العنوان:	تأثير البرمجة ذات التوجه المفاهيمى على جودة البرمجيات
المؤلف الرئيسي:	عثمان، أحمد سيد أحمد علي
مؤلفين آخرين:	حاج على، عوض(مشرف)
التاريخ الميلادي:	2015
موقع:	الخرطوم
الصفحات:	1 - 240
رقم MD:	830496
نوع المحتوى:	رسائل جامعية
اللغة:	Arabic
الدرجة العلمية:	رسالة دكتوراه
الجامعة:	جامعة النيلين
الكلية:	كلية الدراسات العليا
الدولة:	السودان
قواعد المعلومات:	Dissertations
مواضيع:	هندسة البرمجيات، برمجة المفاهيمية، جودة البرمجيات، التحليل البرمجى
رابط:	https://search.mandumah.com/Record/830496

الفصل الأول (الأطار العام)

1.1 مقدمة

البرمجة Programming هى عملية كتابة مجموعة من الشفرات [الأكواد – Codes] بأستخدام احدى لغات البرمجة Programing Languages ومن ثم أختبار الشفرة البرمجية وتصحيح الاخطاء ان وجدت ، وكل شفرة او برنامج أو نظام حاسوبى يتم كتابته للقيام بأداء مهمة محددة.

تطورت عملية البرمجة وكتابة الشفرات البرمجية عبر الزمن تطورا كبيرا ففى الجيل الأول من لغات البرمجة كانت تتم عملية البرمجة بأستخدام الرمزين 0 و 1، وكانت تسمى لغة الالة Machine Language وهى اللغة الوحيدة التي يفهمها كمبيوتر ويقوم بتحويلها الى أشارات كهربية . أذ أن الـ1 يعنى وجود التيار الكهربائى والـ0 يعنى عدم وجوده، وبالتالى نجد أن عملية البرمجة كانت تتطلب معرفة دقيقة وتفصيلية بمعمارية الجهاز الذى سيقوم بتنفيذ البرنامج.

وجاء بعد ذلك الجيل الثاني من اللغات والذى عرف بأسم لغة التجميع Assembly Language ، لغة التجميع تحول تسلسل الـ 0 والـ1 الى كلمات يفهمها الانسان مثل Add وعند كتابة برنامج عن طريق هذه اللغة فأن المجمع Assembler يقوم بتحويل هذه العبارات البسيطة الى ما يقابلها من سلاسل الـ 0 والـ1 وبالتالى الى حد ما أصبحت عملية البرمجة أسهل وأقرب الى الانسان من الالة مقارنة بلغات الجيل الاول، ولكن ما زالت هنالك حوجة الى معرفة البنية الهيكلية للجهاز الذى سيقوم بتنفيذ البرنامج.

بعد ذلك جاء الجيل الثالث وهو ما عرف بلغات المستوى العالى High Level HLL،Languages والتي أصبحت تحتوى على جمل وتعابير أقرب الى الأنسان. وحتى يستطيع الكمبيوتر فهم وتنفيذ هذه البرامج فأن هذه اللغات احتوت على مترجم Compiler يقوم بتحويل هذه الجمل والتعابير الى لغة الالة او الى لغة التجميع. وكما نعلم فأن جميع البرامج يجب فى النهاية أن تترجم الى لغة الالة حتى تستطيع البنية الالكترونية لجهاز الكمبيوتر تنفيذها. هنالك العديد من لغات برمجة المستوى العالى بدءا من لغات البرمجة الهيكلية Structural Language واللغات الوظيفية Functional Language الى اللغات الكائنية Object Oriented Language . فى هذا الجيل أصبحت عملية البرمجة تركز أكثر على المشكلة موضوع الحل عوضا عن التركيز على تفاصيل الالة وأهم خطوة فى هذه الاتجاه كانت التركيز على الكائنات المكونة للنظام وتحديد سلوكها مما أتاح للمبرمجين ان ينتجو برامج اكثر تعقيدا واكبر حجما وبجودة أعلى. ولكن مع ظهور البرامج الكبيرة والمعقدة والتى يمكن أن تحتوى على ملايين الأسطر من الكود والتى تعمل على أنظمة حساسة بدأت تظهر بعض المشاكل فى البرمجة الكائنية وأهمها كان الكود الموزع والمتداخل والذى لا يمكننا الجزم بتبعيتة لكائن محدد من كائنات النظام. فعلى سبيل المثال عندما نتحدث عن الامن Security أو عن التتبع Logging فلا يمكننا ان نتحدث عنهما كوحدة مستقلة بل نجد أنهما يتوزعان داخل جميع الكائنات المكونة للنظام. وعليه فأن عملية برمجة الانظمة المعقدة والكبيرة أصبحت تواجه بعض المشاكل أذ تتطلب أن يكون المبرمج ملما بجميع التفاصيل المتعلقة بالوظائف الاساسية للنظام Functional Requirement زائدا الوظائف غير الوظيفية Non-Functional Requirement مثل الأمن والتتبع والتعامل مع

الاخطاء Error Handling هذا بالاضافة الى المشاكل التى أصبحت تواجهنا عند أجراء الصيانة والتعقيد الذى أصبح موجودا فى البرامج جراء صيانتها لعدة مرات.

ظهرت تقنية البرمجة المفاهيمية Aspect- Oriented Software Development – AOSD – لتعالج مشاكل الكود المتداخل والموزع المرتبطة بالبرمجة الكائنية المنحى، كما تساعد على فصل الاهتمامات المختلفة عن بعضها البعض، كما تقوم بفصل المتطلبات الوظيفية Functional Requirements عن المتطلبات غير الوظيفية Non-Functional Requirements بصورة يمكن بها تتبع كل وظيفة أبتداءا من مرحلة المتطلبات وحتى مرحلة التنفيذ.

1.2 مشكلة الدراسة

مشكلة الدراسة تكمن فى أيجاد الطريقة الأمثل لتطوير البرامج وكتابة الكود, أذ أنه ومنذ ظهور الحاسبات وبرامجها وحتى يومنا هذا ما زال السؤال الشاغل للعاملين فى المجال ماهى الطريقة الامثل لتصميم البرامج وتطويرها والتى يمكن أن تنتج برامج ذات جودة عالية.

1.3 أهداف الدراسة

تهدف الدراسة الى دراسة تاثير البرمجة المفاهيمية Aspect- Oriented Software Development على خصائص جودة البرمجيات (قابلية الصيانة ، الأداء، قابلية أعادة الأستخدام، وقابلية الفهم) ومقارنة هذا التاثير مع البرمجة الكائنية المنحى Object – Oriented Programming.

1.4 أهمية الدراسة

لا شك ان البرمجيات غزت جميع مناحى الحياة ، فاصبحت هى الاساس فى ادارة جميع الاعمال سواءا التجارية او الصناعية او التعليمية او اى مجال اخر. وانتاج برمجيات بجودة عالية هو التحدى الاساسى الذى يواجه هندسة البرمجيات، بالاضافة الى أهمية أنتاج برمجيات تتوفر فيها خاصية الصيانة مستقبلا نسبة لاهمية أستمرارية البرمجيات بالعمل لفترات طويلة مع أمكانية أستيعاب المتغيرات التى يمكن أن تحدث فى أى بيئة عمل نتيجة للتطور المستمر فى الاعمال. تنبع أهمية الدراسة من مساعدة المبرمجين ومهندسي البرمجيات على أختيار المنهجية الافضل لتقسيم النظام والتى تمكنهم من أنتاج برمجيات بجودة عالية

1.5 منهجية الدراسة

لأختبار تأثير البرمجة المفاهيمية على خصائص جودة البرمجيات ستنتهج الدراسة المنهج التحليلى Imperical Study، وذلك من خلال تطبيق المنهجية الكائنية والمنهجية المفاهيمية على مجموعة من البرامج المختلفة ومن ثم مقارنة البرامج مع بعضها البعض بالأضافة الى تحليل نتائج أستبيان قام بملئه مطورو تلك البرامج.

1.6 هيكل البحث

تتكون الدراسة من سبعة فصول حيث يتكون الفصل الاول(هيكل البحث) من مقدمة البحث، ومشكلة الدراسة، واهمية الدراسة، وخطة الدراسة. اما الفصل الثاني (الدراسات السابقة) فيتحدث عن عدد من الدراسات السابقة التي تم اجرائها خلال السنوات الماضية حيث يعرض

كل دراسة النتائج التى توصلت اليها كل دراسة . اما الفصل الثالث (البرمجة المفاهيمية التوجه) فيتكون من مقدمة تاريخية عن البرمجة وتطورها حتى الان ، ومفهوم البرمجة المفاهيمية والمشاكل التى قامت بحلها هذه الطريقة ، بالاضافة الى اللغات البرمجية التى من خلالها يمكن تطبيق البرمجة المفاهيمية. الفصل الرابع (جودة البرمجيات) فيتكون تعريف الجودة من وجهات النظر المختلفة ، كما تناول بعض نماذج جودة البرمجيات بالاضافة الى القياسات المستخدمة فى قياس الجودة ، أما الفصل الخامس (تحليل وتصميم النظام) فيتكون من تحليل وتصميم النظام الذى قام بتطويره الباحث باستخدام البرمجة الكائنية والبرمجة المفاهيمية . أما الفصل السادس (النتائج الأحصائية) فيتعرض الى تحليل الاستبيانات التى قام بملئها المتطوعون بالاضافة الى نتائج تطبيق قياسات جودة البرمجيات على الأنظمة التى قاموا بتطويرها ومقارنة النتائج بين البرمجة المفاهيمية والكائنية . أما الفصل السابع (الخاتمة والتوصيات) فتناول خاتمة البحث، وتوصيات الباحث، بالإضافة الى المصادر والمراجع، واخيرا الملاحق.

العنوان:	تأثير البرمجة ذات التوجه المفاهيمى على جودة البرمجيات
المؤلف الرئيسي:	عثمان، أحمد سيد أحمد علي
مؤلفين آخرين:	حاج على، عوض(مشرف)
التاريخ الميلادي:	2015
موقع:	الخرطوم
الصفحات:	1 - 240
رقم MD:	830496
نوع المحتوى:	رسائل جامعية
اللغة:	Arabic
الدرجة العلمية:	رسالة دكتوراه
الجامعة:	جامعة النيلين
الكلية:	كلية الدراسات العليا
الدولة:	السودان
قواعد المعلومات:	Dissertations
مواضيع:	هندسة البرمجيات، برمجة المفاهيمية، جودة البرمجيات، التحليل البرمجى
رابط:	https://search.mandumah.com/Record/830496

الفصل الثانى (الدراسات السابقة –

(PREVIUOS STUDIES

2.1 مقدمة

أنصب اهتمام الباحثين فى مجال هندسة البرمجيات منذ ظهورها والى يومنا هذا نحو الأهتمام بمعالجة التحديات التى تواجه هندسة البرمجيات. ووفقا للكاتب [Ian Summervil] [2] فأن هذه التحديات تنقسم الى ثلاثه تحديات رئيسية هى:

1- تسليم البرمجيات فى الزمن المحدد [Delivery]

2- أنتاج برمجيات يمكن الاعتماد عليها (الجودة) [Trust]

3- تنوع بيئات التشغيل [Heterogeneity]

وكنتيجة لهذه البحوث ظهرت مجموعة من التظريات Methods والادوات Tools والتقنيات Techniuques فى مجال هندسة البرمجيات والتى سعت بطريقة أو أخرى لمعالجة هذه التحديات، ومن المجالات التى نالت حظا وافرا من الباحثين هى كيفية تقسيم البرنامج Modularity بطريقة تمكن من سهولة كتابته وبالتالى تقليل زمن التطوير كما تزيد من مقروئية الكود – Code Readability وتؤثر أيجابا على امكانية الصيانة Maintainabiliy فى المستقبل، ومن اخر هذه التقنيات فى مجال البرمجة هى البرمجة المفاهيمية والتى بعد ظهورها لاقت اهتماما كبيرا من الباحثين والمطورين وذلك لما تحمله من ميزات واعدة يمكن ان تساهم بصورة ايجابية فى تطوير البرمجيات، ونسبة لأن البرمجة المفاهيمية تعتبر من التقنيات الجديدة والتى لم تبلغ مرحلة النضج بعد، فنجد ان مجتمع البرمجة المفاهيمية أنصب أهتمامه على تطوير هذه الطريقة أكثر من أهتمامه بدراسة التأثيرات التى يمكن أن تنتج عن أستخدامها، ولذلك نلاحظ انه توجد دراسات قليلة [17]

أهتمت بدراسة التأثيرات المحتملة من أستخدام البرمجة المفاهيمية على جوانب تطوير البرمجيات المختلفة مقارنة بتلك الدراسات التى أهتمت بتطوير الطريقة نفسها، فى هذا الفصل سنتناول بعضا من الدراسات التى أهتمت بدراسة تأثير البرمجة المفاهيمية على جودة البرمجيات.

2.2 الدراسات السابقة

2.2.1 البرمجة المفاهيمية وجودة البرمجيات

Rpoger Alexander and James Bieman، 2004، Aspect Oriented Technology ans Software Quality [15]

ذكرت الدراسة ان البرمجة المفاهيمية من الطرق الجديدة فى البرمجة والتى لاقت اهتماما من الباحثين ومجتمع المطورين، وهى عبارة عن طريقة لتطوير البرمجيات تهتم بفصل الاهتمامات عن بعضها البعض، وعلى الرغم من الفوائد المرتبطة بهذه الطريقة والمتمثلة فى تقليل اسطر الكود المكتوبة والتقسيم الجيد للبرنامج الا انها كاى تقنية اخرى تقابل هذه الميزات تكلفة، من خلال الدراسة تعرض الباحثان الى التكلفة المرتبطة بهذه الطريقة وذلك لاظهار الجانب السلبي فى هذه الطريقة

عدد من الباحثين والمطورين قاموا بدراسات عن الفوائد المرتبطة باستخدام البرمجة المفاهيمية، ووجد الباحثان عدد قليل جدا من البحوث عن التكلفة المرتبطة بالبرمجة المفاهيمية، وكأى تقنية جديدة فان الفوائد تكون واعدة، ولكن اى تقنية جديدة تجلب ايضا تكلفة اضافية مرتبطة بها، وأذا ما تم اعتماد البرمجة المفاهيمية فان ذلك سيكون له تاثير كبير على هندسة البرمجيات

ككل، لذلك فان الفهم الجيد لقصور هذه التقنية سيمكن من معرفة التاثيرات الجانبية السيئة لها، ويأمل الباحثان من خلال الاسئلة والجوانب التى تمت مناقشتها فى البحث من المساعدة فى تطوير هذه التقنية الجديدة بتلافى القصور فيها.

البرمجة المفاهيمية هى تطوير للبرمجيات عن طريق فصل الاهتمامات، ففى الطرق التقليدية نجد ان بعض الاهتمامات Concerns لا يمكن تطبيقها الا من خلال توزيع الكود الخاص بها على وحدات النظام المختلفة، كأمنية النظام على سبيل المثال، فنجد ان الكود الذى يقوم بتطبيق سياسة امنية محددة يتوزع على فصائل Classes متعددة، ولكن عن طريق البرمجة المفاهيمية فان مثل هذا الكود سيكتب مرة واحدة وفى وحدة واحدة تسمى بالاسبكت Aspect ، فالاسبكت يمكن من أنشاء وحدة واحدة Module تقوم بتنفيذ اهتمام محدد كان فى السابق تنفيذه يتطلب ان يتم توزيعه على الاهتمامات الرئيسة Primary Concerns ، وبالتالى نجد انه فى البرمجة المفاهيمية ان فصل الاهتمامات الثانوية عن الاهتمامات الرئيسية يجعل الاهتمامات الرئيسية أكثر تماسكا بالاضافة الى عدم الحوجة الى ادارة اجزاء من النظام ليس لها علاقة مباشرة بالاهتمام المحدد. نجد ان التاثير العملى لكتابة الاسبكت Aspect هو كتابة كود اقل، فكل الكود الذى كان موزعا على الفصائل المختلفة الان أصبح فى وحدة واحدة ولاحقا سيتم توزيعه على الفصائل المرتبط بها اثناء التنفيذ من خلال عملية تسمى بالنسج او الحياكة Weaving.

2.2.1.1 التأثير على فهم البرنامج Understandability Effects

أن أحد المفاهيم الرئيسية فى هندسة البرمجيات فى التصميم والتنفيذ هى تقليل الاقتران Coupling، وبصورة عامة فان البرمجيات التى يوجد أقتران اقل بين أجزائها نجد انها أسهل فى الفهم، ولكن هذا المفهوم يتم التضحية به الى حد ما من أجل الحصول على فوائد أخرى من بعض التقنيات الحديثة، ومن الامثلة الواضحة على ذلك أستخدام الوراثة فى البرمجة الكائنية فنجد ان الفصائل الابناء يكونون مقترنين اقترانا قويا بالاباء، ولفهم اى فصيل من الابناء لا بد من فهم الفصيل الاب، بل اكثر من ذلك نجد ان التغيير فى الفصيل الاب ربما يتطلب تغييرا فى الفصائل الابناء تبعا لذلك، وهذه هى تكلفة أستخدام تعدد الاشكال Polymophism فى البرمجة الكائنية.

البرمجة المفاهيمية تواجه نفس الاشكال، أولا نجد ان الاسبكت Aspect لا يمكن ان ينفذ لوحده ، وبالتالى فان فهمه يحتاج الى معرفة بالاهتمامات الرئيسية التى سيتم الحاقه بها، ،والعكس صحيح فمن الجانب الاخر نجد أن فهم الاهتمامات الرئيسية يتطلب أيضا فهم الاسبكت Aspect المرتبط بها والذى يتم الحاقه بها. لفهم اسبكت Aspect واحد فان ذلك يتطلب فهم أجزاء اخرى كثيرة من البرنامج، ويصبح الوضع أسوأ أذا كان لدينا أكثر من اسبكت Aspect يتم نسجهم مع الاهتمامات الرئيسية أثناء التنفيذ، فنجد انه من الصعب فهم الكود كما يصعب التنبؤ بسلوك البرنامج، والنتيجة انه ليس فقط البرنامج يصبح صعب الفهم بل ان ذلك يمكن ان

يولد أخطاء من الصعب تحديدها وبالتالى فأن السؤال المهم المطروح عند هذه النقطة هو هل فوائد هذه التقنية ما زالت جيدة فى مقابل التكلفة ؟

2.2.1.2 الخصائص الناشئة ومعالجة الأخطاء Emergent properties and fault resolution

عند حدوث خطأ ما فأن التحدى الاساسى هو أكتشاف مكان الخطأ، فى طرق التطوير الاخرى نجد ان عملية أكتشاف مكان الاخطاء تطلب فحص الكود وبالتحديد الكود الذى يتوقع ان يوجد فيه الخطأ، مع البرمجة المفاهيمية يمكننا أستخدام نفس الاسلوب فحص الكود، ولكن نجد انه من غير الكافى فحص كود الاهتمامات الرئيسية فقط بل يجب فحص كود الاهتمامات الثانوية كذلك، وكنتيجة لعملية النسيج او الحياكة Waeving Process فان الخطأ يمكن ان يوجد فى أحدى أربع أماكن محتملة وهى:

1- أن يكون الخطأ موجودا فى جزء من الاهتمامات الرئيسية والتى لم تتاثر أصلا باى من الأهتمامات الثانوية
2- أن يكون الخطأ موجودا من ضمن كود الاسبكت Aspect وبصورة مفصولة عن عملية الحياكة او النسيج
3- الخطأ عبارة عن خاصية ناشئة من التفاعل الذى يحدث بين الاسبكت Aspect والاهتمامات الرئيسية، وهذا يحدث بعد عملية الحياكة والتى تضيف عمليات او بيانات اضافية لم تكن موجودة أصلا فى الاهتمامات الرئيسية

4- الخطأ عبارة عن خصائص ناشئة من مجموعة من الاسبكت Aspect التى يتم نسجها مع الاهتمامات الرئيسية

وتنوع اماكن الخطأ ينتج عنها جهد أضافى فى عملية الاختبار لتحقيق مستوى الجودة المطلوب

2.2.1.3 *التغييرات الضمنية فى البنية النحوية والدلالات* *Implicit changes in syntactic structure and semantics*

أعتمادا على كيفية أستخدامها، الاسبكت Aspect يمكن ان تغير البنية النحوية والدلالات للاهتمامات الرئيسية، واحد السيناريوهات لذلك انه اثناء مراجعة لتحسين الكود تم تحويل جزء كبير من الكود الموجود فى الاهتمامات الرئيسية الى الاسبكت Aspect، والتبرير لذلك هو ان جزء كبير من الكود كان عبارة عن كود موزع ، وكانت النتيجة ان كود الاسبكت Aspect اصبح اكبر بكثير من كود الاهتمامات الرئيسية، وعند تنفيذ البرنامج فان الناتج كان مغايرا للكود الاساسي قبل فصل الاهتمامات، أما السيناريو الاخر كان على العكس تماما اذ تم كتابة كود الاسبكت Aspect اولا قبل كتابة كود الاهتمامات الرئيسية ، وفى هذه الحال نجد ان الشخص الذى يقوم بكتابة كود الاسبكت Aspect ينبغى ان يكون على دراية تامة ببنية ودلالات كود الاهتمامات الرئيسية. وبغض النظر عن السيناريوهات التى تم ذكرها نجد ان التحكم واعتمادية البيانات الناتجة عن عملية النسج تكون مختلفة تماما عن تلك الموجودة اصلا فى الاهتمامات

الاساسية. وبالتالى فانه لا يمكن ملاحظة هذه الاعتمادية والتحكم لانها لا تحدث الا أثناء التنفيذ وخلال عملية النسج.

2.2.1.4 *التأثير على البنية المعرفية* *Effects on cognitive burden*

عملية النسج Weaving يمكن ان تغير من البنية الاساسية للاهتمامات الرئيسية، فأذا قام كاتب الكود بوضع أفتراض محدد أثناء كتابة الاهتمام الرئيسي فيمكن للاسبكت Aspect ان يغير من تلك الفرضية وذلك عن طريق الحاق فرضية أخرى بالاهتمام الرئيسي يمكن ان تكون متضاربة مع الفرضية الاساسية التى قام بوضعها كاتب كود الاهتمام الرئيسي، وهذه المشكلة يمكن ان تظهر بصورة جليه اذا ما تمت كتابة كود الاهتمام الرئيسي و الاسبكت Aspect من قبل اكثر من شخص.

خلصت الدراسة الى مجموعة من الاسئلة والتى ينبغى الأجابة عليها، وهى :

- كيف يمكن قياس التعقيد Complexity فى البرمجة المفاهيمية التوجه، بالاخص وان عملية النسج تتم اثناء التنفيذ؟ وهل يجب توقع التعقيد قبل عملية النسج؟
- هل يمكن التحكم فى منع التاثير على البنية المعرفية للاهتمامات الرئيسية؟
- كيف يمكن أجراء صيانة على برنامج مكتوب باستخدام البرمجة المفاهيمية التوجه؟ مع الاخذ فى الاعتبار ان التغيير فى الاسبكت Aspect يمكن ان يولد أخطاء ومشاكل فى الاجزاء التى سيتم الحاقه بها.

- كيف يمكن اختبار برنامج مكتوب بالبرمجة المفاهيمية بكفاءة؟ ما هى التقنيات التى يجب أستخدامها ؟ وهل التقنيات المتوفرة حاليا كافية؟
- كيف يمكن تحليل برنامج مكتوب بالبرمجة المفاهيمية؟ وما هو التمثيل المناسب لبنية النظام؟ البنية الساكنة للنظام قبل عملية النسج مطلوبة ولكنها غير كافية ولا تعكس البرنامج بصورة حقيقية.

