المجلة العراقية للعلوم الإحصائية (29) 2019
عدد خاص بالمؤتمر الطلابي الاول
ص ص [61-72]

توظيف النموذج الحلزوني في تحليل وتصميم أداة لتحقيق أمثلية البرامج المكتوبة بلغة جافا

رحمة سليم داؤود الصواف* د.أسماء ياسين حمو**
rahmasaleemsawaf@yahoo.com asmahamo@yahoo.com

المستخلص

النموذج الحلزوني هو احد النماذج المستخدمة في تطوير البرمجيات . تم استخدام هذا النموذج في تحليل وتصميم وبناء اداة (JCOT) حيث تقوم الاداة باستبدال بعض الجمل البرمجية المكتوبة بلغة JAVA بجمل اخرى تطابقها في العمل لكن بوقت تنفيذ اقل. كما تعرض الاداة مدى الاختصار في الوقت عبر شريط طولي ولوني لسهولة تميز ذلك من قبل المستخدم. اثبت النموذج الحلزوني فاعليته حيث تم التعامل مع كل حالة من حالات الأمثلة كحلقة ثم الانتقال الى الحالة الاخرى الى نهاية تكوين الاداة الامر الذي مكن الباحث من سهولة اضافة حالات جديدة للاداة (JCOT).

Employ spiral model analysis and design tool to optimize software written in Java

ABSTRUCT

The spiral model is one of the models used in software development. This model was used in the analysis, design and construction of the tool (JCOT), where the tool replaces some of the sentences written in JAVA in other sentences that match the work but at the time of implementation less. The tool also displays the shortness of the time in a longitudinal and chromatic strip for ease of user recognition. The spiral model proved its effectiveness. Each case of optimization was treated as a link and then the other state was transferred to the end of the tool. This enabled the researcher to easily add new cases of the tool (JCOT).

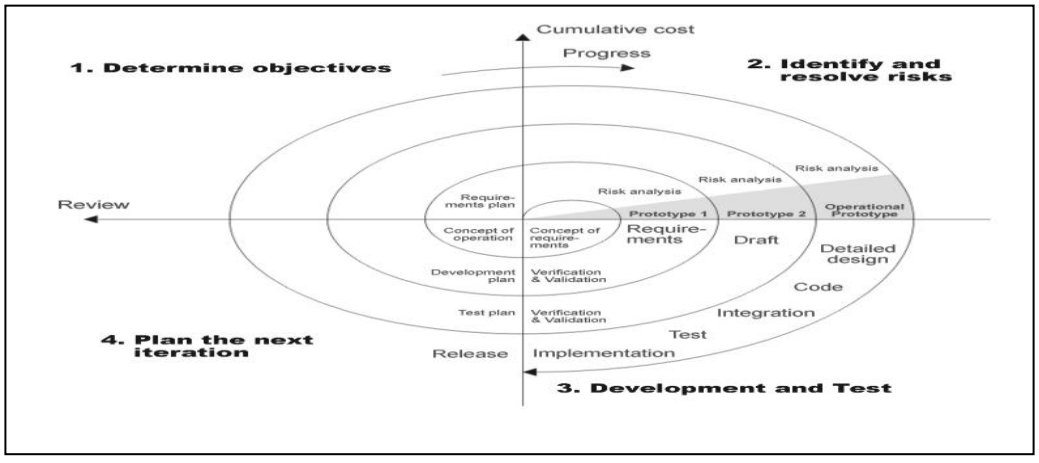
* باحثة / قسم البرمجيات / كلية علوم الحاسوب والرياضيات / جامعة الموصل
** استاذ مساعد / قسم البرمجيات / كلية علوم الحاسوب والرياضيات / جامعة الموصل

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

1- المقدمة:

لسهولة بناء اي منتج تم اقتراح نماذج المعالجة الإلزامية للتخلص من الفوضى في عمليات تطوير البرمجيات ، تلك النماذج تشترك في الإطار العام نفسه ولكنها تختلف من ناحية التركيز على مسائل معينة دون اخر من هذه النماذج هو النموذج الحلزوني ، ففي عام 1986 تم تطوير نموذج شلال الماء Waterfall Model بتقديم عدة دورات التي تبدأ بكلفة صغيرة ثم تنتهي بدوره كبيرة ذلك هو النموذج الحلزوني(Ruparelia,2010). العمليات خلال النموذج الحلزوني تمثل بشكل دوامة أو حلقة بدلا من سلسلة من الأنشطة مع إمكانية التراجع كما في شلال الماء Waterfall Model (Brink,2008). كل دورة خلال النموذج مقسمة إلى أربع أقسام وهي:

تحديد الأهداف.Determine objectives و تقيم البدائل وتعريف وحل المخاطر.Evaluate alternatives, and identify and resolve risks و التطوير والاختبار. Develop and test ثم التخطيط للخطوة التالية.Plan the next iteration . خلال كل دورة داخل النموذج الحلزوني يتم بناء النموذج الأولي Prototype Model: من مرحلة جمع المتطلبات إلى طور الاختبار، المخاطر يتم تعريفها وتحليلها لكي يتم إدارتها. المخاطر تصنف إلى مخاطر الأداء ذات العلاقة أو مخاطر التطوير . إذا كانت المخاطر التي حصلت هي مخاطر التطوير فيتم تطبيق النموذج التزايدي incremental model في الخطوة التالية. من ناحية أخرى إذا كانت مخاطر الأداء قد حصلت يتم إتباع النموذج الحلزوني الخطوة التالية من خلال النموذج التطوري Evolutionary Model . هذا يؤكد إن النموذج الأولي Prototype Model المنتج في ربع الدائرة التالية يحوي الحد الأدنى من المخاطر(Ruparelia,2010).



الشكل (1) يوضح النموذج الحلزوني

ايجابيات النموذج الحلزوني (Sasankar &Dr.ChavanVinay, 2011):

النموذج الحلزوني يحاول حل جميع المخاطر الممكنة المتعلقة بالمشروع ابتدءاً من أعلى مخاطر.

1. المستخدمون النهائيون لهم الفرصة في رؤية المشروع في دورة حياة مبكرة.
2. يعدل المشروع مع كل طور بعد عرضه على الزبون وبهذا النموذج يضمن أفضل جودة للمشروع.
3. النموذج يمكن من استخدام التقنيات مثل إعادة الاستخدام Reuse، النموذج الأولي Prototype وتصميم العنصر الأساس component based design.

ومن مساوئ النموذج الحلزوني(Munassar and A. Govardhan, 2010):

انه يمكن أن يكون نموذجا مكلفا للاستخدام كما ان تحليل المخاطر يتطلب خبرة عالية و نجاح المشروع يعتمد بشكل كبير على مخاطر مرحلة التحليل لا يعمل بشكل جيد للمشاريع الصغيرة.

2- مفهوم تحقيق الامثلية:

تحقيق الامثلية هي عملية تحويل جزء من شفرة البرنامج لجعلها أكثر كفاءة (إما من ناحية الوقت أو المساحة) من دون تغير الإخراج أو حصول تأثيرات جانبية . الاختلافات التي تظهر لشفرة برنامج المستخدم هو سرعة التنفيذ او استخدام مساحة خزنية اقل (Johnson,2008).

يتضمّن تحقيق الامثلية للشفرة تطبيق القواعد والخوارزميات للبرمجة بهدف جعل الشفرة أكثر كفاءة وأصغر وأسرع في التنفيذ. يهدّف تحقيق الامثلية الى تحقيق العديد من الأهداف المرغوبة في هندسة البرامج،غير انه يؤثر بشكل سلبي في أغلب الأحيان على الأهداف المهمة الأخرى مثل الاستقرار ، وقابلية الصيانة والنقل (N Manikandan et,2013.تم في هذا البحث توظيف النموذج الحلزوني في تحليل وتصميم اداة لتحقيق الامثلية لشفرة لغة جافا

3- متطلبات الأداة JCOT من وجهة نظر مهندس البرمجيات : JCOT Requirement From Software Engineer Viewpoint

عندما يقوم المبرمج بكتابة الشفرة قلما يهتم بأفضل صيغة برمجية للحصول على أفضل تنفيذ، ذلك إن غايته هو الحصول على الإخراج المعني من غير الانتباه إلى سرعة التنفيذ للشفرة أو المساحة الخزنية التي تتطلبها تلك الشفرة للتنفيذ، لذلك لابد للمبرمجين ولمهندس البرمجيات من

استخدام أداة لتحسين برامجهم من حيث سرعة التنفيذ وتقليل المساحة الخزنية كونهما دليل جودة البرمجيات. تركز الأداة JCOT على مسالة سرعة التنفيذ حيث تزود المستخدم ببدائل لجمله البرمجية تؤدي نفس الهدف لكن بشكل أسرع. تحسين الجمل البرمجية يتم بتقنيات تسريع التنفيذ لكن هذه التقنيات ليست بنفس المستوى ، حيث أن نسبة زيادة السرعة تختلف من تقنية إلى أخرى، لذلك فان الأداة توفر للمستخدم مؤشر لوني وطولي يدل على مستوى التحسين الذي طرأ على سلسلة الجمل الحالية. حيث يمثل اللون الأخضر أعلى مستوى ومتدرجا إلى السمائي ثم البرتقالي ثم الأصفر الذي يمثل اقل مستوى.

