المجلة العراقية للعلوم الإحصائية (29) 2019
عدد خاص بالمؤتمر الطلابي الاول
ص ص [73-90]

دراسة مراجعة في مقاييس البرمجيات الكيانية

مريم حسين مرعي اللويزي *
mh.white_rose91@yahoo.com

د. محمد عبد الغني طه الدباغ **
m.a.taha@uomosul.edu.iq

المستخلص

مما لا يخفى على أحد ان صناعة البرمجيات في العصر الحالي والمستقبل هي الصناعة الرائدة والعنصر الرئيس في أي تقنية جديدة وتدخل في اغلب الصناعات الحديثة من أصغرها كأجهزة الجوال والساعات الذكية الى اكبرها كالطائرات والمحطات الفضائية والمفاعلات النووية ولارتباط هذه البرمجيات بالحفاظ على حياة الانسان او انقاذها في بعض الأحيان فقد أصبح من المهم التأكد من جودتها وخلوها من الأخطاء فكان لابد من تطوير مقاييس لقياس جودة هذه البرمجيات. سيقدم هذا البحث دراسة مراجعة شاملة لأغلب المقاييس المستخدمة في قياس جودة البرمجيات والتأكيد على البرمجيات الكيانية التصميم والتنفيذ وذلك لأنها الرائدة الان في صناعة البرمجيات. وتعتبر هذه الدراسة بمثابة مرجع لطلبة هندسة البرمجيات المهتمين بمقاييس البرمجيات الكيانية.

الكلمات المفتاحية: جودة هذه البرمجيات، البرمجيات الكيانية

A Review of Object-Oriented Programming Software Metrics

Abstract

It is no secret that the software industry in the present and future time is the leading industry and the main element in any new technology and is involved in most modern industries, from the smallest ones such as mobile phones and smart watches to the largest ones such as airplanes, space stations and nuclear reactors, and the relation of this software to preserving human life or saving in sometimes It was important to make sure that it was quality and error-free. For this reason was must be to develop metrics to measure this software. This research will provide a comprehensive review of most of the standards used in software quality measurement and emphasis on software

* باحثة / قسم البرمجيات / كلية علوم الحاسوب والرياضيات / جامعة الموصل
** مدرس / قسم البرمجيات / كلية علوم الحاسوب والرياضيات / جامعة الموصل

تاريخ استلام البحث 2019/4/29 تأريخ القبول 2019 /7/7

design and implementation because it is now the leader in the software industry. This study serves as a reference for software engineering students interested in software standards.

1. المقدمة

من الاهداف الرئيسية لهندسة البرمجيات هو تطوير أنظمة برمجيات ذات جودة عالية، وان الجودة لا تتحقق الا اذا تم بناء مثل هذه الانظمة وفقاً لمعايير ومقاييس معينة لكي تكون ثابتة الأداء وقابلة للصيانة وغيرها من خصائص الجودة، ولذلك فانه من غير الممكن معرفة جودة البرمجيات من دون تطبيق مقاييس البرمجيات، حيث تعمل هذه المقاييس على دعم وتحسين جودة البرمجيات.

ففي الوقت الحاضر أصبح لمقاييس البرمجيات دور مهم وفعال ليس فقط في قياس الجودة وانما أيضا في قياس تعقيد البرمجيات وانه كلما كان الاعتماد على هذه المقاييس اكبر سيؤدي هذا الى انتاج برمجيات ذات جودة اعلى لكي تكون الجودة هي الفرق الرئيس بين منتجات البرمجيات.

وتعتبر مقاييس البرمجيات مهمة جدا في إدارة مشاريع البرمجيات فبدون معرفة جودة او تعقيد البرمجيات لا يمكن أكتشاف وتحديد المشاكل محتملة الحدوث اثناء عملية التطوير في وقت مبكر وقبل تسليم المشروع.

ويمكن القول بان المقاييس هي توفر مجموعة من النتائج المتوقعة او المخمنة والمحسوبة مسبقا ليتم مقارنتها مع النتائج الواقعية المستحصلة من تطبيق المقاييس على المشروع قيد التطوير لضمان تحقيق رضا الزبون بهذا المنتج. وبالتالي فان مقاييس البرمجيات تستخدم نسب رقمية لتحديد قيمة بعض الخصائص او الصفات لأي كيان برمجي.

ونظرا لعدم وجود مقاييس موحدة يمكن تطبيقها على جميع مراحل تطوير البرمجيات فقد تعددت المقاييس واختلفت أدوارها باختلاف الأشخاص الذين سيستخدمون هذه المقاييس فالجدول (1) يوضح بعض المقاييس المختلفة تبعاً لاختلاف أدوارها.

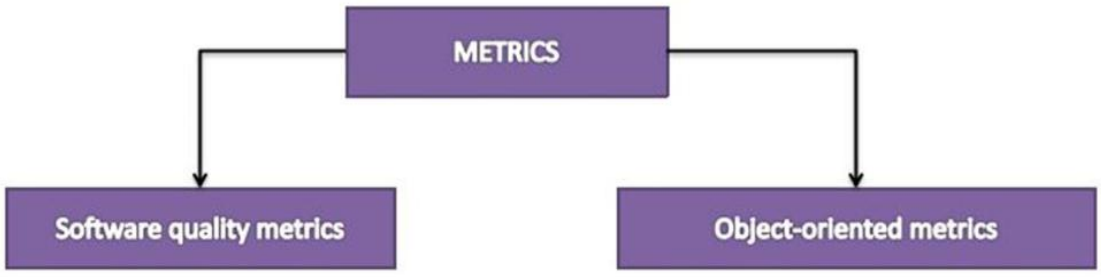
Role	Measurements
User	Usability, Simplicity, Stability, Cost…
Designer	Extendibility, Scalability, Manageability…
Programmer	Complexity, Maintainability…

الجدول (1) : يوضح مقاييس مختلفة تبعا لاختلاف ادوارها

ان المقاييس تستخدم من قبل جميع الأشخاص المهتمين بعملية تطوير البرمجيات من المدراء والمحللين والمصممين والمبرمجين وانتهاءاً بالزبون. فتبعا لاختلاف الشخص الذي يستخدم المقاييس سيتم استخدام مجموعة مختلفة من المقاييس لنفس المنتج البرمجي، فعلى سبيل المثال فان المدير الفني للمشروع سيكون مهتم بـ (عدد الاسطر البرمجية)، بينما يهتم المدير الإداري للمشروع بـ(تخمين عدد الساعات المطلوبة لكتابة نفس عدد الاسطر البرمجية) وذلك لغرض حساب انتاجية المبرمجين.

