

إطار عمل لتخمين جهود فريق العمل في تطوير مشاريع الويب

روان انور نصر

كلية الهندسة المعلوماتية - قسم البرمجيات - جامعة تشرين

طالبة ماجستير علوم الويب في الجامعة الافتراضية السورية

المشرف: د. محمد علي محمد

دكتور مدرس في كلية الهندسة المعلوماتية والاتصالات - الجامعة العربية الدولية

الملخص:

يخوض مجال تقدير جهود إنجاز المشاريع البرمجية مصاعب كبيرة لعدم القدرة على تحديد الجهد المبذول بشكل دقيق ليتم تخمين التكلفة الصحيحة مقارنة بالجهد المبذول، لذا ومنذ عقود تم وضع العديد من الأفكار والخوارزميات والمنهجيات التي تسعى لتقدير الجهد بأقرب شكل ممكن اعتماداً على معايير متعددة، أبرزها COCOMO التي تعتمد على عدد الأسطر البرمجية و Use Case Point التي تعتمد على جداول حالات الاستخدام التي يبنى عليها المشروع البرمجي. وبعد دراسة العديد من الطرق بما فيها النموذجين المذكورين، تم التوصل إلى فكرة جديدة تعتمد على بيانات دراسة المتطلبات والدراسة التحليلية للمشروع بشكل كامل، لتحديد باقي مراحل إنجاز المشروع البرمجي والتي هي (تصميم - تنفيذ - اختبار - نشر - صيانة)، وإدخال جميع تفاصيل البيانات التي تحدد العمل الذي سيتم إنجازه لتنفيذ المشروع بدقة، كتصميم الصفحات والواجهات وقاعدة المعطيات والأكواد البرمجية الواجب كتابتها مع تحديد حجمها، بناءً على هذه الإدخالات سيتم تحديد الوقت اللازم لإنجاز المشروع وبالتالي معرفة عدد الأشخاص اللازمين لإنجاز هذا المشروع. تمت تجربة المشروع عن طريق بناء موقع ويب يسمح

بإدخال بيانات المشروع، وتم مقارنة الناتج مع نواتج COCOMO و Use Case Point، وتبين معنا أنه بتحديد بيانات المشروع بدقة حصلنا على توفير بالوقت المخصص لإنجاز المشروع، وبالتالي توفير في الكادر والتكلفة.

- الكلمات المفتاحية: الجهد - تقدير الجهد - تقدير التكلفة - تحليل متعدد المعايير -
- عامل ضبط القيمة - عدد أسطر الكود البرمجي - نقاط حالات الاستخدام - رأي
- الخبير - التناظر التاريخي.

A Framework for Estimating Efforts in Developing Web Projects

Abstract:

The field of efforts estimating of implementation software projects is facing great difficulties due to the inability to accurately determine the effort expended in order to estimate the correct cost compared to the effort expended. Therefore, for generations, many ideas, studies, algorithms and methodologies have been developed that seek to estimate the effort as closely as possible based on multiple criteria, most notably COCOMO, which Depends on the number of lines of code, and the Use Case Point is based on the use case tables on which the software is built. After studying a lot of solutions (including the past two mentioned models) a new idea was reached that relies on the data of the requirements study and a fully analytical study of the project, to determine the remaining stages of the completion of the software project, which are (design - implementation - testing - publishing - maintenance), and entering all the details of the data that determining the work that will be done to implement the project accurately, such as designing the pages, interfaces, database, and programming codes to be written, with specifying their volumes.

The project was tested by building a website that allows entering project data, and the output was compared with the outputs of COCOMO and Use Case Point, and it was found with us that by accurately identifying the project data, we got a saving in the time allocated to completing the project, and thus saving in staff and cost.

Keywords: Effort - Effort Estimation - Cost Estimation - Multi-Criteria Analysis (MCA) - Value Adjustment Factor (VAF) - Source line of code (SLOC) - Use Case Point (UCP) - Expert Judgment - Analogy Historical.

1- مقدمة البحث

1.1 - المقدمة:

يُعرى إخفاق المشاريع البرمجية في كثير من الحالات الى التقديرات غير الملائمة، أو المُبالغ فيها لعناصر التكلفة، لذلك يجب دراسة الأساليب المتبعة التي تتصف بالعمومية لمعرفة ما إذا كانت تناسب جميع الشركات على حد سواء، وبغض النظر عن خصوصية وتفرد كل شركة، وإن كانت بعض الأساليب تراعي موضوع الخصوصية بشكل بسيط. ونقصد بالخصوصية هنا خبرة الشركة مع نمط معين من المشاريع ومستوى كفاءة عاملها، بالإضافة لسهولة تعاملها مع الزبون، فبعد دراسة خصوصية كل شركة على حدة، ربما يكون من المناسب اقتراح الآلية المناسبة لها بناءً على هذه الخصوصية، وبالتالي من الممكن أن تكون الآلية المقترحة لها لا تتلاءم تماماً مع غيرها. من هنا تبرز أهمية التقدير الملائم لتكلفة المشروع البرمجي، التي تعد من التحديات المهمة في الوقت الحاضر في مجال إدارة المشاريع البرمجية، حيث يتطلب ذلك وضع تقديرات صحيحة لجميع الجوانب، بما في ذلك من تحليل وتصميم النظام وتجزير البرمجيات واختبارها، ولاسيما فيما يتعلق بتقدير الجهد البشري اللازم لإنجاز مثل هذه الأعمال.^[1]

1.2 - المشكلة العلمية ومبررات مشروع البحث:

يعتبر تخمين جهود فريق العمل من أهم التحديات التي يعاني منها مدراء المشاريع، بسبب عدم اعتمادها على أسس محددة، ولما له من أثر كبير على تقدير التكلفة النهائية للمشروع، ففي إطار تقدير تكلفة مشروع برمجي، كيف يمكن أن نأخذ بعين الاعتبار خصوصية الشركة المطورة، وكيفية تنفيذ المشروع، من أجل الحصول على تقديرات أكثر ملائمة من تلك التي تعطيها الطرائق العامة المستخدمة في هذا المجال. ولنستطيع الإجابة على هذا السؤال الجوهرى يجب أن نركز على الفكرة الرئيسية التالية: إن مشكلة تقدير الكلفة في المشاريع البرمجية تؤول في النهاية إلى مسألة تقدير الجهد اللازم للتطوير، فهذه المسألة هي الموضوع الأساسي الذي تركز

عليه الطرائق الشائعة ، أما الانتقال من تقدير الجهد إلى تقدير تكلفة هذا الجهد فهو مرتبط بالبيئة الاقتصادية التي يجري فيها تنفيذ المشروع وخاصة أجور العاملين في مجال التطوير البرمجي.

1. 3- هدف البحث:

- 1- البحث في نقاط الضعف والمشكلات التي تواجه إدارة مشاريع الويب عموماً وخاصة فيما يتعلق بتقدير تكلفة المشروع.
- 2- مراجعة الأدوات والتقنيات والمنهجيات المتوافرة واستكشاف مميزاتا وعيوبها و تحديد الفجوات التي تحتاج لمعالجة.
- 3- تقديم إطار عمل متكامل لتخمين جهود فريق العمل بالاعتماد على معايير متنوعة بعد تحليل الأطر المتوافرة.