تحتاج الأداة إلى قراءة ملفات مكتوبة بلغة Java (إذ إن الأداة مختصة للبرمجيات المكتوبة بلغة Java كما اشرنا سابقا"). وتقوم الأداة JCOT بإرجاع ملفات مكتوبة أيضا بلغة Java ولكنها محسنة من حيث سرعة تنفيذ الشفرة . وفيما يأتي تلخيص المدخلات والعمليات والمخرجات للأداة JCOT :

أ- المدخلات : برمجيات مكتوبة بلغة Java (على شكل ملفات لها امتداد نصي (.txt)).

ب- العمليات : يتم إتباع ألخوارزمية المذكورة في الفقرة 6-3.

ت- المخرجات : وتشمل ما يأتي:

1. ملفات مكتوبة بلغة Java تعمل نفس عمل برامج الإدخال لكن أسرع منها بالتنفيذ.
2. مؤشر لوني وطولي لشدة كل تقنية من تقنيات تحقيق الامثلية يشير إلى درجة التحسين في وقت التنفيذ.

4- متطلبات بناء الأداة المقترحة: Requirements Of The Proposed Tool

إن عملية جمع المتطلبات تلعب دورا"أساسيا" خلال إنشاء أي منتج برمجي. ففي هذا البحث يتطلب إيجاد المتطلبات الخاصة بعمل الامثلية وفهمها وتحديد المتطلبات الوظيفية وغير الوظيفية.

4-1 المتطلبات الوظيفية: Functional Requirement

1. يتطلب وجود حالة او اكثر من حالات الامثلية لكي تتمكن الأداة من العمل بصورة صحيحة.
2. يتطلب وجود الملف الذي يتم القراءة منه وتحديد مكان خزنه.
3. يجب إن تكون الأداة قادرة على تمييز الحالات المذكورة في الملف لتحديد الجمل التي ممكن من خلالها عمل الامثلية.

4. يجب إن تعمل الأداة بصورة مستمرة وبدون أخطاء.
5. يجب إن تظهر الأداة الشدة لكل حالة من حالات الامثلية ليتسنى للمستخدم التعرف على مستوى الزيادة في السرعة في كل حالة.
6. يجب إن تقوم الأداة بحفظ الملف الناتج بعد تنفيذ الامثلية ليتسنى للمبرمج أو مهندس البرمجيات استخدامه فيما بعد وان يعمل البرنامج في الملف الجديد بدون اخطاء ونفس عمل البرنامج في الملف المستخدم.
7. يجب إن يكون للأداة القدرة على الاحتفاظ بالملف السابق في حالة عدم رغبة المستخدم بتطبيق الامثلية.

4-2 المتطلبات غير الوظيفية: Non-Functional Requirement

1. عند تصميم واجهة الأداة فانه يتوجب إتباع اتفاقية التصميم التي تنص على أن تكون الواجهة سهلة للعمل عليها من قبل المستخدم.
2. أن تكون الأداة قادرة على العمل بدون تنصيب لغة جافا ذلك للتسهيل على المستخدم.
3. إن يكون للأداة واجهة رسومية تمكن المستخدم من التعامل معها بسهولة.
4. ان يتبع أسلوب Windows نفسه في استعرض الملفات واختيار الملف المرغوب.
5. ان تظهر للمستخدم مكان المقطع البرمجي الذي يحوي الحالة التي تنطبق عليها إحدى التقنيات.
6. ان تظهر الأداة المقطع البرمجي بعد عمل الأداة لضرورة ذلك للمستخدم.
7. ان تظهر الأداة بشكل أو بآخر مؤشراً على مدى التحسين الذي سيطرأ على السرعة.

5- توظيف النموذج الحلزوني في بناء الأداة المقترحة:

لبناء الأداة المقترحة يتم إتباع خطوات تكوين كل دائرة في النموذج الحلزوني وتحديد الناتج منها وتخمين الدورة التالية والناتج منها وبشكل مفصل.