2. مقاييس البرمجيات

تعتبر مقاييس البرمجيات كأداة قوة في بحوث البرمجيات، وتستخدم كأداة تنبيه مبكر لاحتمالية حدوث مشاكل اثناء عملية تطوير البرمجيات ويجب تعريف كل مقياس على انه نموذج تطوير جودة QIP (quality improvement paradigm) التصميم الجيد والمتكامل (Krishnapriya, et al., 2010).



الشكل (1) : التصنيف العام للمقاييس(Mohamad Akram، 2016)

ولأنه لا يوجد مقاييس موحدة يمكن تطبيقها على جميع مراحل تطوير البرمجيات فان البحوث حول مقاييس البرمجيات عملية مستمرة لعدة عقود، وقد بدأ استخدامها في سبعينيات القرن الماضي باقتراح مقاييس لقياس تعقيد البرمجيات [نيد جابين]1979 (Chapin، 1979) وسرعان ما تغيرت باتجاه قياس الإنتاجية وتقييم وثوقية النماذج وتطورت مؤخراً لقياس قدرة النضج الى ان وصل التطور لوقتنا الحاضر حيث ظهرت مقاييس تعقيد البرمجيات الكيانية.

قام الباحثون الين لوبس تيموتيو واخرون باستعراض ملخص لبحوث مقاييس البرمجيات بالاعتماد على شفرة المصدر (source code)، حيث تم تقسيم الملخص الى فترتين زمنيتين: الأولى ماقبل عام 1991، عندما كان التركيز على المقاييس المعتمدة على تعقيد الشفرة، والفترة الثانية هي مابعد عام 1992 حيث اصبح التركيز على المقاييس المعتمدة على مفاهيم البرمجة الكيانية للأنظمة (Object Oriented systems) تصميماً وتمثيلاً (Timóteo، 2014).

قام الباحثان يو ولامب (Liu، 1999) بدراسة حول مقاييس البرمجيات للسنوات 1978 – 1991 على سبيل المثال: مقياس جابينس كيو يركز على دور البيانات المدخلة والمخرجة لـ (modules) للتصميم المهيكل.

وقام الباحث ميكابي في عام 1989 (McCabe، وآخرون، 1989) وايضا الباحثان ريلي كاب و بوتلر بتطبيق مقياس ميكابي (cyclomatic complexity) للتصميم المهيكل واقترحا ان أجزاء مخطط سريان الأجراءات(procedure's flow graph) (غير المتضمنة في استدعاءات الإجراءات الأخرى) لا تشارك في تعقيد التصميم (Liu، 1999) .

وتشير مقاييس هنري و كافورا لسريان المعلومات الى ان العامل الرئيس لتحديد التعقيد الهيكلي (structural complexity) هو الاتصال او التواصل بين الاجراء وبيئته، وان الإجراءات الأكثر تعقيدا في النظام هي ذات الكمية الأكبر من سريان المعلومات(information flow) (Liu، 1999).

في عام 2003م قامت الباحثة هيلدا بي كلاسكاي (Klasky، 2003)بتقديم دراسة وتمثيل لمختلف مقاييس البرمجيات واوضحت انه يوجد مقاييس محددة لكل مرحلة في دورة تطوير البرمجيات. وعند استخدام مقاييس افضل فان جودة البرمجيات ستزداد تلقائياً. ومن امثلة المقاييس المدروسة في مرحلة المتطلبات: سير العمل (workflow)، خصوصية واكتمال المتطلبات (the Specificity and Completeness of Requirements). وفي مرحلة التصميم: التعقيد السايكلوماتيكي

(Cyclomatic Complexity)، النقاط الوظيفية (Function Points)، سير المعلومات (Information Flow) ومقياس بانج (Bang Metric). وفي مرحلة التمثيل: تخمين 47 من عدد العيوب (the 47 Estimation of Number of Defects)، عدد الاسطر البرمجية (Lines Of Code (LOC) ومقياس هالستيد (Halstead Metrics). وفي مرحلة الفحص والاعتماد: عدد العيوب وتخمين عدد حالات الاختبار وغيرها من المقاييس الاخرى لتخمين الجهد مثل COCOMO II.

في عام 2004م قام الباحث ارمن كروسكو (Krusko، 2004) بتحليل تعقيد برمجيات الزمن الحقيقي، وكان الهدف هو تحديد اي من المقاييس تكون متعلقة بجودة برمجيات الزمن الحقيقي وتحديد اعلى/ أقل قيمة لها، وسيؤدي فرض هذه القيم الى تحسين عملية التطوير وبالتالي تحسين الجودة. وتم التعرف على 8 مقاييس للتعقيد واجراء التحليل لحساب القيم العليا(Max) لها. وتم ايجاد قيمة جديدة تسمى قيمة الحد الاعلى (upper-limit value) التي تشير الى اكثر الوحدات المعرضة للخطأ لكل المقاييس المختارة.