2.2.2 أنتاج برمجيات عالية الجودة بأستخدام البرمجة المفاهيمية التوجه

Andrew Matthews، 2011 Producing High Qulity Software with Aspect Oriented Programming [16]

لسنوات ظل مهندسو البرمجيات مشغولون بالاجابة على السؤال: ماهى الطريقة الانسب لهيكلة الكود؟ لزيادة امكانية أعادة الاستخدام وتقليل الاخطاء، بينما قدمت البرمجة الكائنية أطار عمل قوى لتنظيم الكود ولكنها اخفقت عندما كانت هنالك حوجة لتنفيذ خصائص تتوزع على جميع النظام، فمثل هذه الخصائص تؤدى الى تكرارا غير مرغوب فيه بالنسبة للكود مما يزيد من نسبة الاخطاء وومن التطوير. أضافت البرمجة المفاهيمية درجة عالية من اعادة الاستخدام للبرمجة الكائنية التقليدية، ونجد ان البرمجة المفاهيمية أضافت:

- تقليل التكلفة وتقليل زمن التطوير
- تقليل نسبة الاخطاء
- وزيادة قابلية الصيانة

من خلال الدراسة قام الباحث بتوضيح ميزات أستخدام البرمجة المفاهيمية فى تطوير البرامج والتى لخصها فى:

2.2.2.1 تقليل التكلفة من خلال تقليل عدد اسطر الكود Lowers Cost by Reducing Lines of Code

تكلفة تطوير البرمجيات ترتبط ارتباطا وثيقا بعدد اسطر الكود التى تمت كتابتها، وتكرار اسطر الكود النمطى مثل التراجع والتنبيهات والتحقق يمكن ان تزيد من عدد أسطر الكود بنسبة تصل الى 80% من نسبة كود البرنامج الاصلى.

فى عام 2007 ومن خلال دراسة تطبيقية قام بها Greenwood اثبتت الدراسة ان البرمجة الكائنية يمكن ان تقلل عدد الاسطر فى البرنامج بحوالى 15%، وأهمية البرمجة المفاهيمية تزيد كلما زاد عدد اسطر الكود فى البرنامج.

2.2.2.2 تقليل الاخطاء فى البرنامج Decrease Defects

من خلال دراسة تم اجرائها فى 2008 عن طريق Eddy وجد ان عدد الاخطاء فى خاصية محددة يزيد مع زيادة عدد أسطر الكود ومع زيادة توزيع الكود على اجزاء مختلفة من النظام. البرمجة المفاهيمية عالجت الجانبين ، فعدد اسطر الكود أصبح أقل، كما ان توزيع كود خاصية محددة اصبح اقل من خلال استخدام الاسبكت Aspect. فمع ان الاسبكت Aspect نفسه يمكن ان يحتوى على اخطاء الا اننا نجد ان معالجة هذه الاخطاء أسهل بكثير من تتبع الكود الموزع فى فصائل مختلفة ومن ثم معالجته.

2.2.2.3 سهولة تنفيذ ضبط الجودة *Ease Implementation of Quality Assurance*

البرمجة المفاهيمية تساعد على تقليل عدد الاخطاء لانها تفصل خصائص الجودة عن الاهتمامات الرئيسية فى النظام، وبالتالى تصبح عملية تنفيذ ومراقبة خصائص الجودة فى النظام أسهل، وفى حالة حدوث خطأ فى أحد أكواد خصائص الجودة يمكن أزالته بسهولة او تعديله دون التاثير على كود الاهتمام الرئيسى او ما يعرف بمنطق العمل Business Logic.

2.2.2.4 تحسين قابلية الصيانة *Improves Maintainability*

ان تطوير البرامج لا يمثل الا مرحلة اولى فى دورة تكلفة حياة الانظمة، فالانظمة تحتاج الى ان تتطور خلال ما يعرف بالصيانة، ومن خلال الدراسات وجد ان المطورين يقضون زمنا اطول فى قراءة الكود أثناء الصيانة من الزمن المستغرق فى كتابة كود الصيانة نفسه، تحليل الكود المكتوب عملية مهمة لفهم التغيرات التى قام باجرائها اعضاء الفريق الاخرين، بل حتى للذين قاموا بكتابة الكود يحتاجون الى بعض الوقت حتى يستطيعوا ان يتذكروا بينة النظام.

فريق العمل الذى يقوم بتقليل نسبة الكود الموزع يجعل من السهل على فريق الصيانة ان يجد اجزاء الكود التى تتاثر بتغيير محدد، البرمجة المفاهيمية تمكن من جعل كود منطق العمل نظيفا من خلال الاكواد التقنية وتلك المتعلقة يالجودة. بالاضافة الى ان الدراسات ايضا اثبتت ان البرمجة المفاهيمية تحافظ على فصل الخصائص عن بعضها البعض مع مرور الزمن وخلال الاصدارات المختلفة للبرنامج

2.2.2.5 تحسين روح العمل كفريق *Improves Teamwork*

واحدة من اعظم الفوائد الجانبية للبرمجة المفاهيمية تحسين التعاون بين أعضاء فريق التطوير، وذلك من خلال فصل الاهتمامات الواضح فى التطبيقات الكبيرة والمعقدة، وهذا ما يساعد على يركز كل عضو فى الفريق على العمل الذى يقوم به دون الحوجة الى معرفة كل تفاصيل الاجزاء الاخرى من البرنامج.

2.2.3 دراسة استكشافية لدراسة أثر البرمجة المفاهيمية على قابلية الصيانة

Bartsch، M.، & Harrison، R. (2008). An exploratory study of the effect of aspect-oriented programming on maintainability. Software Quality Journal، 16(1)، 23-44. [17]

الدراسة عبارة عن تجربة لدراسة تأثير البرمجة المفاهيمية على قابلية الصيانة، التجربة تمت عن طريق تكليف 11 مطور بأجراء صيانة على برنامجين مختلفين الاول مكتوب بلغة الجافا والثانى مكتوب بلغة AspectJ، البرنامجين عبارة عن تنفيذ لنفس المتطلبات وبنفس واجهة المستخدم، البرنامج عبارة عن نظام تسوق، تمت دراسة النتائج وتحليلها احصائيا.

فى الدراسة تم اعتماد تعريف Bohem، Brown and Kaspar لقابلية الصيانة فى العام 1978م ، والذى عرف الصيانة على انها سهولة الفهم Undertandability وسهولة التعديل Modifiability، وقابلية الصيانة عبارة عن خاصية جودة خارجية بمعنى انه لا يمكن قياسها مباشرة من خلال الكود ، لذلك سيتم أختبار متغيرات أستجابة مرتبطة بخاصية قابلية الصيانة، وهذه المتغيرات هى :

1- سهولة الفهم Understandability والتى تحتوى على متغيرات الاستجابة

أ. تعريف المكونات Identification of Componenets وتشمل تعريف الفصائل والاسبكت Aspect المكونة للبرنامج وقياس زمن التعريف.

ب. تعريف العلاقات Identification of Relationships وتشمل : تتبع التحكم ومخطط سير البرنامج لتعريف المخرجات والزمن المستغرق فى التعريف

ت. فهم البرنامج ككل، وهو عبارة عن تقييم

2- قابلية التعديل Modefiability والتى تحتوى على المتغيرات التالية:

أ. عدد اسطر الكود من غير التعليقات NCLOC اللازمة لأجراء التعديل

ب. الزمن المطلوب لأجراء التعديل

ولقياس المتغيرات السابقة تم تصميم أستبيان يحتوى على اربعة اسئلة هى

1- تعريف الفصائل والاسبكت Aspect المكونة للبرنامج

2- تعريف مخرجات النظام

3- اضافة متطلب جديد للبرنامج. والمتطلب الجديد عبارة عن خاصية موزعة بمعنى انها تؤثر على كل اجزاء النظام

4- تقييم سهولة النظام بمقياس من 1 الى 5

بالنسبة للمطورين تم أختيارهم بصورة عشوائية وجميعهم لديهم خبرة من سنتين الى 5 سنوات فى البرمجة الكائنية، ولكن كلهم ليس لديهم اى خبرة او سابق معرفة بالبرمجة المفاهيمية، وعند تقسيم الاحدى عشر مطورا على البرنامجين الكائنى والمفاهيمى تم توزيعهم أيضا بصورة عشوائية، تم أعداد كورس مبسط لتعليم المطورين طريقة البرمجة المفاهيمية، الكورس كان على الانترنت وكل مطور قام به على حده.

خلصت التجربة الى ان الزمن المستغرق للاجابة على اسئلة الاستبيان كان اقل لمجموعة المطورين الذين عملوا على برنامج المنهجية الكائنية مقارنة بالذين عملوا على برنامج المنهجية المفاهيمية. أيضا خلصت الدراسة الى انه لا يوجد فرق فى تقييم فهم البرنامج بين المجموعتين على الرغم من ان المطورين المشاركين فى التجربة ليس لديهم خبرة مع البرمجة المفاهيمية ولديهم خبرة مع البرمجة الكائنية. أيضا فيما يتعلق بعدد أسطر الكود من غير التعليقات لم يكن هنالك فرق واضح بين الطريقتين،

2.2.4 أختبار البرمجة المفاهيمية فى مجال الصناعة

Bradley، J. T. (2003). An examination of aspect-oriented programming in industry. Colorado State University، Colorado، USA. [19]

تناولت الدراسة تاثير البرمجة المفاهيمية على خصائص الجودة التعقيد Complecxity ، الصحة Correctness ، وقابلية الاختبار Testability. الطريقة التى تم استخدامها لاجراء الدراسة هى أجراء مقابلات مع المطورين الذين يستخدمون البرمجة المفاهيمية فى تطوير الانظمة على ارض الواقع.

2.2.4.1 البرمجة المفاهيمية وتعقيد البرمجيات Software Complexity

لا شك ان درجة تعقيد البرمجيات له علاقة كبيرة جدا بجودة البرمجيات، ولكن قياس التعقيد لا يزال من المسائل المعقدة والتى لها طرق كثيرة ، اعتمدت الدراسة مقياس التعقيد الهيكلى Structural Comlexity للكود، وعادة ما يحاول المطورون تقليل التعقيد الهيكلى للبرنامج لما له من اثار على جودة البرنامج مثل زمن التنفيذ والمساحة المطوبة عوضا عن نسبة الاخطاء، كما اعتمدت الدراسة مقياس التعقيد الادراكى Cognitive والذى يمكن تعريفة على انه مقروئية الكود Readability وسهولة فهم الكود Understandability بالنسبة للبرمجين، التعقيد الادراكى يؤدى الى تاخير زمن التطوير وصعوبة الصيانة فى المستقبل . على الرغم من انه للوهلة الاولى يستبين بان المقياسين مرتبطين مع بعضهما البعض الا ان ذلك غير صحيح فقد نجد ان برنامجا ما سهل جدا من حيث التعقيد الهيكلى ولكنه صعب جدا للقراءة والفهم من قبل المطورين.

2.2.4.1.1 تأثير البرمجة المفاهيمية على التعقيد الهيكلى

أن تاثير البرمجة المفاهيمية على التعقيد الهيكلى غير واضح بصورة كاملة، ولكن بعض الامثلة الحقيقية من الواقع بينت ان البرمجة المفاهيمية يمكن ان تزيد من التعقيد الهيكلى للبرنامج، على الرغم من ان احد اهداف البرمجة المفاهيمية هو تقليل التعقيد الهيكلى للنظام.

أحد هذه الامثلة كان لنظام كبير به اكثر من مليون سطر كود، والتعقيد الذى يواجهونه انه بسبب ان عملية النسيج تتم بصورة متكاملة على كل اجزاء النظام فانه لا يمكن ان تتم ترجمة جزء من النظام. وبالتالى فان البرمجة المفاهيمية زادت من درجة الاعتمادية بين مكونات النظام المختلفة. وهذه الاعتمادية الكبيرة بين اجزاء النظام المختلفة فى البرمجة المفاهيمية بالاضافة الى التعقيد الذى تضيفة اثناء كتابة الكود والمتمثل فى ان المطورين الذين يكتبون كود الاسبكت Aspect عليهم ان يكونوا على دراية كاملة بالاجزاء الاخرى فى البرنامج التى سيتم الحاق الاسبكت Aspect بها ، فأن هذه الاعتمادية تتطلب ان تتم ترجمة النظام ككل فى كل مرة يتم فيها تغييرات وأذا اخذنا فى الاعتبار ان بعض الانظمة الكبيرة تتطلب ساعات او حتى أيام لاكمال عملية الترجمة Compilation فأن الوضع سيكون أسوأ.

2.2.4.1.2 تأثير البرمجة المفاهيمية على التعقيد الأدراكى Cognitive Complexity

أحد أهداف هندسة البرمجيات هو ازالة التعقيد الادراكى، والبرمجة المفاهيمية أحد الركائز التى قامت عليها هى أزالت التعقيد الأدراكى، ولكن التحليل أوضح ان أستخدام البرمجة المفاهيمية فى المشاريع الحقيقة قد ولد بعض المشاكل التعلقة بفهم البرنامج، وعلى الرغم بانه يمكن ارجاع

بعض هذه المشاكل الى عدم معرفة المطورين بتقنية البرمجة المفاهيمية بصورة كافية وبعضها يرجع الى الالية التى تعمل بها لغات البرمجة التى تدعم البرمجة المفاهيمية.

التحليل اوضح ان استخدام البرمجة المفاهيمية يحد من قدرة أعضاء الفريق على العمل بصورة مستقلة لانها تزيد من التعقيد الادراكى على المطورين، وذلك يرجع لسبب ان المبرمجين يقع عليهم عبء أضافى وهو معرفة التفاعل الجديد بين الاجزاء التى يقومون بتطويرها واجزاء البرنامج الاخرى ، بالأضافة الى معرفة كيف ستتاثر الاجزاء التى يعملون عليها بالاجزاء الاخرى من البرنامج. المشكلة الاخرى التى وضحت من خلال التحليل ان المبرمجين على ارض الواقع فى احد المشاريع واجهوا مشكلة انهم لا يستطيعون التعامل مع النتائج النهائية وذلك بسبب عملية النسج التى تتم أثناء الترجمة او التنفيذ.

2.2.4.2 البرمجة المفاهيمية والصحة Correctness

كتابة برنامج ينفذ بصورة صحيحة واحد من الاهتمامات الرئيسية لاى مطور برمجيات، وهندسة البرمجيات لها عدد من الادوات والتقنيات التى تساعد على تقليل الاخطاء فى البرنامج، ولكن لسوء الحظ نجد ان تقنية كالبرمجة المفاهيمية أضافت بعض التحديات الجديدة عند كتابة برنامج يؤدى وظائفه بصورة صحيحة، واهم هذه التحديات هو كيف سيتم التعامل مع عملية النسج.

أن الخصائص المدمجة هى أحد المشاكل الرئيسية التى تواجه البرمجة المفاهيمية، وبالاخص عندما يكون لدينا أكثر من اسبكت Aspect سيتم نسجه والحاقه بواحد من الاهتمامات

الرئيسية، وهذه المشكلة ترجع الى أن مترجمات البرمجة المفاهيمية وأطر العمل المتوفرة لم تصل درجة النضج الكافى بعد. ففى هذه الحالة نجد ان الاسبكت Aspects التى يتم نسجها والحاقها بأهتمام رئيسي تتفاعل مع بعضها البعض بطريقة غير متوقعة وغالبا ما تنشأ منها خصائص جديدة لم تكن موجودة فى الاهتمام الرئيسي او الاسبكت Aspect. فى أحد المشاريع الحقيقية أوضح الفريق أنه يواجه مشكلة كبيرة مع الخصائص الناشئة والناتجة عن تنفيذ كود الاسبكت Aspect بترتيب عشوائى، وهذه المشكلة من المشاكل التى لا يمكن حلها، ولكن نفس الفريق اوضح ان المشكلة تتعلق بمترجم لغة AspectJ وبالطريقة التى يكتبون بها الاسبكت Aspect.

فى بعض المشاريع قام المطورون بحل المشكلة عن طريق التاكد من انه لا يوجد أكثر من اسبكت Aspect واحد ستم نسجه والحاقه بأى أهتمام رئيسي.

2.2.4.3 البرمجة المفاهيمي وقابلية الاختبار Software Testability

البرمجة المفاهيمية ولدت مشاكل ليس مع صحة البرنامج فقط بل أيضا مع أمكانية أختبار البرنامج، ويرجع ذلك ايضا الى نفس السبب وهو عملية النسج، فنجد أن أختبار الوحدات Unit Testing من الصعب بل يستحيل تطبيقه.

أن عملية أختبار الاسبكت Aspect بصورة مستقلة تعتبر صعبة ومستحيلة ولا توجد حتى الان طريقة محددة يمكن من خلالها اختبار الاسبكت Aspect على عكس الفصائل والدوال والاجراءات والتى يمكن أختبارها بصورة مستقلة، فى أحد المشاريع الحقيقية أوضح فريق العمل بأن المشكلة التى يواجهونها مع أختبار الاسبكت Aspect يقومون بحلها عن طريق

أختبار الاسبكت Aspect فى بيئة شبيهة بالبيئة التى سيتم أدماجه فيها لاحقا، ولكن هذا الأختبار أيضا لا يقوم بأختبار الاسبكت Aspect بصورة مستقلة كما انه لا يضع فى الاعتبار الخصائص الناشئة من تفاعل الاسبكت Aspect مع الاهتمامات الرئيسية فى البرنامج الحقيقى، وتكون المشكلة الاكبر عندما يوجد أكثر من اسبكت Aspect يتم الحاقهم بنفس الاهتمام الرئيسي، فأذا حدث خطأ ما فانه يحتمل ان يوجد فى كود الاسبكت Aspect او فى الخصائص الناتجة أو فى كود الأهتمام الرئيسي.

على الجانب الاخر نجد ان البرمجة المفاهيمية لها أستخدام أيجابي فى اختبار البرمجيات فمن خلال الدراسة أوضح احد فرق التطوير انهم يستخدمون الاسبكت Aspect لاختبار وحدات البرنامج الاخرى.

خلصت الدراسة الى ان أستخدام البرمجة المفاهيمية غير كبير فى الحياة التطبيقية العملية بعد وذلك يرجع الى ان هذه التقنية وادواتها لم تبلغ مرحلة النضج الكافى بعد مقارنة بالطرق والادوات الاخرى. بالأضافة الى عدم انتشارها بصورة كبيرة فى المجال التطبيقى فمن أصل خمسة فرق عمل تم اختيارهم للدراسة فقط اثنين منهم أستطاعوا الاستجابة للمقابلات.

2.2.5 وجود مفارقة فى تطور البرمجيات التى تستخدم البرمجة المفاهيمية

Tourwé، T.، Brichau، J.، & Gybels، K. (2003). On the existence of the AOSD-evolution paradox. SPLAT: Software engineering Properties of Languages for Aspect Technologies. [36]

من المسلمات ان تطور البرمجيات Software Evolving من المراحل الحرجة والمهمة من مراحل وعلميات تطوير البرمجيات، وعلى الرغم من ان البرمجة المفاهيمية قدمت حلول جيدة لمشكلة الكود الموزع والمتداخل الا انها لم تثبت كفاءتها فى التطور المستقبلى للبرامج، فتقنيات البرمجة المفاهيمية الحالية تقوم بأنتاج برامج يكون من الصعب تطويرها فى المستقبل.

البرمجة المفاهيمية قامت بتحسين بنية البرامج، وذلك عن طريق الاسبكت Aspect والذى يحتوى على الكود الموزع، ومن المعلوم ان بنية البرنامج وطريقة تقسيم الكود لها أثر كبير على امكانية تطويره فى المستقبل، وعليه فان البرمجة المفاهيمية تبدو واعدة فى هذا المجال.

لسوء الحظ ان معظم الابحاث الموجودة الان فى مجتمع البرمجة المفاهيمية أهتمت بأنتاج البرامج فقط، وبالتالى تركز على المدى القصير فى خطوات بناء وتطوير البرمجيات وليس على المدى البعيد. والمفارقة التى تم أكتشافها ان البرمجة المفاهيمية تقوم بتسليم برامج من الصعب تطويرها فى المستقبل وذلك يرجع للأسباب التالية فى نظر الباحث:

1- بيئة التطوير المتوفرة حاليا والتى لا تلقى الدعم الكافى

2- الأرتباط الكبير الذى يكون بين الاسبكت Aspect وأجزاء البرنامج الاخرى

تقنيات البرمجة المفاهيمية الحالية لا تتيح الية جيدة لتعريف نقاط القطع Cross-Cuts فى الكود، وبالتالى عند تعريفعها لا بد من معرفة البنية التفصيلية للكود الذى سيتم قطعه وذلك من خلال معرفة الاسماء أو وجود أتفاقيات مسبقة، وذلك يولد ثلاثة أنواع من المشاكل:

1- من المسلمات ان الاتفاقات المسبقة غالبا ما يتم خرقها بالاخص عند تطوير البرامج الكبيرة والعقدة، وذلك يرجع لعدم أكتمال التوثيق أو لضيق زمن التطوير

2- لدينا أحساس بان البرنامج يجب ان لا تتم مراجعته للتحسين، وذلك بنية ان نمكن المطور من كتابة كود الاسبكت Aspect . وبالتالى فأن البرنامج يجب ان يبقى دون تعديل.

3- فى البرامج الكبيرة والتى يمكن ان تحتوى على أكثر من أسبكت Aspect نجد أن مسألة الأتفاق المسبق تكون صعبة للغاية وذلك نسبة للتغيرات التى يمكن أن تطرا على نقاط القطع.

أقترحت الدراسة ان يتم أستخدام تقنيات تساعد فى تعريف نقاط القطع بصورة أفضل دون أن يكون هنالك تقييد على المطورين بمعرفة الاجزاء الاخرى.

2.2.6 أستخدام البرمجة المفاهيمية فى المحاكاة المتقطعة

Chibani، M.، Belattar، B.، & Bourouis، A. (2014). Practical benefits of aspect-oriented programming paradigm in discrete event simulation. Modelling and Simulation in Engineering، 2014، 47. [33]

عند تصميم البرمجيات، لا بد من الاهتمام بعوامل الجودة مثل المتانة والقدرة على التكيف، وإعادة استخدام. وتسمى هذه المتطلبات بالمتطلبات غير الوظيفية او الأهتمامات. وجاءت البرمجة المفاهيمية لتدعم فصل الاهتمامات عن بعضها البعض. وعالجت البرمجة المفاهيمية مشكلة الكود الموزع مثل كود الامنية والذى يتوزع على وحدات مختلفة من البرنامج كما عالجت مشكلة الكود المتداخل عندما تحتوى وحدة واحدة فى البرنامج على أكثر من أهتمام. ونجد ان الكود المتداخل والموزع يقلل من سهولة صيانة البرامج فى المستقبل.

البرمجة المفاهيمية لها أكثر من أطار عمل ،وأكثر من لغة برمجة هذه الدراسة طبقت على نظام محاكاة متقطع بأستخدام لغة البرمجة AspectJ

خلصت الدراسة الى ان أستخدام البرمجة المفاهيمية فى المحاكاة المتقطعة من المجالات الواعدة التى تستخدم فيها البرمجة المفاهيمية، والتى أستطاعت ان تعالج أستخدام البرمجة الكائنية فى المحاكاة المتقطعة، فقد مكنت البرمجة المفاهيمية من فصل أحداث المحاكاة المتقاطعة، كما ان برامج المحاكاة بأستخدام البرمجة المفاهيمية تميزت بالتقسيم الجيد للوحدات Modularity ، أعادة الأستخدام Reusability ، قابلية الصيانة Maintainability والوضوح Visibility

2.2.7 ملائمة أعادة الأستخدام للبرمجة المفاهيمية، تقييم وتحليل

Chaudhary، R.، & Chatterjee، R. (2014، February). Reusability in AOSD-The aptness، assessment and analysis. In Optimization، Reliabilty، and Information Technology (ICROIT)، 2014 International Conference on (pp. 34-39). IEEE. [35]

البرمجة المفاهيمية تقنية لمعالجة المفاهيم المتقاطعة، هدفها هو تسهيل الصيانة ودعم أعادة الأستخدام. ونعنى بأعادة الأستخدام هو تكلفة نقل وحدة الى برنامج اخر، وهى من المناطق المهمة لتطوير البرامج، وذلك لان الوحدات التى تم أعادة أستخدامها تساعد فى فهم الكود بصورة أفضل وتقليل مجهود الصيانة بالنسبة للبرامج. لذلك من المهم تقييم تقييم أعادة أستخدام مكون قبل أن يتم أدماجه فى النظام، الورقة أهتمت بتقييم أعادة الأستخدام لبرامج مكتوبة بلغة الجافا التى تدعم البرمجة المفاهيمية، تم أستخدام المنطق الضبابي Fuzzy Logic وبرنامج الـ MATLAB فى تقييم أعادة الأستخدام.

