UNIVERSITY OF MOSUL
COLLEGE OF COMPUTER SCIENCES
AND MATHEMATICS

An Improved Complexity Metric for Java OOP Software with a Corresponding Tool Implementation

Marwa Najm Abd Jader

M.Sc./Thesis

Software Engineering

Supervised By

Dr. Riyadh Zaghlool Mahmood
Lecturer

2017 A.D. **1439 A.H.**

ABSTRACT

Most software developers aims to having a high-quality software that is easily maintainable, easily understandable, well structured, reliable, etc.. Measuring software complexity is quite important as high complexity has been identified to be the reason behind difficult understanding and reading and problematic changes in the future. Added to that, it has been the source of defects and poor software quality in the present time. Accordingly, high complexity entails the presence of serious faults in the software and duly higher costs to maintain and fix.

Software complexity metrics determine the improvement of software quality and project controllability to a large extent. Hence, many different software metrics, namely Lines of Code (LOC), Halstead complexity, Cyclomatic Complexity are used to measure software complexity. However, both LOC and Halstead's measure complexity metrics suffer from certain drawbacks. Hence, McCabe's Cyclomatic metric, which is viewed as the strongest, has been introduced to overcome these problems. Unfortunately, there had been no concept of object oriented when Cyclomatic complexity was introduced. Because of the disregard of this concept in cyclomatic complexity ,it was considered as a weak metric to measure the complexity of programming language like Object Oriented Language.

McCabe's Cyclomatic Complexity (CC) metric does not consider or measure the exact software complexity. It does not measures the complexity of interaction, if any, between two or three object classes in software. This is why there is a need to consider the concept of coupling since coupling complexity is considered as a change in one affect others. A great benefit is entailed from the addition of the concept of coupling between objects (CBO), in the calculation of McCabe's Cyclomatic Complexity (CC) metric, when the exact value of the complexity of the program (software) is calculated, we should realize that this

complexity is closely related to a number of quality factors, namely maintenance cost and effort, understanability, reusability, testability, reliability, testing effort, time prediction and many others as well.

In the proposed system, the initial metric (McCabe's Cyclomatic Complexity (CC), i.e. without coupling, using one method of calculation of the CC has been calculated. Then the (Coupling Between Objects “CBO”) metric that occurs through a number of cases has been calculated. These cases are:

(Inheritance, Arguments, Return types, Exception, Instruction type, methods calls (i.e. Constructor call)).

Of the 6 algorithms we have made, each algorithm was used to calculate a specific case of the CBO already mentioned. Then the cases were collected to get the value of CBO, called "TCBO", i.e ."Total Coupling Between Objects" in the proposed system. Finally, the proposed final metric "TCC", i.e. "Total Cyclomatic Complexity", (the new value of CC after adding the concept of CBO to it) has been calculated to obtain the value of this metric "TCC" by collecting the two previous metrics (CC, TCBO).

These results have been explaind by specific tables and diagrams to illustrate the values and information of the metrics in detail. The main objectives of the proposed system are: Implementation of (Cyclomatic Complexity CC) metric for java programs, implementation of(Coupling Between Objects CBO) metric for java programs and Also analysis and discussion of the results of combining (CC ,CBO) metrics and study their impact on the quality concepts such as reliability risk level, testability level, cost and effort of maintenance, and bad fix probability.This contribute to the improvement of the concepts for the quality of the software to a large extent and also help in preparing strategies or certain technologies to control the complexity of the software by depending on these concepts which depend on the exact value of the complexity of the software by combining these two metrics (CC, CBO).

جامعة الموصل

كلية علوم الحاسوب والرياضيات

مقياس التعقيد المحسّن لبرامج البرمجة الشيئية جافا مع تنفيذ أداة المقياس

مروة نجم عبد جادر

رسالة ماجستير

هندسة البرمجيات

بإشراف

د. رياض زغلول محمود

مدرّس

١٤٣٩ هـ

٢٠١٧ م

الملخص

يهدف معظم مطوري البرمجيات الى ايجاد برامج عالية الجودة والتي تكون سهلة الصيانة، سهلة الفهم، منظمة جيدا، موثوقة ,وغيرها من الصفات.

ان قياس تعقيد البرامج مهم جداً، حيث تم تحديد تعقيد البرامج العالي ليكون السبب وراء صعوبة الفهم ، القراءة، ومشاكل التغيير في المستقبل. بالإضافة اللى ذلك ,فقد كان مصدر العيوب وجودة البرمجيات الرديئة في الوقت الحالي. وبناءاً على ذلك ,ينطوي التعقيد العالي على ظهور أخطاء فادحة في البرنامج وهذا يؤدي إلى ارتفاع في تكاليف صيانة وإصلاح هذه الأخطاء.