في عام 2014م قام الباحثان اميت كومار وجاكهار (Jakhar, Amit Kumar; Rajnish, Kumar, 2014) بنمذجة احد اهم الخصائص الأساسية لتعقيد البرمجيات عن طريق دراسة الاوزان المعرفية لهياكل التحكم الأساسية في البرمجيات، وتم تطوير مفهوم جديد لطريقة التعقيد الموزونة للبرمجيات((New Weighted Method Complexity (NWMC) ، حيث تم توزيع 20 برنامج لخمسة طلاب دراسات عليا وتم ملاحظة وقت التطوير لهم جميعا وتم اعتبار معدل الوقت بانة الوقت الحقيقي اللازم لتطوير البرنامج وفهم الشفرة ،وقد استند هذا البحث في دراسته على الحجم الوظيفي المعرفي((Cognitive Functional Size (CFS) للبرمجيات، وللتحقق من صحة المقاييس الجديدة للتعقيد تم حساب الارتباط بين المقاييس المقترحة والحجم الوظيفي المعرفي (CFS) فيما يتعلق بوقت التطوير الفعلي، وتم اجراء تحليل لطريقة التعقيد الموزونة الجديدة(NWMC) والحجم الوظيفي المعرفي (CFS)، معدل الخطأ النسبي(MRE) و الانحراف المعياري(.Std) . وفي النهاية وجد انه دقة تخمين وقت التطوير بالطريقة المقترحة افضل بكثير من الحجم الوظيفي المعرفي.

في عام2018م قام الباحثون ميشيل لانزا واخرون(Frunzio, et al., 2018) بتقديم إضافة(plugin) ((REal TIme Code qUaLity Assessment (RETICULA تقييم جودة شفرة

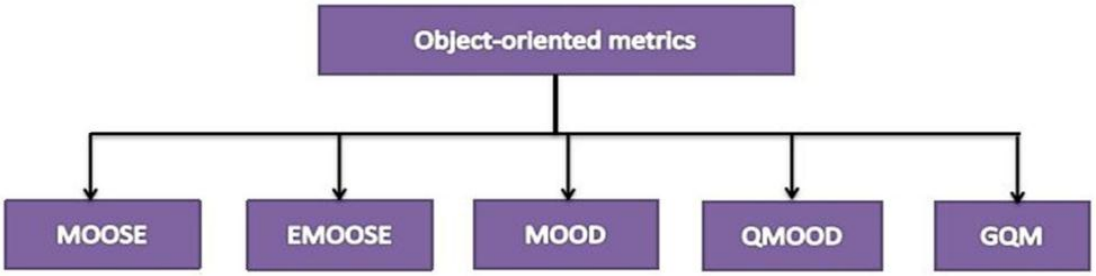
الزمن الحقيقي) لمساعدة المطورين في ادارك جودة الشفرة اثناء تطوير البرمجيات حيث تقوم الأداة بمقارنة جودة المشروع قيد التطوير مع الأنظمة ذات المصدر المفتوح والمشابهة لها والتي تم تحليلها مسبقا مع نتائج مرئية ومن ميزات هذه الإضافة انها توفر حاسبة مقاييس قابلة للتخصص وفلاتر للمشاريع والتي تمكن المطورين من مقارنة جودة شفرتهم مع تلك المشاريع المشابهة لها والمفتوحة المصدر ويمكن لهذه الأداة ان تكون بمثابة ارشاد للمطورين لمراقبة جودة الشفرة بصورة مستمرة

3. البرمجيات الكيانية

ظهرت البرمجة الكيانية في بداية تسعينيات القرن الماضي نظراً لتطور وتعقيد أنظمة البرمجيات فكان لا بد من إيجاد مقاييس جديدة تتلائم مع هذا التطور الزمني حيث اتجه اهتمام الباحثين لأيجاد مقاييس جديدة تتناسب مع مفهوم التصميم الكائني بدلا من المقاييس التقليدية وذلك لمساعدة المصممين في تقييم ومعرفة مدى تعقيد وجودة تصميم البرمجيات الكيانية وكانت أولى المحاولات لايجاد مقاييس البرمجيات الكيانية عام 1991 للباحثَين كيدامبير وكيمرير (Chidamber, et al., 1991).

فالبرمجة الكيانية هي محاولة لجعل البرامج اقرب الى الفهم الدقيق للمشكلة، فهي تصور للناس طريقة التفكير والتعامل مع العالم من حولهم، في السابق عندما كان المبرمج يواجه معوقات معينة فإنه يحتاج تعريف مهمة حساب (computing task) لحل المشكلة، ويحتاج الى سلسلة من الايعازات لاتمام هذه المهمة، ولكن عند الاعتماد على مبدأ البرمجة الكيانية فيوجد كيانات_تواجدات(objects–entities) بدلاً من المهام، وهذه الكيانات لديها تصرفات (behaviors)، حفظ المعلومات (hold information) ويمكنها التفاعل(interact) مع بعضها البعض. وعليه فتكون البرمجة مكونة من تصميم مجموعة من الكيانات التي ستتكفل بحل المشكلة المعنية. وهذا يمنح مصمم البرنامج اكثر واقعية وسهولة في الفهم (Sarker، 2005).

ويوجد هنالك انواع مختلفة من مقاييس البرمجة الكيانية منذ ظهورها في نهايات القرن الماضي واستمر تطويرها مع مرور الزمن الى وقتنا الحاضر كما في الشكل (2) الذي يوضح أشهر أنواع مقاييس البرمجيات الكيانية.