1 . 4- فرضيات البحث:

باعتبار كون تكلفة الجهد للمشاريع هي الأكثر جدلاً في الآونة الأخيرة، نهدف لأن نقدم في هذا المشروع نموذج عمل جديد لتقدير الجهد اعتماداً على الدراسة التحليلية للمشروع، لذا من المهم جداً أن تكون الدراسة التحليلية مكتوبة بدقة وبخبرة لتمكنا من إعطاء النتيجة الأدق للجهد المطلوب لتنفيذ المشروع وتحديد عدد الساعات الكلي وعدد الأشخاص المطلوب توافرهم لإنجاز المشروع خلال فترة معينة.

1 . 5- النتائج المتوقعة والجهات المستفيدة:

- بناء إطار عمل متكامل يساعد مدراء المشاريع والمطورين في التقدير الأمثل لجهود فريق العمل في تنفيذ مشاريع الويب بأفضل كفاءة وجودة ممكنة.
- حل المشكلات التي تعاني منها شركات البرمجة السورية الصغيرة منها والكبيرة من أجل الاستفادة القصوى من مهارات فريق العمل في مشروع الويب.

2- الجزء النظري

يمكن تصنيف التخمين البرمجي إلى ثلاث مراحل:

- المرحلة الأولى: تشمل تخمين الحجم
- المرحلة الثانية: تتمثل في تخمين الوقت وتخمين الجهد
- المرحلة الثالثة: تتمثل في تخمين الكادر وتخمين الكلفة.



شكل رقم 1 تسلسل مراحل التخمين في دورة حياة تطوير البرمجيات

2 . 1- المنهجيات العامة لتقدير الجهد:

هي عديدة كما ذكرها Boehm عام 1981 حيث منها ما هو نماذج حسابية Models Algorithmic ومنها غير حسابية Non-Algorithmic Models ويمكن تلخيصها بالتالي:

2. 1. 1 - رأي الخبير (Expert Judgment) :

هنا يتم الاستفادة من الخبراء المتخصصين في مجال تطوير المشاريع البرمجية حيث يقوم كل منهم بإعطاء تقدير تكلفة المشروع ومن ثم يجري مناقشة التقديرات والوصول إلى رأي نهائي متفق عليه لهذه التكاليف.

2. 1. 2- التقدير المبني على التشابه (Estimation by Analogy):

هنا يتم تقدير تكلفة المشروع بناء على تكلفة مشاريع مماثلة سابقة ، حيث يتم تقدير تكلفة المشروع قيد الدراسة بالتشابه مع ما سبقه، وتمتاز هذه الطريقة بالدقة إذا توافرت بيانات لمشاريع مماثلة للمشروع الذي هو قيد التقدير، أما ما يعيبها أنها مستحيلة الحدوث إذا لم يتواجد هناك مشاريع مماثلة سابقة.

2. 1 . 3- قانون باركنسون (Parkinson's Law) :

هو قانون قائم على فكرة أن العمل يتوسع كي يكون على اتساع الوقت المتاح لإنجازه، وهذا يعني أنه عند تحديد وقت أكبر لإنجاز عمل ما، فإن هذا العمل يميل إلى استغراق الوقت المحدد له، فإذا تم تحديد ثلاث ساعات على سبيل المثال لمجموعة من الأشخاص لتنفيذ مهمة محددة مقابل أربعة ساعات لمجموعة أخرى من الأشخاص لتنفيذ المهمة نفسها، فإنّ كلتا المجموعتين تنتهيان المهمة في حدود الوقت المحدد لها، وهذا يعني أن تقدير التكاليف يتم بناء على المدة المحددة لتنفيذ المشروع والموارد المتاحة، مثال على ذلك: إذا طلب تسليم مشروع برمجي خلال 12 شهراً ولدينا 5 مبرمجين فإنّ الجهد اللازم للإنجاز يتم تقديره بـ 60 شخص*شهر.

2. 1 . 4- التسعير للفوز (Pricing to Win):

هنا يتم تقدير تكلفة المشروع البرمجي بناءً على مقدرة العميل على الدفع عليه، أي أنّ الجهد يقدر بناء على ميزانية العميل وليس على أداء البرمجية، يتم اللجوء إليها في حالة عدم توافر مواصفات تفصيلية للمشروع المطلوب تنفيذه، وتقدر التكاليف بحيث تتوافق مع أساسيات العرض المقدم بدون النظر إلى جودة أداء النظام المستهدف.

2. 1 . 5- تقدير التكلفة (Cost Estimation) :

يتطلب تخطيط المشروع البرمجي بشكل عام تقديم تقديرات بخصوص المعالم الأساسية للمشروع مثل الحجم، الموارد، فريق التطوير، الجداول الزمنية، والتكلفة. ويتحكم بتطوير المشاريع البرمجية أربعة عوامل أساسية هي: الوقت، والمتطلبات، والمخاطر، والموارد التي تشتمل على فريق العمل، العامل المادي، والتجهيزات والبنية التحتية، وأي تغيير غير متوقع في هذه العوامل سوف يؤثر مباشرة على خطة تطوير المشروع.

هنا يجب التأكيد أنه في الغالب التكاليف في معظم المشاريع هي تكلفة الجهد، حيث أن تكلفة العتاد اللازمة تعد نسبياً بسيطة من ناحية التقدير، أما تكاليف الجهد لا تشمل فقط رواتب وأجور فريق تطوير المشروع البرمجي بل تحمّل الجهات المطورة التكاليف الأخرى ذات الصلة بنسب معينة على الرواتب والأجور بما يضمن الوصول إلى تكلفة العمل الكلية.

2. 1 . 6- عدد أسطر الكود البرمجي (SLOC) Source line of code :

هو مقياس يستخدم لتقدير حجم برنامج عن طريق تعداد الأسطر البرمجية في نص البرنامج. يستخدم SLOC للتنبؤ بمقدار الجهد الذي سيكون مطلوباً لتطوير البرنامج، وكذلك لتقدير إنتاجية البرمجة أو الصيانة بمجرد أن يتم إنتاج البرنامج. وهناك نوعان رئيسيان لقياس ال SLOC: قياس فيزيائي (مادي) وقياس منطقي، حيث المقياس المادي هو عدد أسطر الكود في البرنامج بعد إهمال أسطر التعليقات، أما المقياس المنطقي فيُعرّف بأنه عدد التعليمات القابلة للتنفيذ ويرتبط ذلك بلغة البرمجة فمثلاً في لغة البرمجة باسكال يمكن تعريفه بأنه عدد الفواصل المنقوطة التي تنتهي بها التعليمات.

2. 1 . 7- نموذج بناء التكلفة (COCOMO) Constructive Cost Model: [2]

تم تطوير نموذج بناء التكلفة من قبل Boehm عام 1981 وهو من النماذج التي تُستخدم لتقدير الجهد اللازم لإنتاج نظام برمجي محدد وبالتالي التكلفة والوقت الزمني له محددين. عملية الحساب في هذا النموذج تتم باستخدام معادلة الجهد وذلك لتقدير حجم العمل المطلوب مقاساً بالواحدة شخص*شهر وبقية نتائج هذا النموذج مثل تقدير المتطلبات والصيانة تشتق من نتيجة هذه المعادلة. وافترض Boehm أنه يمكن تصنيف المشاريع البرمجية حسب تعقيدها وحجمها إلى ثلاث فئات :

أ- المشاريع العضوية organic : وهي المشاريع البرمجية البسيطة والصغيرة نسبياً، وتعمل فيها فرق برمجية صغيرة العدد ذات خبرة جيدة بتطوير التطبيقات، وتكون متطلبات المشروع مفهومة والمتطلبات أقل صرامة .