نجد أن البرمجة المفاهيمية جاءت لتدعم مفهوم فصل الأهتمامات Seperation of Concerns – SOC ، وفصل الاهتمامات نعنى به تقسيم البرنامج الى وحدات وخصائص يكون التداخل الوظيفى بينها أقل ما يمكن، فصل الأهتمامات يمكن تحقيقه من خلال تقنيات تقسيم البرنامج الى وحدات Modularity وأخفاء المعلومات، توجد كثير من البحوث التى تقيم أعادة الأستخدام بالنسبة للبرمجة الكائنية، ولكن بالنسبة للبرمجة المفاهيمية ما زالت تحت الدراسة،

على الرغم من أن البرمجة المفاهيمية قامت بتقسيم البرنامج الى وحدات بطريقة لا يمكن تحقيقها من خلال الطرق الاخرى وبالأخص مع الاهتمامات الموزعة، وبتطبيق البرمجة المفاهيمية نجد أن اهتمامات مثل الأمنية يمكن أن تفصل من البرنامج وتؤدى الى زيادة قابلية الصيانة وزيادة أعادة الأستخدام.

أقترحت الدراسة نموذج لقياس أعادة الأستخدام قام بالربط بين أعادة الأستخدام وبعض خصائص الجودة الخارجية ، ومن ثم تم ربطها ببعض خصائص الجودة الداخلية التى يمكن قياسها مباشرة من خلال قياسات محددة. وبناءا على دراسات سابقة فأن أعادة الاستخدام ترتبط بالخصائص الخارجية سهولة الفهم Understandability ، تقسيم البرنامج Modularity ، سهولة الصيانة Maintainability ، وقابلية التكيف Adaptability. تم أستخدام هذه الخصائص الخارجية الاربعة كمدخل لماكينة المنطق الضبابي ومن ثم تقوم بالمعالجة وأخراج رقم واحد وهو الاخراج الذى يمثل قابلية أعادة الأستخدام والذى هو عبارة رقم فى المدى من 0 الى 50.

الدراسة قامت بأستخدام برنامجين الاول برنامج مصرفى والاخر نظام تسوق ، يحتوى الاول على 11 مكون بينما يحتوى الثانى على 12 مكون. تم أستخدام أداة لتقييم أعادة الأستخدام من خلال قياس بعض الخصائص الداخلية للكود مثل الارتباط والاستقلالية والتعقيد.

خلصت الدراسة الى أنه أذا كانت قياسات الخصائص الداخلية عالية فأن قابلية أعادة الأستخدام ستكون عالية أيضا تبعا لذلك، فى هذه الدراسة تم ربط أعادة الأستخدام بأربعة خصائص

خارجية فقط من أصل عشرة، وعند ربطها مع اعادة الاستخدام باستخدام ماكينة المنطق الضبابي فأن الاخراج كان عبارة عن 35 وهو يمثل أمكانية أعادة أستخدام عالية.

2.3 أصالة الدراسة

نلاحظ من خلال الدراسات السابقة أنها توصلت الى نتائج متضاربة بشأن تأثير البرمجة ذات التوجه المفاهيمى على جودة البرمجيات, بينما هنالك دراسات توصلت الى أيجابية التأثير نجد أن هنالك دراسات أخرى خلصت الى النقيض تماما, مما يتطلب مزيد من البحوث فى دراسة تأثير البرمجة ذات التوجه المفاهيمى على جودة البرمجيات. كما نجد أيضا أن الدراسات السابقة أعتمدت فى عملية المقارنة بين البرمجة ذات التوجه المفاهيمى والكائنى على عينات صغيرة كان أكبرها 11 عينة والتى أستخدمت فى الدراسة الاستكشافية لدراسة تأثير البرمجة المفاهيمية على قابلية الصيانة [17] , بينما فى هذه الدراسة بلغ حجم عينة الدراسة 32. أضف الى ذلك أن الدراسات السابقة أستخدمت إحدى طريقتين فى التقييم وهما أجراء قياسات على الكود المكتوب بالطريقتين أو أجراء أستطلاعات وأستبيانات, فى هذه الدراسة تم أستخدام الطريقتين معا فبالأضافة إلى أجراء قياسات على الكود المكتوب بالطريقتين تم أستطلاع أراء المتطوعين الذين قاموا بكتابة هذه الأكواد بالنسبة لتأثير البرمجة ذات التوجه المفاهيمى على الجودة.

العنوان:	تأثير البرمجة ذات التوجه المفاهيمى على جودة البرمجيات
المؤلف الرئيسي:	عثمان، أحمد سيد أحمد علي
مؤلفين آخرين:	حاج على، عوض(مشرف)
التاريخ الميلادي:	2015
موقع:	الخرطوم
الصفحات:	1 - 240
رقم MD:	830496
نوع المحتوى:	رسائل جامعية
اللغة:	Arabic
الدرجة العلمية:	رسالة دكتوراه
الجامعة:	جامعة النيلين
الكلية:	كلية الدراسات العليا
الدولة:	السودان
قواعد المعلومات:	Dissertations
مواضيع:	هندسة البرمجيات، برمجة المفاهيمية، جودة البرمجيات، التحليل البرمجى
رابط:	https://search.mandumah.com/Record/830496

الفصل الثالث (البرمجة مفاهيمية التوجه

ASPECT ORIENTED SOFTWARE -

(DEVELOPMENT

3.1 مقدمة [26]

ظهرت أولى لغات البرمجة قبل صناعة الحاسوب الحديث، وقد كانت في البداية عبارة عن شيفرات (codes) في عام 1801م اخترع العالم جوزيف – ميري جاكاردJacquard) ، (Joseph–Marie نولاً ميكانيكياً (آلة للحياكة)، يتمّ التّحكم به عن طريق البطاقات المثقّبة (Punch Cards) والبطاقة المثقّبة عبارة عن بطاقة صغيرة الحجم مصنوعة من الورق المقوّى، تحتوي على عدّة ثقوب مرتّبة بنسق معين. وتمثل هذه الثقوب "البيانات" اللازمة لتغذية الآلة بالبرنامج المحدّد للحركة.

الكثير من مصنعي الحواسيب الأوائل أدركوا أهمية البطاقة المثقبة في تزويد آلاتهم بالأوامر اللازمة للعمل، ففي عام 1820م أو 1821م تبنّى العالم البريطاني تشارلز بابيج (Charles Babbabe) فكرة البطاقة المثقّبة لصناعة أول كمبيوتر ميكانيكي حقيقي عرفه التاريخ يدار بواسطة محرّك بخاري. وقد أسماه الآلة التحليليّة.(Analytical Engine)

كما صمم الأمريكي هيرمان هوليرث (Herman Hollerith) آلة خاصة لإجراء التعداد السكاني في العام 1890م تتم تغذيتها بواسطة بطاقة مثقبة بحجم ورقة الدولار، وتحوي البطاقة عددا من الثقوب يمثل فيها موضع الثقب معلومة محددة عن الشخص كالجنس أو مكان الولادة إلخ... وقد ساعدت هذه الطريقة الحكومة الأمريكية على إنهاء التعداد في عامين ونصف بدلا من السبعة أعوام ونصف التي احتاجها التعداد السابق

وقد استخدمت البطاقة المثقبة لفترة طويلة خلال القرن العشرين (حتى بداية السبعينات تقريباً) كأداة أساسية لتغذية الحاسوب الحديث بالبيانات [26].

3.2 تطور لغات البرمجة

3.2.1 لغات الجيل الاول

أعتمد الأساس في تطور لغة البرمجة كما نعرفها اليوم لاستعمال نظام العد الثنائي (Binary System)،(0-1) لتمثيل الأوامر والعميات الحسابية والمنطقية في الحاسوب الحديث (الذي يدار بالطاقة الكهربائية)، وتلك تعد ثورة في عالم الحاسوب وأساس لنجاحه بعد فشل عدة حواسيب تستخدم نظام العد العشري كحاسوب تشارلز بابيج.

وتعرف الأوامر المكتوبة بنظام العد الثنائي بلغة الآلة (machine language) أو الجيل الأول للغات البرمجة. وفي البدايات البرمجية كان على المبرمج كتابة البرنامج كاملاً مستخدما لغة الآلة، ولكن هذه العملية كانت صعبة ومرهقة وعرضة لكثير من الأخطاء [25]، لان المبرمج كان يقع على عاتقه الالمام بتفاصيل المكون المادى للحاسوب (Hardware) حتى يتمكن من أستخدام لغة الالة، بالاضافة الى ذلك نجد أن تنفيذ عملية بسيطة كعملية الجمع مثلا تحتاج منا الى كتابة عدد من الاسطر كما هو واضح فى المثال ادناه [24]

Location Hex	Instruction Code Binary	Instruction Code Hex	Instruction	Comments
100	0010 0001 0000 0100	2104	LDA 104	Load first operand into AC
101	0001 0001 0000 0101	1105	ADD 105	Add second operand to AC
102	0011 0001 0000 0110	3106	STA 106	Store sum in location 106
103	0111 0000 0000 0001	7001	HLT	Halt computer
104	0000 0000 0101 0011	0053	operand	83 decimal
105	1111 1111 1111 1110	FFFE	operand	-2 decimal
106	0000 0000 0000 0000	0000	operand	Store sum here

شكل 0-1 برنامج بلغة الالة لجمع عددين عشريين

3.2.2 لغات الجيل الثانى

كان لا بد من إيجاد طريقة لتمثيل الأوامر البرمجية (op–code) بعيداً عن تعقيد رموز لغة الآلة. وتمّ التفكير باستخدام شيفرة نصية مكونة من عدة حروف (من 1– 5 أحرف) لكتابة هذه الأوامر ووصف مواقع الذاكرة، عرفت باسم .(mnemonics). عند استخدام هذه التقنية في البداية كان المبرمج يستخدم الشيفرة النصية لتصميم البرنامج على الورق، ومن ثم يقوم بترجمته إلى لغة الآلة عند إدخاله إلى جهاز الحاسوب. ولكن، وبعد بعض الوقت، تم التوصل لطريقة تمكن الحاسوب من القيام بعملية الترجمة بنفسه، حيث تم إنشاء برنامج خاص سمّي باسم المجمّع (assembler) مهمته تجميع الأوامر المكتوبة بلغة الآلة من الأوامر المكتوبة على شكل شيفرة

رمزية أو نصية (mnemonics) ومن ثم تحويلها الى مصفوفة من الاحاد والاصفار او ما يعرف بلغة الالة (Machine Language). [26]

هذا التطور في عملية ترميز البرامج وترجمتها أدّى لنشوء لغة برمجة خاصة عرفت باسم لغة التجميع (assembly language) والتي تمثل الجيل الثاني من لغات البرمجة والتى عرفت ايضا بالغات متدنية المستوى (Low-Level Programming Languages). واعتبرت هذه اللغة قفزة عملاقة في عالم لغات البرمجة اذ انها الى حد ما اصبحت اقرب الى لغة الانسان من لغة الالة مما أتاح للمبرمجين ان يكتبوا برامج اكبر حجما واكثر تعقيدا. ولكن مع ذلك نجد ان هنالك عدد من العقبات التى واجهت هذا الجيل من أهمها :

- البرنامج المكتوب بلغة التجميع تتم كتابته ليتناسب مع خصائص الآلة (الحاسوب) التي سيتم تطبيقه عليها. بمعنى أن البرنامج المكتوب بلغة التجميع معتمد على الآلة التي يتم تنفيذه عليها ولا يمكن استخدامه على آلة أخرى، إلا بعد إعادة كتابته ليتلاءم مع تكوين هذه الآلة الجديدة (مثل أسماء المسجلات ومواقع الذاكرة..).
- عملية تصميم البرنامج تتطلب من المبرمج التفكير بدقائق البرنامج جميعها، من حيث تعريف اسم المسجل (register) وأسماء مواقع الذاكرة التي سيتم تطبيق أمر معين عليها، وهذا يعني أن على المبرمج أن يبني البرنامج خطوة خطوة من أدنى مستوى الى المستويات الاعلى .

الشكل ادناه يوضح شفرة برمجية مكتوبة بلغة التجميع لجمع عددين [24]

```
.model small
.data
 opr1 dw 1234h
 opr2 dw 0002h
 result dw 01 dup(?),'$'
.code
        mov ax,@data
        mov ds,ax
        mov ax,opr1
        mov bx,opr2
        clc
        add ax,bx
        mov di,offset result
        mov [di], ax

        mov ah,09h
        mov dx,offset result
        int 21h

        mov ah,4ch
        int 21h
        end
```

شكل 2-0 برنامج بلغة التجميع لجمع عددين

3.2.3 لغات الجيل الثالث

عرفت لغات برمجة الجيل الثالث باسم لغات المستوى العالى (High-level languages). فى هذا الجيل اصبحت البرامج تكتب باستخدام اللغات العادية مثل الانجليزية بالاضافة الى الرموز. وبالتالى اصبحت عملية البرمجة تركز على المشكلة موضع الحل اكثر من التركيز على تفاصيل لغة الالة. وبعد كتابة البرنامج تقوم اللغة بتحويل البرنامج الى صيغة يمكن تنفيذها وفهمها من قبل الالة (الحاسوب) ويتم ذلك باحدى طريقتين:

- المفسر (Interpreter) يقوم بتحويل الشفرة البرمجية الى لغة الالة وتنفيذها ولكنه يقوم بهذه العملية سطرا بسطر بمعنى انه يقوم بتحويل السطر الاول الى لغة الالة وتنفيذه ثم ينتقل الى السطر التالى وهكذا. وبالتالى فأذا وجد خطأ فى الشفرة البرمجية سيتم اكتشافه فقط عند التنفيذ وعندها سيتوقف التنفيذ.
- المترجم (Compiler) أيضا مهمته هى تحويل الشفرة البرمجية الى لغة الالة ولكنه يقوم بمسح البرنامج كله ومن ثم تحويله الى لغة الالة ومن ثم يقوم بعملية التنفيذ [23]

الشكل ادناه يوضح برنامج مكتوب بلغة الـ C++ لجمع رقمين

```
#include <iostream>

using namespace std;

int main()
{
   int a, b, c;

   cout << "Enter two numbers to add\n";
   cin >> a >> b;

   c = a + b;
   cout <<"Sum of entered numbers = " << c << endl;

   return 0;
}
```

شكل 0- 3 برنامج لجمع رقمين بلغة ال (C++)

3.2.4 لغات الجيل الرابع

كما نعلم فأن عملية تطوير البرمجيات شهدت تغييرات كبيرة وبالاخص بعد ظهور الانترنت ودخول البرمجيات فى أدارة الاعمال والحاجة الى أنظمة يمكن الوثوق بها والاعتماد

عليها لفترات طويلة. كل هذه التحديات فرضت على مطورى البرمجيات أن يستحدثوا طرقا جديدة أفضل لتطوير البرمجيات أبتداءا من عمليات التطوير والبرمجة والتحقق من الجودة والصيانة.

البرمجيات هى بناء نماذج حاسوبية لجزء من نظام المعلومات. وفى معظم الحالات من أجل تطوير البرمجيات نحتاج الى تقسيمها الى عدة وحدات يمكن تطويرها وأدارتها. فى العقود الاخيرة من القرن العشرين شهدنا ظهور تقنية البرمجة الكائنية والتى أحدثت تغييرا كبيرا فى طريقة تعاملنا مع انتاج البرمجيات. من خلال البرمجة الكائنية فأننا نركز على الكائنات المكونة للنظام ومن ثم نقوم بربط سلوك النظام بهذه الكائنات. وهذه التقنية أضافت الى عالم تطوير البرمجيات فوائد كبيرة لعل من أهمها أعادة الاستخدام والكبسلة. ولكن مع كل الفوائد التى جلبتها البرمجة الكائنية لكن ظهرت بعد المشاكل مثل وجود بعض المعالجات التى لا تنتمى الى كائن محدد بمعنى أنه لا يمكننا ان نضعها فى قالب محدد. على سبيل المثال أذا كان لدينا كائن وكان يحتاج الى ادارة الاتصال بقاعدة البيانات لاجراء بعض العمليات فأننا لا يمكننا وضع الكود الخاص بأدارة الاتصال بقواعد البيانات تبع هذا الكائن لان جميع الكائنات الاخرى ستكون بحاجة الى هذا الكود. وفى الماضى كنا نقوم بنسخ مثل هذا الكود ووضعه فى جميع الكائنات التى تختاج اليه مما يؤدى الى ضعف فى تصميم النظام بوحدات مستقلة. مما سيؤدى الى زيادة مجهود الصيانة مستقبلا أذ أن أجراء أى تعديل على هذا الكود سيتطلب منا أجراء تعديلات فى أماكن مختلفة من البرنامج لنفس الكود. مثل هذا الكود يعرف بالكود الموزع (Scattering Code) وهو عبارة الكود الخاص بمتطلب من المتطلبات ولكنه يتوزع على مجموعة من الوحدات المكونة للنظام. وغالبا

ما يكون الكود الموزع مرتبط بالاهتمامات الغير رئيسية فى النظام مثل المتطلبات الغير وظيفية المرتبطة بالجودة كالامنية وتسجيل الاحداث المختلفة فى النظام، عند دمج الاهتمامات الثانوية هذه مع الاهتمامات الرئيسية فى النظام او المتطلبات الوظيفية او ما يعرف أيضا بمنطق العمل Business Logic فأن ذلك نظريا يؤدى الى تعقيد البرنامج وتقليل فرص أعادة الاستخدام سواءا للأهتمامات الرئيسية أو الثانوية.

بالاضافة الى ذلك ومن ناحية أعادة الاستخدام نجد أن الوحدة التى تحتوى على كود له علاقة بأكثر من متطلب من الصعب بمكان أعادة أستخدامها فى حالات مختلفة. وهذا النوع من الكود يعرف بالكود المتداخل (Tangling Code).

بالطبع عند أستخدام أنماط الاستخدام (Design Patterns) يمكن أن تساعد فى كبسلة النظام بصورة كبيرة ويمكن أن تقلل من الكود المتداخل والموزع ولكن مع ذلك نجد أن هنالك حالات لا يمكن أن نتعامل معها.

البرمجة المفاهيمية جاءت لتعالج مشاكل الكود المتداخل والكود الموزع الذى المذكور اعلاه. بأستخدام هذه التقنية نقوم بتصميم النظام دون أن نكترث كثيرا الى مشكلة تجميع النظام فيما بعد. فعند كتابتنا لكود أى وحدة برمجية سواء ان كانت فصيل ، دالة ، الخ.. فأننا لا نهتم بالكود المتداخل لأنه سيتم كتابته لاحقا فى وحدة برمجية مستقلة تعرف بأسم الاسبكت Aspect.

3.4 ما هو الأسبكت Aspect

ببساطة هو عبارة عن نوع معين من الوظائف التى يهتم بها المتعاملون مع النظام التى لها علاقة بأهداف النظام أو خصائصه أو وظائفه. ويتم أطلاق تنفيذه من خلال وظيفة أخرى. أذن الأسبكت Aspect هو عبارة عن كود متعلق بمتطلب محدد ولكن يتم تنفيذه من خلال كود اخر مرتبط بمتطلب اخر. برمجيا الأسبكت Aspect عبارة عن وحدة برمجية مستقلة تحتوى على الكود المتداخل او الموزع والذى كان يتم تنفيذه فى السابق عن طريق أضافيه لمجموعة من الوحدات او الفصائل الاخرى . الأسبكت Aspect يشبه الى حد كبير الفصيل Class مع بعض الاختلافات. الجدول 3.1 يوضح أوجه الشبه والاختلاف بين الأسبكت Aspect والفصيل Class

الخاصية	Class	Aspect
أنشاء كائن مباشر منه	نعم	لا
يمكن أن يورث فصيل	نعم	نعم
يمكن أن يرث واجهة (Implement interface]	نعم	نعم
يمكن أن يورث [Aspect]	لا	نعم
الأعلان عنه داخله	نعم	نعم

جدول 3-1 مقارنة بين الـ Aspect والفصيل

وتحتوى هذه الوحدة البرمجية على ثلاثة مكونات رئيسية هى :

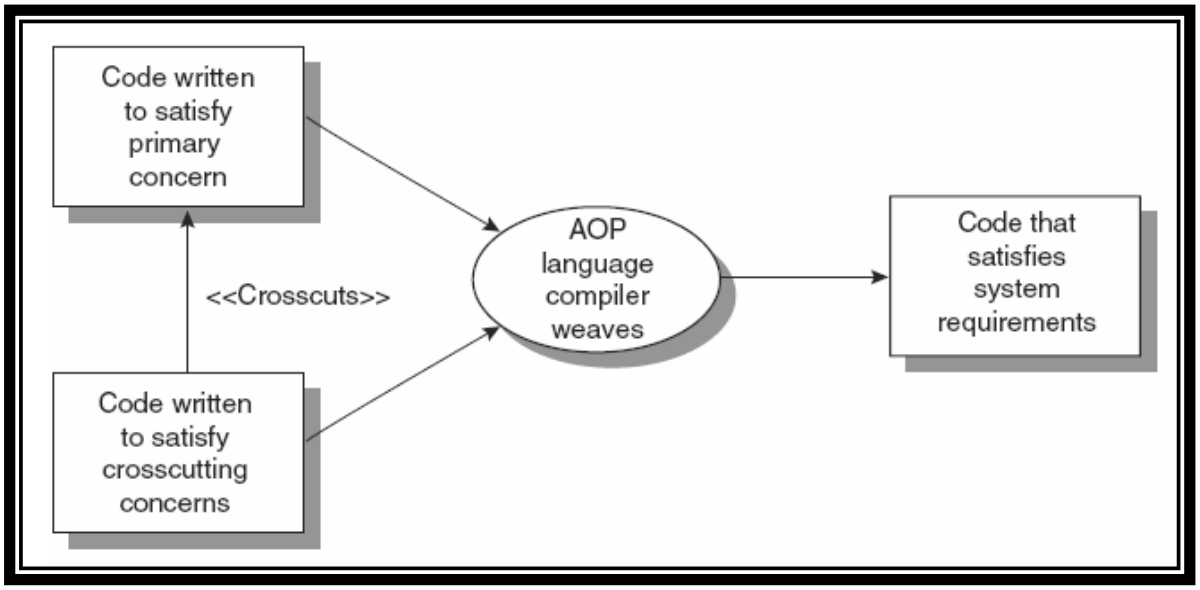
- Join Point
- Point Cut
- Advice

الـ Join Point هى المكان فى الكود الرئيسى تم تعريفه بدقة يحدد أين يجب أن يكون الكود الموزع أو أين يجب تنفيذه ويمكن أن تكون الـ Join Point عبارة نداء دالة أو تعامل مع الاخطاء Exception Handler أو أى نقطة أخرى فى تنفيذ البرنامج. بينما الـ Point Cut هى متى نريد لهذا الكود ان ينفذ، مثلا عند نداء الدالة أو عند أنشاء الفصيل أو عند أرسال المعاملات.. الخ. أما الـ Advice فهى الكود الذى يجب أن يتم تنفيذه ولها دائما ثلاث حالات هى قبل أو أثناء أو بعد Around، After،Before

Aspect Weaver 3.5

هو بمثابة المترجم للغات التى تدعم البرمجة المفاهيمية بأستخدام الأسبكت Aspect أذ يقوم الحائك او الناسج Weaver بترجمة الأسبكت Aspect وتوزيع الكود الخاص به على الوحدات الرئيسية للنظام فى الاماكن التى يجب أن ينفذ فيها الكود مستفيدا من المعلومات المتوفرة عن نقاط القطع والألتقاء الموجودة فى الأسبكت Aspect. أذ عادة يتم أستخدام اى لغة برمجة لكاتبة الوظائف الرئيسية للنظام والتى لا يوجد بها كود متداخل وبعد ذلك يتم كتابة الأسبكت Aspect ومن ثم يستخدم الحائك Weaver من أجل توزيع كود الأسبكت Aspect على الوحدات

الرئيسية للنظام. فى معظم الحالات يتم أستخدام نفس اللغة التى من خلالها تمت كتابت الوظائف الرئيسية فى كتابة الأسبكت Aspect الشكل 3.1 يبين العمليات الاساسية فى ترجمة البرامج التى تستخدم الأسبكت Aspect دائما ما يكون عمل الحائك Weaver عمل سابق لعملية الترجمة Pre-Compiling Step وعندما يقوم الحائك Weaver بتوزيع الكود المتداخل على الفصائل المختلفة، ولتوزيع الكود فأنه أما أن يقوم بتوليد فصيل جديد يحتوى على ذلك الكود المتداخل فى الاماكن المحددة وبعد ذلك يقوم مترجم اللغة المعنية بترجمة هذا الفصيل الى بايت كود Byte Code أو ان يقوم الحائك Weaver مباشرة بأنتاج البايت كود.