الشكل (2) : تصنيف المقاييس الكيانية(Mohamad Akram، 2016)

ان التوجه الكياني Object-Orientation هو ملائمة جيدة لـ "سلاسة التطوير". فالكيانات (Objects) هي تغليف للمعلومات والتصرفات information and behavior لبعض التواجدات في مجال التطبيق المأخوذ بعين الاعتبار، وفي الأنظمة الحقيقية يمكنك ان تجد العديد من الكيانات والتي يمكن ان تتشابه من حيث البيانات والوظائف. والصنف class هو مختصر يصور واقعية هذه الكيانات، ويمكن ان يُنظر له كمختصر لنوع البيانات abstract data type . وتعريف الصنف يتضمن على الأقل نوعين من الخصائص: الصفات attributes التي تعني البيانات المخزونة (المغلفة ضمن الصنف) والدوال methods تمثل التصرف (الدوال التي تؤدي وظيفة هذا الصنف). يمكن تعريف العديد من هذه الخصائص ضمن نفس الصنف لإتمام وظائف معينة خاصة بهذا الصنف، وهذه الخصائص يجب ان تكون مخفية عن المبرمجين الطرفيينclient programmers وهم المبرمجون ضمن فريق التطوير والذين يستخدمون صنفاً لم يقوموا بانشائه بأنفسهم (Fernando، وآخرون، 1994).

التوجه الكياني هو مجموعة من الادوات او الطرق التي تمكن مهندس البرمجيات من بناء نظام برمجي موثوق، قابل للصيانة، موثق بصورة جيدة، قابل لإعادة الاستخدام وصديق للمستخدم user friendly حيث يلبي جميع متطلباته. والتوجه الكياني يزود نظرة جديدة للحسابات. والنظام البرمجي يُرى على انه مجموعة من الكيانات التي تتعاون عن طريق تمرير الرسائل passing messages فيما بينها لحل المشكلة.

4. خصائص البرمجيات الكيانية

ان اكثر ما يميز البرمجيات الكيانية هو ان لديها بعض الخصائص او الصفات التي تميزها عن غيرها من البرمجيات ولأجل هذا زاد انتشارها والاعتماد عليها بشكل كبير في الوقت الحاضر لما تقدمه

من تسهيلات لمطوري البرمجيات تساعدهم في انجاز مهامهم بصورة ادق وأكفأ وفيما يلي استعراض لأابرز هذه الخصائص:

1.5. التغليف واخفاء المعلومات Encapsulation and Information Hiding

وكما ذُكر سابقا بان البرمجة الكيانية هي عبارة عن مجموعة من الكيانات وان الكيان هو بحد ذاته تغليف للصفات والدوال الخاصة بصنف معين، فيمكن تعريف اكثر من خاصية ضمن نفس الصنف لتؤدي وظيفة معينة تكون خاصة بالصنف نفسه ويجب ان تكون مخفية عن المبرمجين العملاء، وان الفائدة المرجوة من كل هذا لتساعدهم على عدم الانشغال بالتعقيد الداخلي للوظيفة المصممة، والفائدة الاخرى هي بان يكون الصنف مستقلا في بنائه أي انه يسمح بالتعديل بدون أي تأثيرات جانبية.

2.4. الوراثة Inheritance

هي طريقة للتعبير عن التشابه بين الأصناف، حيث تسمح بتصوير مبدأ التعميم والتخصيص، وبخصوص التصميم فانها تسمح بتبسيط تعريف وراثة الأصناف، أي ان الصنف عندما يُرث من صنف اخر فانه يستطيع استخدام الدوال والصفات الخاصة بالصنف الأول مالم يتم تعريفهما بنطاق خاص.

3.4. الارتباط والتكتل Coupling and Clustering

التكتل هي اليه تمكن الباحثين من تجميع بيانات الكيانات المتشابهة فينا بينها بكتلة واحدة، ويجب ان يكون التقارب الداخلي للكتل عالي جدا بينما يكون التقارب بين الكتل المنفصلة قليل جداً، بمعنى اخر يجب ان ترتبط مجموعة نقاط البيانات العائدة لنفس الكتلة ارتباطا وثيقاً ، بينما لاتكون البيانات العائدة لكتل مختلفة مرتبطة فيما بينها (Kaushik, et al., 2015).

4.4. التعددية الشكلية Polymorphism

وتعني " تعدد الصيغ". ففي البرمجة الكيانية تشير الى إمكانية إرسال رسالة دون معرفة ما هي الصيغة (الصنف) الخاص بالكائن الذي سيربطbind تلك الرسالة بأحد أساليب الواجهة الخاصة به interface methods. حيث تنتمي جميع الاصناف المحتملة لاستقبال الرسالة إلى نفس شجرة الوراثة الهرمية. ويمكن أن يكون الربط ثابتًا (في وقت التجميع) أو ديناميكيًا (في وقت التشغيل)

(Fernando، وآخرون، 1994). هنالك نوعين من التعددية الشكلة هما: تعددية التحميلOverloading التي تسمح بوجود أكثر من دالة بنفس الاسم مع اختلاف المعلماتparameters في عددها او نوع تعريفها. والنوع الثاني هو تعددية التجاوزOverriding والذي يعني تغير التصرف للدالة المعينة والعائدة للصنف الرئيسي عن طريق دالة أخرى عائدة للصنف المشتق.

5.4. إعادة الاستخدام Reuse

ان إعادة الاستخدام لها تأثيراً كبيراً في إنتاجية وجودة التطوير، فهي توفر الكثير من الوقت وتقلل من تكاليف النظام، وذلك لان المكونات التي تكون قابلة لاعادة الاستخدام غالبا ماتكون مصممة بعناية فائقة اكثر من البرامج العادية، حيث ان الاستخدام المتكرر لهذه الكونات سيبرز من العيوب المتكررة في التصميم او التمثيل، وهذا السبب الذي يجعلها ذات جودة عالية وهذه الجودة ستكون مضمنة في الأنظمة التي ستقوم بدمج هذه المكونات. وان إعادة الاستخدام تكون بشكل رئيسي اما عن طريق إعادة استخدام مكونات مكتبية library components او إعادة الاستخدام عن طريق الوراثة inheritance (Fernando، وآخرون، 1994).