ب- المشاريع القليلة الترابط semi-detached : وهي المشاريع البرمجية المتوسطة التعقيد والحجم ، وتعمل فيها فرق برمجية لديهم مستويات خبرة متفاوتة، وتكون متطلبات المشروع هنا متنوعة بين صارمة وأقل صرامة.

ج- المشاريع المضمنة embedded : هي المشاريع التي لها خصوصية معينة، وهي خاضعة لمجموعة قيود عملية وبرمجية خاصة.

وقد تم اقتراح نماذج مختلفة من Cocomo وذلك بناء على مقدار الدقة المطلوبة وهم:

أ- النموذج الأساسي Basic Cocomo

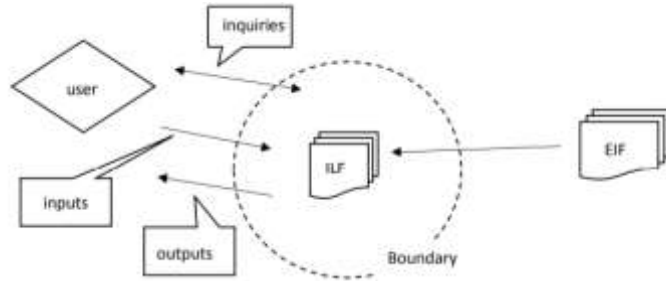
ب- النموذج المتوسط Intermediate Cocomo

ج- النموذج المفصل Detailed Cocomo

2. 1 . 8 - تحليل النقاط الوظيفية Function point analysis [3]:

وهو كما عرفه Albrecht مقياس كمّي يستخدم للتعبير عن كمية المهام التي يجب أن يحققها نظام المعلومات المستهدف حيث يُنظر إلى النقطة الوظيفية كوحدة قياس لتقدير حجم كل وظيفة على حدى في المشروع (FSM) Function Size Measurement، ويعتمد حساب تكلفة كل وحدة استناداً إلى مشاريع سابقة. يتم تحديد المتطلبات الوظيفية الخاصة بالمنتج البرمجي، ومن ثم يتم تصنيف كل مطلب ضمن واحد من خمسة أنواع:

مخرجات - مدخلات - ملفات داخلية - واجهات خارجية - استعلامات، والشكل الآتي يوضح ما سبق:



شكل رقم 2 أنواع المتطلبات الوظيفية

حيث يجب في البداية تحديد النطاق الذي يفصل بين المشروع الذي يجب قياسه وبين التطبيقات الخارجية والمستخدمين، ثم يتم تقييم تعقيدات هذه المتطلبات وتحديد عدد النقاط الوظيفية غير المعدلة (UFP) Unadjusted Function Point.

2. 1 . 9 - نقاط حالة الاستخدام Use Case Point (UCP): [4]

تم تطوير هذه الطريقة من قبل Karner عام 1993 وهي تقنية تُستخدم لتوقع حجم البرمجيات المستخدمة لإنهاء تطوير مشروع برمجي، وهي تشبه ال FPA من حيث المبادئ لكنها مُوجهة لتقدير حجوم المشاريع البرمجية غرضية التوجه Projects

Unified Object Oriented، وهي تعمل حصراً عندما يتم استخدام منهجية Unified Modeling Language (UML) و Rational Unified Process (RUP) عند تصميم البرمجيات وتطويرها، وذلك بسبب أن UCP تعتمد في حساب حجم المنتج البرمجي على متطلبات النظام التي تتم نمذجتها باستخدام حالات الاستخدام Use Case والتي هي جزء من تقنيات ال UML ، فضلاً عن أن هذه التقنية تأخذ بالاعتبار عند تقدير الجهد اللازم للمشروع العوامل التقنية و بيئة العمل . وتعتمد هذه التقنية في حساب حجم المشروع البرمجي على عملية توازن بين أوزان العناصر التالية:

- عدد حالات الاستخدام في النظام وتعقيدها.
- عدد الفاعلين في النظام وتعقيدهم.
- متطلبات غير وظيفية.
- البيئة التي سيتم تطوير المشروع فيها.

3- الدراسات المرجعية

لقد قَدّم الباحثون في هذا المجال عدة دراسات بما يخص تقدير الجهود، ومنها الدراسات التالية:

دراسة أعدها (Suresh Nageswaran، 2001) بعنوان Test Effort Estimation Using Use Case Points^[5] قدمت هذه الدراسة تجارب على تقدير الجهود باستخدام نموذج Use Case Points واستخلصت أنه هناك مزايا عملية كبيرة من استخدام هذا النموذج، ولكنه بحاجة للمزيد من البحث والتجريب للحصول على المزيد من الفوائد الجوهرية في الوصول إلى طريقة موضوعية للتحقق من صحة التقديرات.

دراسة أعدها (Taghi M. Khoshgoftar، Ali Idri ، Alain Abran، 2002) بعنوان Estimating software project effort by analogy based on linguistic values^[6]: في هذه الدراسة قام الباحثون بتصميم أسلوب جديد بالاعتماد

على المنطق الضبابي والمحددات (quantifiers and fuzzy logic) كذلك على الاستدلال اللغوي باستخدام أسلوب المقارنة (linguistic reasoning by analogy) لتخمين جهد المشروع البرمجي وأطلقوا على هذا الأسلوب اسم المقارنة الضبابية (Fuzzy Analogy). ووفقاً لهذا البحث تم تطبيق النموذج المقترح على COCOMO dataset إذ لوحظ أن الطريقة المقترحة قد حسنت عملية التخمين و حصلت على نتائج أفضل من الطرائق التقليدية وأكثر دقة.

دراسة أعدها (Dharmender Singh Kushwaha, Ashish Sharma, 2012) بعنوان Estimation of Software Development Effort from Requirements Based Complexity [7] هدفت هذه الدراسة إلى اقتراح مقياس لتقدير جهد تطوير المشاريع بناءً على تعقيد متطلبات المشروع التي سيتم تطويرها والتي بدورها تعتمد على توصيف متطلبات النظام للمشروع المقترح (software specification requirements)، وللتحقق من صحة النتائج تمت مقارنتها مع تقديرات بطرق أخرى (algorithmic use case point, function point count, models, lines of code)، وتم في هذه الدراسة إنشاء نموذج يستند إلى وضع علامات أو تصنيفات لمتطلبات النظام المطلوبة وقد ثبت أن تعقيد متطلبات البرنامج له تأثير مباشر على الجهد اللازم وبناءً على ذلك اقترح Kushwaha & Sharma معيار تحسين تقدير المتطلبات على أساس تعقيدها وذلك بناءً على الـ SRS المُعطى وتبين أن التقديرات التي تعتمد على هذه المنهجية أكثر فائدة لأنه يتم معرفة تعقيد المتطلبات و بالتالي تقدير الجهد بمرحلة مبكرة جداً.