شكل 3-4 خطوات ترجمة البرامج التى تحتوى على (Aspect)

3.6 لغات البرمجة المفاهيمية *Aspect Oriented Languages*

اليوم نجد أن هنالك العديد من لغات البرمجة التى تدعم هذه التقنية عن طريق أضافة الأسبكت Aspect الى اللغة الموجودة أصلا ، ومن هذه اللغات لغة الجافا Java والسى بلس بلس C++ فى هذا القسم سنتطرق بأختصار الى عدد من لغات البرمجة التى تدعم هذه التقنية:

AspectR 3.6.1

هذه اللغة قامت بأضافة البرمجة المفاهيمية الى لغى ربى Ruby Language عن طريق توزيع الكود المتداخل على البرنامج الذى تمت كتابته بأستخدام لغة ربى. لكى تعمل الـ AspectR يجب أن تكون لدينا على الاقل النسخة الثانية من لغة ربى. الاسطر التالية تبين مثال لكود تمت كتابته بأستخدام الـAspectR :

```
require 'aspectr'

include AspectR

class Verify < Aspect

def log_enter(method، object، exitstatus، *args)

$stderr.puts "#}self.class}##}method}: args = #}args.inspect}"

end
```

```
def log_exit(method، object، exitstatus، *args)

$stderr.print "#}self.class}##}method}: exited "

end

end

class HelloWorldClass

def sayHelloWorld

puts "Hello World"

end

end

Verify.new.wrap(HelloWorldClass، :verify_enter، :verify_exit،

/say/)

HelloWorldClass.new.sayHelloWorld

End
```

AspectS 3.6.2

تم بناء هذه اللغة على لغة Squeak/Smalltalk . الـ Squeak عبارة عن نسخة مفتوحة من لغة الـSmalltalk ومتوفرة على الموقع www.squeak.org .الـ Squeak تعتمد على ماكينة أفتراضية Virtual Machine وبالتالى هى متوفرة لعدة بنيات وانظمة تشغيل مختلفة. ولكى نستخدم هذه اللغة يجب أن تتوفر لدينا الـSquea .

Apostle 3.6.3

عبارة عن بحث ماجستير من اجل أضافة البرمجة المفاهيمية الى لغة الـSmalltalk . ومتطلبات هذه اللغة صممت لتعمل مع IBM's VisualAge for Smalltalk 4.5 والتى تتوفر منها نسخ للعمل مع جميع أنظمة التشغيل على الموقغ www-3.ibm.com/software/ad/smalltalk/?c=0035016165&n=befree_affiliate&t=aff.

AspectC 3.6.4

عبارة عن محاولة لأضافة البرمجة المفاهيمية الى لغة السى. ويشترك فى هذا المشروع عدد من المؤسسين للبرمجة المفاهيمية ومطورى لغة AspectJ. تعمل هذه اللغة بنفس الطريقة التى تعمل بها الـ AspectJ وهى أنه يتم فصل الكود الرئيسيى عن الكود المتداخل ويتم دمجهما فى المرحلة الاخيرة للكود. الاسطر التالية توضح كود تمت كتابته بهذه اللغة:

```
aspect MainTest {
```

```
pointcut hello() : calls(void sayHello());

before() : hello() {

printf(–Before sayHello() call");

}

}

#include <stdio.h>

void sayHello() {

printf("Hello World!");

}

void main() {

sayHello();

}
```

AspectC++ 3.6.5

قامت هذه اللغة على لغة السى بلس بلس. وهى تعمل بنفس الطريقة التى تعمل بها الـ AspectJ أذ أنها تقوم بعملية الحياكة Weaving قبل الترجمة عن طريق توزيع الكود المتداخل على الكود الرئيسي وبعد ذلك يتم ترجمته بأستخدام مترجم لغة السى سى بلس. هذه اللغة تدعم نظم تشغيل الويندوز واللينيكس والسولارس. الاسطر التالية توضح مثال بسيط لكود تمت كتابته بأستخدام هذه اللغة:

```
#include <iostream.h>

pointcut hello() =

call ("void HelloWorld::sayHello()");

aspect HelloWorldAspect {

public:

advice hello() : void after () {

cout << ("Saying Hello");

}

};

class HelloWorld {
```

```
public:

void sayHello() {

cout << "Hello";

}

};

int main () {

HelloWorld hello = new HelloWorld();

hello.sayHello();

}
```

هذا الكود ستتم ترجمته أولا بأستخدام الـWeaver وبعد ذلك سيتم تنفيذه من خلال مترجم السى بلس بلس العادى.

Pythius 3.6.6

هذه اللغة تمكن من استخدام البرمجة المفاهيمية مع لغة Python. ولتشغيلها تتطلب وجود Python 2.2.1 أو نسخة أحدث. الاسطر التالية توضح كود مكتوب باستخدام هذه اللغة:

```
import aop

class Check(aop.Aspect):

def __init__(self):

aop.Aspect.__init__(self) # don't forget this!

self.after('getattr', 'area', self.log)

def log(self, cxt):

value = cxt['value']

name = cxt['name']

print 'Attribute %s (value of %s)' % (name, value)

class DoubleIt:

def __init__(self, x):

self.x = x*x

self.quad = x*x*x*x

check = Check()
```

```
DoubleIt = check.affect(DoubleIt)

doubleit = DoubleIt(4)

print 'Double Value = %d' % doubleit.x

print 'Quad Value = %d' % doubleit.quad
```

AspectJ 3.6.7

خلال هذه الدراسة تم أستخدام هذه اللغة التى أضافت مفهوم البرمجة المفاهيمية الى لغة الجافا وذلك للميزات العديدة للغة الجافا التى من اهمها أنها لا تعتمد على بنية حاسوبية أو نظام تشغيل محدد. هذه اللغة تقوم بترجمة الـ Aspect وتوزيعه على الفصائل الرئيسة للنظام وذلك فى خطوة تسبق عملية الترجمة وبعد ذلك تتم ترجمة الكود بأستخدام ماكينة الجافا الافتراضية (JVM – Java Virtual Machine) . تتطلب هذه اللغة وجود لغة الجافا الاصدار 1.2 أو أحدث ومثال لهذه اللغة الكود ادناه:

```
public aspect Showcase

{

pointcut int_A_a_int(): call(int A.a(int));

pointcut int_A_all_int(): call(int A.*(int));
```

```
pointcut all_all_c_all(): call(* *.c(*));

before(): int_A_a_int()

{

System.out.println(“Before: “ + thisJoinPoint);

}

after(): int_A_all_int() || all_all_c_all()

{

System.out.println(“After: “ + thisJoinPoint);

}

Object around(): all_all_c_all()

{

System.out.println(“Start around: “ + thisJoinPoint);

Object o = proceed();

System.out.println(“End around: “ + thisJoinPoint);
```

```
return o;

}

}
```

لتنفيذ هذا الكود يجب أن نستخدم الـ .AspectJ weaver عن طريق الامر ajc والصيغة العامة لتنفيذ امر الترجمة بأستخدام الـ AspectJ compiler هى:

```
>ajc class/aspect-name.java
```

والذى يقوم بأنتاج ملف بايت بأمتداد class. يحتوى على الكود الرئيسى زائدا الـ[Aspect] ومن ثم يتم تنفيذه بأستخدام ماكينة الجافا الافتراضية عن طريق الامر java والصيغة العامة له كالاتى

```
>java class-name
```

3.6.7.1 التعامل مع الاسبكت Aspect فى لغة AspectJ

الاسبكت Aspect هو الوحدة الاساسية فى التى من خلالها يتم كتابة الكود المتداخل. وكما ذكرنا سابقا فأن الاسبكت Aspect يشبه الى حد كبير الفصيل. يتم حفظ الملف بامتداد java. مثل الفصيل. ولكن الاختلاف الرئيسي هو أن الاسبكت Aspect يستطيع أن يتفذ الكود الخاص به داخل الكائنات مما يتيح له أمكانية تغيير الكائنات الاخرى.

أن مترجم الـ AspectJ يقوم بترجمة الاسبكت Aspect الى فصيل جافا Java class لذلك عند كتابة برامج بأستخدام الـ .AspectJ يجب أن لا يحتوى البرنامج على اسبكت Aspect يحمل نفس أسم أحد الفصائل فى نفس الحزمة Package.

الصيغة العامة General Syntax لتعريف الاسبكت Aspect هى كالاتى :

Access-Modifier aspect aspect-name {. . . }

بحيث أن:

Access-Modifier: عبارة عن محدد الوصول للكائن والذى يحدد مستوى الوصول المسموح به للكائن مثل (public، private ،etc.… static).

aspect: عبارة عن كلمة محجوزة تستخدم لتعريف الاسبكت Aspect فى لغة الجافا.

Aspect-name: أسم الاسبكت Aspect وهو عبارة عن أسم أختيارى ولكن يجب الالتزام بشروط التسمية المعروفة فى لغة الجافا ولغات البرمجة عموما مثل ان لا يحتوى الاسم على مساحات فارغة أو ان يكون كلمة محجوزة فى اللغة.

AspectJ Join Point 3.6.7.2

واحدة من أهم المكونات فى البرمجة المفاهيمية هى نقاط الارتباط join point. وهى كما ذكرنا سابقا عبارة نقطة تم تعريفها جيدا فى تنفيذ البرنامج من خلالها يعرف مترجم لغة

البرمجة المفاهيمية المكان الذى يجب ان يوزع فيه الكود. النسخة الحالية من الـ AspectJ عرفت عدد من نقاط الارتباط منها:

نداء الدالة(Method call): تعرف عندما تتم عملية نداء لاى دالة عن طريق كائن أو نداء مباشر للدالة أذا كانت الدالة ساكنة static الصيغة العامة لها كالاتى:

Call(method-name(parameters))

نداء دالة البناء (constructor call): تعرف عندما يتم نداء دالة البناء عند انشاء اى كائن من الفصيل والصيغة العامة لها كالاتى:

initialization(method-name(parameters))

قراءة حقل (Field get): تعرف عند قراءة خاصية كائن محدد والصيغة العامة لها كالاتى :

get(field-name)

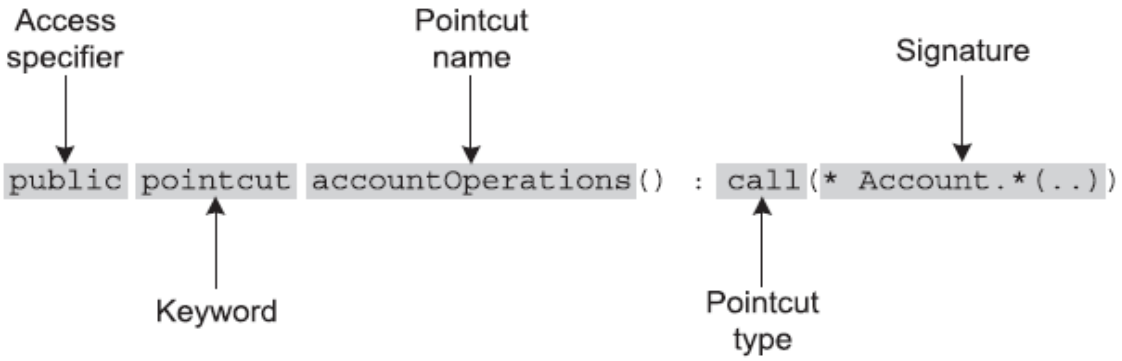
كتابة حقل (set field): تعرف عندما تتم كتابة حقل محدد يتبع لكائن محدد والصيغة العامة لها كالاتى:

set(field name)

فى نقاط الربط عند الحاجة الى كتابة اكثر من دالة او اكثر من حقل فاننا يمكن أن نستخدم النجمة * والتى تعنى اى عدد من الحروف عند أستخدامها مع أسم الدالة، وعند أستخدامها مكان معاملات الدالة فأنها تعنى أى عدد من المعاملات للدالة.

AspectJ Pointcuts 3.6.7.3

كما نعلم فان نقاط القطع عبارة عن نقاط تم تعريفها بدقة فى تنفيذ البرنامج والتى يمكن أن تكون عبارة عن نداء دالة أو قراءة أو كتابة حقل محدد. نقاط الوصول وحدها لا تعمل ولكن عند ربطها مع نقاط القطع Point Cut يمكن أن توضح لنا كيف سيتم توزيع الكود داخل البرنامج فيما بعد. نقاط القطع عبارة عن بنية يتم أستخدامها لتعريف نقاط القطع والتى يمكن أستخدامها لتعريف أكثر من نقطة وصول والصيغة العامة لها كالاتى:



الجدول 3.2 يوضح أنواع نقاط القطع Pointcut فى لغة الـ AspectJ والصيغ العامة لها

Join Point Category	Pointcut Syntax
Method execution	`execution(MethodSignature)`
Method call	`call(MethodSignature)`
Constructor execution	`execution(ConstructorSignature)`
Constructor call	`call(ConstructorSignature)`
Class initialization	`staticinitialization(TypeSignature)`
Field read access	`get(FieldSignature)`
Field write access	`set(FieldSignature)`
Exception handler execution	`handler(TypeSignature)`
Object initialization	`initialization(ConstructorSignature)`
Object pre-initialization	`preinitialization(ConstructorSignature)`
Advice execution	`adviceexecution()`

جدول 3-2 أنواع نقاط القطع والصيغ العامة لها

AspectJ Advise 3.6.7.4

النصيحة Advice تحدد الكود الذى يجب تنفيذه عند نقطة ربط محددة تم اخيارها عن طريق نقطة القطع أو بكلمات أخرى هى تحتوى على الكود الموزع الذي يجب ان ينفذ فى عدة كائنات. توجد ثلاثة أنواع رئيسية من النصائح وهى :

- before: الصيغة العامة لها كالتى

before(Formal-parameter) : Pointcut {Body}

فى هذه الحالة يتم تنفيذ الكود الموجود فى الـ Body قبل أن تتم مطابقة نقطة الوصول.

- after: الهدف الاساسي منها هو تنفيذ الـ Advice بعد مطابقة نقطة الوصول والصيغة العامة لها كالاتى:

after(Formal-parameters): pointcut {Body}

- around: هى واحدة من الخصائص المهمة فى البرمجة المفاهيمية. وهى تمكننا من تبديل الكود الموجود فى نقطة الربط بالكود الموجود فى النصيحة Advice والصيغة العامة لها كالاتى:

Return-type around(Formal-parameters) throws List-of-

exception : pointcut {Body}

العنوان:	تأثير البرمجة ذات التوجه المفاهيمى على جودة البرمجيات
المؤلف الرئيسي:	عثمان، أحمد سيد أحمد علي
مؤلفين آخرين:	حاج على، عوض(مشرف)
التاريخ الميلادي:	2015
موقع:	الخرطوم
الصفحات:	1 - 240
رقم MD:	830496
نوع المحتوى:	رسائل جامعية
اللغة:	Arabic
الدرجة العلمية:	رسالة دكتوراه
الجامعة:	جامعة النيلين
الكلية:	كلية الدراسات العليا
الدولة:	السودان
قواعد المعلومات:	Dissertations
مواضيع:	هندسة البرمجيات، برمجة المفاهيمية، جودة البرمجيات، التحليل البرمجى
رابط:	https://search.mandumah.com/Record/830496

الفصل الرابع (جودة البرمجيات -

(SOFTWARE QUALITY

4.1 مقدمة (هندسة البرمجيات)

لأول مرة تم استخدام مصطلح هندسة البرمجيات عام 1968 فى مؤتمر للناتو NATO لمناقشة مشاكل تطوير البرمجيات او ما عرف بازمة البرمجيات Software Crisis [2]، فالبرمجيات الكبيرة يتم تسليمها متاخرا، ولا تقوم باداء المهام والوظائف المطلوبة بالاضافة الى زيادة التكلفة عن ما هو موضوع من البداية.أضف الى ذلك مشاكل صيانة وتطوير البرامج فى المستقبل او ما يعرف بصيانة الـ (Legacy Systems) وهى عبارة عن البرمجيات التى تم تطويرها أعتمادا على متطلبات محددة ومع مرور الزمن وتغير المتطلبات وتطورها تم أجراء صيانة متكررة لها الى ان وصلت الى مرحلة يستحيل أجراء صيانه لها نسبة للتعقيد الذى أصبحت عليه أو أن تكلفة الصيانة أصبحت عالية لدرجة أنها قد تفوق تكلفة التطوير الأساسية، وبالتالى اصبح من الواضح جدا ان الاعتماد على المهارات الفردية فى البرمجة ليس كافيا وغير فعال فى البرامج الكبيرة. ومن هنا ظهرت الحوجة الى علم هندسي يستخدم نظريات وأسس وقواعد بالاضافة الى ادوات وتقنيات معرفة مسبقا لأنتاج برمجيات ذات جودة عالية.

خلال السبعينات والثمانيات من القرن الماضى ظهرت العديد من تقنيات وادوات ونظريات هندسة البرمجيات التى هدفت الى حل مشاكل البرمجيات وزيادة جودتها مثل البرمجة الهيكلية والموجهة ولغة النمذجة الموحدة.

4.2 جودة البرمجيات:

الجودة عبارة عن تعبير نوعى Qualitative Term متعدد الابعاد مبنى على ملائمة الغرض الذى يخدمه . وهذا الغرض يمكن أن يكون منتج Product أو خدمة Service أو مفهوم أو أى شئ اخر. والجودة من الكلمات المتداولة كثيرا والتى تختلف فى تقيمها من شخص لاخر. ولان النجاح والقدرة على المنافسة فى السوق تعتمد بصورة أساسية على التكلفة والجودة لا بد أذن من وضع تعريف علمى منهجى قابل للقياس للجودة. ، والجودة من المفاهيم الاساسية التى تدخل فى جميع مناحى الحياة.

قبل ظهور البرمجيات وتقنية الأتصالات الى الوجود كانت الجودة من المفاهيم المرتبطة بالصناعات الفيزيائية مثل صناعة التلفزيونات والمسجلات والسيارات الى اخره. ولكن نتيجة لأزدهار صناعة البرمجيات وانتشارها كان لا بد وأن تتركز جهود الباحثين لتعريف جودة المنتج البرمجى وأدارته وتطوير مقاييس فاعلة ومؤثرة. ولأن الجودة بطبيعتها من المفاهيم الديناميكية التى تجارى تطور حاجات المستخدم والمنتج والادارة فما زالت البحوث مستمرة فى هذا الجانب فى النظريات والمفاهيم والادوات والمقاييس. كذلك أهتمت مؤسسات المواصفات العالمية مثل منظمة [ISO9001] و [IEEE] والتين قامتا بوضع تعاريف ونماذج لجودة البرمجيات.

4.3 تعريف الجودة:

الجودة بأعتباره مفهوم أساسى يستخدمه الانسان عند أقتناءه حاجاته اليومية أو حصوله على الخدمات التى يحتاجها يمكن تعريفه ببساطة بأنها ملائمة الغرض Fitness to purpose

. منطقيا نجد أن هذا التعريف مقبول والغرض Purpose فى العديد من المنتجات البسيطة قد نجده واضحا من أسم المنتج فعلى سبيل المثال فالسكين الغرض منها القطع ، أما المنتجات الاكثر تعقيدا يمكن أن يكون الغرض موضحا فى دليل التشغيل . أما بالنسبة للبرمجيات فأن الغرض يكون موضح فى وثيقة المتطلبات Software Requirement Specification. ولكن نسبة للخصوصية التى تتفرد بها البرمجيات عن مجالات الصناعة الاخرى فأن التعريف البسيط للجودة لا ينسجم مع المعنى العصرى للجودة والذى يتجاوز ملائمة الغرض وبالاخص مع الخصوصية التى تتمتع بها البرمجيات.

أن جودة المنتج البرمجى لها أهمية خاصة فى هندسة البرمجيات فمن غير الواقعى أن تكون هنالك صناعة دون وجود مفهوم مهنى واضح للجودة. والمفهوم المهنى الواضح نقصد به أن يكون قابلا للقياس.

لذلك نجد ان التعريفات لجودة البرمجيات كانت متباينة تباينا كبيرا فعلى سبيل المثال يعرفها كروسبى [Crosby] بانها موافقة المتطلبات conformance to the requirements [10] ولكن هذا التعريف لم يضع فى الحسبان الفروقات التى يمكن ان تكون فى الجودة بين المنتجات التى لبت نفس المتطلبات. لذلك عرفها جوران Juran بانها ملائمة الاستخدام fitness for use ومع ان هذا التعريف افضل من سابقه الا انه لم يوضح الية يمكن بها الحكم على منتجين توفرت فيهما ملائمة الاستخدام [10].

لذلك نجد أن السؤال "ما هي جودة الرمجيات؟ ، لا بد وأن يولد إجابات مختلفة ، اعتمادا على من تسأل ، وتحت أي ظرف من الظروف ، وعن أي نوع من الأنظمة والبرمجيات ، وهلم جرا. لذلك بدلا من ذلك فأن السؤال الاكثر تحديدا هو: ما هى خصائص المنتج البرمجى عالى الجودة؟ فبالتركيز على خصائص المنتج البرمجى يمكن أن تعطينا فكرة أوضح عن جودة البرمجيات.

هنالك عدة مستويات لجودة البرمجيات بناءا على الأدوار والمسئوليات يمكن تقسيمها ببساطة الى خمس مستويات رئيسية هي: التجريدى ، المستخدم ، المنتج، الصناعة ، القيمة ، كما هو مبين أدناه :

• المستوى التجريدى من الصعب تحديد أو وصف الجودة بعبارات مجردة ، ولكن يمكن التعرف عليها إذا كانت موجودة. وترتبط مع بعض العناصر الملموسة التى ترضى المستخدم.

• على مستوى المستخدم ، هى مدى ملائمة المنتج لمتطلباته أو مقابلة أحتياجاته.

• مستوى التصنيع هى مطابقة معايير عمليات الانتاج.

• مستوى المنتج هى التركيز على خصائص المنتج الداخلية التى تنعكس على الاداء الكلى للمنتج.

• ومن ناحية القيمة هى مدى قابلية الزبائن للدفع مقابل البرنامج.

أما تعريف الجودة فنجد أن كل شخص يمكن أن ينظر الى الجودة من زاوية مختلفة ولكن بصورة عامة يمكن أن نقسم الاشخاص الهتمين بجودة البرمجيات الى :

- المستهلكون: ويقصد بهم الاشخاص المتعاملين مع البرنامج أذا كانو عبارة عن مشغلين أو أدارة عليا.
- المنتجون: هم المطورون للبرنامج أو كل شخص شارك فى عملية التطوير أو الصيانة أو الاختبار أو الاعدادات.

4.3.1 توقعات المستهلكين للجودة:

الجودة من وجهة نظر المستخدمين هى أن يؤدى البرنامج الوظائف المطلوبة منه كما هو محدد. وللقيام بذلك هناك نقطتين رئيسيتين هما: أن يقوم البرنامج بأداء العمل الصحيح كما هو مطلوب وتسمى الصلاحية للأستخدام ، أما النقطة الثانية فهى أداء العمل الصحيح المطلوب بطريقة صحيحة ومدى ثباتها مع مرور الزمن وهى الأعتمادية. وهاتين النقطتين لهما علاقة بالمصادقة والتحقق Validation & Verfication فى ضبط جودة البرمجيات Quality Assurance – QA

4.3.2 توقعات المنتجين للجودة:

بالنسبة لمنتجي البرمجيات ، فان التحدى الاساسى للجودة هو الوفاء بالتزاماتهم التعاقدية من خلال انتاج منتجات تتوافق مع المواصفات أو تقديم الخدمات التي تتوافق مع الاتفاق. وبالتالي ، فأن أهتمامهم بالخصائص الداخلية للبرمجيات التى تجعل من السهل أن يتوافق المنتج

مع المواصفات المطلوبة. مثل النماذج الجيدة التى تحافظ على خصائص المنتجات المختلفة الداخلية والتي تجعل من السهل تطويره ليتوافق مع مواصفات المنتجات ، مثل التصميم الجيد للبرنامج الذى يسهل من عملية تجميع البرنامج وتكامليته.