5-1 تحديد الأهداف: Determine Objective

الهدف الأساس للأداة المقترحة هو زيادة سرعة تنفيذ لغة Java وذلك عن طريق دراسة كل المقاطع البرمجية (المقطع قد يكون جملة او اكثر) وتحديد وقت التنفيذ لكل مقطع وكذلك عمل مقارنة مع وقت تنفيذ البرنامج قبل تطبيق حالات الامثلية وبعدها وملاحظة التغير الناتج في سرعة التنفيذ للبرنامج.

5-2 تحديد وحل المخاطر: Identify & Resolve Risk

1. ان البحوث التي اعتمدت في جميع التقنيات ليست كلها تتناول لغة Java وبما ان الأداة المقترحة خاصة بالبرامج المكتوبة بلغة Java لذلك ينبغي اختبار كل تقنية على حدى بتنفيذها وقياس الوقت ومن ثم تبنيها في الأداة أو رفضها.

2. يُعد حساب وقت التنفيذ الفعلي لبرنامج الاختبار من التحديات نظرا لوجود برامج مضادات الفايروسات Anti-Various وغيرها التي تعمل في الخلفية Background وقد تؤثر على دقة حساب الوقت. لذلك ينبغي قياس الوقت في حاسبة مثالية لا تحوي تلك البرامج.

3.ان تنفيذ الجمل في الحاسبات المتطورة يستغرق وقتاً قصيراً جدا لذلك تم اللجوء الى تكرار الجمل تحت الاختبار مائة ألف مرة للتمكن من حساب الوقت المستغرق لتنفيذ تلك الجمل.

5-3 التطوير والاختبار: Development & Test

لتصميم الأداة المقترحة تم جمع عدة مصادر حول الامثلية ودراسة الحالات المقترحة فيها وتطبيقها للتأكد من فعالية تلك الحالات في زيادة سرعة التنفيذ للمعالج ودراسة شدة الحالة المقترحة ليتسنى للأداة تحديد ذلك من خلال إضافة مؤشر لوني وطولي للحالة عند التنفيذ.

5-4 التخطيط للدورة التالية: Plan The Next Iteration

بعد تنفيذ كل حالة وفحصها وتحديد فعاليتها يتم الانتقال إلى حالة الامثلية التالية وأيضا يتم اختبارها وفحصها وتحديد فعاليتها في زيادة السرعة فإذا كانت الحالة تزيد من سرعة تنفيذ البرنامج يتم إدراجها ضمن حالات الامثلية الخاصة بالأداة.

6- بناء الأداة :

تم استخدام عدة ادوات حاسوبية مساعدة لهندسة البرمجيات (CASE TOOls) لغرض القيام بكل مرحلة من مراحل هندسة البرمجيات وكما هي موضحة في الجدول (1-1) .

6-1: مرحلة التحليل لعملية بناء الأداة:

بدايةً تم دراسة الموضوع بشكل كامل وجمع عدة مصادر حول ذلك ، وتناولت المصادر عمل الامثلية من حيث السرعة عند التنفيذ وكيفية تقليص ذلك كما هو موضح في مخطط التسلسل الزمني ، و يظهر الشكل (2) ان موضوع عمل الامثلية يركز على سرعة التنفيذ للمعالج

والمصادر المستخدمة تتناول الامثلية لبرامج لغتي C و Java . تم دراسة الحالات المذكورة في لغة C وتطبيقها بلغة Java . وبغية الوقوف على فاعلية التقنيات المستخدمة في البحوث للغة C تحت لغة Java تم حساب زمن التنفيذ لكل حالة من حالات تحقيق الامثلية عن طريق إجراء التجربة المذكورة في المصدر (Asmaa & Rahma,2014).