6.4 . مقاييس البرمجة الكيانية

وبعد ما تم طرحه بخصوص البرمجة الكيانية وخصائصها التي تميزها عن غيرها من النماذج فان لكل خاصية من هذه الخصائص هنالك مقياس يحدد قيمته العددية وان مقاييس تصميم البرمجيات الكيانيةMOOD Metrics for Object Oriented design هي من اهم المقاييس التي تدعم معظم خصائص البرمجة الكيانية والتي اقُترحت من قبل الباحث فيرناندو بريتو عام 1994 (Fernando، وآخرون، 1994) م لزيادة جودة البرمجيات، حيث ان المقياسان (MHF , AHF) يتعلقان بتغليف البيانات و(MIF , AIF) يتعلقان بالوراثة، و(COF , CLF) يتعلقان بتمرير الرسائل، و(POF) التعددية الشكلية، و(RF) لإعادة الاستخدام، ويمكن تلخيص هذه المقاييس كما يلي (Deepak، وآخرون، 2011):

- عامل إخفاء الدوال Method Hiding Factor (MHF) كما في المعادلة رقم (1).
- عامل إخفاء الصفات Attribute Hiding Factor (AHF) كما في المعادلة رقم (2).
- عامل وراثة الدوال Method Inheritance Factor (MIF) كما في المعادلة رقم (3).

- عامل وراثة الصفات Attribute Inheritance Factor (AIF) كما في المعادلة رقم (4).
- عامل الارتباط Coupling Factor (COF) كما في المعادلة رقم (5).
- عامل التكتل Clustering Factor (CLF) كما في المعادلة رقم (6).
- عامل التعددية الشكلية Polymorphism Factor (PF) كما في المعادلة رقم (7).
- عامل إعادة الاستخدام Reuse Factor (RF) كما في المعادلة رقم (8)

ويمكن حساب قيمة كل عامل من هذه العوامل وفقاً للمعادلات التالية (Fernando، وآخرون، 1994):

رقمها	المعادلة
1	$$MHF = \frac{\sum_{i=1}^{TC} M_h(C_i)}{\sum_{i=1}^{TC} M_d(C_i)}$$
2	$$AHF = \frac{\sum_{i=1}^{TC} A_h(C_i)}{\sum_{i=1}^{TC} A_d(C_i)}$$
3	$$MIF = \frac{TM_i}{TM_a}$$
4	$$AIF = \frac{TA_i}{TA_a}$$

5	$$COF = \frac{\sum_{i=1}^{TC}\left[\sum_{j=1}^{TC} is\_client(C_i, C_j)\right]}{TC^2 - TC}$$
6	$$CLF = \frac{TCC}{TC}$$
7	$$PF = \frac{\sum_{i=1}^{TC}\left[\sum_{j=1}^{DC(C_i)} M_o(C_j)\right]}{\sum_{i=1}^{TC}\left[M_d(C_i) * DC(C_i)\right]}$$
8	$$= \frac{\sum_{i=1}^{TC} in\_library(C_i)}{TC} + \frac{MIF * \sum_{i=1}^{TC}\left[1 - in\_library(C_i)\right]}{TC}$$

الجدول (2): يوضح معادلات مقاييس البرمجيات الكيانية MOOD

5. الاعمال المتعلقة بمقاييس البرمجيات الكيانية

في عام 1996م قام كل من الباحثين كارين ارني وكلاوس ليورنتز (Erni, et al., 1996) بتفصيل استخدام نموذج المقاييس المتعددة (multi-metrics approach) لتصميم وتحسين اطار العمل للصناعة واستخدامه في التطبيقات الرسومية. استخدمت أدوات تعدد المقاييس مع عدة إصدارات لاطار العمل. واستخدمت نتائج القياس في مراجعة التصميم لتحديد تأثير الجهد لإعادة التنظيم اطار العمل. واظهرت النتائج ان هذا النهج كان فعالا للغاية في إعطاء تغذية عكسية حتى لخبراء مطوري البرمجيات وتساعدهم على تحسين برامجهم وانشاء تصاميم ثابتة للنظام.

في عام 2005م قام الباحث مكتامي ساركر بدراسة مجموعة من المقاييس لمعرفة جودة التصميم للبرمجيات الكيانية . حيث قام بدراسة متقنة لمقاييس (CK) شيدامبير وكامرير (Sarker,

(2005.والاعمال المشابهة لهذا العمل والتي اقترحت خمسة من اصل ست مقاييس للـ (CK) وهي (WMCالدوال الموزونة للصنف وDITعمق شجرة الوراثة وRFC الاستجابة للصنف وNOCعدد الابناء و CBO الترابط بين الكيانات) بأنها مؤشرات مفيدة للجودة للتنبؤ بالأصناف المعرضة للفشل، واقترح بأن يتم فقط استخدام المقاييس التي تم التحقق منها تجريبيا، ونصح بان تكون المقاييس بسيطة وقابلة للحساب وبلغة برمجية مستقلة وان يكون دائما هنالك شيء جديد لقياسه.

وفي عام 2005م قام الباحث رادو مارينسيو (Marinescu, 2005).بتقديم الية جديدة تسمى ستراتيجية الكشف لأجل زيادة ملائمة امكانية استخدام المقاييس في التصميم الكائني المنحى من خلال توفير وسائل ذات مستوى اعلى لتفسير نتائج القياس وتم تعريف ستراتيجية الكشف على النحو التالي (بانه تعبير قابلية تقييم القانون من قبل هيكل التصميم الذي يتلائم (يتجانس) مع ذلك القانون يمكن معرفته في الشفرة المصدرية). وكان الهدف الرئيسي من الستراتيجة هو تزويد مهندس البرمجيات باليه تسمح له بالعمل مع المقاييس على مستوى اكثر تجريدية وهو اقرب من الناجية النظرية الى نواياه الحقيقة في استخدام المقاييس. قد تشير هذه القوانين الى فهم وتحديد عيوب التصميم.