4.4 خصائص جودة البرمجيات العامة

4.4.1 الوظيفية " Functionality":

هي مجموعة من الصفات التي تحمل مجموعة من الوظائف والخصائص المحددة ، والتى تعني ان البرنامج يجب ان يوفر الوظائف والخدمات "وفقا للمتطلبات" عند استخدامها تحت شرط محدد . الخصائص الفرعية للوظيفية هي:

4.4.1.2 الملائمة " Suitability"

الملائمة تعبر عن مدي ملائمة البرنامج لمتطلبات المطور ، المتطلبات وتكون معروفة فقط لمطور النظام ولايمكن للمطور قياس الملائمة اثناء تطوير النظام وأنما يمكن قياسها بعد الأنتهاء من تطوير النظام ككل

4.4.1.2 الدقة " Accuracy"

هي تقييم دقة البرنامج مع مستوي الدقة المطلوب من قبل مطور النظام.

4.4.1.3 إمكانية التشغيل المتداخل " Interoperability"

تشير الي توافق تنسيق البيانات ومعالجتها من قبل المبرمج مع المعايير العالمية .

4.4.1.4 الامن "Security"

يشير الي ان البرنامج قادر علي التحكم في الوصول غير المصرح به لخدماته.

4.4.1.5 الامتثال "Compliance"

تشير هذه الخاصية الي اذا ما كان البرنامج مطابق لأي معيار او شهادة دولية.

4.4.2 الفعالية "Efficiency":

تعبر هذه الخاصية عن قدرة البرنامج علي تقديم الاداء المناسب حسب كمية الموارد المستخدمة. وتتأثر الفعالية بالتكنولوجيا بالرغم من استخدام الموارد الخام وقت التشغيل الا انها تتأثر بآلية التفاعل ، حيث يمكن تحسين اداء وحدة نمطية دون التأثير علي مواصفاتها ، وينبغي اختبار الوحدات النمطية علي منصات مختلفة للتحقق من الاداء.

للفعالية خاصيتين فرعيتين يتم تعريفهما علي النحو التالي :

4.4.2.1 Time behaviour:

هذه الخاصية تشير الي القدرة علي تنفيذ مهمة محددة بصورة صحيحة وفى زمن معقول في ظل ظروف محددة.

4.4.2.2 Resource behaviour:

تشير الي كمية الموارد المستخدمة من قبل النظام في ظل ظروف محددة .

4.4.3 الموثوقية " Reliability "

الموثوقية هي احتمال ان النظام او البرنامج سينتج فشل خلال فترة زمنية معينة ، وبعبارة اخري الموثوقية تعبر عن قدرة البرنامج للحفاظ علي مستوي معين من التسامح مع الخطاء وفقا لشروط محددة . لذلك نجد أن اعادة استخدام جانب محدد من البرنامج في تطبيقات متعددة يزيد من موثوقية هذا البرنامج كما يكمن ملاحظة ان الجزء الذى يتم أعادة أستخدامه يكون قد تم اختباره بدقة قبل اعادة استخدامه في تطبيقات اخري .

للموثوقية خصائص تم تقسيمها الي:

4.4.3.1 النضج " Maturity "

النضج في سياق البرنامج يتعامل مع عدد الاصدارات التجارية للبرمجيات والفاصل الزمني بين كل اصدار.

4.4.3.2 قابلية الاسترداد " Recoverability "

مستوي قدرة البرنامج أو المنتج البرمجي علي استعادة مستوي الأداء المطلوب والبيانات في حالة حدوث أي فشل للنظام، تعتبر هذه الميزة من أهم الميزات والمكونات في أي منتج برمجي تعرف بتقنية النسخ الاحتياطي والتي تعمل علي استعادة البيانات ووضعية النظام للعمل بعد حدوث أي خطآ..

4.4.3.3 التسامح مع الخطاء " Fault Tolerance "

تشير هذه الخاصية الي امكانية البرنامج الحفاظ علي مستوي معين من الاداء في حالة الاعطال.

4.4.4 سهولة الاستخدام " Usability "

هي قدرة البرنامج علي ان يفهم ، يستفاد منه، يستخدم ، يتم اعداده وتكوينه ، وينفذ وفقا لشروط محدده . ونعني بالمستخدم هنا مطور النظام او البرنامج بدلا عن المستخدم النهائي ، وبالتالي فان امكانية استخدام البرنامج يجب ان تكون اقل تعقيدا وأكثر قابلية لاعادة الاستخدام وصديقة للمطور بحيث يمكن تجميعها بشكل صحيح في النظام . وتعرف الخصائص الفرعية لقابلية الاستخدام كما يلي:

4.4.4.1 سهولة الفهم " Understandability "

وهي قدرة البرنامج علي تمكين المستخدم"مطور النظام" لفهم اذا ما كان البرنامج مناسب وكيف يمكن استخدامه لتنفيذ مهام وشروط تتعلق بالاستخدام.

4.4.4.2 امكانية التعلم " Learn ability "

تشير هذه الخاصية الي قدرة البرنامج علي تمكين مطور النظام من تعلم البرنامج ، علي سبيل المثال يجب ان تكون وثائق المستخدم و مساعدات النظام كاملة وشرح كيفية تحقيق المهام المشتركة.

4.4.4.3 الجاذبية "Attractiveness "

يشير الي قدرة البرنامج ان يكون جاذب للمستخدم ، كما ذكرنا سابقا ان المستخدم هو المطور فان هذه الخاصية تكون مهمة عند تصميم واجهات البرامج API لذلك يمكن تجاهل هذه الخاصية عند تصميم النموذج.

4.4.5 امكانية التعديل "Modifiability ":

حيث تشمل جانبين هما:

4.4.5.1 الصيانة "Maintainability ":

هي سهولة تعديل او تصحيح أخطاء في نظام برمجي وتحسيين الاداء والتكييف مع تغيير البيئة. كما يقصد بها أيضا أمكانية أضافة خصائص أو متطلبات جديدة للنظام ليتوافق مع أحتياجات المستخدم كما تشمل أيضا تعديل النظام ليتوافق مع بيئة تشغيل جديدة .

قابلية الصيانة تحتوى على الخصائص التالية :

- التخصيص "Customizability":

تشير الي تعديل البرامج من خلال معلومات محدودة متاحة مثل واجهات او مؤشرات.

- قابلية الاختبار "Testability":

نعني بها قابلية فحص وظائف النظام النهائي .

- الثبات "Stability":

هي قدرة النظام علي التعامل مع التغيرات الغير متوقعة خلال فترة الصيانة ، او هي الدرجة التي ينعدم فيها تأثير مكون علي مكون اخر.

- ***قابلية التحليل " Analyzability":***

تشير الي التحليل الالي للبرمجيات والتي نادرا ما تكون موجودة في البرمجيات.

4.4.5.2 المرونة " Flexibility":

هي سهولة تعديل نظام او مكون من اجل الاستخدام في تطبيقات او بيئات اخري.

تشمل هذه الخاصية:

- التمدد " Extensibility " مقياس يمثل قدرة البرنامج على تطويره وتركيب إضافات عليه دون الحاجة إلى ترقيته إلى إصدارات جديدة ،لا تقتصر قابلية الامتداد على المستخدمين بل حتى على المبرمجين أنفسهم، فقد تبين قابلية الامتداد مدى قدرة الشفرة المصدرية على تقبل التطوير وإضافة أجزاء إضافية بها دون التأثير على الشيفرات الأساسية.
- التبسيط" Simplification " نعني بها تبسيط وتقليل الوظائف.
- اعادة الهيكلة " Restructuring "نعني بها ترشيد الخدمات بإنشاء مكونات قابلة للاستخدام.
- وقت النشر " Time to deploy " هو الوقت المستغرق لتحديد متطلبات لتوفير قدرات جديدة.

- قابلية التوسع الوظيفية " Functional scalability " هي القدرة علي توسيع نطاق اعلي واسفل النظام من حيث عدد المستخدمين.
- المرونة الوظيفية " Functional flexibility " نعني بها تحويل قدرات سابقة الي استخدامات جديدة.

4.4.6 امكانية النقل " Portability"

تعرف هذه الخاصية علي قدرة نقل البرامج من بيئة الي اخري مع تعديل خفيف اذا لزم الامر مع الاخذ في الاعتبار الحد الادني للتكاليف والجدول الزمني .

الخصائص الفرعية لامكانية النقل هي:

4.4.6.1 امكانية الاستبدال " Replace ability"

تشير الي اذا ما كان البرنامج متوافق مع الاصدارات السابقة ، وهذا يعني ان البرنامج الجديد يمكن ان يكون بديل لسابقه بدون مجهودات كبيرة.

4.4.6.2 القدرة علي التكيف " Adaptability"

يشير الي قابلية البرنامج علي التكييف مع منصات مختلفة .

4.4.6.3 قابلية التنصيب " Install ability"

يشير الي سهولة تنصيب البرنامج علي منصات عمل مختلفة.

4.4.7 الاداء " Performance"

هي قدرة النظام علي انجاز مهامه المعينة من قيود معينة مثل قيود السرعة والدقة او استخدام الذاكرة. الخصائص الفرعية للاداء هي:

Latency 4.4.7.1

هو الوقت المستغرق للرد علي حدث معين.

Throughput 4.4.7.2

هو عدد الاحداث التي تمت الأستجابة لها في وقت محدد.

Capacity 4.4.7.3

هو الطلب الذي يمكن ادراجه في النظام لمقابلة متطلبات Latency و Throughput.

Modes 4.4.7.4:

هو التغيرات في الطلبات والموارد بمرور الزمن .

4.4.8 الاعتمادية " Dependability"

هي خاصية للنظام حيث يمكن الاعتماد عليه في ايصال الخدمات. خصائص الاعتمادية هي:

4.4.8.1 التوفرية " Availability"

نعني بها الاستعداد للاستخدام.

4.4.8.2 الموثوقية" Reliability"

قدرة النظام علي العمل بصورة صحيحة والمحافظة علي مستوي الأداء، توفر أي مكون من مكونات النظام للمستخدم عند الحاجة كما يجب أن يضمن إمكانية تجنب فشل النظام (توقف الخدمة) وإمكانية استعادة مستوي الأداء والبيانات في حالة حدوث أي فشل .

4.4.8.3 السلامة " Safety"

عدم وقوع احداث مع عواقب كارثية للبيئة.

4.4.8.4السرية " Confidentiality"

عدم حدوث كشف غير مصرح به للمعلومات.

4.4.9 اعادة الاستخدام " Reusability"

هي قدرة المكونات والنظم الفرعية لتكون مناسبة للاستخدام في تطبيقات اخري وسيناريوهات اخري ، اعادة الاستخدام يقلل من الازدواجية في المكونات ويقلل وقت التنفيذ .

4.4.10 قابلية التوسع " Scalability"

يمثل قدرة البرنامج على العمل عندما تكثر حجم البيانات والمستخدمين التي يتعامل معها البرنامج مع مرور الوقت. فعند تجربة البرنامج لحظة التطوير Development، يعتمد المبرمجون على بيانات صغيرة الحجم قد تغرر بهم بأن برامجهم ناجحة، وبعد اعتمادها في مرحلة الإنتاج Production وتكثر البيانات، يبدأ عجز البرنامج يشيخ شيئا فشيئا وتبدأ المشاكل بالظهور.

4.4.11 الكمالية " Integrity"

يحدد الاتساق والتماسك لتصميم النظام العام ويتضمن طريقة تصميم الوحدات والمكونات.

4.4.12 سهولة الادارة " Manageability"

نعني بها سهولة ادارة النظام لمديري النظام .

4.4.13 امكانية الدعم " Supportability"

نعني بها قدرة النظام علي تقديم معلومات مفيدة لتحديد وحل المشاكل عندما يحدث فشل

4.4.14 التوافقية Compatibility :

قدرة اثنين أو أكثر من مكونات المنتج البرمجي علي تبادل المعلومات أو أداء المهام المطلوبة تنفيذها فيما بينهم والتي تعمل علي نفس الأجهزة وتتشارك نفس بيئة العمل.

4.4.15 التوفر Availability :

مستوي إمكانية توفر المنتج البرمجي أو أي مكون من مكوناته للمستخدم عند الحاجة، يتم تقييمها بناء علي نسبة إجمالي الوقت اللازم لتوفر النظام أو أي المكونات بعد حدوث أي خطآ وبالتالي تعتبر مزيج من القدرة علي التحكم في وتيرة حدوث الأخطاء ومعالجتها.

4.4.16 الموائمة أو الالتزام Appropriateness:

مستوي التزام المنتج البرمجي وموائمته لتلبية الاحتياجات الوظيفية الأساسية للعميل والتي توثر بصورة مباشرة في تشغيل وتنفيذ مهام الوظيفية للنظام.

4.4.17قابلية تكامل الأجزاء Modularity :

مستوي قدرة المنتج البرمجي المكون من أجزاء منفصلة علي الاستمرار في العمل في حالة تم إجراء أي تعديلات علي أي من المكونات دون التأثير علي بقية الأجزاء.

4.5 نماذج جودة البرمجيات (SOFTWAR QUALITY MODELS)

نماذج الجودة [Quality model] يعرف عادة كمجموعة من الخصائص والعلاقات بينها والتي توفر قاعدة لتحديد متطلبات الجودة و تقدير او تقييم الجودة [39]. نماذج جودة البرمجيات تستخدم لعرض بنية واضحة لجودة البرمجيات قابلة للقياس

4.5.15 نموذج MCCALL:

هذا النموذج اقترح بواسطة Jim McCall فى العام 1977 كأول نموذج جودة برمجيات [39] ، هذا النموذج يفرق بين مستويين من مستويات خصائص جودة البرمجيات، وهما خصائص الجودة الخارجية والتى يمكن قياسها مباشرة ، والمستوى الثانى هو معايير الجودة Quality Criteria والتى يتم قياسها نوعية يحاول McCall تجسير الفجوة بين المستخدمين والمطورين بالتركيز علي عدد من عوامل جودة البرمجيات التي تعكس وجهة نظر المستخدمين و تعطي اولوية لي المطورين.

هذا النموذج لدية ثلاثة وجهات نظر لتعريف وتميز جودة منتجات البرمجيات :

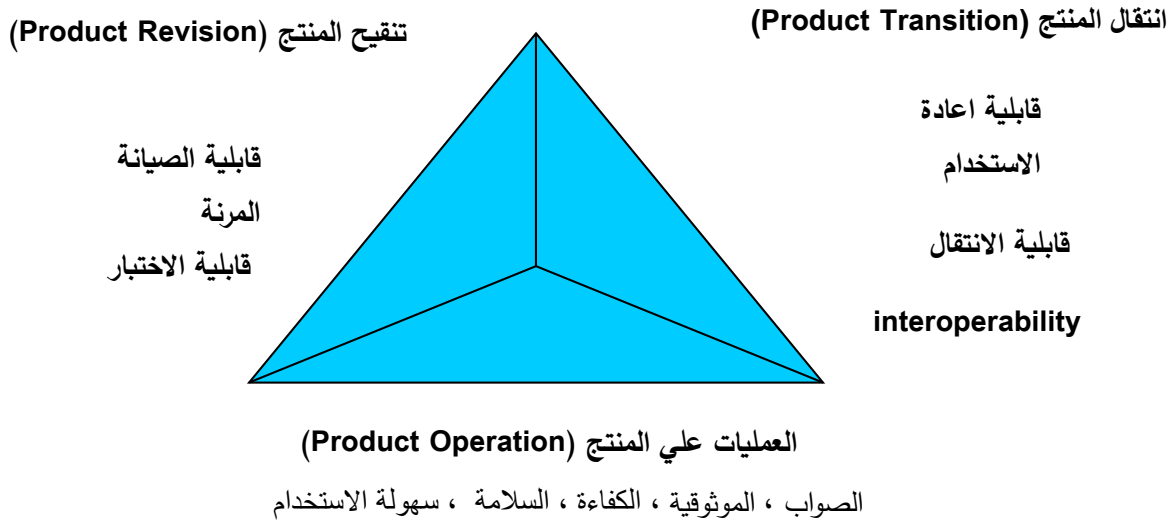
1. تنقيح او تعديل المنتج (Product Revision): خصائص الجودة في هذا المنظور هي قابلية الصيانة (maintainability) و المرونة (flexibility) و قابلية الاختبار (testability).

2. العمليات علي المنتج (Product Operation): خصائص الجودة في هذا المنظور هي الصواب (correctness)، الموثوقية (reliability)، الكفاءة (efficiency)، السلامة (integrity) ، سهولة الاستخدام (usability).

3. انتقال المنتج (Product Transition): خصائص الجودة في هذا المنظور هي قابلية الانتقال(portability) ، قابلية اعادة الاستخدام (Reusability) ، وقابلية الاستخدام المتعدد (Interoperatability)

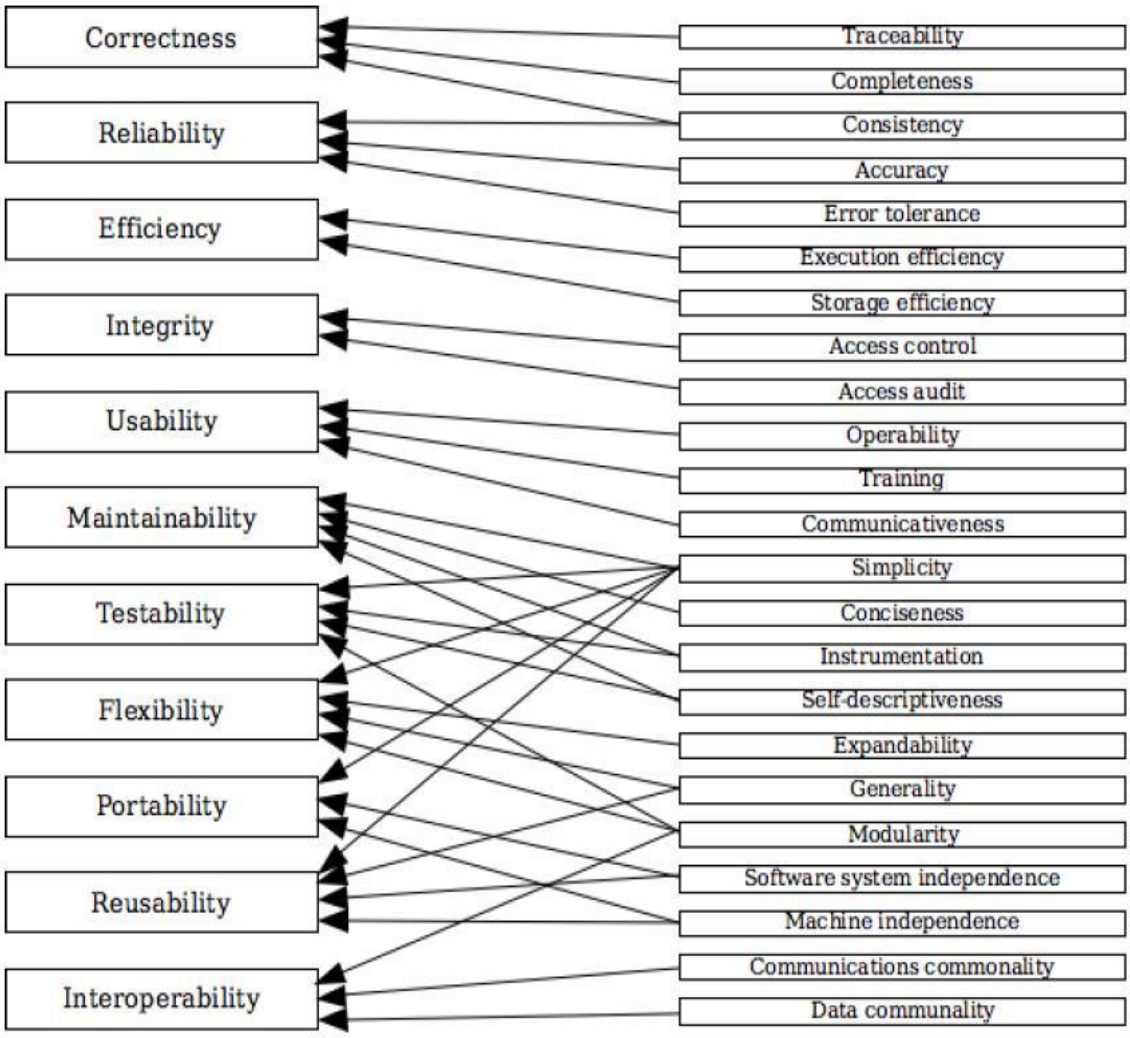
نموذج Mc Call يحتوي علي ثلاثة وجهات نظر وكل وجهة تحتوي علي عوامل او خصائص جودة وكل خاصية ترتبط بمعيار او اكثر ومن خلال هذا المعايير يتم قياس الجودة.

الشكل 4-1 يوضح الأبعاد الثلاثة لنموذج MC Call .



شكل 4-1 نموذج MC Call للجودة

أعتمد نموذج MC Call مجموعة من المعايير والتى قام بربطها مع خصائص جودة المنتج ، الشكل 4-2 يوضح العلاقة بين خصائص الجودة المختلفة ومقاييس الجودة.



شكل 4-2 خصائص الجودة والمعايير المرتبطة بها نموذج MC Call

4.5.2 نموذج Boehm

نموذج الجودة الثانى قدم بواسطة Barry W. Boehm فى العام 1978م [39] ، وكسابقه أيضا نجد ان نموذج بوهم عبارة عن نموذج هرمى تم تقسيم خصائص الجودة فيه الى ثلاثة مستويات المستوى العالى high-level characteristics ، والمستوى المتوسط

primitive والخصائص الأساسية ، intermediate level characteristics characteristics والتى تساهم كلها مع بعضها البعض فى نموذج الجودة الشاملة.

خصائص المستوى العالى تمثل متطلبات الجودة الاساسية للاستخدام والتى على أساسها يتم تقييم جودة المنتج البرمجى، خصائص المستوى العالى أجابت على الأسئلة الأساسية الثلاثة التى يحتاجها مشترى البرنامج وهى:

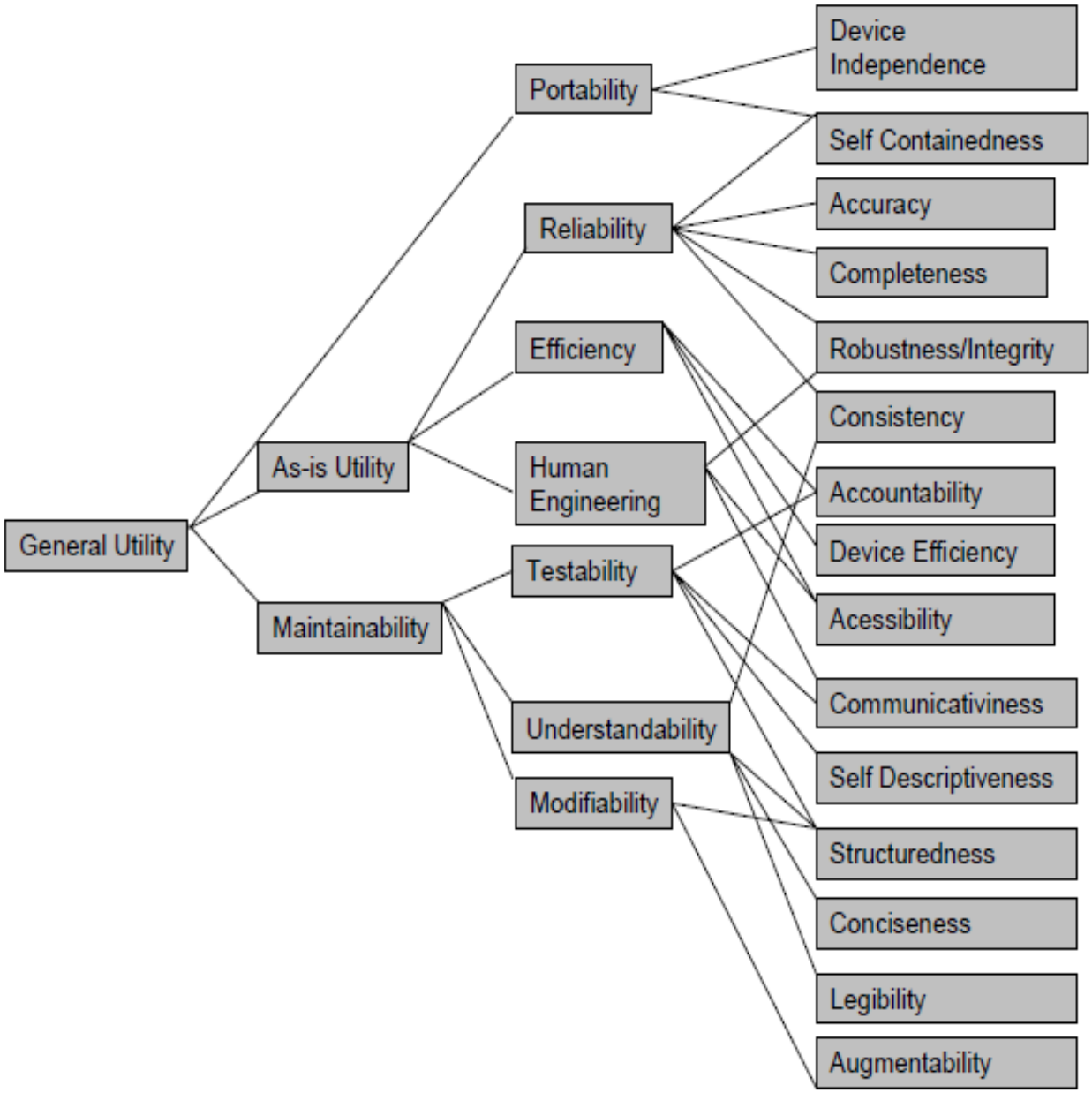
1- ما مدى (سهولة، كفاءة ، وأعتمادية) البرنامج ؟

2- قابلية الصيانة : ما مدى سهولة فهم وتعديل واعادة أختبار البرنامج؟

3- النقل : هل بالامكان أستخدام البرنامج أذا قمت بتغيير بيئتى؟

أما خصائص المستوى المتوسط فتمثل عوامل الجودة السبعة فى نموذج بوهيم والتى تمثل مجتمعة مستوى الجودة المطلوب فى المنتج البرمجى وهى : امكانية النقل ، الأعتمادية ، الكفاءة ، سهولة الاستخدام، قابلية الاختبار، قابلية الفهم ، والمرونة.