دراسة أعدها (Abdul Wahid, Mudasir Manzoor Kirmani, 2015) بعنوان Revised Use Case Point (Re-UCP) Model for Software Effort Estimation [8] أكدت هذه الدراسة أن الجانب الذي لا غنى عنه في أي شركة تطوير للبرمجيات هو تشكيل آلية للتعامل مع المشكلات التي تؤدي إلى الفوضى. يتمثل أحد المجالات للتعامل مع هذه المشكلات في جدولة عملية التطوير بأكملها للخضوع لعملية

تقدير مناسبة وفعالة ، حيث يمكن تقدير جميع الموارد مسبقًا بشكل جيد من أجل التحقق مما إذا كان المشروع المتصور مجدياً وضمن الموارد المتاحة. اللبنة الأساسية في أي تصميم موجه للكائنات هي مخططات حالة الاستخدام التي يتم إعدادها في المراحل الأولى من التصميم بعد فهم المتطلبات بوضوح. تعتبر مخططات حالة الاستخدام مفيدة لتقريب تقديرات مشروع تطوير البرمجيات. يعطي هذا العمل البحثي نظرة عامة مفصلة عن طريقة Re-UCP (Revised Use Case Point) (نقطة حالة الاستخدام المنقحة) لتقدير الجهد لمشروعات البرمجيات. طريقة Re-UCP هي طريقة مُعدلة تعتمد على طريقة UCP لتقدير الجهد. في هذه الدراسة البحثية ، خضع 14 مشروعًا لتقدير الجهود باستخدام طريقة Re-UCP وتمت مقارنة النتائج مع نماذج UCP و e-UCP. وتظهر المقارنة بين الـ 14 مشروعًا أن Re-UCP تفوقت بشكل ملحوظ على تقنيات UCP الحالية و (e-UCP) Effort Estimation Techniques.

دراسة أعدها (Alessio Petrozziell, Federica Sarro ، 2018) بعنوان Linear Programming as a Baseline for Software Effort Estimation: [9] قدمت هذه الدراسة طريقة جديدة تعتمد البرمجة الخطية وتم تسميتها ب Linear Programming for Effort Estimation (LPEF) وقد تم مقارنة هذه الطريقة بنموذج سابق المستخدم الوحيد في البرمجة الخطية وهو (Automatically Transformed Linear Model (ATLM) وبينت الدراسة أن LP4EE أكثر دقة من ATLM بنسبة 17% من التجارب ، وذلك من حيث تقسيم البيانات المختلفة في 44% من الحالات.

دراسة أعدها (Chastine , Daniel O. Siahaan, Rahmi Rizkiana Putri) بعنوان Improve the Accuracy of Software Project Effort and Cost Estimates in COCOMO II Using GWO: [10] في هذه الدراسة يتم توضيح أنّ قيمة دقة COCOMO II أقل دقة لأنه لا يزال هناك فرق كبير بين جهد المشروع الفعلي وقيمة التكلفة المقدرة. ولتحسين دقة COCOMO II ، يتم

استخدام طريقة (GWO) Gray Wolf Optimization التي تعتمد على سلوك الذئب في اصطیاد الفرائس. في هذه الدراسة ، يتم استخدام COCOMO II GWO للحصول على مستوى أعلى وأكثر دقة من دقة التقدير وتقليل قيمة الخطأ الإجمالية أو متوسط الخطأ النسبي (MMRE) لمشاريع البرمجيات. إن نتيجة الاختبار MMRE التي أنتجتها الدراسات السابقة لهذه الدراسة التي تستخدم COCOMO II BCO (تحسين مستعمرة النحل) كان 12.92%. وفي الوقت نفسه ، فإن معدل MMRE بالطريقة المقترحة COCOMO II GWO هو 1.731%. هذا يعني أن الطريقة المقترحة يمكن أن تقلل من قيمة الخطأ في MMRE بنسبة 11.19%.

دراسة أعدها (Preeti ، Verma Aditya ، 2021) بعنوان Calibrating Intermediate COCOMO Model Using Genetic Algorithm: [11] هدفت هذه الدراسة إلى تعديل معاملات نموذج COCOMO الوسيط باستخدام خوارزمية الكشف عن مجريات الأمور. ستم مقارنة التكلفة المقدرة بالبيانات الحقيقية. يتم تطبيق المنهجية على مجموعة بيانات برنامج COCOMO NASA ، وتم استخلاص أنه لا بُد من استخدام تقنية البرمجة الجينية (Genetic Programming) لبناء هيكل نموذج مناسب لتقدير جهد البرنامج.

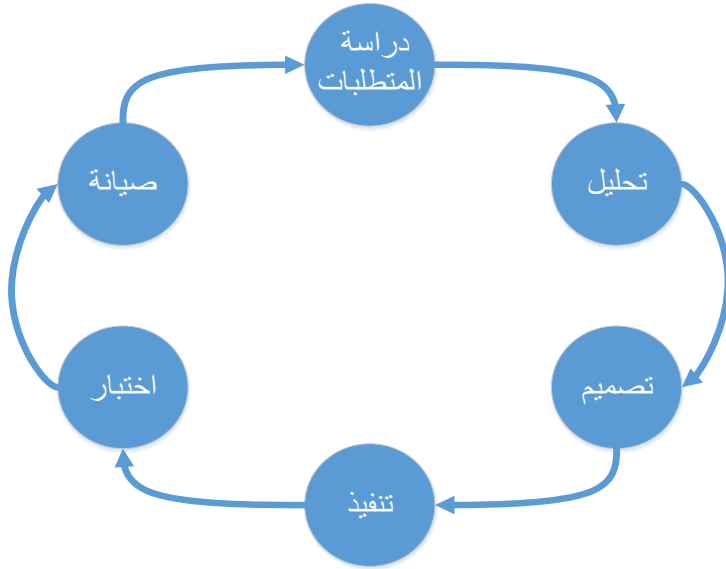
بالرغم من جميع الدراسات ولكن لم يتم التوصل إلى قيم دقيقة وقريبة من الواقع في هذا المجال، وأيضاً لم يتم التوصل إلى نسب توفير كبيرة يمكن اعتمادها لتوفير الوقت المخصص لانجاز المشاريع وبالتالي التكلفة الكلية للمشاريع.

4- الحلول المقدمه:

بعد التمعن في الدراسة المرجعية السابقة ودراسة المنهجيات المتبعة لتقدير الجهد تبين أنه من الممكن الاستفادة من عدة عوامل لإنتاج أسلوب عمل مميز لتقدير الجهد البرمجي المبذول لأي مشروع بأكثر قيمة تقريبية للعمل الحقيقي.

تم ملاحظة أن تقدير الجهد يجب أن يكون على مستوى المشروع بشكل كامل، وليس فقط على مستوى مرحلة واحدة وهي البرمجة، بل أن التقدير الصحيح للمشروع يجب أن يكون شامل لجميع المراحل التي يمر بها منذ البداية حتى النهاية، وهي [12]:

- دراسة المتطلبات
- تحليل المشروع
- تصميم المشروع
- التنفيذ والتحقق البرمجي للمشروع
- اختبار المشروع
- نشر وصيانة المشروع



شكل رقم 7 مراحل انجاز المشروع البرمجي

وبصرف النظر عن المنهجية المتبعة لتطوير المشروع إن كانت شلالية أم حلزونية، حيث أن الحلزونية هي تكرار لخطوات العمل منذ البداية ومن الممكن في بعض الحالات أن تحصل على نتيجة المنهجية الشلالية مضروبة بعدد دورات المشروع.

