشروع حيث انه هناك ثلاثة أنماط للمشروع (عضوي (*Organic*)، شبه منفصل (*Semi-detached*) أو ضمني (*Embedded*)) والجدول (1) يبين هذه القيم:

الجدول (1). قيم الـ a والـ b الخاصة بالـ *COCOMO* الوسطي

نمط المشروع	a	b
عضوي (*Organic*)	**3.2**	**1.05**
شبه منفصل (*Semi-detached*)	**3.0**	**1.12**
ضمني (*Embedded*)	**2.8**	**1.20**

وعوامل الكلفة التابعة للمشروع البرمجي والخاصة بهذا الأنموذج ومضاعفات الجهد الخاصة بها موضحة في الجدول (2):

الجدول (2). مضاعفات الجهد (*Effort Multiplier*) الخاصة بتصنيفات عوامل الكلفة

التصنيف / عوامل الكلفة	*Very low*	*low*	*nominal*	*high*	*Very high*	*Extra high*
RELY	0.75	0.88	1.00	1.15	1.40	
DATA		0.94	1.00	1.08	1.16	
CPLX	0.70	0.85	1.00	1.15	1.30	1.65
TIME			1.00	1.11	1.30	1.66
STOR			1.00	1.06	1.21	1.56
VIRT		0.87	1.00	1.15	1.30	
TURN		0.87	1.00	1.07	1.15	
ACAP	1.46	1.19	1.00	0.86	0.71	
AEXP	1.29	1.13	1.00	0.91	0.82	
PCAP	1.42	1.17	1.00	0.86	0.70	
VEXP	1.21	1.10	1.00	0.90		

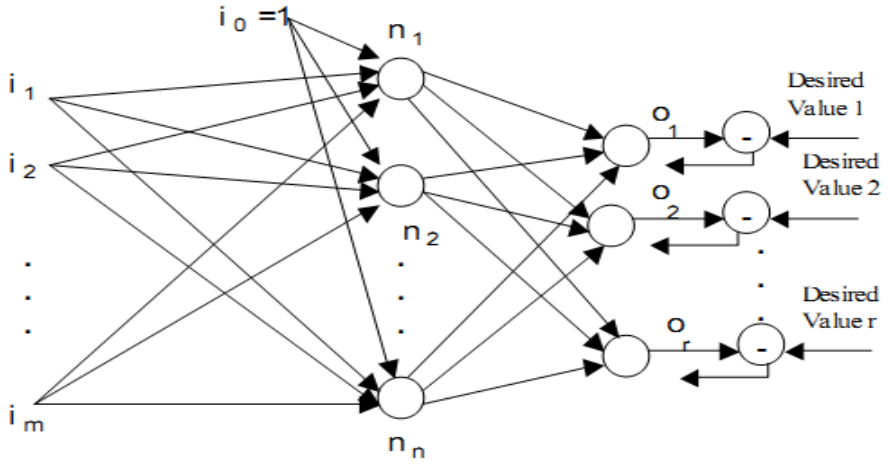
		0.95	1.00	1.07	1.14	***LEXP***
	0.82	0.91	1.00	1.10	1.24	***MODP***
	0.83	0.91	1.00	1.10	1.24	***TOOL***
	1.10	1.04	1.00	1.08	1.23	***SCED***

5- التخمين والشبكات العصبية Estimation and neural networks

بسبب صعوبة تحديد العلاقة الدقيقة بين صفات تخمين الجهد فيمكن اتباع نهج الشبكة العصبية بمثابة أداة لتوليد نموذج من خلال صياغة العلاقات بالاعتماد على تدريبها [7]. ومنذ العقدين الأخيرين اصبحت الشبكات العصبية الاصطناعية (*ANN*) تستخدم على نطاق واسع وفي تطبيقات متنوعة والشبكات العصبية تم تمييزها لقدرتها على إنتاج تنبؤات مقبولة ودقيقة في الحالات التي يكون بها علاقات معقدة بين الإدخالات والإخراجات الموجودة وعندما تكون بيانات الإدخال مشوهة من قبل مستويات ضوضاء عالية [1]

5-1 الشبكات العصبية ذات التغذية الأمامية Feed Forward Neural Networks (FFNN)

تتألف الشبكة ذات التغذية الأمامية على الأقل من ثلاث طبقات من الخلايا: طبقة الإدخال *Input layer*، والطبقة الوسطى وتسمى الطبقة المخفية *hidden layer*، وطبقة الإخراج *output layer*. و ترتبط كل طبقة في الشبكة بالطبقة التي تليها وهذا يعني أن أية خلية في طبقة الإدخال ترسل إخراجها إلى الخلايا كلها في الطبقة الوسطى، و ترسل خلايا الطبقة الوسطى إخراجها إلى كل خلية في طبقة الإخراج. ويعتمد عدد الخلايا في الطبقة الوسطى على درجة تعقيد المسألة وحجم معلومات الإدخال. يوضح الشكل (2) الشبكة العصبية ذات التغذية الأمامية (*FFNN*) المتكونة من ثلاث طبقات وهي طبقة الإدخال والطبقة الوسطى وطبقة الإخراج على الترتيب.