الخصائص الاساسية فهى تمثل الاساس لتعريف قياسات الجودة، وهى أحد الاهداف الاساسية من بناء النموذج ، الشكل 4-3 يوضح الخصائص الاساسية وارتباطها مع خصائص المنتج البرمجى السبعة .

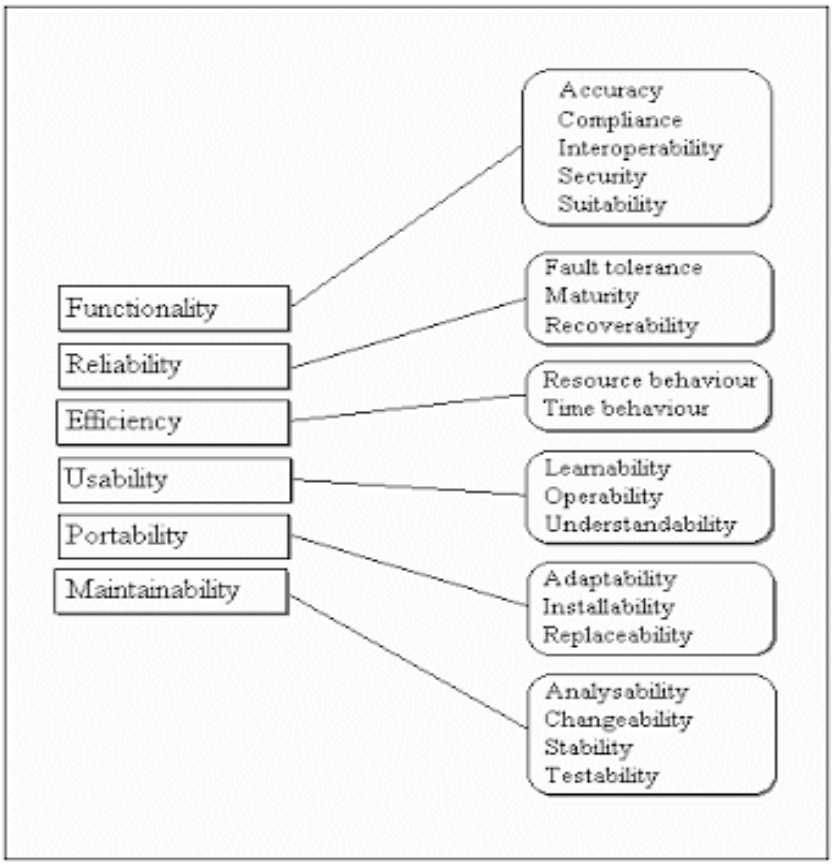


شكل 4-3 الخصائص الاساسية فى نموذج بوهم Boehm

4.5.3 نموذج FURPS

يقسم هذا النموذج الخصائص الي فئتين من المتطلبات هما: متطلبات وظيفية ومتطلبات غير وظيفية .

المتطلبات الوظيفية يتم تعريفها بواسطة مدخلات ومخرجات متوقعة وتمثل حرف Fبينما المتطلبات الغير وظيفية تعرف بـ (URPS) حيث تشير U سهولة الاستخدام و R للموثوقية و P للاداء و S لامكانية الدعم . الشكل 4-4 يبين الخصائص الرئيسية والفرعية المرتبطة بها



شكل 4-4 خصائص الجودة فى نموذج FRUP

4.5.4 نموذج Dromey

نموذج جودة يستند علي معايير التقييم والهدف من هذا المنتج هو تقييم جودة المنتج عندما يكون لكل منتج برمجي جودة تختلف من الاخر ، يصف Dromey مجموعة من تقنيات التطوير الديناميكية المطلوبة لتغطي اكبر قدر ممكن من انظمة البرمجيات ، تم تصميم هذا النموذج علي العلاقة بين عوامل الجودة والعوامل الفرعية بين خصائص البرمجيات وعوامل جودة البرمجيات.

نموذج Dromey يتعامل مع ثلاثة عناصر اساسية تتضمن:

1- خصائص المنتج التي تؤثر علي الجودة.

2- جودة عالية المستوي.

3- وسيلة للربط بينهم.

تم بناء هذا النموذج حول خمسة خطوات من العمليات:

1- اختيار مجموعة من سمات الجودة رفيعة المستوي الضرورية للتقييم.

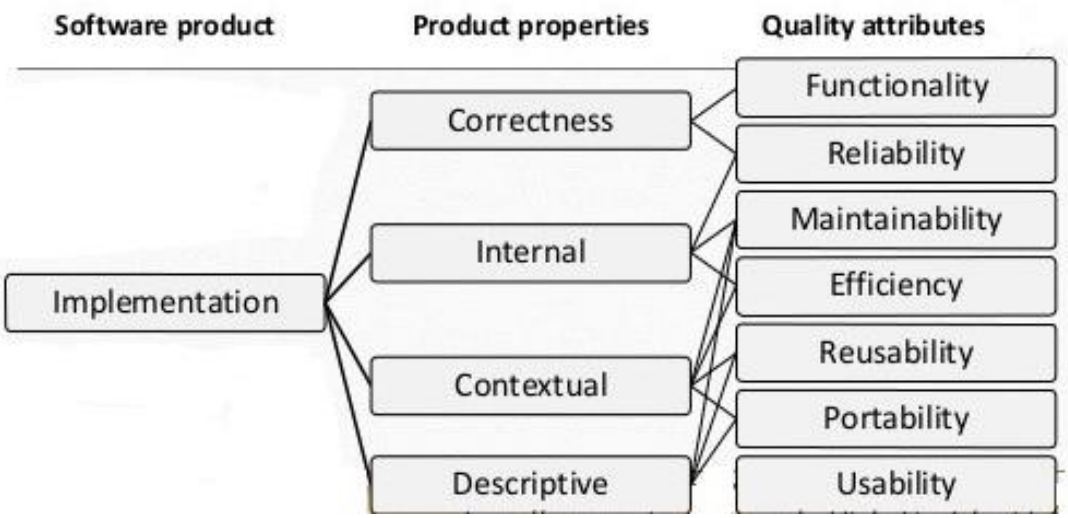
2- قائمة بالمكونات والوحدات للنظام الخاص بك.

3- تحديد الخصائص التي تنفذ الجودة للمكون او للوحدة.

4- تحديد كيفية تأثير كل خاصية علي صفات الجودة .

5- تقييم النموذج وتحديد نقاط الضعف.

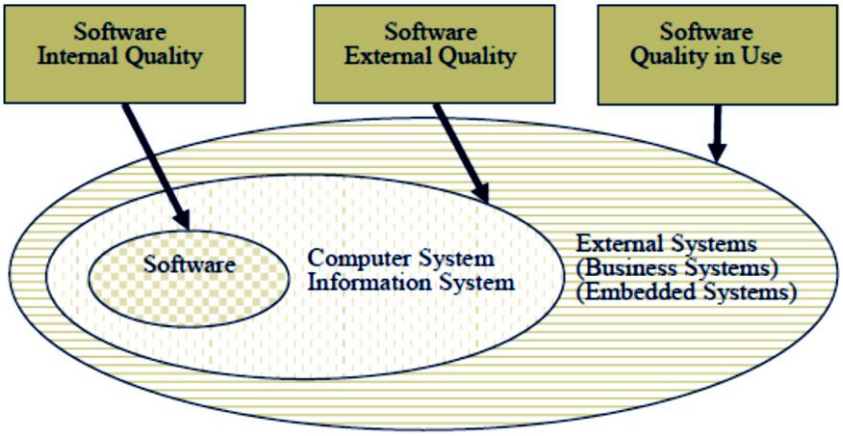
الشكل 4-5 يوضح خصائص نموذج Dromey



شكل 4-5 خصائص نموذج Dromey

4.5.6 نموذج ISO 9126-1:

نموذج الجودة ISO 9126-1 يستند علي نموذجي McCall و Boehm وقسم الجودة الى خصائص جودة داخلية وخصائص جودة خارجية وخضائص الجودة قيد الاستخدام كما هو موضح بالشكل 4-6 أدناه [37]

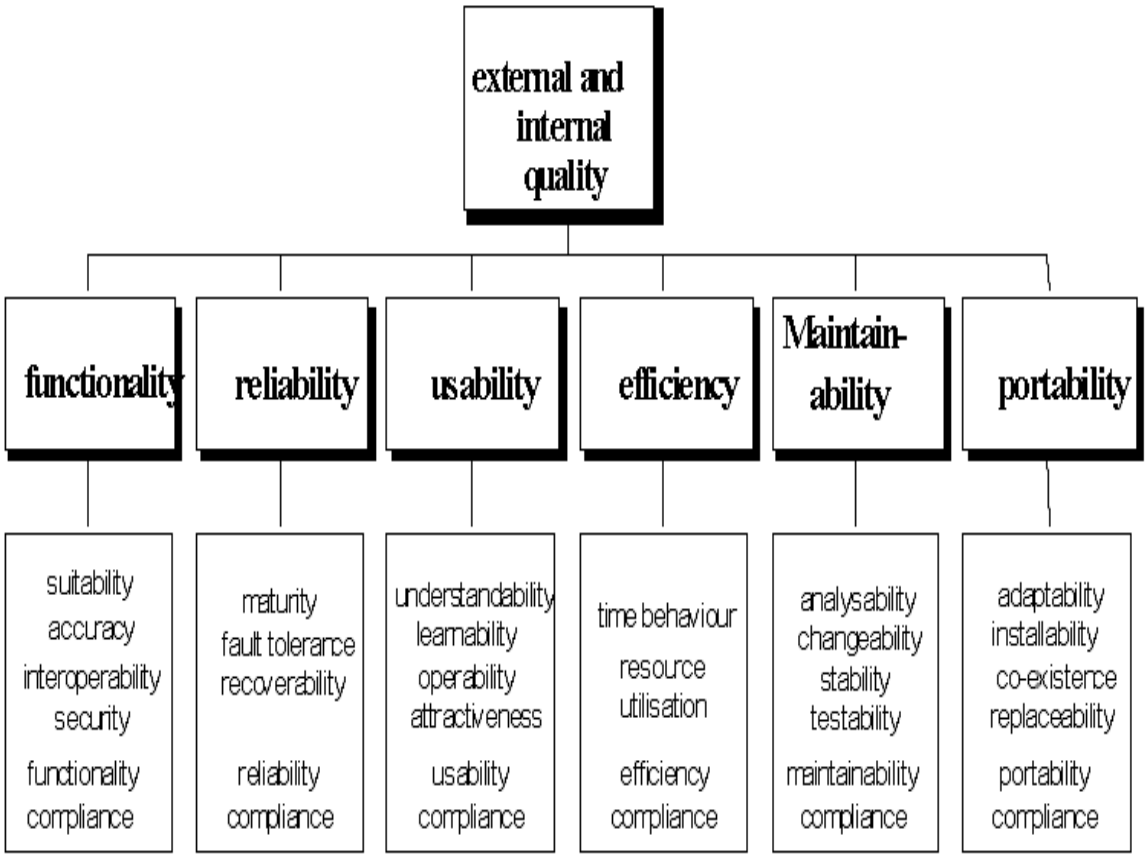


شكل 4-6 خصائص الجودة فى نموذج ISO 9126-1

تشير خصائص الجودة الداخلية الي خواص النظام التي يمكن تقيمها بدون تنفيذ النظام ،اما خصائص الجودة الخارجية تشير الي خواص النظام التي يمكن تقيمها بملاحظة النظام اثناء تنفيذه. بينما تشير خصائص الجودة قيد الاستخدام الي خواص النظام المجربة من قبل مستخدم النظام عندما يكون النظام اثناء عملة وايضا اثناء صيانتة.

خصائص جودة المنتج البرمجى فى هذا النموذج هى : الكفاءة ، الوظيفة ، قابلية الصيانة، قابلية النقل، الموثقية ،قابلية الاستخدام.

فى هذا النموذج تم ربط خصائص الجودة بمجموعة من الخصائص الفرعية ، الشكل 4-7 يوضح الخصائص الرئيسية للجودة والخصائص الفرعية المرتبطة بها



شكل 4-7 خصائص الجودة الرئيسية والخصائص الفرعية المرتبطة بها نموذج ISO 9126-1

4.5.7 نموذج ISO/IEC 25010:2011

وهو عبارة عن تطوير للنموذج السابق ، وقد أحتوى على نفس الخصائص الموجودة فى النموذج السابق مع بعض التغييرات والتمثلة فى الاتى:[38]

- تم توسيع نطاق نموذج الجودة ليشمل أنظمة الكمبيوتر بصورة عامة عوضا عن البرمجيات فقط، كما أيضا شمل الجودة أثناء الأسنخدام من وجهة نظر النظام.
- تم أضافة تغطية السياق Context Coverage كواحدة من خصائص الجودة أثناء الأستخدام مع الخصائص الفرعية : الخصوصية confidentiality ، التكاملية integrity ، عدم التنصل non– repudiation ، المسئولية accountability ، والأصالة authenticity .
- تمت إضافة التوافق Compatibility (بما في ذلك العمل المشترك والتعايش) كخاصية.
- تمت أضافة الخصائص الفرعية التالية : أكتمال الوظيفية functional completeness ، القدرة capacity ، حماية المستخدم من الاخطاء user error protection ، الوصول accessibility ، الاتاحية availability ، النمطية modularity ، وأعادة الاستخدام reusability.
- تم حذف الخصائص الفرعية لخاصية الجودة الامتثال Compliance ، فالامتثال للوائح والقوانين هو جزء من المتطلبات العامة وليس بالضرورة جزء من الجودة.
- تم دمج نموذجى الجودة الداخلية والخارجية وأعتبارها نموذج جودة المنتج.
- متى ما كان مناسبا تم أستخدام تعاريف عامة عوضا عن التعاريف الخاصة بالبرامج.

- عدد من الخصائص الرئيسية والفرعية تم اعادة تسميتها واعطائها أسماء أكثر دقة.

4.6 مقاييس جودة البرمجيات

مما سبق نجد أنه لا بد من وجود قياسات تمكن المنتج والمستهلك على متابعة المنتج والتأكد من مطابقته للمواصفات. ونجد أن مقاييس البرمجيات قد تطورت مع مرور الوقت تطورا واكب التطور الكبير الذى حدث فى مجال البرمجيات. ولعل من اهم المقاييس التى ركز عليها العلماء والباحثون هى تلك المقاييس التى تقيس التعقيد أذ أن التعقيد هو أحد العوامل الرئيسية والتى من خلالها يمكن أن نحدد الخصائص الاخرى كقابلية الصيانة والأعتمادية.

من أشهر مقاييس البرمجيات:

1- التعقيد.

2- الاخطاء مقابل عدد اسطر الكود.

3- قابلية الصيانة.

4- قابلية الاستخدام

4.6.1 التعقيد Complexity

كما ذكرنا سابقا فأن التعقيد يعتبر من أهم مقاييس جودة البرمجيات لانه من العوامل المؤثرة على جودة البرمجيات ككل. فالتقيد يؤثر على الصحة وقابلية الصيانة وقابلية الاختبار والاعتمادية. وبصورة عامة يمكن تقسيم التعقيد الى نوعين هما:

1- التعقيد الهيكلى Structural Complexity ويقصد به التعقيد المرتبط ببنية البرنامج أو الكود.

2- التغقيد الادراكى Cognitive Complexity .ويقصد به مدى سهولة وأدراك الانسان للنظام.

طور العلماء والباحثون عدة مقاييس لقياس تعقيد البرمجيات منها:

- LOC- Lines Of Code يعتبر هذا المقياس من ابسط المقاييس وتعتمد فكرته الاساسية على أنه كلما زاد عدد أسطر الكود كلما زاد تعقيد النظام ويقوم بحساب عدد الاسطر فى الكود.
- Cyclomatic complexity تم تطوير هذا المقياس من قبل Thomas J. McCabe فى العام 1976 وتقوم فكرته الاساسية على قياس المسارات المستقلة خطيا من خلال الكود وهو يشبه الى حد كبير مقياس تعقيد النص الذى طوره Flesch-Kincaid Readability ويتم حسابه من خلال مخطط التدفق للبرنامج Flow-Chart.

- مقاييس التعقيد Coupling و Cohesion والتى تعنى بمدى أرتباط وأتصال وحدات البرنامج المختلفة مع بعضها البعض، يوجد عدد كبير من المقاييس التى تنتمى الى هذين التصنيفين.
- مقاييس هولستيد Halstead's Metrics تم تطوير هذا المقياس من قبل العالم Halstead فى العام 1977م لقياس تعقيد البرمجيات. وهو من المقاييس الاستاتيكية والتى يتم تطبيقها على الكود مباشرة ولا تتطلب ان يتم تنفيذ البرنامج، فى هذا المقياس يتم حساب أربعة قيم أساسية ومن ثم يمكن حساب عدد من المقاييس أعتمادا على هذه القيم ، القيم الاربعة هى:
  - n1 : عدد المعاملات الاستثنائية distinct operators
  - n2 : عدد العوامل الاستثنائية distinict operands
  - N1 : العدد الكلى للمعاملات
  - N2 : العدد الكلى للعوامل

من خلال هذه القيم الاربعة يمكن حساب القياسات التالية:

1- مفردات البرنامج Program Vocabulary :

$$n = n1 + n2$$

2- طول البرنامج Program Length:

$$N = N1 + N2$$

3- طول البرنامج المحسوب Calculated Program Length:

$$N^{\wedge} = n1\ \log_2 n1 + n2\ \log_2 n2$$

4- الحجم Volume:

$$V = N\ \log_2 n$$

5- الصعوبة Difficulty:

$$D = \frac{n1}{2} \times \frac{N2}{n2}$$

6- المجهود Effort :

$$E = D \times V$$

7- الزمن المطلوب للتطوير Time required to program:

$$T = \frac{E}{18}\ seconds$$

8- الاخطاء عند التسليم Number of delivered bugs :

$$B = \frac{E^{\frac{2}{3}}}{3000}$$

4.6.2 الاخطاء مقابل اسطر الكود

هذا المقياس يقوم بحساب عدد الأخطاء مقابل كل الف سطر من الشفرة البرمجية KLOC. وستدل به على صحة النظام والتى تعنى مقدار أداء النظام لوظائفه الرئيسية.

4.6.3 قابلية الصيانة

قابلية الصيانة تعنى مقدار الجهد والزمن المطلوب من أجل أستعادة النظام الى وضعه الطبيعى بعد حدوث خطأ ويتم قياسه من خلال ما يعرف بـ MTTC- Mean Time To Change وهو يقيس عند حدوث خطأ كم من الزمن نحتاج حتى نتمكن من أكتشافه وتحليه ومعالجته وأختباره.

4.6.4 قابلية الاستخدام

ومن خلاله يتم قياس مدى سهولة أستخدام النظام وهو يعتبر من المقاييس الاساسية. ويتم قياس ذلك من خلال:

- الجهد الفيزيائى والفكرى المطلوب من أجل تعلم النظام.
- الزمن المطلوب من أجل الوصول الى درجة الاحترافية فى التعامل مع النظام.
- الزيادة فى الانتاجية من خلال استخدام النظام الجديد.
- التقييم الذاتى من خلال استخدام الاستبيانات.

العنوان:	تأثير البرمجة ذات التوجه المفاهيمى على جودة البرمجيات
المؤلف الرئيسي:	عثمان، أحمد سيد أحمد علي
مؤلفين آخرين:	حاج على، عوض(مشرف)
التاريخ الميلادي:	2015
موقع:	الخرطوم
الصفحات:	1 - 240
رقم MD:	830496
نوع المحتوى:	رسائل جامعية
اللغة:	Arabic
الدرجة العلمية:	رسالة دكتوراه
الجامعة:	جامعة النيلين
الكلية:	كلية الدراسات العليا
الدولة:	السودان
قواعد المعلومات:	Dissertations
مواضيع:	هندسة البرمجيات، برمجة المفاهيمية، جودة البرمجيات، التحليل البرمجى
رابط:	https://search.mandumah.com/Record/830496

الفصل السادس (التجربة التطبيقية

ومناقشة النتائج)

6.1 مقدمة

التجربة التطبيقية كانت عن طريق سؤال عدد من المتطوعين لكتابة برنامجين الاول بأستخدام البرمجة الكائنية والثانى بأستخدام البرمجة المفاهيمية لنفس مجموعة المتطلبات. البرنامج الاول تم تطويره بأستخدام لغة الجافا Java والثانى بأستخدام لغة AspectJ المبنية على لغة الجافا. وبعد الانتهاء من تطوير التطبيقين قاموا بملء أستبيان عن البرمجة المفاهيمية.

الهدف من هذه الدراسة التطبيقية هو أختبار مدى تأثير البرمجة المفاهيمية على خصائص الجودة التالية: قابلية الأختبار، قابلية الصيانة، الأداء ، وقابلية أعادة الأستخدام وسهولة الفهم. تم أختيار خصائص الجودة أعلاه بناءا على العوامل التالية:

1- مدى أهمية خاصية الجودة المحددة وتأثيرها على خصائص الجودة الأخرى وتكلفة البرنامج.
2- أن تكون من خصائص الجودة التى يمكن أن تتأثر بالمنهجية المتبعة بالتطوير، فبعض الخصائص لا تتأثر بالمنهجية كالوظيفية على سبيل المثال.
3- أن تكوم من الخصائص التى يمكن قياسها من خلال البيانات المتاحة للدراسة، فبعض خصائص الجودة تتطلب مثلا أن يتم أستخدام البرنامج لفترة زمنية طويلة كالصحة والأعتمادية مثلا.

تمت دراسة التأثير عن طريق تحليل الاستبيان الذى قام بملئه المتطوعون بالأضافة الى أجراء قياسات على الكود بالنسبة للبرنامجين الذان قام بتطويرهما كل متطوع.

من أصل 32 برنامج تم أستبعاد 7 برامج لعدم أكتمالها أو لعدم تطابق المتطلبات بالنسبة للبرنامجين المكتوبين بالبرمجة المفاهيمية والكائنية. وبالتالى أصبح عدد البرامج التى سيتم تطبيق التجربة عليها هى 25 برنامج مكتوب بأستخدام البرمجة المفاهيمية و 25 برنامج مكتوب بأستخدام البرمجة الكائنية.