وفي النهاية سيتم اعتماد فكريتي النقاط والتثقييل، بالنسبة للنقاط فيتم حسابها من كل عامل على حدى، وبعد ذلك يتم تثقييل هذه النقاط ليحصل المشروع على نتيجة معينة لنتمكن من مقارنتها مع مشاريع أخرى لمعرفة الفروقات بين الجهود المبذولة على كل مشروع. يتطلب تخطيط المشروع البرمجي بشكل عام تقديم تقديرات بخصوص المعالم الأساسية للمشروع مثل الحجم، الموارد، فريق التطوير، الجداول الزمنية، والتكلفة. ستكون الآلية البرمجية تعتمد على مجموعة من الأسئلة تشمل جميع حيثيات المشروع على جميع مراحل الستة، وستكون مجموعة كبيرة جداً من الأسئلة وذلك لضمان الحصول على أكبر قدر ممكن من الدقة في تقدير التكلفة للمشروع البرمجي. سيتم الاعتماد على مفهوم الكتل البرمجية المطلوب إنجازها بوضع مجموعة متدرجة من الأصناف ووضع تثقييل لكل صنف كتلة برمجية مطلوب. وتكون الأصناف كالتالي:

- 1- صنف بسيط: يكون لعمليات الإدخال البسيطة، مثل تسجيل معلومات طالب، أو إنشاء حساب مستخدم (الاسم - العنوان - الهاتف - الايميل ... الخ).
- 2- صنف علمي (يتطلب إنجاز معادلات رياضية).
- 3- صنف معقد (خوارزميات).
- 4- صنف دلالي (عمليات دلالية أو شبكات عصبونية بدرجات تعقيد متغيرة).
- 5- صنف معالجة (يتطلب معالجة بيانات موزعة أو متزامنة ... الخ).
- 6- صنف مرئي (معالجة صور).

مع إمكانية تحديد حجم هذا الصنف (صغير - متوسط - كبير). سيتم الاعتماد على البيانات المقدمة من دراسة المتطلبات الخاصة بالمشروع بالإضافة للدراسة التحليلية، أي أنه سيتم تنفيذ حسابات تقدير الجهد بعد تلك المرحلتين، بحساب الجهد المُقدم للمرحلتين السابقتين (دراسة متطلبات - دراسة تحليلية)، بالإضافة لتقدير جهد المراحل التالية (تصميم وتنفيذ واختبار ونشر وصيانة). إحدى أهم النقاط الواجب أخذها بعين الاعتبار هو وجود الكيان المادي الممكن أن يكون جزء من العمل ويتطلب مهام إضافية (تجميع - إعداد - برمجة خاصة).

أما بالنسبة لعدد الأسطر البرمجية، فلا يمكن الوصول إليه إلا عند انتهاء العمل البرمجي، لذا لا يمكن الاعتماد عليه في تقدير تكلفة العمل البرمجي عند بدايته، بينما من الممكن الاعتماد عليه لتقدير جهد مشروع تم إنشاؤه مسبقاً ليتم مقارنة مشروع آخر معه، لذا تم استثناءه من فكرة المشروع.

احتساب نتيجة تقدير الجهود Effort Estimation Result Calculate:

في نهاية تقييم الجهود وحساب التتقيات يتم تحديد الجهد بالنسبة للوقت وعدد الأشخاص بإعطاء عدد من الساعات بناءً على عدد الأشخاص. وسيكون الناتج عبارة عن عدد طبيعي أي موجب، وبدون فواصل وذلك للسماح بإجراء عمليات رياضية على الناتج بشكل بسيط وواضح.

وبعد ذلك يتم تحديد إما الوقت أو عدد الأشخاص ليتم تحديد المعامل الآخر بشكل تلقائي. حيث أنه عند زيادة الوقت يقل عدد الأشخاص، بينما في حال إنقاص الوقت يتم زيادة عدد الأشخاص المطلوبين لإنجاز العمل. كما يتم تحديد عدد الأيام اعتماداً على نوع العمل الموكل إلى العاملين إن كان بوقت جزئي Part Time أو بوقت كامل Full Time، وبذلك يتم إعطاء التقدير النهائي حسب نوع العمل وعدد الأشخاص.

مثال توضيحي:

ليكن لدينا مشروع تم تقدير جهده ب 800 ساعة عمل، عند تقسيم هذا الناتج على عدد الأشخاص يعطي عدد الأيام.

تقدير الجهد اعتماداً على عدد الأشخاص:

عند تحديد عدد الأشخاص سينتج معنا عدد الأيام. فرضاً 10 أشخاص سيقومون بهذا العمل، أي أن العمل يستغرق 80 ساعة للإنجاز، وهذا لا يعني أن نقوم بتقسيم 80 على 24 لمعرفة عدد الأيام اللازمة لإنجاز العمل، أي أن نتيجة القسمة وهي 3 أيام و8 ساعات تعتبر نتيجة خاطئة للعمل، بالرغم من كونها القيمة الفعلية المقدره لإنجاز العمل.

ولتقدير العدد الكلي للأيام سيتم توزيع هذه الساعات حسب نوع العمل إن كان Part Time أو Full Time، حيث أن العمل بوقت جزئي Part Time يعتبر 4 ساعات يومياً، بينما العمل الكامل Full Time هو 8 ساعات يومياً.

على هذا الأساس فإن العمل المقدر بـ 800 ساعة عمل وموزع على 10 أشخاص 80 ساعة عمل كوقت إجمالي لكل شخص، يكون تقديره بالأيام بالنسبة للعمل بوقت جزئي Part Time هو حاصل قسمته على 4 أي 20 يوم، بينما يكون تقديره بالأيام بالنسبة للعمل بوقت كامل Full Time هو حاصل قسمته على 8 أي 10 أيام.

تقدير الجهد اعتماداً على عدد الأيام:

عند تحديد عدد الأيام سينتج معنا عدد الأشخاص.

على سبيل المثال تم تحديد عدد الأيام بـ 25 يوم، فيكون عدد الأشخاص الواجب تعيينهم بوقت جزئي للعمل Part Time هو حاصل قسمة 800 على 25 وهو عدد الأيام وبعدها القسمة على 4 وهو عدد الساعات في اليوم كالتالي:

$800 \div 25 = 32 = 4 \div 8$ أي أنه يكفي تعيين ثمانية أشخاص لتنفيذ المشروع خلال 25 يوم. أما عدد الأشخاص الواجب تعيينهم بوقت كامل للعمل Full Time هو حاصل قسمة 800 على 25 وهو عدد الأيام وبعدها القسمة على 8 وهو عدد الساعات في اليوم كالتالي: $800 \div 25 = 32 = 8 \div 4$ أي أن أربع أشخاص كافيين للقيام بالعمل خلال 25 يوم.

5- التنفيذ والاختبار:

بهدف الحصول على قيم واقعية، قمنا أولاً بالاستعانة بخبراء (في عدة مننديات برمجية والغالبية لم يذكروا أسماؤهم) لتحديد قيم التنقيلات وحصلنا على آراء متقاربة، ولتحديد القيم بشكل أدق قمنا بإنشاء استبيان الكتروني للوصول إلى القيم الصحيحة لتنقيت كل قيمة يتم اختيارها. وبعدها قمنا بمعالجة النتائج واختيار قيم التنقيت اعتماداً على الاختيار الأكثر انتقاءً أو قيمة تنقيت متوسطة بين قيمتين متقاربتين الانتقاء. وتم تخزين جميع القيم

ضمن قاعدة المعطيات، للتعامل معها برمجياً مع إمكانية تغيير القيم دون العودة إلى الشيفرات البرمجية.