الشكل(2). الشبكة العصبية ذات التغذية الأمامية (*FFNN*)

إذ أن:

$I = (i_1, i_2, i_3, ..., i_n)$ يمثل متجه الإدخال.

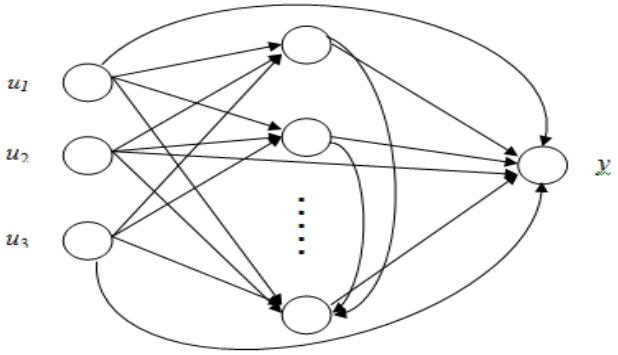
$O = (o_1, o_2, o_3, ..., o_m)$ يمثل متجه الإخراج.

$N = (n_1, n_2, n_3, ..., n_m)$ يمثل متجه الطبقة الوسطى.

و m و n و r أبعاد متجهات الإدخال والطبقة الوسطى والإخراج على التوالي.

2-5 الشبكات العصبية المتتالية (CNN) The Cascade Neural Network (CNN)

شبكة الـ *CNN* مشابهة لشبكات التغذية الأمامية ولكن تحتوي على ترابط بالأوزان من طبقة الإدخال إلى كل طبقة تالية لها ومن كل طبقة إلى الطبقات التالية لها. على سبيل المثال شبكة ذات ثلاث طبقات تملك ارتباطات من الطبقة 1 إلى الطبقة2، ومن الطبقة 2 إلى الطبقة 3، ومن الطبقة 1 إلى الطبقة 3. وهذه الشبكة أيضا تملك ارتباطات من الإدخال إلى الطبقات الثلاثة كلها وهذه الارتباطات الإضافية قد تحسن من سرعة تعليم الشبكة للإخراج المرغوب [53, 54]. إن شبكة الذكاء الاصطناعي *CNN* مشابهة للشبكات العصبية ذات التغذية الأمامية (*FFNN*) في أنها تستخدم خوارزمية الانتشار خلفاً لتعديل الأوزان ولكن العلامة الفارقة الأساسية لهذه الشبكة هي انه كل طبقة من العقد تكون مترابطة مع كل الطبقات السابقة من العقد وتستخدم هذه الشبكة دالة التحويل (*Tan-sigmoid*) و(*log - sigmoid*) للوصول إلى الحالة الأمثل [55]. والشكل (3) يبين الهيئة العامة لشبكة *CNN*.