تحدد مقاييس تعقيد البرمجيات تحسين جودة البرامج والتحكم اوالسيطرة على المشروع الى حد كبير. وبذلك ,فأن العديد من مقاييس البرمجيات المختلفة وهي :

"Lines of Code (LOC), Halstead complexity, Cyclomatic Complexity"

تستخدم في قياس تعقيد البرمجيات.ومع ذلك ,يعاني كل من مقاييس التعقيد(LOC and Halstead) من بعض المشاكل او السلبيات ولذلك السبب, فقد تم ادخال او تقديم مقياس التعقيد مكابي McCabe's) Cyclomatic Complexity(CC) والذي يُنظر اليه بإعتباره المقياس الأقوى للتغلب على هذه المشاكل.

لسوء الحظ,لم يكن هناك مفهوم الشيئية ,عندما تم ادخال او تقديم مقياس التعقيد مكابي CC .وبسبب تجاهل هذا المفهوم في حساب مقياس التعقيد مكابي , فقد اعتُبر مقياساً ضعيفاً لقياس تعقيد اللغات البرمجية مثل :OOPs اي (Object Oriented Programming).

لا يدرس أو يقيس مقياس التعقيد مكابي, تعقيد البرمجيات بشكل أساسي, وهو لا يقيس تعقيد التفاعل_ ان وجد_ بين اثنين أو ثلاثة فئات كائن(object classes) في البرنامج,وهذا هو السبب في الحاجة للنظر في مفهوم الاقتران او الارتباط (coupling). حيث أن تعقيد الاقتران يحتاج أن يؤخذ في نظر الاعتبار لأن التغيير في أحدهما(object classes) يؤثر على الآخرين.

هناك فائدة كبيرة من اضافة مفهوم الاقتران بين الكائنات في حساب مقياس التعقيد مكابي,عندما قمنا بحساب القيمة الدقيقة لتعقيد البرنامج, ينبغي علينا ان ندرك بأن هذا التعقيد يرتبط ارتباطا وثيقا ومباشرا بعدد من عوامل الجودة(اي بمعنى انه يؤثر على العوامل ذاتها) وهي :

جهد وكلفة الصيانة ، قابلية الفهم ، إعادة الاستخدام ، قابلية الاختبار، الموثوقية ,جهد الاختبار , تخمين الوقت وغيرها الكثير أيضا.

في النظام المقترح ، تم حساب المقياس الابتدائي اي بدون أقتران,وذلك بالاعتماد على احدى طرق حساب ال CC وهي بالاعتماد على ال (decisional statements). وبعدها تم حساب المقياس (Coupling Between Objects "CBO") والذي يحدث من خلال عدد من الحالات,وهذه الحالات هي :

(Inheritance, Arguments, Return types, Exception, Instruction type, methods calls (i.e. Constructor call)).

تم عمل 6 خوارزميات ,كل خوارزمية تختص بحساب حالة معينة من حالات ال CBO والتي ذكرت سابقا.وتم جمع هذه الحالات للحصول على قيمة هذا المقياس CBO المسمى في نظام العمل المقترح "TCBO" اي "Total Coupling Between Objects". واخيرا تم حساب المقياس النهائي المقترح , TCC اي "Total Cyclomatic Complexity"وهو القيمة الجديدة لل CC بعد اضافة مفهوم ال CBO اليه ,والحصول على قيمة هذا المقياس TCC بجمع المقياسين السابقين (CC,TCBO).

كما تم توضيح هذه النتائج بجداول معينة ومخططات لبيان قيم المقاييس ومعلوماتها بشكل تفصيلي .

تتمثل الأهداف الرئيسية للنظام المقترح في: تنفيذ مقياس التعقيد مكابي CC لبرامج جافا ، تنفيذ مقياس (الاقتران بين الكائنات CBO) لبرامج جافا وأيضا تحليل ومناقشة نتائج دمج هذين المقياسين (CC، CBO) و دراسة تأثيره على مفاهيم الجودة مثل:

مستوى مخاطر الوثوقية ,مستوى قابلية الاختبار, جهد وتكلفة الصيانة واحتمالية الإصلاح السيئة للاخطاء. ويساهم هذا في تحسين مفاهيم جودة البرامج إلى حد كبير ويساعد أيضًا في إعداد استراتيجيات أو تقنيات معينة للسيطرة على تعقيد البرامج من خلال الاعتماد على هذه المفاهيم التي تعتمد على القيمة الدقيقة لتعقيد البرنامج بواسطة دمج هذين المقياسين (CC، CBO).