بعد تحليل نتائج الاستبيان وجدنا من خلال الإجابات أن:

أ- المعايير الأساسية الواجب تقدير الجهود حسبها هي: نوع المشروع، حجم المشروع، العتاد الصلب.

ب- في حال تم تقسيم حجم المشروع إلى 5 مستويات (صغير جداً، صغير، متوسط، كبير، كبير جداً) فإن الزمن المُقدّر لكل مستوى: صغير جداً (خلال مدة أقصاها أسبوع)، صغير (خلال مدة أقصاها شهر)، متوسط (خلال مدة أقصاها 3 أشهر)، كبير (خلال مدة أقصاها سنة)، كبير جداً (خلال مدة أقصاها 5 سنوات).

ج- في حال برمجة تطبيق Desktop: بالنسبة لكل نموذج يتم إنشاؤه هناك وقت مخصص لتصميم النموذج، وتقسيم حجم العمل التصميمي لكل عنصر إلى خمسة مستويات للتصميم (بسيط جداً - بسيط - متوسط - متقدم - متقدم جداً) فإن الوقت المُقدّر لكل مستوى بالدقائق: بسيط جداً: دقيقة، بسيط: دقيقتان ونصف، متوسط: خمس دقائق، متقدم: سبع دقائق ونصف، متقدم جداً: عشر دقائق.

د- فيما يخص التعديل الفوتوغرافي على الصور: في حال تم التقسيم إلى ثلاث مستويات للتصميم (بسيط - متوسط - متقدم) فإن الوقت المُقدّر لكل مستوى بالدقائق: بسيط: 10 دقائق، متوسط: 50 دقيقة، متقدم: 100 دقيقة.

هـ- في حال برمجة موقع ويب: فإن الوقت المُتوقع لتصميم صفحة: تم اعتماد النتيجة: ساعة.

ح- في حال برمجة موقع ويب: بالنسبة لكل صفحة يتم إنشاؤها هناك وقت مخصص لتصميم الصفحة، وتقسيم حجم العمل التصميمي إلى خمسة مستويات للتصميم (بسيط جداً - بسيط - متوسط - متقدم - متقدم جداً) فإن الوقت المُقدّر لكل مستوى بالدقائق: بسيط جداً: دقيقة، بسيط: دقيقتان ونصف، متوسط: خمس دقائق، متقدم: سبع دقائق ونصف، متقدم جداً: عشر دقائق.

ط- في حال برمجة موقع ويب : لتحسين مظهر الإظهار قمنا بافتراض ثلاث تقنيات، (CSS -Bootstrap - كل عنصر له تنسيقه اليدوي الخاص)فإنّ الوقت المُقدّر لكل تقنية بالدقائق: Bootstrap : ربع ساعة، CSS: نصف ساعة، تنسيق يدوي: ساعة.

ي- الوقت المُقدّر لإضافة جدول والوقت المُقدّر لإضافة حقل إلى هذا الجدول مع تحديد القيود الخاصة به: (القيم بالثواني): تم اعتماد الناتج 15 ثانية.

ك- بعد إنشاء جدول وتحديد حقوله، ووجود بيانات بحاجة للإدخال ضمنه.فإنّ الوقت المُقدّر لإضافة قيمة ضمن خلية في جدول ما (بالثواني): 10 ثواني.

ل- بعد إنشاء جدول وتحديد حقوله،فإنّ الوقت المقدر لإنشاء مفتاح أجنبي (بالثواني): تم اعتماد القيمة 60 ثانية.

م- الوقت المتوسط المُقدّر لإنشاء عملية (كالإجراءات والوظائف والقوالب) متضمنة وقت التفكير بالعملية(القيم بالدقائق): بسيط: 5 دقائق، متوسط: ربع ساعة، متقدم: ساعة.

ن- الوقت المُخصّص لكتابة الكود الخاص بالكتلة البرمجية اعتماداً على تعقيد العملية مُوزّعة على المستويات التالية(بسيطة جداً - بسيطة - متوسطة - معقدة - معقدة جداً) : بسيطة جداً : ربع ساعة، بسيطة : ساعة، متوسطة: ساعتين ونصف، معقدة: خمس ساعات، معقدة جداً : عشر ساعات.

س- الوقت المخصص لكتابة الكود الخاص بالكتلة البرمجية اعتماداً على صنف العملية مُوزّعة على المستويات التالية (بسيط - علمي - معقد - دلالي - معالجة - مرئي): بسيط: ربع ساعة، علمي: ساعة، معقد: ساعتين ونصف، دلالي: خمس ساعات، معالجة: عشر ساعات، مرئي: خمسة عشر ساعة.

ع- فيما يخص العتاد الصلب: فإنّ الوقت المُقدّر لإنجاز أبسط مشروع باستخدام التقنيات التالية: آردوينو: ساعتان, Pic16: 5 ساعات, متحكم آخر: 10 ساعات, متحكمات تحتاج إلى تصميم دائرة الكترونية: 25 ساعة, PLC: 30 ساعة.

ف- أيضاً فيما يخص العتاد الصلب : حسب حجم المشروع، فإنّ معامل المضاعفة الذي سيؤثر على إعداد العتاد الصلب: صغير جداً: 1, صغير: 2, متوسط: 5, كبير: 10, كبير جداً: 25.

ص- الوقت المُقدّر للصيانة والاختبار والنشر, تم اعتماد القيم التالية: زمن الصيانة 10% من زمن المشروع, زمن الاختبار 10% من زمن المشروع, زمن نشر التطبيق ساعة واحدة, زمن نشر الموقع 4 ساعات.

ولتنفيذ العمل فُمنّا بتطوير موقع ويب يحتوي على التبويبات التالية:

- 1- إضافة مشروع جديد Add New Project: لإضافة مشروع برمجي واحتساب الجهد اللازم لتنفيذه.
- 2- عرض المشاريع View Projects : لعرض قائمة بالمشاريع التي تمت إضافتها مسبقاً، مع إمكانية عرض تفاصيلها.
- 3- التثقيلات Widths: لعرض التثقيلات التي تم تخزينها في قاعدة المعطيات، مع إمكانية تغييرها.
- 4- COCOMO: لاحتساب تثقيل المشروع اعتماداً على نموذج COCOMO
- 5- Use Case Point: لاحتساب تثقيل المشروع اعتماداً على نموذج Use Case Point

حيث يتم أولاً إضافة مشروع جديد ووضع تفاصيل المشروع بناءً على الدراسة التحليلية والتي تم حصرها بالبنود التالية:

1- اختيار بيانات رئيسية عن المشروع (اسم المشروع - نوع المشروع - حجم المشروع - العتاد الصلب المستخدم)

2- إن كان نوع المشروع تطبيق، يقوم بإدخال تصميم النماذج الموجودة فيه، كل نموذج على حدى وتحديد البيانات الخاصة بكل نموذج والتي هي (اسم النموذج - مستوى التصميم - عدد العناصر - مستوى التعديل الفوتوغرافي - عدد الصور التي بحاجة إلى تعديل فوتوغرافي).