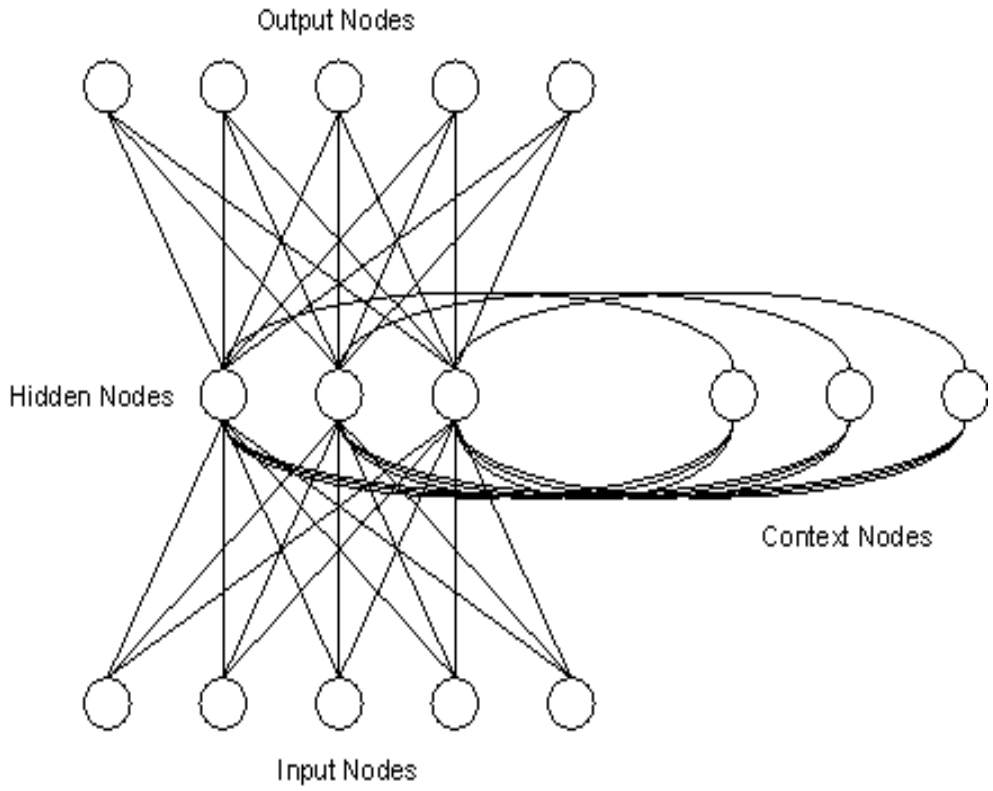
الشكل (3). الهيئة العامة لشبكة *CNN*

إن الشبكات العصبية المتتالية تحوي على ثلاث طبقات : طبقة الإدخال ، والطبقة المخفية وطبقة الإخراج. والتي هي $p, ..., 2,1=i$ ، u_i ، $n, ..., 2,1=i$ ، e_i و $4, ..., 2,1=i$ ، y_i على التتابع.

3-5 الشبكة العصبية إيلمان (ENN) Elman Neural Network(ENN)

تتألف شبكة إيلمان *Elman* على الأقل من ثلاث طبقات من الخلايا: طبقة الإدخال، والطبقة الوسطى وتسمى الطبقة المخفية، وطبقة الإخراج. و ترتبط كل طبقة في الشبكة بالطبقة التي تليها مع ارتباط خلايا ألطبقه الوسطى أيضا مع طبقة الإدخال وهذا يعني أن أية خلية في طبقة الإدخال ترسل إخراجها إلى الخلايا كلها في الطبقة الوسطى، و ترسل خلايا الطبقة الوسطى إخراجها إلى كل خلية في طبقة الإخراج وإلى طبقة الإدخال. ويعتمد عدد الخلايا في الطبقة الوسطى على درجة تعقيد المسألة وحجم معلومات الإدخال [12].

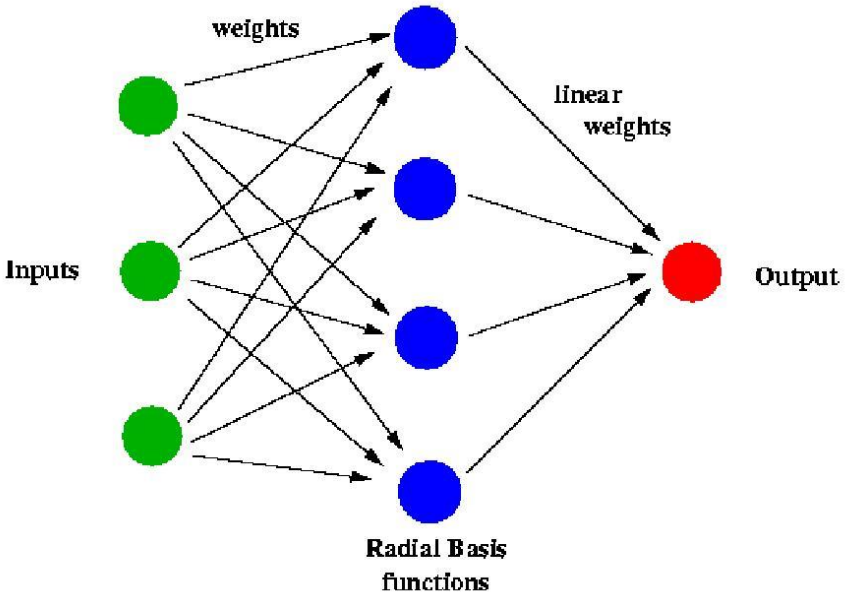
ويوضح الشكل (4) التركيب العام للشبكة العصبية *Elman* المتكونة من ثلاث طبقات وهي طبقة الإدخال والطبقة الوسطى وطبقة الإخراج على الترتيب.



الشكل(4). التركيب العام للشبكة العصبية إيلمان (*ENN*)

4-5 شبكة دالة القاعدة الشعاعية RBFN Radial Basis Functions netwoek RBFN

تعد من شبكات التغذية الأمامية *feed forward* وتحتوي على طبقة مخفية واحدة ودالة اللياقة لهذه الطبقة تسمى *basis functions* والشكل(5) يمثل الهيكلية العامة لشبكة دالة القاعدة الشعاعية [13].