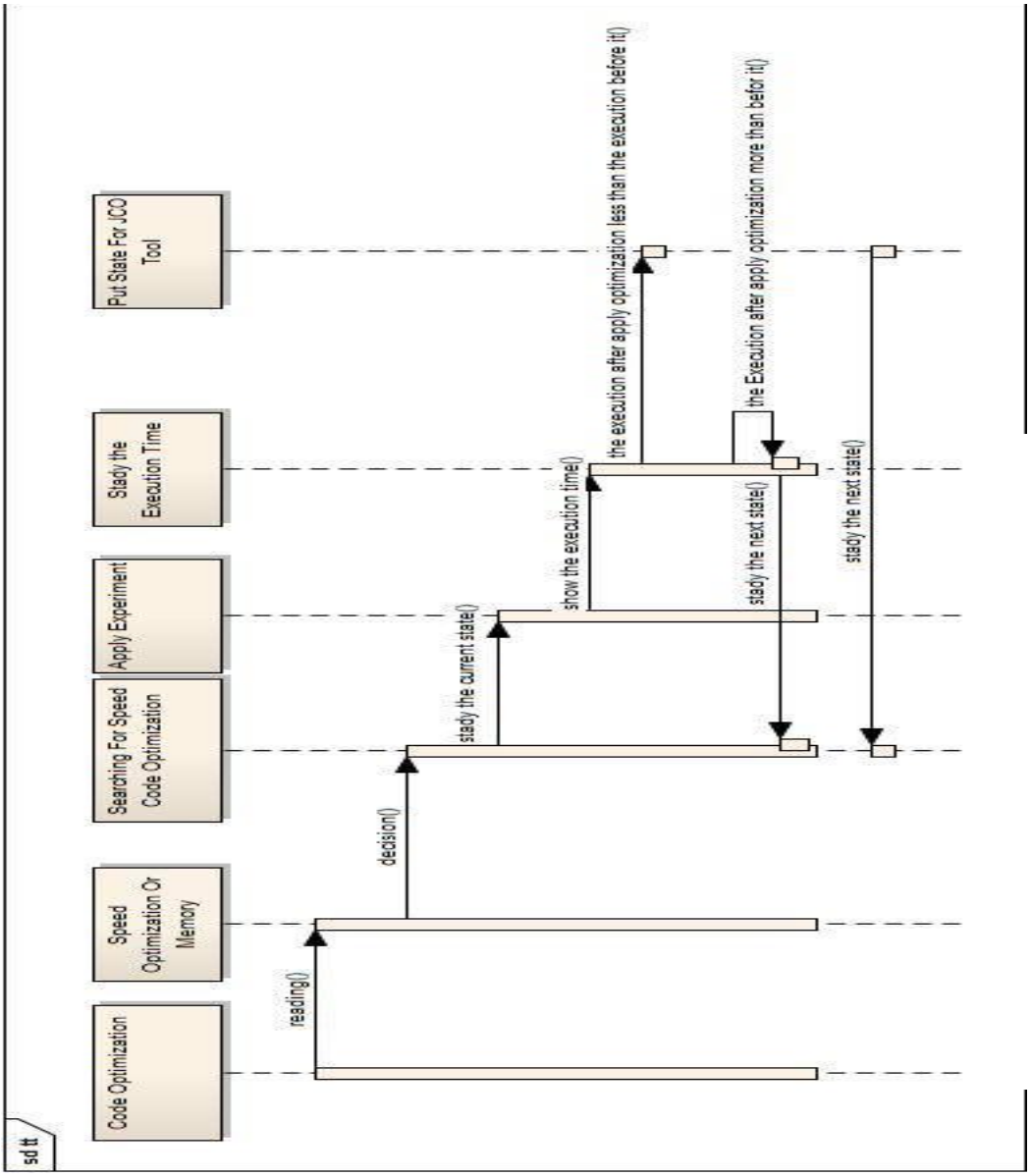
جدول (1) :أدوات هندسة البرمجيات المساعدة للحاسوب المستخدمة في بناء العمل

المرحلة	أدوات هندسة البرمجيات المساعدة للحاسوب	المخططات المستخدمة	الشركة المنتجة
التحليل	Enterprise Architect	مخطط التسلسل الزمني	Sparx System
التصميم	Enterprise Architect	مخطط التسلسل الزمني مخطط الحالة مخطط الفعالية	Sparx System
	Microsoft Vision Version 2010	المخططات الانسيابية	Microsoft
البرمجة	NET Beans IDE 7.4	————————	Sun Micro-Sisters
تنفيذ الواجهات	NET Beans IDE 7.4	————————	Sun Micro-Sisters
الاختبار	NET Beans IDE 7.4	————————	Sun Micro-Sisters