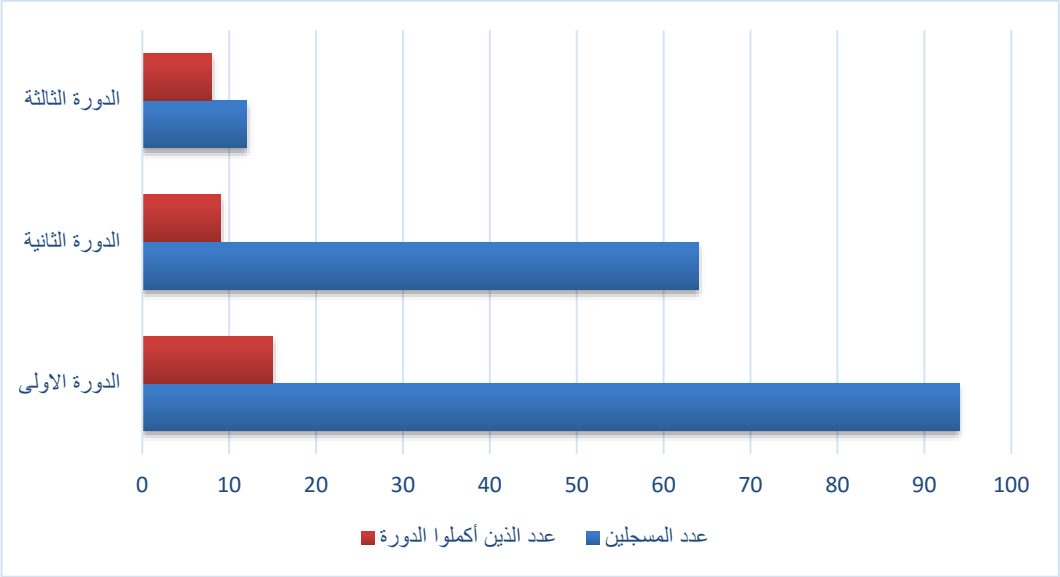
المتطوعون كان عددهم الاجمالى 32 متطوع مقسمون على 3 مجموعات المجموعة الاولى كانت عبارة عن 15 متطوع والمجموعة الثانية عبارة 9 متطوعين والثالثة 8 متطوعين.

جميع المتطوعين من طلاب البكالوريوس او الماجستير والمتخصصين فى مجالات الحاسوب، وتتراوح مستوياتهم بين الفرقة الثالثة وطلاب دراسات عليا (ماجستير) ، المتطوعون لديهم معرفة جيدة بلغة البرمجة جافا بالاضافة الى البرمجة الكائنية ، ولكن ليس لاى احد منهم معرفة مسبقة بالبرمجة المفاهيمية. المتطوعون ينتمون الى 5 جامعات مختلفة هى : جامعة النيلين، جامعة الخرطوم، جامعة السودان للعلوم والتكنولوجيا ، كلية الخرطوم التطبيقية، جامعة الرباط الوطنى.

6.2 المعوقات التى واجهت الدراسة

واحدة من أكبر المعوقات التى واجهت الدراسة هى عدم أنتشار أستخدام البرمجة المفاهيمية، مما أضطر الباحث الى تنظيم كورسات لتدريب المتطوعين على البرمجة المفاهيمية، ولكن الاقبال كان ضعيفا للغاية، تم عقد ثلاثة دورات تدريبية بعد ان تم الاعلان عنها فى جامعة النيلين بكلية علوم الحاسوب كما تم أستخدام وسائط التواصل الاجتماعى للاعلان عن الدورة

التدريبية، الاقبال والتسجيل كان كبيرا فللدورة الاولى قام بالتسجيل للدورة عدد 94 متطوع فى حين أن الذين أكملوا الدورة بنجاح حتى النهاية كان 15 متطوعا، أما بالنسبة للدورة الثانية فأن عدد المسجلين للدورة بلغ 64 متطوعا فى حين ان الذين أكملوا الدورة بلغ 9 ، أما بالنسبة للدورة الثالثة فبلغ عدد المسجلين للدورة 12 متطوعا بينما بلغ عدد الذين أكملوا الدورة 8 متطوعين، الشكل 6-1 يوضح الفرق بين عدد المسجلين للدورات التدريبية والذين أكملوا الدورات



شكل 6-1 نسبة الذين أكملوا الدورات لعدد الذين سجلوا للدورات

نخلص من البيانات أعلاه الى أن نسبة الذين أكملوا الدورات بالنسبة للعد الكلى الذى قام بالتسجيل هى 18.8% فقط ويرجع الباحث ذلك للاسباب التالية:

1- عدم معرفة الطلاب عموما او سماعهم بالبرمجة المفاهيمية.

2- عدم أجادة لغة البرمجة جافا كأحد المتطلبات لأكمال الدورة.

3- عدم أجادة البرمجة الكائنية كأحد متطلبات الدورة.

4- عدم وجود شهادات بنهاية الدورة. معظم الذين قاموا بالتسجيل للدورة كانوا يبحثون عن شهادات يدعمون بها السيرة الذاتية أكثر من المعرفة.

6.3 الدورات التدريبية

تم عفد ثلاثة دورات تدريبية فى البرمجة المفاهيمية للمتطوعين ، كل دورة أخذت ما بين 15 ألى 20 ساعة تدريبية ، أحتوت الدورات على المواضيع التالية :

1- مقدمة عن البرمجة الكائنية ولغة الجافا من خلال تطبيق برنامج بأستخدام البرمجة الكائنية.

2- مراجعة خصائص جودة البرمجيات.

3- مقدمة للبرمجة المفاهيمية ، والاهتمامات المتقاطعة والموزعة.

4- مصطلحات البرمجة المفاهيمية.

5- كيفية تنزيل وتشغيل لغة AspectJ

6- كتابة برنامج بأستخدام البرمجة المفاهيمية

كل الدورات التدريبية تم عقدها فى جامعة النيلين بكلية علوم الحاسوب وتقانة المعلومات

6.4 تأثير البرمجة المفاهيمية على قابلية الصيانة Maintainability

قابلية الصيانة تعنى مدى الجهد المبذول من أجل صيانة النظام (أصلاح النظام أو أضافة متطلبات جديدة) خلال فترة زمنية محددة. أو بعبارات أخرى فأن سهولة الصيانة تقيس مدي سهولة وسرعة أستعادة النظام الى وضعه الطبيعى بعد حدوث خطأ.

الصيانة تعتبر من النشاطات المركزية فى عملية تطوير البرمجيات. وهى تكلف أكثر من نصف تكلفة البرمجيات ككل.

وقابلية الصيانة من خصائص الجودة الخارجية للبرمجيات، وبالتالى فانه لا يمكن قياسها بصورة مباشرة ولكن يمكن قياسها عن طريق ربطها مع بعض خصائص الجودة الداخلية لبرمجيات، ووفقا لنموذج ISO/IEC 9126 فأن قابلية الصيانة تتاثر بالخصائص التالية: قابلية الاختبار Testability، قابلية الفهم Undertandability، قابلية التعديل Moifiability، والاستقرار Stability. ولكل واحدة من الخصائص أعلاه يوجد عدد من المقاييس المنفصلة ولكن من خلالها لا يمكن الحصول على رقم واحد نستطيع من خلاله الحكم على قابلية الصيانة.

فى العام 1992م قام كل [51] من Paul Oman و Jack Hagemeister بتطوير مقياس أداء الصيانة Maintainability Index MI والذى تم عرضه فى مؤتمر International Conference on Software Maintianance وتمت مراجعت المقياس بعد ذلك عن طريق ورقة قدمت لمنظمة IEEE، هذا المقياس قام بدمج عدد من المقاييس وهى مقياس الحجم لهولستيد Halstead's Volume Metric (HV) ، مقياس تعقيد موكابي McCabe's

cylcomatic complexity (CC) ، ومقياس عدد أسطر الكود Line of Code (LOC) ، ونسبة عدد أسطر التعليق Percentage of Comments (COM). يتم أخذ المتوسط او المجموع للقياسات اعلاه على المكونات المختلفة فى البرنامج ومن ثم حساب أداء الصيانة من خلال المعادلة

$$MI = 171 - 5.2\ \ln(HV) - 0.23\ (CC) - 16.2\ \ln(LOC) + 50.0\ \sin\sqrt{2.46\ \times (COM)}$$

أذا كانت النتيجة التى حصلنا عليها كبيرة فأن ذلك يعنى قابلية صيانة أعلى، على سبيل المثال نجد شركة مايكروسوفت قامت بتضمين هذا المقياس واعتماده منذ العام 2007م فى أصدارات الفيجوال ستديو Visual Studio، فأذا حصل البرنامج المكتوب باحدى لغات الفيجوال ستديو على اداء صيانة اكبر من او يساوى 20 [51] فان ذلك يعنى قابلية صيانة عالية ومن 10 وحتى 20 يعتبر متوسط القابلية للصيانة، واقل من ذلك من الصعب صيانته.

عند حساب القيم للبرامج المكتوبة سواءا بأستخدام البرمجة المفاهيمية او الكائنية ولغرض المقارنة بين الطريقتين تم تم تجاهل المعامل الاخير فى المعادلة والمتعلق بعدد أسطر التعليقات فى الكود وتم أدخاله بالقيمة 0 ، وذلك لأن عدد أسطر الكود عبارة عن مجهود بشرى وليس له علاقة بالتقنية المستخدمة. أما بالنسبة للقيم الاخرى فكما تم ذكره سابقا هنالك طريقتان لحساب القيم الاخرى أما بحساب المتوسطات بالنسبة لمكونات البرنامج المختلفة أو بحساب المجموع، ونسبة لان عدد المكونات غير ثابت فى البرامج موضوع المقارنة فأنه قد تم أستخدام المتوسط.

الجدول التالى 6-1 يوضح القياسات الداخلية التى تم أجرائها واختصاراتها :

Symbol	Level	Name
A	package	Abstractness
AC	package	Afferent Coupling
C	package	Coverage
D	package	Distance
EC	package	Efferent Coupling
I	package	Instability
LOC	package	Lines Of Code
LOCm	package	Lines Of Comments
NCP	package	Number Of Classes in Package
NIP	package	Number Of Interfaces in Package
LCC	class	Loose Class Coupling
LCOM1	class	Lack Of Cohesion in Methods 1
LCOM2	class	Lack Of Cohesion in Methods 2
LCOM3	class	Lack Of Cohesion in Methods 3
LCOM4	class	Lack Of Cohesion in Methods 4
LCOM5	class	Lack Of Cohesion in Methods 5
LOC	class	Lines Of Code
LOCm	class	Lines Of Comments
NAK	class	Number of Assertions per KLOC
NOC	class	Number Of Children
NOF	class	Number Of Fields
NOM	class	Number Of Methods
NOSF	class	Number Of Static Fields
NOSM	class	Number Of Static Methods
NTM	class	Number of Test Methods
TCC	class	Tight Class Coupling
WMC	class	Weighted Method Count
LOC	method	Lines Of Code
LOCm	method	Lines Of Comments
NBD	method	Nested Block Depth
NOP	method	Number Of Parameters

VG	method	McGabe's Cyclomatic Complexity

جدول 6-1 القياسات التى تم تطبيقها على البرامج

أما الجدول التالى 6-2 فيوضح نتائج القياسات التى تم تطبيقها على البرامج الخمسة وعشرون بالاضافة الى حساب مقياس أداء الصيانة Maintanance Index للبرامج المكتوبة بالبرمجة الكائنية

NO	Package	Class	A	AC	D	NCP	LCC	LOCM1	LOCM2	LOCM3	LOCM4	LOCM5	NOC	NOF	NOM	NOSF	NOSM	NTM	TCC	WMC	LOC	LOCm	NBD	NOP	VG	HV	MI
1	javaapplication1		0	0	0.70711	1	0.267	11	7	3	3	0.7	0	2	6	0	1	0	0.26667	9	41	0	1	4	1	347.08	80.19244
		AccountSimple					0.267	11	7	3	3	0.7	0	2	6	0	1	0	0.26667	9	41	0	1	4	1	347.08	
2	(default package)		0	0	0.70711	1	0	10	10	5	5	0	0	0	5	0	1	0	0	6	38	0	1	9	1	1411.27	74.12943
		cal					0	10	10	5	5	0	0	0	5	0	1	0	0	6	36	0	1	9	1	1411.27	
3	(default package)		0	0	0.70711	2	0.5	7	0	1	1	0.34524	0	6	9	1	1	0	0.25	11	81	9	1	6	1	2033.5	59.96885
		Student					1	14	0	1	1	0.69048	0	6	8	1	0	0	0.5	10	68	9	1	5	1	2048	
		TestStudent					0	0	0	1	1	0	0	0	1	0	1	0	0	1	11	0	1	1	1	2019	
4	(default package)		0	0	0.70711	1	0	6	6	4	4	0	0	0	4	0	1	1	0	5	41	0	1	3	1	2057.45	70.93817
		test					0	6	6	4	4	0	0	0	4	0	1	1	0	5	37	0	1	3	1	2057.45	
5	(default package)		0	0	0.70711	1	0	3	3	3	3	0	0	0	3	0	3	0	0	3	23	0	1	4	1	2038.56	80.351
		Hello					0	3	3	3	3	0	0	0	3	0	3	0	0	3	21	0	1	4	1	2038.56	
6	(default package)		0	0	0.70711	3	0.6	15	0	1	1	0.24444	0	4	27	0	1	0	0.48148	27	150	7	0	22	1	2237.973	49.48841
		MyCircle					1	32	0	1	1	0.4	0	2	16	0	0	0	0.73333	16	76	3	0	12	1	2228.4	
		MyPoint					0.8	13	0	2	2	0.33333	0	2	10	0	0	0	0.71111	10	55	3	0	9	1	2276.2	
		TestMyCircle					0	0	0	1	1	0	0	0	1	0	1	0	0	1	16	1	1	1	1	2209.32	
7	(default package)		0	0	0.70711	1	0.19048	17	13	4	4	0.75	0	2	7	0	2	0	0.19048	9	45	0	1	5	1	2391.43	68.6479
		AccountSimple					0.19048	17	13	4	4	0.75	0	2	7	0	2	0	0.19048	9	43	0	1	5	1	2391.43	
8	(default package)		0	0	0.70711	3	0.66667	8	0	1	1	0.39653	0	7	16	0	1	0	0.37963	16	109	8	0	14	1	2430.117	54.23255
		Author					1	5	0	1	1	0.53333	0	3	6	0	0	0	0.66667	6	36	4	0	4	1	2343.33	
		Book					1	19	2	1	1	0.65625	0	4	9	0	0	0	0.47222	9	53	4	0	9	1	2439.65	
		Test					0	0	0	1	1	0	0	0	1	0	1	0	0	1	15	0	1	1	1	2507.37	
9	javaapplication1		0	0	0.70711	1	0	1	1	2	2	1.5	0	2	2	1	1	0	0	3	25	0	1	2	1	2633.74	77.66818
10		JFact					0	1	1	2	2	1.5	0	2	2	1	1	0	0	3	21	0	1	2	1	2633.74	
10	javaapplication1		0	0	0.70711	1	0	1	1	2	2	1	0	1	2	0	1	1	0	2	26	0	1	1	1	303.58	88.26748
		test					0	1	1	2	2	1	0	1	2	0	1	1	0	2	17	0	1	1	1	303.58	
11	pkg11		0	0	0.70711	1	0	0	0	1	1	0	0	0	1	0	1	0	0	1	39	3	1	1	1	513.53	78.9655
		Test					0	0	0	1	1	0	0	0	1	0	1	0	0	1	28	3	1	1	1	513.53	

12	(default package)		0	0	0.70711	1	0	3	3	3	3	0	0	0	3	0	1	0	0	3	32	0	1	3	1	521.32	82.09199
		Test					0	3	3	3	3	0	0	0	3	0	1	0	0	3	30	0	1	3	1	521.32	
13	(default package)		0	0	0.70711	1	0	6	6	4	4	1	0	1	4	0	1	0	0	4	36	0	1	3	1	664.39	78.92287
		Test					0	6	6	4	4	1	0	1	4	0	1	0	0	4	34	0	1	3	1	664.39	
14	(default package)		0	0	0.70711	1	0	6	6	4	4	0	0	0	4	0	1	0	0	5	37	0	1	3	1	429.53	80.74713
		Test					0	6	6	4	4	0	0	0	4	0	1	0	0	5	35	0	1	3	1	429.53	
15	(default package)		0	0	0.70711	2	0.5	6	0	1	1	0.22222	0	3	11	0	1	0	0.36667	17	97	6	1	10	1	925.63	61.14121
		MyTime					1	12	0	1	1	0.44444	0	3	10	0	0	0	0.73333	16	78	6	1	9	1	938.35	
		TestMyTime					0	0	0	1	1	0	0	0	1	0	1	0	0	1	21	0	2	1	1	912.91	
16	(default package)		0	0	0.70711	1	0	3	3	3	3	0	0	0	3	0	1	0	0	3	27	0	1	2	1	946.86	81.74106
		hello					0	3	3	3	3	0	0	0	3	0	1	0	0	3	23	0	1	2	1	946.86	
17	(default package)		0	0	0.70711	4	0.58333	0	0	1	1	0.125	2	5	10	0	1	0	0.58333	10	83	7	1	8	1	1071.68	62.90447
		Rectangle					1	0	0	1	1	0	0	2	3	0	0	0	1	3	23	2	1	3	1	1049.71	
		Shape					0.33333	2	1	2	2	0.5	2	1	3	0	0	0	0.33333	3	21	3	1	1	1	1023.86	
		TestShape					0	0	0	1	1	0	0	0	1	0	1	0	0	1	12	0	1	1	1	1085.35	
		Triangle					1	0	0	1	1	0	0	2	3	0	0	0	1	3	23	2	1	3	1	1127.8	
18	(default package)		0	0	0.70711	1	1	0	0	1	1	0	0	1	2	1	1	0	1	2	23	0	1	1	1	1162.75	83.27057
		Test					1	0	0	1	1	0	0	1	2	1	1	0	1	2	19	0	1	1	1	1162.75	
19	(default package)		0	0	0.70711	3	0.66667	1	0	1	1	0.11111	1	3	13	0	1	0	0.60317	13	80	8	0	9	1	1238.99	62.7465
		Point					1	4	0	1	1	0.33333	1	2	7	0	0	0	0.80952	7	34	4	0	4	1	1215.44	
		Point3D					1	0	0	1	1	0	0	1	5	0	0	0	1	5	29	4	0	4	1	1268.45	
		TestPoint3D					0	0	0	1	1	0	0	0	1	0	1	0	0	1	14	0	1	1	1	1233.08	
20	(default package)		0	0	0.70711	1	0	0	0	1	1	0	0	1	1	1	1	0	0	1	49	0	1	1	1	1474.38	69.78335
		LinkedListDemo					0	0	0	1	1	0	0	1	1	1	1	0	0	1	44	0	1	1	1	1474.38	
21	(default package)		0	0	0.70711	1	0	1	1	2	2	1	0	1	2	0	1	0	0	2	73	6	2	1	1	1776.63	62.35569
		m					0	1	1	2	2	1	0	1	2	0	1	0	0	2	61	6	2	1	1	1776.63	
22	(default package)		0	0	0.70711	2	0.5	1	0	1	1	0.25	0	2	5	0	1	0	0.33333	5	39	0	0	3	1	1781.3	72.49779
		Circle					1	2	0	1	1	0.5	0	2	4	0	0	0	0.66667	4	23	0	0	2	1	1748.85	
		TestCircle					0	0	0	1	1	0	0	0	1	0	1	0	0	1	14	0	1	1	1	1813.75	
23	hotel.		0.07692	0	0.65271	13	0.40935	20	12	3	2	0.58693	0	182	80	14	8	0	0.27563	92	3105	242	1	51	0	18058.62	-10.2276
		CurrentCustomer					0.81818	27	0	3	2	0.895	0	20	11	1	1	0	0.6	13	375	14	2	7	1	8517.86	

	CustomerServices					0.77206	107	78	5	5	0.97635	0	37	17	1	1	0	0.22794	18	602	25	1	14	1	8652.51	
	DBTable_1					0	0	0	0	0	0	0	3	0	1	0	0	0	0	49	6	0	0	0	8249.28	
	Frame4all					0	3	3	3	3	0	0	0	3	0	1	0	0	4	73	20	2	1	1	8159.97	
	GuestData					0.14286	20	19	6	5	0.96429	0	14	7	1	0	0	0.14286	7	197	14	0	5	1	8260.45	
	JInternal1					1	1	1	2	1	1	0	1	2	0	0	0	1	2	48	11	1	0	1	8081.91	
	Login					0.41667	31	26	5	4	0.95192	0	13	9	1	1	0	0.19444	11	242	17	1	6	1	8104.2	
	MDIApplication					0.70513	35	0	3	3	0.96558	0	46	13	1	1	0	0.55128	15	712	36	2	9	1	8349.88	
	MySQL_DB					0	0	0	1	1	0	0	5	1	5	1	0	0	1	25	0	2	0	1	8821.15	
	NewCustomer					0.46667	38	31	6	6	0.94841	0	28	10	1	1	0	0.2	12	394	20	1	7	1	8148.81	
	NewCustomerInternal					0	1	1	2	2	0	0	0	2	0	0	0	0	2	41	10	1	0	1	8551.5	
	SignUp					1	4	2	2	1	0.92857	0	14	4	1	1	0	0.66667	6	202	13	2	2	1	8081.91	
	logger					0	0	0	1	1	0	0	1	1	1	0	0	0	1	16	1	2	0	1	8372.26	
24	re		0	0	0.70711	13	0.68578	47	18	3	2	0.7735	0	299	131	0	7	0	0.55072	270	6805	297	1	83	1	22540.87	-24.3217
	CategoryDetails					1	5	0	2	1	0.76667	0	12	6	0	0	0	1	9	209	8	1	4	1	8271.62	
	Connect					0	0	0	1	1	0	0	1	1	0	1	0	0	1	21	0	2	0	1	8461.82	
	Delivery_view					0.6	6	2	3	2	0.85	0	10	5	0	1	0	0.4	11	273	18	2	2	2	8473.03	
	Employee					0.66667	9	3	3	2	0.76667	0	24	6	0	0	0	0.66667	15	331	13	1	3	2	8866.18	
	KitchenTable					0.72857	151	92	8	4	0.94878	0	41	21	0	1	0	0.34762	28	803	22	1	13	1	8809.89	
	Login					0.28571	17	13	4	4	0.94048	0	14	7	0	1	0	0.19048	16	278	19	1	6	2	9950.1	
	MainMenu					0.56897	235	64	10	8	0.80263	0	38	29	0	1	0	0.42611	44	1433	22	2	17	1	9261.29	
	Meals					0.71429	11	1	3	2	0.77778	0	18	7	0	0	0	0.71429	14	290	9	1	3	2	10308.68	
	MenuMeal					0.75099	152	51	5	4	0.90246	0	48	23	0	1	0	0.40711	83	1349	30	2	14	3	10710.32	
	OrdersView					0.6	6	2	2	2	0.85	0	10	5	0	1	0	0.4	11	272	18	2	2	2	11773.83	
	PaymentView1					1	20	12	3	1	0.93728	0	41	8	0	0	0	0.60714	14	540	9	1	4	1	11866.82	
	RegisterNewAccount					1	6	0	2	1	0.76282	0	26	7	0	0	0	1	13	411	8	1	9	1	13730.79	
	RegisterResturant					1	5	0	2	1	0.75	0	16	6	0	0	0	1	11	264	8	1	6	1	14760.84	
25	libaray1		0	0	0.70711	9	0.59207	23	9	3	2	0.74154	0	151	77	0	10	0	0.54048	188	2979	44	1	49	1	24574.31	-11.3885
	Libaray1					0	0	0	1	1	0	0	0	1	0	1	0	0	1	7	0	0	1	1	15838.11	
	book1					0.8	20	0	3	2	0.7971	0	23	10	0	1	0	0.73333	29	478	4	1	6	2	15862.1	
	borrow1					1	7	0	2	1	0.76389	0	12	7	0	1	0	0.95238	12	221	3	2	3	1	15886.1	
	first					0.50909	34	13	5	4	0.94545	0	11	11	0	1	0	0.50909	11	183	3	1	9	1	15706.23	

	logger					0.28571	19	17	5	4	0.95	0	10	7	0	2	0	0.19048	10	161	5	1	3	1	16270.53	
	return1					1	5	0	2	1	0.78125	0	8	5	0	1	0	0.9	6	138	3	2	1	1	16234.45	
	searchb					0.77778	17	0	3	2	0.80208	0	24	9	0	1	0	0.69444	29	454	8	1	6	3	16643.87	
	searchs					0.46154	52	26	6	5	0.81771	0	32	13	0	1	0	0.42308	45	636	11	1	10	3	17441.63	
	stu					0.49451	58	25	6	5	0.81638	0	31	14	0	1	0	0.46154	45	655	7	1	10	3	17562.84	

جدول 6-2 القياسات الثابتة Static Metrics على برامج المنهجية الكائنية

من خلال الجدول السابق نجد أن أصغر قيمة لمقياس أداء الصيانة هى: **-24.32173278** وأكبر قيمة هى : **88.26748121**

بينما متوسط المقياس للبرامج هو : **61.40458688** .