الشكل (5). الهيكلية العامة لدالة القاعدة الشعاعية

تقوم هذه الشبكة بتحويل المدخلات بطريقة غير خطية ثم إيجاد المنحني المناسب لإعطاء النتائج الصحيحة. تمزج هذه الشبكة نوعي التعليم للشبكات العصبية (*hybrid of unsupervised and supervised learning*) بحيث يكون التعليم بين طبقة الإدخال والطبقة المخفية هو تعليم بدون مُعَلم *unsupervised* ويتم عنقدة البيانات إلى مجاميع حسب قانون المسافة الإقليدية بين بيانات الإدخال وأوزان الطبقة المخفية التي يتم في البداية اختيارها بشكل عشوائي وبدون الحاجة إلى معرفة المخرجات وتسمى دالة اللياقة الخاصة بهذه الطبقة *Gaussian radial basis functions* [14]، أما التعليم بين الطبقة المخفية وطبقة الإخراج فيكون تعليماً بِمُعَلم *supervised* ويعتمد على نسبة الخطأ بالاعتماد على المخرجات. المزايا الرئيسية لشبكة دالة القاعدة الشعاعية هي بساطة الدالة المستخدمة ومنحني الدالة يكون سلساً جداً وشعاعي التناظر وغالباً ما يتم اختيار دالة *Gaussian* لتكون هي دالة القاعدة الشعاعية حيث أنها تستطيع أن تحدث تقارباً *approximate* في أي وظيفة ثابتة من دون الاعتماد على نموذج النظام [14].

6- معايير التقييم Evaluation criteria

لإجراء مقارنة بين النماذج المستخدمة سيتم استخدام المقاييس التالية:

1- جذر متوسط مربع الخطأ [7] ***Root Mean Square Error (RMSE)***

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(E_i - \hat{E}_i)^2} \quad \ldots(3)$$

2- متوسط حجم الخطأ النسبي [16] ***Mean Magnitude Relative Error(MMRE)***

$$MMRE = \frac{1}{N}\sum_{i=1}^{N} MRE_i \times 100 \quad \ldots(4)$$

حيث:

$$MRE = \left|\frac{E-\hat{E}}{E}\right| \quad \ldots(5)$$

3- خطأ التوازن النسبي [16] ***Balanced Relative Error BRE***

$$BRE = \frac{\left|E-\hat{E}\right|}{\min(E,\hat{E})} \quad \ldots(6)$$

4- نسبة التباين لـ [17] ***Variance-Accounted-For (VAF)***

$$VAF = \left[1 - \frac{var\left(E-\hat{E}\right)}{var(E)}\right] \times 100\% \quad \ldots(7)$$

حيث في جميع المقاييس الـ $(\hat{E})$ هي الجهد المخمن *Estimated Effort* والـ (E) هي الجهد الحقيقي *Actual Effort* و N هي عدد المشاريع الكلي.

ولجميع المقاييس كلما قلت قيمة المقياس كلما كانت النتيجة أفضل عدا مقياس الـ *VAF* فكلما زادت قيمته كانت النتيجة أفضل.

7- تطبيق الطرق المستخدمة

1- تجهيز البيانات Data preparation

في هذا البحث تم استخدام قاعدة بيانات تاريخية (*historical data*) جاهزة لمشاريع سابقة منفذة ومكتملة وهي عبارة عن 60 مشروع وتم حساب الجهد البرمجي المبذول لكل مشروع فيها بشكل دقيق وهذه البيانات توفرها وكالة *NASA* الفضائية. وكل مشروع من هذه المشاريع يحوي على 15 قيمة تمثل عوامل الكلفة *cost driver* وقيمها والتي تكون قيماً مضببة ويتم تحويلها إلى عددية باستخدام الجدول (2).

2- تجهيز الـ COCOMO الوسطي Intermediate COCOMO preparation

إن طريقة الـ *COCOMO* الوسطي تتطلب إيجاد قيمة عامل تعديل الجهد *EAF* والذي عن طريقه يتم تحديد نمط المشروع المطلوب إيجاد الجهد البرمجي له. ويتم حساب قيمة الـ *EAF* عن طريق ضرب قيم عوامل الكلفة الـ 15 الخاصة بالمشروع ببعضها.

3- تجهيز الشبكة العصبية Neural Network preparation

نظراً للتقارب الشديد بين قيم عوامل الكلفة تم في هذا البحث اقتراح استخدام قيمة واحدة بدل قيم الـ 15 عامل وهذه القيمة عبارة عن مضروب هذه العوامل وهي قيمة الـ *EAF* والتي استخدمت مدخلاً للشبكة العصبية إضافة للـ *KLOC* إذ تؤخذ جميع قيم العوامل بالحسبان ولا يتم تجاهل أي عامل من العوامل المؤثرة بالمشروع البرمجي. ويعد الجهد الحقيقي هو الهدف *Target* .