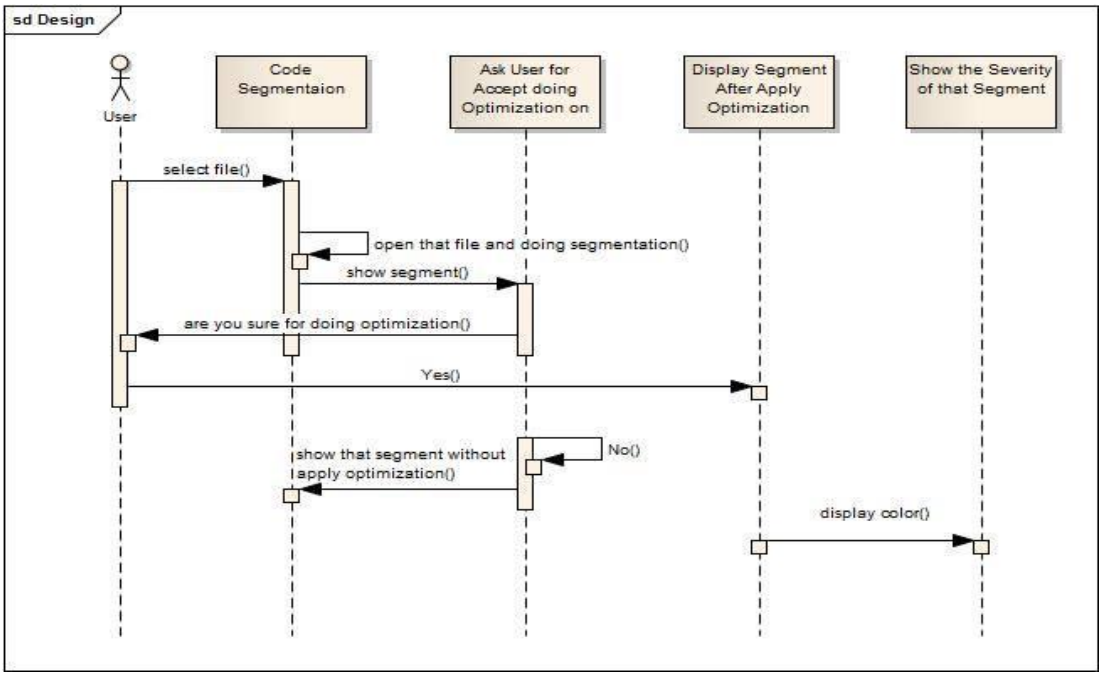
2-6 مرحلة التصميم للأداة المقترحة JCOT:

في هذه المرحلة تم تصميم الأداة وتم توضيحها من خلال استخدام مخطط التسلسل الزمني Sequence Diagram إذ يوضح الشكل (2) ان العمل يبدأ مع بداية اختبار الشفرة للمشروع المعني وتقطيعها حسب حالات الامثلية التي تم تضمينها لتصميم الأداة في مرحلة التحليل وعرض الشفرة المقطعة للمستخدم ثم تطبيق الامثلية على الشفرة المقطعة مع ظهور مؤشر لوني وطولي لمدى تحسين سرعة تنفيذ البرنامج للمقطع المحدد وسؤال المستخدم ما اذا كان يرغب بحفظ المقطع بعد تطبيق حالة الامثلية فإذا كان جواب المستخدم ب (نعم) يتم تطبيق الامثلية

على المقطع وعرضه بعد ذلك على الشاشة وكتابته داخل الملف ليتسنى للمستخدم استعماله فيما بعد أما إذا كان جواب المستخدم ب (لا) فسوف تحتفظ الأداة بالمقطع ذاته من دون إجراء التغير عليه.