وفي عام 2006م قام الباحث عبد اللطيف الاحمدي(El-Ahmadi, 2006) بتحليل مفهوم جودة البرمجيات وكيف يتم رؤيتها من قبل انواع مختلفة من مطورين البرمجيات في تعريف (Maersk Data Defense) حيث اوجد اداة بامكانها استخلاص وحساب مقاييس مختلفة للبرمجيات وأوضح اذا كانت تُستخدم في عملية التطوير أم لا. وقام بتطوير نظام يمكن استخدامه كأساس لاستخلاص وتقييم كل المقاييس لـ (Maersk Data Defense) ويجب ان يكون جزء من عملية البناء والتطوير. يسمى هذا النظام ببرنامج ملائمة المقياس (MCP) ويدمج مع مقاييس برمجيات خارجية التي تحسب مقاييس البرمجيات ومزود بواجهة صورية للمستخدم ولمختلف انواع النشاطات.

وفي عام 2011 م قام كل من الباحثون ديباك اورا (Deepak, et al., 2011) واخرون بدراسة ثلاث مقاييس للبرمجيات الكيانية وهي مقاييس تصميم البرمجيات الكيانية MOOD و CK و QMOOD وقدموا دراسة حالة توضح فائدة هذه المقاييس في تحديد جودة اي برنامج مصمم بنموذج كائنية المنحى وهذه المقاييس هي MOOD: ومنها MHF وAHF العاملان اللذان يتعلقان بتغليف البيانات للدوال والصفات، وMIF وAIF عاملان لوراثة الدوال والصفات وزيادتها تجعل البرنامج اقل فهما واقل قابلية للصيانة وPF عامل التعددية الشكلية اي اعادة استخدام الشفرة. ومقاييس CK:

هيDIT يشير لعمق شجرة الوراثة وNOC يشير لعدد الأبناء وMPC ارتباطية تمرير الرسائل. ومقاييس QMOOD: ومنها NOA عدد الصفات، و NOM عدد الدوال، وANA معدل عدد الإباء، ووجدوا ان تصميم اصناف استدعاء الدوال عن بعد (JAVA RMI) قد اجتاز معايير الجودة المختلفة. ويظهر خصائص تصميم جيدة. فقد تم اختيار مجموعة محددة من المقاييس التي تعتبر هي الاهم للإشارة الى استدعاء جافا للدوال عن بعد والتي تظهر تبني جيد للتغيرات الجديدة ودرجة اعلى من قابلية التوسيع والقدرة الفعالة لتمرير الرسائل.

في عام 2011م قام الباحثان نيشا مالك وراجتير جيلار (Nisha, et al., 2011).باقتراح اربع مقاييس للتصميم على مستوى الصنف للبرمجيات الكيانية وهذه المقاييس هي : قياس تعقيد عضوية الصنف(CMCM) و قياس تعقيد وراثة الصنف(CICM) وقياس مستوى تجميع الصنف(CALM) وقياس تماسك الصنف (CCOM) و التي تقيس التعقيد وفقا لبعض مفاهيم البرمجيات كائنية المنحى وهي قياس إخفاء البيانات والوراثة والتجميع والتماسك ، تم تقييم هذه المقاييس الأربعة عن طريق مخرجات مرحلة التصميم أي قبل البدء بمرحلة البرمجة لتقليل الجهد الكلي من ناحية الوقت والكلفة والقوى العاملة. ووجد ان القيمة الأصغر لهذه المقاييس تشير الى ان المصمم لم يصرف الجهد الكافي لإعادة استخدام عناصر البيانات والوظائف للصنف.

في عام 2012م قام الباحث مصطفى غانم الحيالي (Al-heyalley, 2012).باقتراح نموذج لقياس جودة البرمجيات بتصنيف التعقيد بثلاث مستويات من التحليل وهي: مستوى الحزمة البرمجية (Program Package) ومستوى الصنف (Program Class) ومستوى الدالة (Program Method)، وتم ذلك بالإعتماد على مقاييس تعقيدات الشفرة البرمجية المصدرية (Source Code Complexity Metrics) وحدود العتبة (Threshold Limits)، وتمت دراسة مفهوم الشفرة البرمجية النظيفة(Clean Code) واثرها في تطوير البرمجيات. وقام بنمذجة وبناء اداة (AECC_تحليل وتقييم الشفرة النظيفة) حيث استخدمت هذه الأداة لتقييم جودة الشفرة البرمجية المكتوبة بلغة جافا بشكل تلقائي، وتدعم الاداة اسلوب التحليل الرسومي لمحتوى البرمجيات والتي تعطي نظرة للمهندس عن مدى تعقيد هذه البرمجيات

وفي عام 2013م قام الباحث توفيق مقداد توفيق (Tawfeeq, 2013) باقتراح بناء اداة هندسة البرمجيات بمساعدة الحاسوب اطلق عليها اسم (Transmigration of OOP to AOP

(Tool(TOAT) اداة ترحيل البرمجة الكائنية المنحى الى اسلوب البرمجة جانبية المنحى. ودمجت هذه الاداة ما بين عمل الهندسة العكسية واعادة التنظيم. الهندسة العكسية لعرض تفاصيل البرمجيات الكبيرة بشكل خالي من الغموض والتعقيد، واعادة التنظيم للشفرة المصدرية عن طريق تطبيق البرمجة جانبية المنحى على الكل المتقاطعة والمرشحة لتكون جانبا، وتم اختبار الاداة TOAT وتقييم اداءها و قياس جودة البرمجيات قبل وبعد اعادة التنظيم (ترحيل الشفرة المصدرية من البرمجة كائنية المنحى الى جانبية المنحى) باستخدام مقياسي : Degree Of Scattering (DOS) و Degree Of Tangling (DOT) لقياس تناثر وتشابك الشفرة المصدرية للبرمجيات حيث اظهرت النتائج على ان البرمجة جانبية المنحى قد تغلبت على مشكلتي التشابك والتناثر في الشفرة المصدرية للبرمجيات.