3- إن كان نوع المشروع موقع، يقوم أولاً بإدخال بيانات التصميم الخارجي للصفحات بدون العناصر، كل صفحة على حدى والتي هي: (هل الصفحة MasterPage - عدد الصفحات - مستوى التصميم - أسلوب التنسيق المعتمد لكل صفحة)، ومن ثم يقوم بإدخال بيانات تصميم الواجهات التي يمكن أن يتم وضعها في صفحة واحدة أو عدة صفحات في حال الحاجة والبيانات هي (اسم الواجهة - مستوى التصميم - عدد العناصر - مستوى التعديل الفوتوغرافي للصور - عدد الصور التي بحاجة إلى تعديل فوتوغرافي).

4- ادخال بيانات قاعدة المعطيات في حال وجودها، والتي هي (عدد الجداول - عدد حقول كل جدول - عدد الجداول المترابطة - عدد الجداول التي بحاجة إلى ادخال بيانات مسبق - حجم البيانات المدخل - عدد العمليات المستخدمة (الاجراءات والقوادح) - حجم العمليات (بسيط - متوسط - متقدم).

5- ادخال بيانات الكتل البرمجية كل كتلة على حدى، وهي (اسم الكتلة - وصف الكتلة - تعقيد العملية - نوع العمليات ضمنها حسب الأصناف).

يتم تخزين قيم تثقيلات البنود التي تم اختيارها في قاعدة المعطيات ومن ثم إجراء عمليات حسابية لاحتساب الناتج النهائي لتثقيل الجهود، ومن ثم تحويل قيمة التثقيل إلى ناتج بالدقائق والساعات، مع إمكانية إضافة احتساب مراحل الاختبار والصيانة والنشر إلى الناتج النهائي، ليصبح بالإمكان احتساب تقدير الجهد بعد توزيعه على الطاقم البرمجي واحتساب عدد المبرمجين والأيام اللازمة لإنجاز المشروع. تكمن المرحلة الثانية

بحساب تثقيل المشروع باستخدام خوارزمية (COCOMO و Use Case Point). أما المرحلة الثالثة والأخيرة، فهي تحويل تثقيل المشروع الناتج عن النموذجين السابقين إلى ساعات ثم إدخاله إلى واجهة المقارنة، ليتم عرض مقارنة بين هذين الناتجين، والناتج الذي توصلنا إليه ليتم عرض مخطط يبين الفروقات بين الناتج الثلاث.

6- النتائج:

تم تجريب العديد من المشاريع البرمجية موثقة بدراسة تحليلية، وتم اعتماد مثال مشروع برمجي وهو موقع خاص بدار للأيتام¹، عن طريق إدخال البيانات اعتماداً على الدراسة التحليلية ضمن واجهات موقع الويب الذي قمنا بتطويره. وكانت النتيجة كالتالي:

تثقيل المشروع بشكل كامل: 32405
الوقت النهائي المقدر بالدقائق: 19443
الوقت النهائي المقدر بالساعات: 324:3

شكل رقم 8 نتيجة تقدير الجهد للمشروع

وعند توزيع العمل على طاقم العمل فإنه عندما يعمل الطاقم عمل جزئي ستكون النتيجة كالتالي:

¹ <https://github.com/rawannsr/orphans>

تجهيز العتاد الصلب: لا يوجد
التصميم: يحتاج إلى 1 شخص/أشخاص خلال 11 يوم
قاعدة المعطيات: يحتاج إلى 1 شخص/أشخاص خلال 5 يوم
التحقيق البرمجي: يحتاج إلى 1 شخص/أشخاص خلال 50 يوم
اختبار المشروع: يحتاج إلى 1 شخص/أشخاص خلال 7 يوم
صيانة المشروع: يحتاج إلى 1 شخص/أشخاص خلال 7 يوم
نشر المشروع: يحتاج إلى 1 شخص خلال 1 يوم
المشروع بشكل كامل: يحتاج إلى 5 شخص/أشخاص خلال 75 يوم

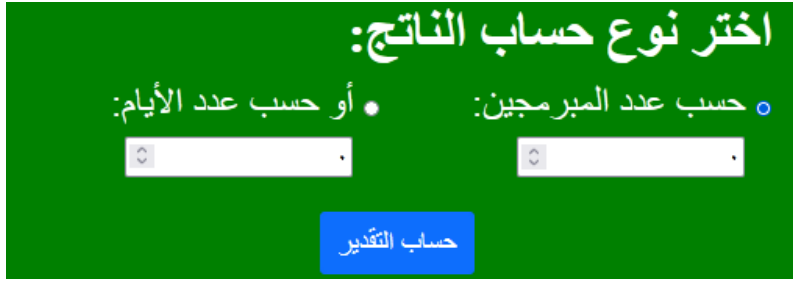
شكل رقم 9 توزيع العمل على طاقم يعمل جزئي

أما عندما يعمل الطاقم عمل كامل تكون النتيجة كالتالي:

تجهيز العتاد الصلب: لا يوجد
التصميم: يحتاج إلى 1 شخص/أشخاص خلال 5 يوم
قاعدة المعطيات: يحتاج إلى 1 شخص/أشخاص خلال 3 يوم
التحقيق البرمجي: يحتاج إلى 1 شخص/أشخاص خلال 25 يوم
اختبار المشروع: يحتاج إلى 1 شخص/أشخاص خلال 3 يوم
صيانة المشروع: يحتاج إلى 1 شخص/أشخاص خلال 3 يوم
نشر المشروع: يحتاج إلى 1 شخص خلال 1 يوم
المشروع بشكل كامل: يحتاج إلى 5 شخص/أشخاص خلال 37 يوم

شكل رقم 10 توزيع العمل على طاقم يعمل كامل

كما من الممكن تحديد عدد المبرمجين أو عدد الأيام واحتساب ناتج تقدير الجهد اعتماداً على ذلك، كالتالي:



شكل رقم 11 حساب التقدير

فرضاً تم تحديد عدد المبرمجين ب 5 مبرمجين، ستكون النتيجة كالتالي:



شكل رقم 12 حساب التقدير حسب عدد المبرمجين

وفي حال تحديد عدد الأيام ب 10 أيام، ستكون النتيجة كالتالي:

شكل رقم 13 حساب التقدير حسب عدد الأيام

من الضروري بعد ذلك الذهاب إلى تبويب COCOMO و Use Case Point لاحتساب تقدير الجهد حسب النموذجين ووضع النواتج ضمن واجهة المقارنة كالتالي:



شكل رقم 14 مقارنة النتائج

تم تنفيذ التجريب على مشروع آخر خاص بتوثيق أجهزة الصرافة²، وكانت النتائج كالتالي:

Effort Estimation: 39, COCOMO: 200, Use Case Point:370

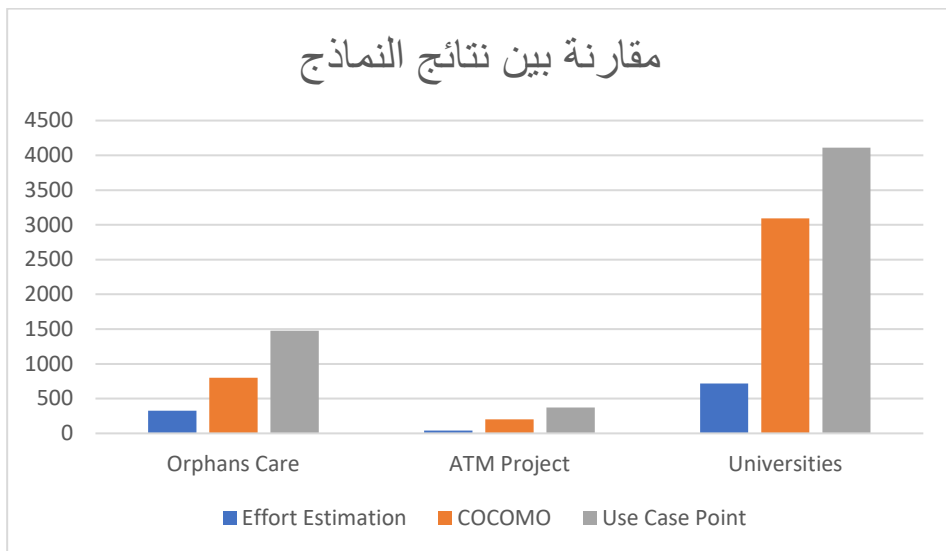
وأيضاً على مشروع دلالي للوصول إلى معلومات حول الجامعات³، وكانت النتائج كالتالي:

² https://github.com/rawannsr/ATM_Project

³ <https://github.com/rawannsr/Universities>

Effort Estimation: 715, COCOMO: 3091, Use Case Point: 4112

ليتم إظهار مخطط يبين الفروقات بين النواتج كالتالي:



شكل رقم 15 مخطط توضيحي لمقارنة النتائج

وكما هو ملاحظ، فقد توصلنا لفارق كبير بتحديد الجهد اللازم لإنجاز المشروع عن النموذجين المشهورين COCOMO، و Use Case Point. حيث تم تحقيق نسبة توفير بمقدار 60% عن نموذج COCOMO، ونسبة توفير بمقدار 78% عن نموذج Use Case Point بالنسبة لمشروع Orphans Care، أما بالنسبة لمشروع الصرافة، فتم توفير 81% عن نموذج COCOMO (حيث نتجت لدينا النسب وفق التالي: بتقسيم ناتج برنامجنا على ناتج النموذج فينتج لدينا نسبة هي نسبة إنجاز المشروع المطبق مقارنة بنتيجة النموذج الثاني وفق التالي: $0.195 = 200 \div 39$ وبالتالي نسبة التحسن تُحسب وفق $1 - 0.195 = 0.805$ وحتى نأخذها كنسبة مئوية نضرب بـ 100 فينتج 80.5% أي تقريبا 81%)، ونسبة توفير بمقدار 90% عن نموذج Use Case Point، (حيث طريقة حساب النسب السابقة هي نفسها كما ذكر في الأعلى)، ونسبة توفير بمقدار 90% عن نموذج Use Case Point. أما بالنسبة لمشروع الجامعات، فتم توفير 77% عن نموذج COCOMO، ونسبة توفير بمقدار 83% عن نموذج

Use Case Point. حيث أن هذا التوفير هو توفير الوقت المخصص لإنجاز المشروع.

7- الخاتمة والتوصيات:

قمنا في هذا البحث بالتوصل إلى خوارزمية تعمل على تخمين جهود فريق العمل للمشاريع البرمجية وذلك بعد إجراء استبيان فيه مجموعة كبيرة من الأسئلة تشمل جميع حيثيات المشروع من دراسة المتطلبات، تحليل، تصميم المشروع، التنفيذ، التحقيق البرمجي، الاختبار، نشر وصيانة المشروع، وتم عرضه على عدد كبير من الخبراء في مجال التطوير البرمجي، ثم استنتجنا بواسطته عدد كبير جداً من العوامل التي تؤثر على تقدير الجهود بالإضافة إلى تنقيحات هذه العوامل ثم بناء موقع برمجي للشركات البرمجية يحسب تقدير الجهود البرمجية لأي مشروع ويب وفق عدد من الأسئلة الدقيقة التي يعرضها الموقع عما يحتاجه المشروع من متطلبات دقيقة ثم يعرض لنا الجهد المتوقع لفريق العمل مع إمكانية تحديد عدد المبرمجين لإنجاز العمل عندها يعرض عدد الأيام التي يحتاجونها أو تحديد عدد الأيام التي يحتاجها المشروع لينتهي خلاله عندها يعرض عدد المبرمجين الذين نحتاجهم لإنجازه مع الأخذ بعين الاعتبار الدوام إن كان جزئي أم كامل، ثم قارنًا هذه الطريقة بمنهجين سابقين في هذا المجال وذلك بإدخال مجموعة كبيرة من المشاريع البرمجية وتم ذكر مشروعين برمجيين كمثالين على خوارزمتنا وعلى المنهجين السابقين وتوصلنا بواسطة خوارزمتنا إلى نتائج مُحسنة. يُنصح بتوفير خدمة حساب الجهد بما يخص الشيفرات المكتوبة بحساب عدد الأسطر البرمجية باعتبار البنى الشرطية و الحلقات التكرارية والاجرائيات المستخدمة، وذلك لكي يتم تحديد عدد الأسطر التي سيتم تنفيذها بأقرب تقدير ممكن. كما يُنصح بإضافة قدرات لفريق العمل، حيث أن المبرمج الخبير من شأنه أن ينجز العمل المؤكل إلى مبرمج متوسط بفارق كبير في الوقت، لذا من الجيد اعتبار الخبرة البرمجية لكل فرد من فريق العمل.

المراجع

1. JOWHRY, D., 2018 - **Effort Estimation of software development**. The Higher Institute for Applied Sciences and Technology.
2. BOEHM, B., 1981 - **Software Engineering Economics**. New York, 197.
3. RASK, R., LAAMANEN, P., & LYYTINEN, K., 1992 - **A Comparison of Albrecht's Function Point and Symons' Mark II Metrics**.
4. SAROHA, M., & SAHU, S., 2015 - **Tools & Methods for Software Effort Estimation Using Use Case Points Model—A Review**. In International Conference on Computing, Communication & Automation, pp. 874-879.
5. NAGESWARAN, S., 2001 - **Test Effort Estimation Using Use Case Points**. In Quality Week, Vol. 6, pp. 1-6.
6. IDRI, A., ABRAN, A., & KHOSHGOFTAAR, T. M., 2002 - **Estimating Software Project Effort by Analogy Based on Linguistic Values**. In Proceedings Eighth IEEE Symposium on Software Metrics, pp. 21-30.
7. SHARMA, A., & KUSHWAHA, D. S., 2012 - **Estimation of Software Development Effort from Requirements Based Complexity**. Procedia Technology, 4, 716-722.
8. KIRMANI, M. M., & WAHID, A., 2015 - **Revised Use Case Point (Re-UCP) Model for Software Effort Estimation**. International Journal of Advanced Computer Science and Applications, 6(3), 65-71.
9. PETROZZIELL, F. S., 2018 - **Linear Programming as a Baseline for Software Effort Estimation**. ACM Transactions on Software Engineering and Methodology.
10. PUTRI, R. R., SIAHAAN, D. O., & FATICHAH, C., 2021 - **Improve the Accuracy of Software Project Effort and Cost Estimates in COCOMO II Using GWO**. In ICICoS, pp. 128-133.
11. VERMA, A., & PREETI., 2021 - **Calibrating Intermediate COCOMO Model using Genetic Algorithm**. IEEE.
12. RIBDAWI, G., 2018 - **Software Engineering**. Syrian Virtual University.