الشكل (2) : مخطط التسلسل الزمني لعملية التحليل

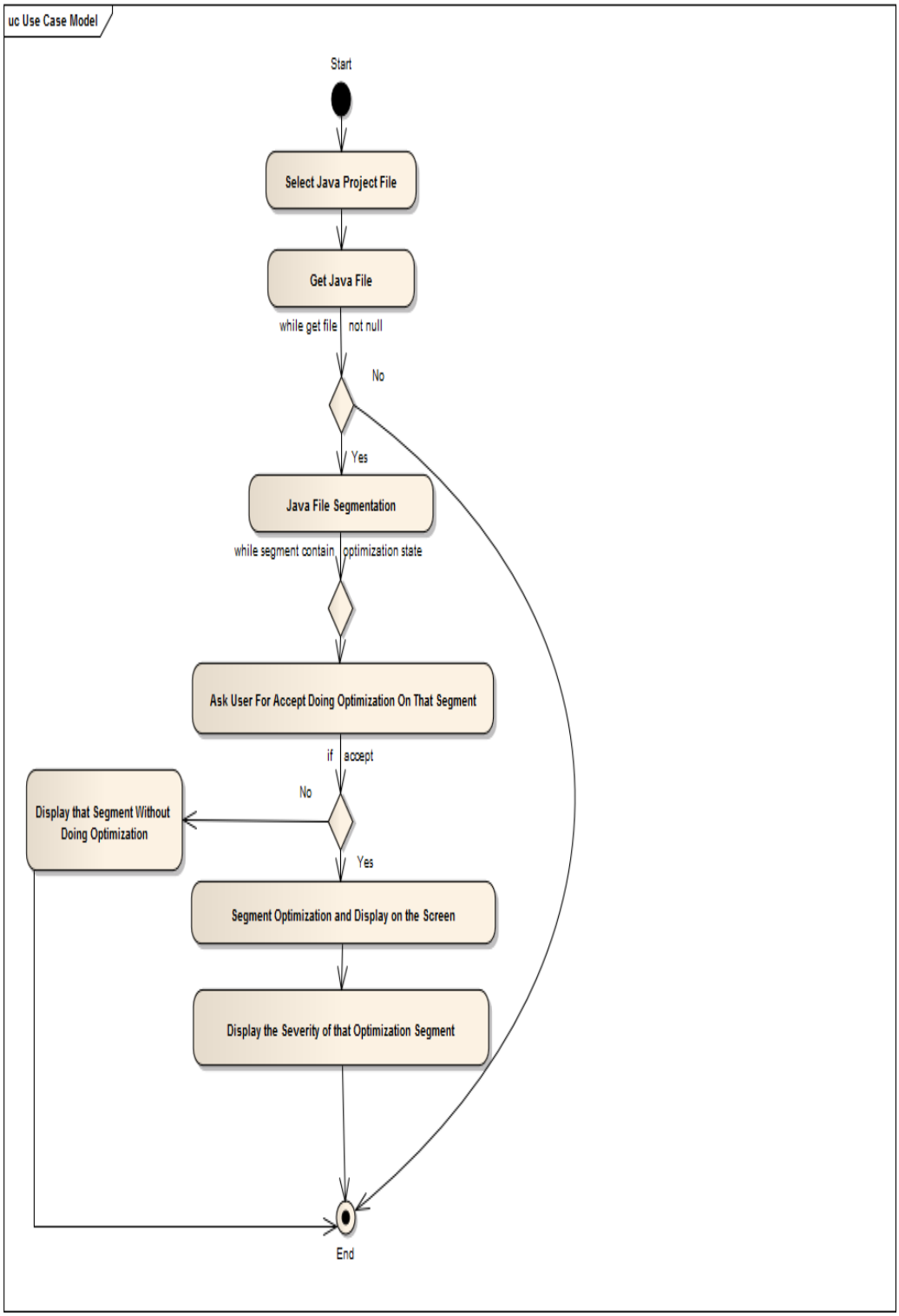


الشكل (3): مخطط التسلسل الزمني لعملية التصميم

ويوضح الشكل(3) ان مستخدم الأداة النهائي يستطيع ان يرى الأداة مكونة من أربعة أجزاء رئيسة وهي كالأتي:

1- تحليل الشفرة Code Analysis : تمكن هذه الخاصية من دراسة الشفرة بشكل كامل والتعرف على حالات الامثلية الممكن تطبيقها.
2- تقطيع الشفرة Code Segmentation : هذه الخاصية تمكن المستخدم من الاستفادة من عملية التحليل لتقطيع حالات الامثلية المكتشفة في تلك المرحلة.
3- تطبيق الامثلية على المقطع المحدد Extract Code Optimization: هذه الخاصية تقوم بتنفيذ الامثليه على الشفرة المقطعة لتحصيل الهدف الأساس من الأداة وهو زيادة سرعة تنفيذ البرنامج.
4- شدة الامثليةOptimization Severity : تمكن هذه الخاصية المستخدم من معرفة التغير الحاصل في

الوقت بعد تطبيق الامثلية من خلال عرض مؤشر لوني وطولي يشير إلى سرعة تنفيذ البرنامج.



الشكل (5): مخطط الفعالية للاداة المقترحة JCOT

6-3 تصميم خوارزمية الأداة المقترحة: Design The Proposed Tool Algorithm

الخطوة الأولى : اختيار احد المشاريع (الملفات) المكتوبة بلغة Java لقراءة الشفرة الخاصة بها وذلك لغرض تطبيق الامثلية عليها.

الخطوة الثانية : قراءة الشفرة مع التقطيع لتلك الشفرة حسب حالات توليد الامثلية المحددة.