وفي عام 2013م قام الباحث خليل ابراهيم الحديدي (Al-Hadidi, 2013) ببناء اداة للتحقق من جودة التصميم كائني المنحى. حيث قام بتطوير ثلاث ادوات الاولى (KDM) تاخذ وثيقة XML كادخال وتعتبر اخراج لمخطط الصنف تعالجها وتقوم باحتساب وتصوير المقاييس وتقدم التوصيات حول التسميات الخاصة بالتصميم وتقوم ايضاً بتصدير المقاييس كوثيقة XML من اجل التواصل مع الأدوات الاخرى وهما KRS وKDB، والاداة الثانية هي KRS التي يكون ادخالها وثيقة XML حيث تقوم بمعالجتها وتقديم تقرير كإخراج لها، حيث يساعد هذا التقرير في مراقبة العمل وتوثيق المقاييس. والاداة الثالثة هي KDB ويكون إدخالها هو وثيقة الـ XML نفسها وتقوم بمعالجتها وخزن المقاييس في قاعدة بيانات للاستفادة في مقارنة مشاريع لاحقة، وان الأدوات الثلاثة متكاملة مع برنامج الـ (Enterprise Architect)، وطورت هذا الأدوات باللغة البرمجية C# . حيث تم استخدام نموذجين لقياس التصميم الكائني المنحى هما: MOOD وهو النموذج الخاص بقياس التصميم الكائني المنحى والذي يقيس التغليف، الوراثة، تعددية الاشكال والارتباط. والنموذج الثاني هو MEMOOD وهو الذي يقيم قابلية الصيانة للبرمجيات في مرحلة التصميم. يقيس هذا النموذج قابلية الفهم، قابلية التعديل وقابلية الصيانة. وتم اثبات النموذجين نظرياً وتجريبياً.

وفي العام 2013م قام الباحثون براديب كومار واخرون (Pradeep Kumar, 2013) بتقديم نظرة مختصرة عن جودة البرمجيات، طرق مقاييس هندسة البرمجيات التي تتنبأ وتقيس عوامل الجودة المحددة للبرمجيات، وهي أيضا تناقش الجودة المحددة وفقا للمعايير القياسية مثل ISO، العناصر الأساسية المطلوبة لمقاييس البرمجيات وجودتها كأسلوب قياس للتنبؤ بجودة البرمجيات. تم تنفيذ هذا البحث من خلال تقييم شفرة المصدر التي تم تطويرها بلغة جافا Java، باستخدام مقاييس البرمجيات،

مثل مقاييس الحجم (Metrics Size) ومقاييس التعقيد(Complexity Metrics) ومقاييس العيوب (Defect Metrics). واظهرت النتائج أنه باستخدام مقاييس البرمجيات يمكن تحليل جودة البرمجيات ودراستها وتحسينها.

وفي عام 2013 قام الباحثان دير سونال وجاجان ديب(Dr Sonal Chawla, 2013) بوصف أنواع مختلفة من المقاييس تماشيا مع مقاييس الشفرة الثابتة ومقاييس كائنية المنحى، حيث تم تلخيص المقاييس على أساس ملائمتها في إيجاد التعقيد ومساعدتها في قابلية صيانه شفرة البرنامج ، مع الحفاظ على الجودة وجعل كلفتها مناسبة، ومع التقدم في صناعة البرمجيات لابما كان اسلوب قياس جودة البرمجيات معقدا لذلك ومع تطور الزمن ازداد الحاجة لتطوير مقاييس افضل لاتها تلعب دورا مهما في تحديد التعقيد وقابلية الصيانة للشفرة البرمجية، ولهذا استنتج الباحثان انه يجب اجراء دراسة استقصائية مناسبة لاختيار افضل المقاييس للشفرة، مع وصف للخصائص المهمة لكل مقياس مثل : كيفية الاستخدام، وتفسير المبادئ الارشادية، وحدود العتبة إن أمكن، وبذلك يتم تقدير فائدتها وملائمتها . وكل هذا سيساعد في الارشاد على انتاج برمجيات رصينة وذات جودة عالية مما يعزز احتمالية إعادة استخدام البرمجيات وتقليل كلفة صيانتها.

في عام 2013 قام الباحثان خليل احمد إبراهيم ود.لهيب الزبيدي (Laheeb M. Al-Zobaidy, 2013) بتقديم دراسة ومقارنة مجموعة من المقاييس الكائنية التوجه التي يمكن استخدامه القياس جودة التصميم، حيث تم دراسة المقاييس الحالية للتصميم C&K و MOOD و L&K والتركيز على نموذج MOOD.

في عام 2015م قام الباحث وون دي واخرون (Wu, et al., 2015) بدراسة مقارنة بين لغات البرمجة (C++ و JAVA و C#) لايجاد اللغة الأفضل في كتابة البرمجيات الكيانية ووجدوا ان C++اكبر من ناحية حجم الاصناف(classes size) من JAVA و C# , وان C# اكثر اقترانا (Coupling) من JAVA و C++، بينما C++ اقل تماسكا (Cohesion) من JAVA و C#، وان JAVA و C# تتفوق على C++ من ناحية بناء شجرة وراثة ذات عمق اكبر (deep inheritance trees)، ولا يوجد اختلاف كبير من ناحية التعددية الشكلية وإعادة الاستخدام والتغليف.