إن هيكلية الشبكة العصبية يجب أن تتلاءم مع المشكلة المراد حلها وقد تم تقسيم البيانات حيث يتم إدخال 50 مشروع من قاعدة بيانات *NASA* لغرض التدريب *Training set* و10 للاختبار *Testing set*. والشبكات العصبية المطبقة في البحث تحتاج إلى عقدتين في طبقة الإدخال *input vector* لكي تتلاءم مع مدخلات بيانات التدريب وعقدة واحدة تمثل متجه الإخراج للشبكة *output vector* والذي يمثل قيمة التخمين وكذلك تحوي الشبكة على عدد العقد في الطبقة المخفية فبعد إجراء عدة محاولات لتحديد أفضل عدد للعقد في الطبقة المخفية بداية بالعدد 1 إلى العدد 20 تم اختيار العدد 4 كأفضل عد العقد المخفية في الشبكات المستخدمة. ويتم في هذه المرحلة تهيئة الإدخالات جميعها الخاصة بالشبكة وهي عدد دورات التدريب حيث: *Epoch=10000* ونسبة الخطأ المسموح *Goal=0.0001* الذي يُستخدم أيضا في شرط التوقف. ويتم أيضا تحديد دالة التفعيل لكل طبقة والتي كانت كما يأتي:

1. طبقة الإدخال (الدالة السجماوية (*logsig*)).
2. الطبقة المخفية (الدالة السجماوية (*logsig*)).
3. طبقة الإخراج(دالة (*purelin*)).

كما ويتم إدخال نوع خوارزمية التدريب المستخدمة حيث تم إجراء محاولات عديدة لاختيار خوارزمية التدريب لتدريب الشبكة من بين عدة خوارزميات والتي تضمنت خوارزمية (*traingdx, trainlm, trainrp*) ولوحظ انه خوارزمية (*Resilient backpropagation*) *trainrp* كانت الأسرع في تدريب الشبكة وحصلت على نتائج أفضل من الخوارزميات الأخرى لذلك تم اختيارها لتدريب هذه الشبكة.

بعد الانتهاء من تدريب الشبكة على المدخلات السابقة والتحقق من صحة تدريبها عن طريق عمل *Validation* لها تم اختبارها على 10 مشاريع *Testing set* وان الإخراج الأساسي المطلوب من الشبكة المدربة هو التخمين النهائي لجهد المشروع والذي يكون ناتج وحدة الاختبار.

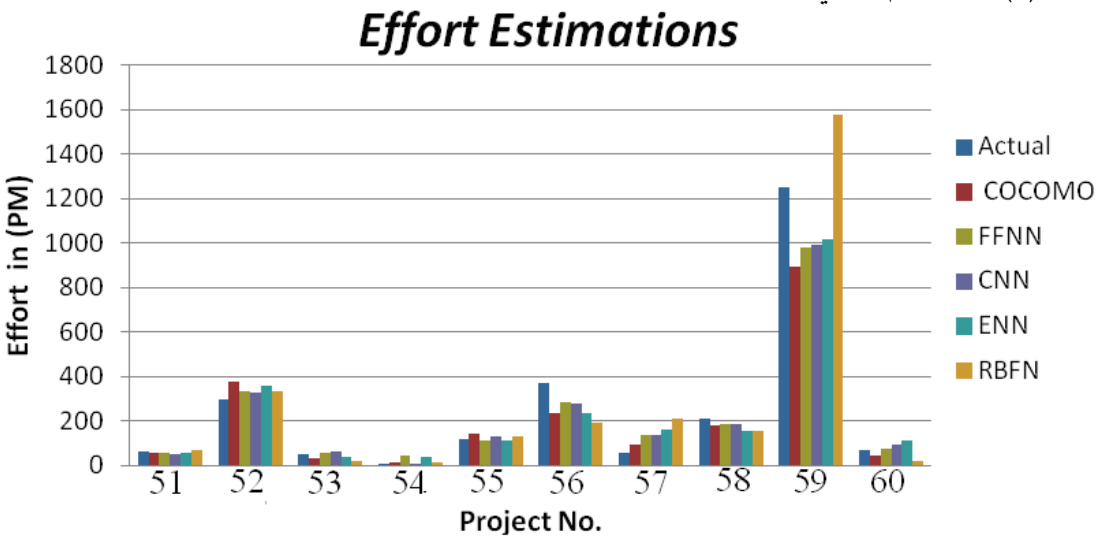
8- النتائج The results

بعد أن تم تصميم وتنفيذ الطرق التقليدية والطرائق الذكائية المستخدمة في البحث كانت النتائج النهائية للتخمين كما موضح في الجدول (3).

الجدول (3). نتائج الطرائق المستخدمة في البحث

آلية التخمين					Actual effort	رقم المشروع
RBFN	*ENN*	*CNN*	*FFNN*	*COCOMO*		
70.3	58.8	50.9593	55.1815	57.8513	62	51
331.8	355.6	330.4601	331.0092	379.3830	300	52
22	38.9	63.9359	58.5792	34.0296	48	53
10.9	38.3	7.9962	46.2691	11.0607	10.8	54
127.8	109.8	132.1036	115.1767	142.7862	120	55
195.1	234	280.8418	286.242	234.1576	370	56
211.7	161.5	137.6326	136.0855	91.9847	60	57
156.5	158.5	186.1757	184.7979	182.5637	210	58
1579.4	1018.3	990.0450	981.3089	892.6617	1248	59
20.9	114.9	94.8860	72.9487	42.1172	72	60

والشكل (6) يمثل الرسم البياني لتخمينات الجهد الناتجة.