الخطوة الثالثة : عرض مقطع الشفرة الذي يحوي إحدى تقنيات الامثلية.

الخطوة الرابعة : تطبيق الامثلية على الشفرة المقطعة الناتجة من الخطوة الثالثة مع اظهار مؤشر لوني وطولي على شدة الامثلية لذلك المقطع .

الخطوة الخامسة : اعلام المستخدم باختيار خزن المقطع الناتج بعد تحقيق الامثلية فاذا كان جواب المستخدم ب "نعم" يتم حفظ المقطع الناتج داخل ملف بنفس مسار خزن الملف الرئيسي ذلك ليتسنى للمستخدم استعماله فيما بعد اما اذا كان جواب المستخدم ب "لا" تقوم الاداة بحفظ المقطع ذاته من دون تطبيق حالة الامثلية عليه ثم الذهاب الى الخطوة الثالثة.

الخطوة السادسة : هل انتهت الشفرة اذا كان جواب المستخدم ب "لا" اذهب الى الخطوة الثالثة واذا كان جواب المستخدم ب "نعم" يتم انهاء العمل للاداة .

ويمكن الاطلاع على الاداة كاملة في المصدر (Asmaa & Rahma,2015)

7- الاستنتاجات:

من خلال تطبيق الأداةJCOT ضمن منصة التشغيل (Windows) او من خلال بيئة التشغيل (Net Beans) على مجموعة من البرامج المكتوبة بلغة Javaوملاحظة التغيرات التي طرأت على زمن التنفيذ تم التوصل الى الاستنتاجات الآتية :

1. إمكانية الأداة JCOT على توليد برامج تتميز بسرعة تنفيذ عالية نظرا للتغيرات التي تقوم بها الأداة على البرامج لجعله يعمل بشكل أسرع.
2. توضح أهمية دور تقنيات تحقيق الامثلية عن طريق عرض الأداة لمؤشر لوني وطولي يشير الى سرعة تنفيذ الأداة.
3. تعطي انتباه للمبرمج الى ضرورة التأكد من أسلوب صياغة الشفرة للمشروع المطلوب بحيث يمكن أن تحتوي على تقنيات تحقيق الامثلية ولا تستغرق وقت كبير في التنفيذ.

4. أن استخدام الأداة المقترحة يقلل من الوقت وهذا العامل مهم في مشاريع هندسة البرمجيات والمشاريع خاصة ، وبالمجالات الأخرى عموما.
5. من الممكن استخدام الأداة لإغراض التعليم في الفصول الدراسية الأولى للبرمجة بلغة Java.
6. اثبت النموذج الحلزوني فاعلية في هذا البحث من خلال امكانية اضافة حالة جديدة او حلقة جديدة على الاداة بعد مرحلة الاختبار واكتشاف حالات اخرى لم تذكر في الاختبار الاول.

المصادر:

1. Asmaa Hamo& Rahma Alsawaf,2014," Estimation Benefit of Java Optimization Techniques", International Journal of Enhanced Research in Science Technology & Engineering, Vol. 3 Issue 5, www.erpublications.com.
2. Asmaa Hamo& Rahma Alsawaf,2015," Designing And Implementation Of A Tool For Java Code Optimization", International Journal of Innovative Research and Creative Technology (IJIRCT), Volume 1| Issue 4, www.ijirct.org .
3. Brink huib,2008,"Optimization techniques used in java hotspot complier", Institute of Information and Computing Sciences, Utrecht University, The Netherlands.
4. IBM Corp.,2012,"Optimization and Programming Guide", Copyright IBM Corporation,Version 12.1,pp1-117.
5. Johnson Maggie,2008,"Code Optimization" ,Handout 20, CS143
6. Munassar Nabil and A. Govardhan, 2010," A Comparison Between Five Models Of Software Engineering", IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 5,pp 94-101.
7. N Manikandan et,2013,"Modularization Based Code Optimization Technique for Ensuring Software Product Quality",International Journal of Engineering and Technology (IJET),Vol 5 No 2,pp1252-1259.
8. Ruparelia Nayan B. ، 2010،"Software Development Lifecycle Models", ACM SIGSOFT Software Engineering Notes , Vol 35, No. 3,pp.8-13.
9. Sasankar Ashish B. &Dr.ChavanVinay, 2011,"Survey of Software Life Cycle Models by VariousDocumented Standards",Dept. of Computer Science India, IJCST Vol. 2, Issue 4,pp.137-144.