وضمن نفس المجال في عام 2018م قام الباحثون موهيت شارما واخرون (Mohit Kumar Sharma, 2018) بتحليل مقاييس مختلفة كائنية المنحى والتي تكون مفيدة في ضمان التصميم

بتماسك قوي وارتباط قليل، حيث تمت دراسة مقاييس مختلفة في بحوث مختلفة ومن بينها C&K و MOOD و L&K، وكان الاختلاف بين البحثين (Al-Zobaidy, 2013) و (Mohit Kumar Sharma, 2018) في مقاييس L&K حيث تضمن البحث الأول 11 مقياساً غير ان البحث اللاحق لم يشمل سوى 6 مقاييس.

وأخيرا في عام 2018 قام الباحثان مروة نجم ود.رياض زغلول (Jader, et al., 2018) باقتراح طريقة محسنة لمقياس تعقيد البرمجيات الكيانية وذلك بحساب قيمة مقياس مكابي (CC) بالاعتماد على جمل القرار (decisional statements) وأيضا حساب قيمة مقياس الاقتران بين الكيانات (CBO) بعدة حالات مختلفة وذلك للحصول على قيمة جديدة ناتجة من دمج هذين المقياسين حيث سميت النتيجة النهائية بـ (TCC) (Total Cyclomatic Complexity).

المصادر References :

A.Aloysius & L. Arockiam, 2011 , "A Survey on Metric of Software Cognitive Complexity for OO design", World Academy of Science, Engineering and Technology, International Journal of Computer and Information Engineering, Vol:5, No:10.

Abreu Fernando Britoe & Carapuça Rogério, October -1994, "Object-Oriented Software Engineering: Measuring and Controlling the Development Process" ,4th Int. Conf. on Software Quality.

Al-Hadidi Khalil Ahmed Ibrahim, 2013, "Constructing a Tool for Measuring a Quality of Object Oriented Design to Enterprise Architect", Master Thesis, University of Mosul, Mosul -Iraq.

Al-heyalley Mostafa Ghanem Saeed, 2012, "The Automated Analysis of Source Code Complexity in Software" , Master Thesis ,University of Mosul, Mosul -Iraq.

Al-Zobaidy Laheeb M., Ibrahim Khalil A., 2013, "Existing Object Oriented Design Metrics a Study and Comparison" ,Fifth Scientific Conference Information Technology, 2012 Dec. 19-20. - 1 : Vol. 10.

Chapin Ned, 1979, "A measure of software complexity ", National Computer Conference , Menlo Park, California.

Chawla Dr Sonal, Kaur Gagandeep, 2013, "Comparative Study of the Software Metrics for the complexity and Maintainability of Software Development", (IJACSA) International Journal of Advanced Computer Science and Applications, No. 9 , Vol. 4.

Dallal Jehad Al And Briand Lionel C, 2012, "A Precise Method-Method Interaction-Based Cohesion Metric for Object-Oriented Classes", ACM Transactions on Software Engineering and Methodology, Vol. 21 No. 2.

Deepak Arora, Khanna Pooja, Tripathi Alpika, Sharma Shipra & Shukla Sanchika, April-4-2011, "Software Quality Estimation through Object Oriented Design Metrics" IJCSNS International Journal of Computer Science and Network Security, Vol. 11.

Eck David J, 2014, "Introduction to Programming Using Java", Vol. Version 7.0, Hobart and William Smith Colleges ,p. 189.

El-Ahmadi Abdellatif, 2006, "Software quality metrics for object oriented systems", Technical University of Denmark.

Erni Karin, Claus Lewerentz, 2014, "Applying design-metrics to object-oriented frameworks" , April 1996. May 31,

Frunzio Luigi, Lin Bin, Lanza Michele, Bavota Gabriele, 2018, "RETICUL `A Real-Time Code Quality Assessment" , IEEE.

Jakhar Amit Kumar, Rajnish Kumar, 2014, "Measuring Complexity, Development Time and Understandability of a Program A Cognitive Approach", I.J. Information Technology and Computer Science.

Kaushik Dr. Manju, Mathur Mrs. Bhawana, 2015, "The Role and Issue of Clustering Techniques in Designing Maintainable Object Oriented System" International Journal of Computer Science and Software Engineering,No.1, Vol.1.

Klasky Hilda B, 2003, "A Study Of Software Metrics", Master Thesis, The State University of New Jersey.

Krishnapriya V., Ramar Dr. K, December 2010, "Comparison of Class Inheritance and Interface Usage in Object Oriented Programming through Complexity Measures", International Journal of Computer Science & Information Technology (IJCSIT), Vol. 2, No. 6.

Krusko Armin, 2004, "Complexity Analysis of Real Time Software– Using Software Complexity Metrics to Improve the Quality of Real Time Software".

Liu Xiaowei, 1999, "Object-Oriented Software Metrics", Master Thesis, Department of Computer Science, University of Manitoba ,Winnipeg, Manitoba, Canada.

Marwa Najm Abd Jader, Dr. Riyadh Zaghlool Mahmood ,2018, "An Improved Complexity Metric for Java OOP Software with a Corresponding Tool Implementation" , Master Thesis, University of Mosul , Mosul-Iraq.

McCabe Thomas J., Butler Charles W, December -12-1989, "Design Complexity Measurement and Testing", Artificial Intelligence and Language Processing Vol.32.

Timóteo Aline Lopes, Álvaro Alexandre, Almeida Eduardo Santana de, Meira Silvio Romero de Lemos, May-21-2014, "Software Metrics A_Survey", Researchgate , https://www.researchgate.net/publication/228698773_ .

Wu Di, Chen Lin, Zhou Yuming, Xu Baowen, 2015, "A metrics-based comparative study on object-oriented programming languages".