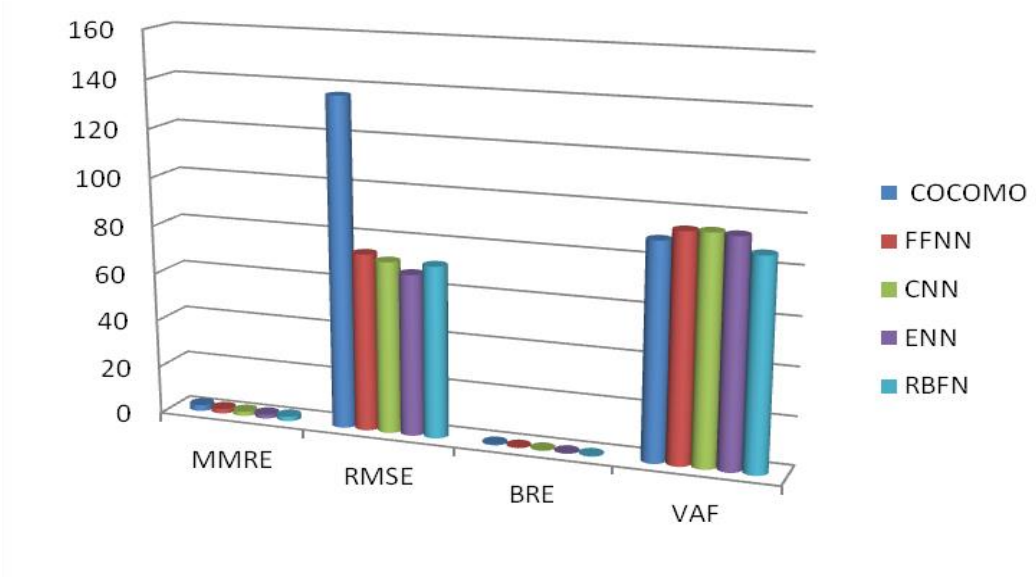
الشكل (6). مخطط شريط الرسم البياني (*Bar chart*) لتخمينات الجهد الناتجة

ولغرض عمل مقارنة بين نتائج هذه التخمينات تم عمل جدول بقيم المقاييس الناتجة من جميع الطرائق المستخدمة والجدول (4) يبين نسب هذه المقاييس:

الجدول (4). مقارنة النتائج

المقياس	آلية التخمين				
	COCOMO	*FFNN*	*CNN*	*ENN*	*RBFN*
MMRE	**2.5517**	**1.9123**	**1.8460**	**1.7408**	**1.8988**
RMSE	**136.6750**	**73.7447**	**71.3928**	**67.0649**	**71.3193**
BRE	**0.3847**	**0.2558**	**0.2501**	**0.2447**	**0.2757**
VAF	**88.9870**	**93.3855**	**93.6742**	**93.0551**	**86.5640**

والشكل (7) يوضح المخطط البياني لقيم المقاييس لجميع الطرائق

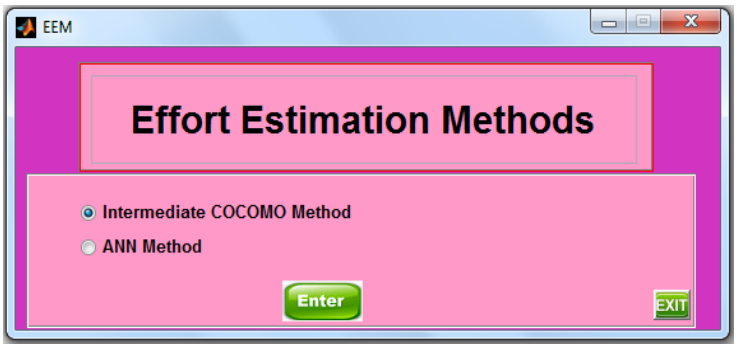


الشكل (7). مخطط شريط الرسم البياني (*Bar chart*) لقيم المقاييس لجميع الطرق

9- واجهات النظام

لغرض استخدام النظام من قبل المستخدم بشكل واضح يسهل عليه التعامل مع النظام المصمم، تم هنا توضيح الواجهات الرسومية للنظام وكيفية استخدامها. وقد تم استخدام مخطط واجهة المستخدم الرسومية لإظهار كيفية تفاعل الأدوات الرسومية في بيئة التطوير المرئية للمستخدم وكيفية عملها.

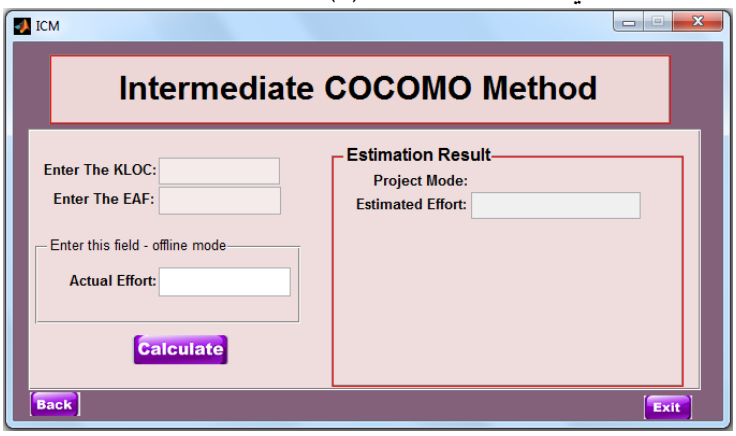
وكانت الواجهة الرئيسة للنظام كما موضحة في الشكل (8) مخطط تصميم الواجهات.



الشكل (8). واجهة المستخدم الرئيسية للنظام.

حيث إنها تحوي على خيارين إما اختيار واجهة طريقة الـ *COCOMO* الوسطي أو اختيار واجهة الشبكات العصبية وبعدها نضغط على *Enter* للدخول إلى الواجهة المطلوبة. وعند الضغط على الزر *EXIT* يتم الخروج من البرنامج نهائياً.

عند اختيار الـ *COCOMO* الوسطي ستظهر لنا الواجهة (9).



الشكل (9). واجهة طريقة الـ *COCOMO* الوسطي.

حيث تحوي هذه الواجهة على الخانات الآتية:

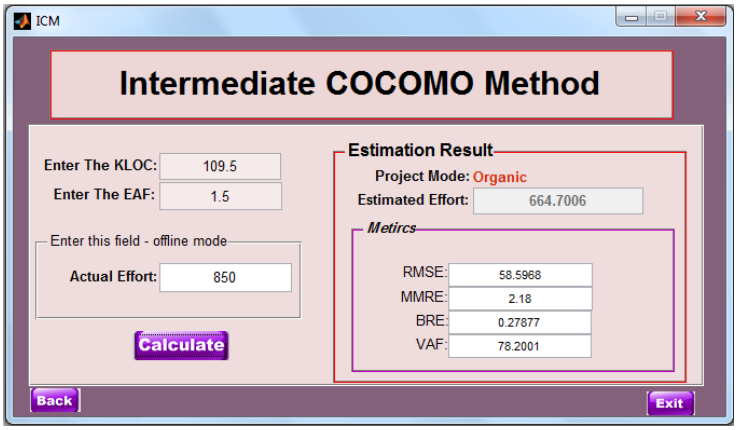
- خانة *Enter the KLOC* والتي يتم فيها إدخال قيمة الـ *KLOC* للمشروع المطلوب تخمين الجهد له.
- خانة *Enter the EAF* والتي يتم فيها إدخال قيمة الـ *EAF* للمشروع المطلوب تخمين الجهد له.
- خانة *Estimation Result* وتحوي هذه الخانة على نتيجة عملية التخمين حيث يظهر فيها نمط المشروع *Project Mode* وأيضا تحوي على خانة *Estimated Effort* والتي تحوي على ناتج الجهد المخمن.

وهذه الخانات السابقة كافية لعمل النظام في حالة الـ *Online* أي لدينا مشروع برمجي نريد معرفة تخمين الجهد له.

أما في حالة الـ *Offline* فيجب إدخال قيمة الجهد الحقيقي إذ أن هذه الحالة تمثل عملية اختبار لمدى دقة التخمين، ذلك أن إضافة لنتيجة التخمين النهائي ونمط المشروع تعطينا هذه الحالة نتيجة أربعة مقاييس لقياس دقة التخمين.

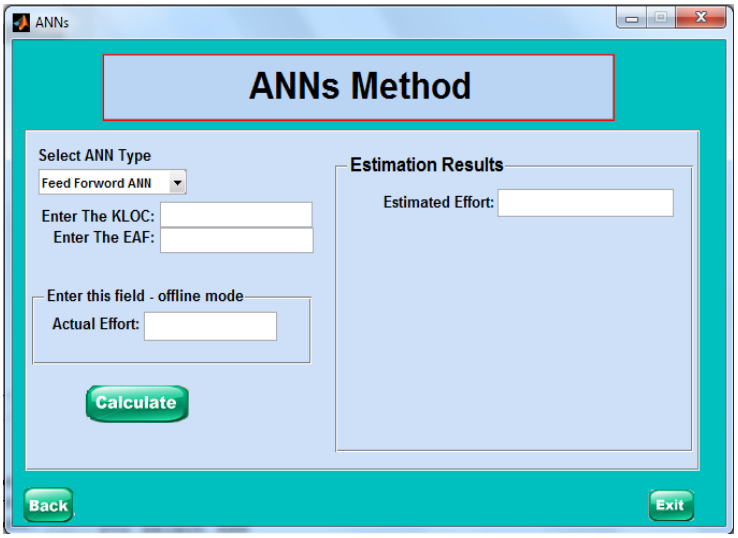
- خانة Actual Effort والتي تعمل في حالة الـ Offline حيث عند إدخال قيمة في هذه الخانة ستتفعل لدينا قيمة المقاييس المستخدمة في خانة الـ Estimation Result كما في الشكل (10).
- زر Calculate حيث عند الضغط عليه ستظهر لنا نتيجة التخمين.
- زر Exit والذي يمثل الخروج من النظام.

- زر Back والذي يمثل العودة إلى الواجهة الرئيسة.



الشكل (10). واجهة طريقة الـ *COCOMO* الوسطي مع المقاييس.

وعند اختيار الـ *ANN Method* من الواجهة الرئيسة ستظهر لنا الواجهة (11).



الشكل (11). واجهة طريقة الـ *ANN method*.

والتي بدورها تحوي على قائمة منسدلة لاختيار نوع الشبكة العصبية المراد تنفيذها إضافة إلى الخانات السابقة الموجودة في واجهة الـ *COCOMO* الوسطي نفسها.

10- الاستنتاجات Conclusions

التخمين البرمجي (*software estimation*) هو عملية أساسية في هندسة البرمجيات وهو عنصر مهم من عناصر تخطيط وإدارة المشروع البرمجي. وفي هذا البحث تم الاعتماد على بيانات *NASA* لتخمين الجهد في طريقة الـ *COCOMO* ولتدريب واختبار الشبكات العصبية المستخدمة تم الاعتماد على لغة *MATLAB11* في برمجة هذه الطرائق. وبصورة عامة برهنت نتائج هذا البحث بان الشبكات العصبية حققت نتائج في تخمين الجهد أفضل بكثير من الطريقة التقليدية *COCOMO* وكانت شبكة *ENN* هي الأفضل بين الشبكات العصبية وتليها شبكة *CNN* ثم شبكة *FFNN* وشبكة *RBFN* وحصل الـ *COCOMO* على أسوأ تخمين بين الطرق المستخدمة.

المصادر

[1] Roheet Bhatnagar, Mrinal Kanti Ghose, 2012, "Early Stage Software Development Effort Estimations–Mamdani Fis Vs Neural Network Models", CS & IT , pp. 377–384.

[2] Guillermo Sebastian Donatti, 2005, "Software Development Effort Estimations Through Neural Networks", Faculty of Mathematics, Astronomy and Physics Cordoba National University.

[3] Jagannath Singh, Bibhudatta Sahoo, 2011, "Software Effort Estimation with Different Artificial Neural Network", IJCA, 2nd National Conference-Computing, Communication and Sensor Network" CCSN.

[4] Daniel Meier, 2006, "E-Learning for Effort Estimation in Software Projects", Master's Thesis in Computer Science and Business Administration, Department of Informatics, University of Zurich, Switzerland

[5] Ali Idri, Alain Abran, Taghi M. Khoshgoftaar, 2012, "Estimating Software Project Effort by Analogy Based on Linguistic Values", IEEE.

[6] Parvinder S. Sandhu, Porush Bassi, and Amanpreet Singh Brar, 2008, "Software Effort Estimation Using Soft Computing Techniques", World Academy of Science, Engineering and Technology.

[7] Jaswinder Kaur, Satwinder Singh, Dr. Karanjeet Singh Kahlon, Pourush Bassi, "Neural Network-A Novel Technique for Software Effort Estimation", International Journal of Computer Theory and Engineering, Vol. 2, No. 1 February, 2010 1793-8201.

[8] Prasad Reddy, Sudha K. R, Rama Sree p, 2011, "Application of Fuzzy Logic Approach to Software Effort Estimation", (IJACSA) International Journal of Advanced Computer Science and Applications.

[9] Zhiwei Xu, Taghi M. Khoshgoftaar, 2003, "Identification of fuzzy models of software cost estimation", Elsevier B.V,141-163.

[10] Barry W. Boehm, 1983, "Software Engineering Economics", Software Information Systems Division, TRW Defense Systems Group, Redondo Beach.

[11] Lionel C. Briand and Isabella Wieczorek, 2001, "Resource Estimation in Software Engineering", Wiley.

[12] Holk Cruse, 1996, "Neural Networks as Cybernetic Systems", 2nd and revised edition, Brains, Minds & Media ISSN.

[13] محمد مهدي ,كرم ، تصميم وتنفيذ نظام كشف التطفل باستخدام التقنيات الذكائي" "،.2012

[14] إبراهيم خليل ، شهباء،" الاسترجاع الكفوء للوسائط المتعددة المعتمد على المحتوى بالشبكات العصبية"، 2006.

[15] Prasad Reddy, Sudha K.R, Rama Sree and Ramesh, 2010, "Software Effort Estimation using Radial Basis and Generalized Regression Neural Networks", journal of computing, ISSN 2151-9617.

[16] Alaa F. Sheta, Alaa Al-Afeef, 2010, "A GP Effort Estimation Model Utilizing Line of Code and Methodology for NASA Software Projects", IEEE, International Conference on Intelligent Systems Design and Applications.