الجدول التالى 6-3 يبين قيم نفس المقاييس عند تطبيقها على البرامج التى تم تطويرها بأستخدام البرمجة المفاهيمية

NO	Package	Class	A	D	NCP	LCC	LOCM1	LOCM2	LOCM3	LOCM4	LOCM5	NOC	NOF	NOM	NOSF	NOSM	NTM	TCC	WMC	LOC	LOCm	NBD	NOP	VG	HV	MI
1	(default package)		0.00	0.71	2	0.13	5	3	1	1	0.35	0	3	6	0	1	0	0.13	6	50	12	0	4	0	240.805	79.10849
		AccountAspect				0.00	0	0	0	0	0.00	0	1	0	0	0	0	0.00	0	4	0	0	0	0	199.42	
		AccountSimple				0.27	11	7	3	3	0.70	0	2	6	0	1	0	0.27	6	33	3	0	4	1	282.19	
2	(default package)		0.00	0.71	2	0.00	5	5	2	2	0.00	0	0	5	0	1	0	0.00	5	43	9	0	9	0	362.355	79.42691
		cal				0.00	10	10	5	5	0.00	0	0	5	0	1	0	0.00	5	30	0	1	9	1	413.95	
		divConstrint				0.00	0	0	0	0	0.00	0	0	0	0	0	0	0.00	0	4	0	0	0	0	310.76	
3	(default package)		0.00	0.71	2	0.38	8	3	1	1	0.37	0	6	9	1	1	0	0.20	11	93	18	1	7	1	593.77	64.13213
		Student				0.75	17	6	2	2	0.74	0	6	8	1	1	0	0.39	9	66	9	0	6	1	482.54	
		asp				0.00	0	0	1	1	0.00	0	0	1	0	0	0	0.00	2	16	0	2	1	2	705	
4	(default package)		0.00	0.71	2	0.00	0	0	1	1	1.00	0	2	3	0	1	0	0.00	3	41	9	0	2	1	632.185	77.07439
		AccessPrivate				0.00	1	1	2	2	2.00	0	1	2	0	0	0	0.00	2	15	0	0	1	1	616.13	
		ClassPrivate				0.00	0	0	1	1	0.00	0	1	1	0	1	0	0.00	1	12	0	1	1	1	648.24	
5	(default package)		0.00	0.71	2	0.00	3	3	3	3	0.75	0	1	6	0	3	0	0.00	6	42	9	0	4	1	688.72	76.23861
		HelloWorld				0.00	3	3	3	3	0.00	0	0	3	0	3	0	0.00	3	17	0	1	4	1	680.59	
		MannerAspect				0.00	3	3	3	3	1.50	0	1	3	0	0	0	0.00	3	16	0	0	0	1	696.85	
6	(default package)		0.00	0.71	3	0.56	20	0	1	1	0.26	0	4	28	0	1	0	0.45	28	164	16	0	23	1	4141.13	44.8428
		MyCircle				0.88	48	0	2	2	0.44	0	2	17	0	1	0	0.65	17	90	4	0	13	1	729.53	
		MyPoint				0.80	13	0	2	2	0.33	0	2	10	0	0	0	0.71	10	55	3	0	9	1	1023.86	
		asp				0.00	0	0	1	1	0.00	0	0	1	0	0	0	0.00	1	12	0	1	1	1	10670	
7	(default package)		0.00	0.71	2	0.13	8	6	3	3	1.02	0	3	10	0	1	0	0.13	12	68	9	0	10	1	1097.35	66.01058

		AccountSimple				0.27	11	7	3	3	0.70	0	2	6	0	1	0	0.27	7	36	0	0	4	1	1084.32	
		SafeWithdrawal				0.00	6	6	4	4	1.33	0	1	4	0	0	0	0.00	5	20	0	0	6	1	1110.38	
8	(default package)		0.00	0.71	3	0.48	11	5	2	2	0.48	0	7	17	0	2	0	0.23	17	125	15	0	17	1	1230.637	55.55183
		Author				0.67	10	5	2	2	0.73	0	3	6	0	1	0	0.33	6	41	3	0	5	1	1127.8	
		Book				0.78	23	10	2	2	0.72	0	4	9	0	1	0	0.36	9	61	3	0	10	1	1206.64	
		asp_au				0.00	1	1	2	2	0.00	0	0	2	0	0	0	0.00	2	12	0	1	2	1	1357.47	
9	(default package)		0.00	0.71	2	0.00	3	3	3	3	0.67	0	1	6	0	1	0	0.00	7	38	9	0	6	1	1388.985	74.2122
		JFact				0.00	1	1	2	2	0.00	0	0	2	0	1	0	0.00	3	15	0	1	2	1	1330.68	
		ajFact				0.00	6	6	4	4	1.33	0	1	4	0	0	0	0.00	4	11	0	0	4	1	1447.29	
10	(default package)		0.00	0.71	2	0.00	1	1	2	2	0.50	0	1	4	0	1	0	0.00	4	41	9	0	3	1	1478.885	72.65511
		Test				0.00	1	1	2	2	1.00	0	1	2	0	1	0	0.00	2	16	0	1	1	1	1492.48	
		testAspect				0.00	1	1	2	2	0.00	0	0	2	0	0	0	0.00	2	14	0	0	2	1	1465.29	
11	(default package)		0.00	0.71	2	0.00	0	0	1	1	0.00	0	0	3	0	1	0	0.00	3	59	12	0	1	1	1657.335	66.16647
		LogUIAspect				0.00	1	1	2	2	0.00	0	0	2	0	0	0	0.00	2	13	0	0	0	1	1528.77	
		Test				0.00	0	0	1	1	0.00	0	0	1	0	1	0	0.00	1	28	3	1	1	1	1785.9	
12	(default package)		0.00	0.71	2	0.08	3	2	2	2	0.33	0	1	6	0	1	0	0.08	6	58	9	0	4	1	1781.3	66.06831
		AspectTest				0.00	1	1	2	2	0.00	0	0	2	0	0	0	0.00	2	19	0	0	0	1	1748.85	
		Test				0.17	5	4	3	3	0.67	0	1	4	0	1	0	0.17	4	30	0	0	4	1	1813.75	
13	(default package)		0.00	0.71	2	0.00	9	9	4	4	0.60	0	2	9	0	1	0	0.00	9	67	9	0	9	1	1888.33	63.42805
		ATest				0.00	15	15	6	6	1.20	0	2	6	0	0	0	0.00	6	29	0	0	6	1	1860.3	
		Test				0.00	3	3	3	3	0.00	0	0	3	0	1	0	0.00	3	29	0	1	3	1	1916.36	
14	(default package)		0.00	0.71	2	0.00	3	3	3	3	1.00	0	2	6	0	1	0	0.00	7	60	9	0	3	1	1925.835	65.11342
		Asp				0.00	1	1	2	2	2.00	0	2	2	0	0	0	0.00	2	18	0	0	0	1	1869.63	
		Test1				0.00	6	6	4	4	0.00	0	0	4	0	1	0	0.00	5	33	0	1	3	1	1982.04	
15	(default package)		0.00	0.71	2	0.41	11	0	2	2	1.25	0	4	13	0	1	0	0.30	17	134	15	0	13	1	2166.665	51.48408
		MyTime				0.82	22	0	2	2	0.50	0	3	11	0	1	0	0.60	14	90	6	0	10	1	2085.82	
		asp3				0.00	1	1	2	2	2.00	0	1	2	0	0	0	0.00	3	33	0	1	3	1	2247.51	

16	(default package)		0.00	0.71	2	0.00	2	2	2	2	0.00	0	0	5	0	1	0	0.00	5	40	9	0	2	1	2190.285	71.01286
		asp				0.00	1	1	2	2	0.00	0	0	2	0	0	0	0.00	2	13	0	0	0	1	2161.71	
		hello				0.00	3	3	3	3	0.00	0	0	3	0	1	0	0.00	3	18	0	1	2	1	2218.86	
17	(default package)		0.00	0.71	5	0.47	1	0	1	1	0.23	2	6	11	0	1	0	0.47	11	99	16	1	11	1	2344.894	55.97708
		Rectangle				1.00	0	0	1	1	0.00	0	2	2	0	0	0	1.00	2	20	2	1	3	1	2343.33	
		Shape				0.33	2	1	2	2	0.50	2	1	3	0	0	0	0.33	3	21	3	1	1	1	2341.53	
		TestShape				0.00	0	0	1	1	0.00	0	0	1	0	1	0	0.00	1	12	0	1	1	1	2324.12	
		Triangle				1.00	0	0	1	1	0.67	0	3	2	0	0	0	1.00	2	21	2	1	3	1	2352.94	
		aasp				0.00	3	3	3	3	0.00	0	0	3	0	0	0	0.00	3	17	0	1	3	1	2362.55	
18	com		0.00	0.71	2	0.00	1	1	2	2	0.50	0	1	4	1	1	0	0.00	4	46	9	0	1	1	2420.37	68.2293
		HelloWorld1				0.00	1	1	2	2	0.00	0	0	2	0	1	0	0.00	2	14	0	1	1	1	2391.43	
		LogAspect				0.00	1	1	2	2	1.00	0	1	2	1	0	0	0.00	2	15	0	0	0	1	2449.31	
19	(default package)		0.00	0.71	4	0.50	1	0	1	1	0.10	1	3	13	0	1	0	0.43	13	96	17	0	11	1	2497.695	56.14732
		Point				1.00	4	0	1	1	0.40	1	2	6	0	0	0	0.73	6	33	4	0	4	1	2488	
		Point3D				1.00	0	0	1	1	0.00	0	1	4	0	0	0	1.00	4	27	4	0	4	1	2468.65	
		TestPoint3D				0.00	0	0	1	1	0.00	0	0	1	0	1	0	0.00	1	14	0	1	1	1	2507.37	
		as				0.00	1	1	2	2	0.00	0	0	2	0	0	0	0.00	2	16	0	1	2	1	2526.76	
20	(default package)		0.00	0.71	2	0.00	0	0	1	1	0.00	0	0	2	0	1	0	0.00	2	71	9	1	2	1	2614.53	60.79662
		LinkedListDemo				0.00	0	0	1	1	0.00	0	0	1	0	1	0	0.00	1	44	0	1	1	1	2507.37	
		asp				0.00	0	0	1	1	0.00	0	0	1	0	0	0	0.00	1	12	0	1	1	1	2721.69	
21	(default package)		0.00	0.71	2	0.00	1	1	2	2	0.50	0	1	4	0	1	0	0.00	5	108	15	1	3	1	2943.725	53.38483
		a				0.00	1	1	2	2	0.00	0	0	2	0	0	0	0.00	3	17	0	1	2	1	2770.7	
		m				0.00	1	1	2	2	1.00	0	1	2	0	1	0	0.00	2	61	6	2	1	1	3116.75	
22	(default package)		0.00	0.71	2	0.30	3	1	1	1	0.31	0	2	6	0	1	0	0.20	6	59	9	0	4	1	3106.8	62.89888
		Circle				0.60	6	2	2	2	0.63	0	2	5	0	1	0	0.40	5	33	0	0	3	1	3086.89	
		aspectest				0.00	0	0	1	1	0.00	0	0	1	0	0	0	0.00	1	18	0	1	1	1	3126.71	
23	test1_ao.src.hotel		0.09	0.64	11	0.32	24	17	3	3	0.61	0	176	73	12	7	1	0.20	84	2683	135	1	45	0	6835.452	-2.80936

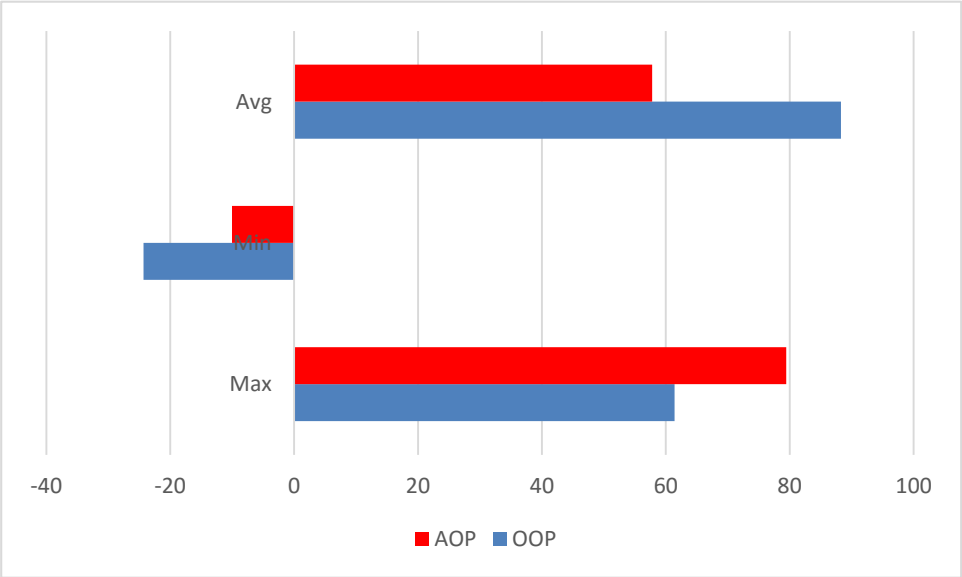
		CurrentCustomer				0.42	49	32	6	5	0.94	0	20	12	1	1	1	0.33	14	351	14	1	7	1	5783.36	
		CustomerServices				0.67	##	80	6	6	0.98	0	37	17	1	1	0	0.22	18	594	25	1	14	1	5966.03	
		DBTable_1				0.00	0	0	0	0	0.00	0	4	0	2	0	0	0.00	0	44	0	0	0	0	5804.82	
		Frame4all				0.00	3	3	3	3	0.00	0	0	3	0	1	0	0.00	4	73	21	2	1	1	5740.48	
		GuestData				0.14	20	19	6	5	0.96	0	13	7	0	0	0	0.14	7	184	6	0	5	1	6030.66	
		LogginAs				0.00	6	6	4	4	0.00	0	0	4	0	0	0	0.00	5	35	2	1	0	1	6149.36	
		Login				0.17	34	32	7	6	0.98	0	13	9	1	1	0	0.11	11	232	17	1	6	1	6637.64	
		MDI_AOP				0.65	37	19	3	3	0.97	0	43	11	0	1	0	0.33	12	579	14	1	7	1	8249.28	
		MySQL_DB				0.00	0	0	1	1	0.00	0	5	1	5	1	0	0.00	1	25	0	2	0	1	7803.83	
		NewCustomer				1.00	6	2	2	2	0.91	0	27	5	1	0	0	0.60	6	277	7	1	3	1	8484.23	
		SignUp				0.50	5	4	3	2	0.98	0	14	4	1	1	0	0.50	6	198	13	2	2	1	8540.28	
	imagegui		0.00	0.71	5	0.41	28	20	3	3	0.83	0	16	41	2	3	0	0.15	43	337	41	0	31	1	12463.41	27.44579
		ImagePanel				0.10	9	8	4	4	0.75	0	1	5	0	1	0	0.10	6	35	4	0	3	1	8461.82	
		JScrollPane2				0.65	43	31	3	3	0.80	0	4	11	0	0	0	0.22	11	56	0	0	9	1	8652.51	
		JTable2				0.65	43	31	3	3	0.80	0	4	11	0	0	0	0.22	11	56	0	0	9	1	8663.74	
		LineNumbering				0.00	3	3	3	3	1.00	0	3	3	2	2	0	0.00	4	64	5	1	1	1	9057.85	
		TransparentPanel				0.65	43	31	3	3	0.80	0	4	11	0	0	0	0.22	11	56	0	0	9	1	9001.43	
	resturantmanagement_		0.00	0.71	14	0.63	44	17	3	2	0.77	0	301	135	0	7	0	0.52	272	6832	308	1	92	1		
24		CategoryDetails				1.00	5	0	2	1	0.77	0	12	6	0	0	0	1.00	9	209	8	1	4	1	9806.21	
		Connect				0.00	0	0	1	1	0.00	0	1	1	0	1	0	0.00	1	21	0	2	0	1	9476.57	
		Delivery_view				0.60	6	2	3	2	0.85	0	10	5	0	1	0	0.40	11	273	18	2	2	2	10537.98	
		Employee				0.67	9	3	3	2	0.77	0	24	6	0	0	0	0.67	15	331	13	1	3	2	10940.58	
		KitchenTable				0.65	##	92	8	5	0.95	0	41	21	0	1	0	0.34	27	785	22	1	14	1	11924.98	
		Login				0.29	17	13	4	4	0.96	0	14	7	0	1	0	0.19	15	265	19	1	6	2	11553.3	
		MainMenu				0.52	##	64	10	9	0.80	0	38	29	0	1	0	0.42	43	1425	22	2	18	1	12087.99	
		Meals				0.71	11	1	3	2	0.78	0	18	7	0	0	0	0.71	14	290	9	1	3	2	13943.61	
		MenuMeal				0.68	##	51	5	5	0.90	0	48	23	0	1	0	0.40	82	1334	30	2	15	3	13447.58	
		OrdersView				0.60	6	2	2	2	0.85	0	10	5	0	1	0	0.40	11	272	18	2	2	2	15586.44	

		PaymentView1				1.00	20	12	3	1	0.94	0	41	8	0	0	0	0.61	14	540	9	1	4	1	14834.04	
		RegisterNewAccount				1.00	6	0	2	1	0.76	0	26	7	0	0	0	1.00	13	411	8	1	9	1	15215.72	
		RegisterRestaurant				1.00	5	0	2	1	0.75	0	16	6	0	0	0	1.00	11	264	8	1	6	1	15610.39	
		logger				0.17	5	4	3	3	0.67	0	2	4	0	0	0	0.17	6	60	2	1	6	1	15538.56	
25	libaray1		0.00	0.71	10	0.54	24	11	3	3	0.75	0	154	86	0	10	0	0.49	199	3106	53	1	64	1	16627.92	-10.0336
		Libaray1				0.00	0	0	1	1	0.00	0	0	1	0	1	0	0.00	1	7	0	0	1	1	16138.27	
		Log				0.05	20	19	6	6	0.83	0	2	7	0	0	0	0.05	9	88	0	0	12	1	16162.31	
		book1				0.80	20	0	3	2	0.80	0	23	10	0	1	0	0.73	29	478	4	1	6	2	16210.4	
		borrow1				1.00	7	0	2	1	0.76	0	12	7	0	1	0	0.95	12	221	3	2	3	1	16018.14	
		first				0.44	44	22	5	4	0.95	0	12	12	0	1	0	0.44	12	191	3	1	10	1	16294.59	
		logger				0.36	24	20	5	4	0.93	0	10	8	0	2	0	0.25	11	165	5	1	5	1	16169.75	
		return1				1.00	5	0	2	1	0.78	0	8	5	0	1	0	0.90	6	138	3	2	1	1	16583.59	
		searchb				0.78	17	0	3	2	0.80	0	24	9	0	1	0	0.69	29	454	8	1	6	3	16993.91	
		searchs				0.46	52	26	6	5	0.82	0	32	13	0	1	0	0.42	45	636	11	1	10	3	17793.39	
		stu				0.49	58	25	6	5	0.82	0	31	14	0	1	0	0.46	45	655	7	1	10	3	17914.85	

جدول 6-3 القياسات الثابتة Static Metrics على برامج المنهجية المفاهيمية

من خلال الجدول السابق نجد أن أصغر قيمة لمقياس أداء الصيانة هى: **-10.0336344** وأكبر قيمة هى : **79.42691128** بينما متوسط المقياس للبرامج هو : **57.78252199** .

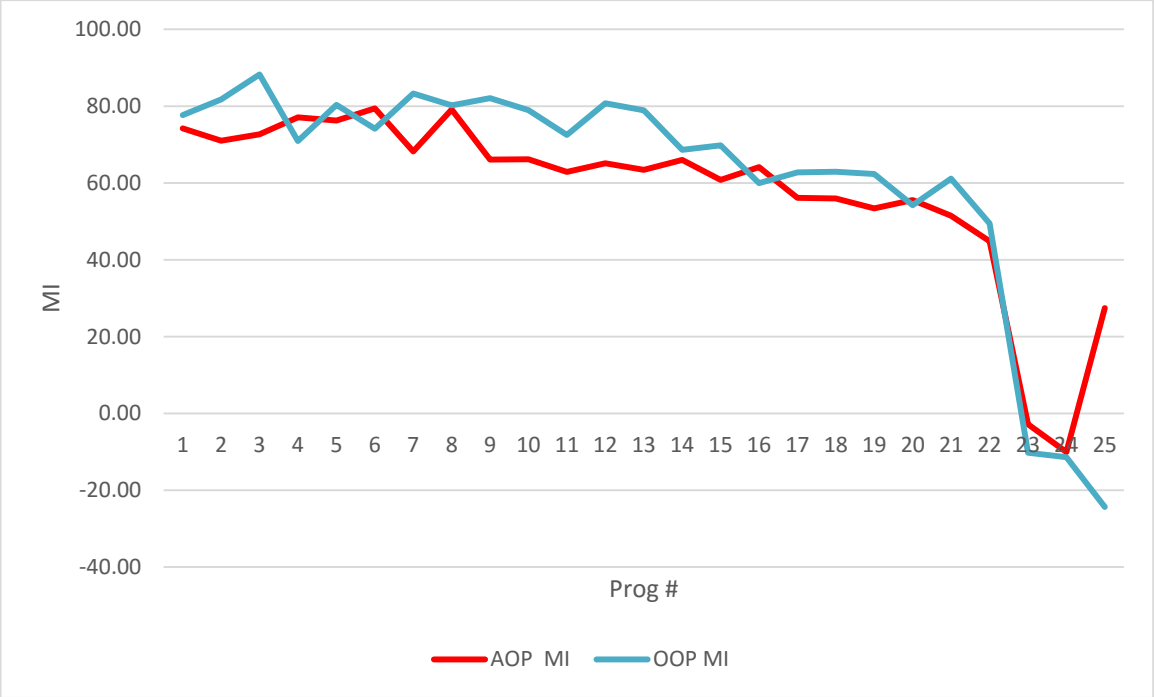
نلاحظ من البيانات السابقة ان متوسط مقياس أداء الصيانة بالنسبة للبرمجة الكائنية أكبر من مقياس الصيانة بالنسبة للبرمجة المفاهيمية، الشكل التالى 6-2 يوضح المقارنة بين قيم مقياس أداء الصيانة فى الطريقتين.



شكل 6-2 مقارنة مقياس أداء الصيانة بين البرمجة الكائنية والمفاهيمية

نلاحظ انه على الرغم من متوسط مقياس الاداء للبرامج كان أعلى بالنسبة للبرمجة الكائنية الا ان أعلى قيمة لمقياس اداء الصيانة كان فى صالح البرمجة المفاهيمية حيث كانت أعلى قيمة فى البرمجة المفاهيمية هى 79.4 بينما كانت فى البرمجة الكائنية 61.4 كما نلاحظ أيضا أن أدنى قيمة لمقياس أداء الصيانة كانت أيضا فى صالح البرمجة المفاهيمية أذ بلغت للبرمجة المفاهيمية -10 بينما كانت فى البرمجة الكائنية -24.3.

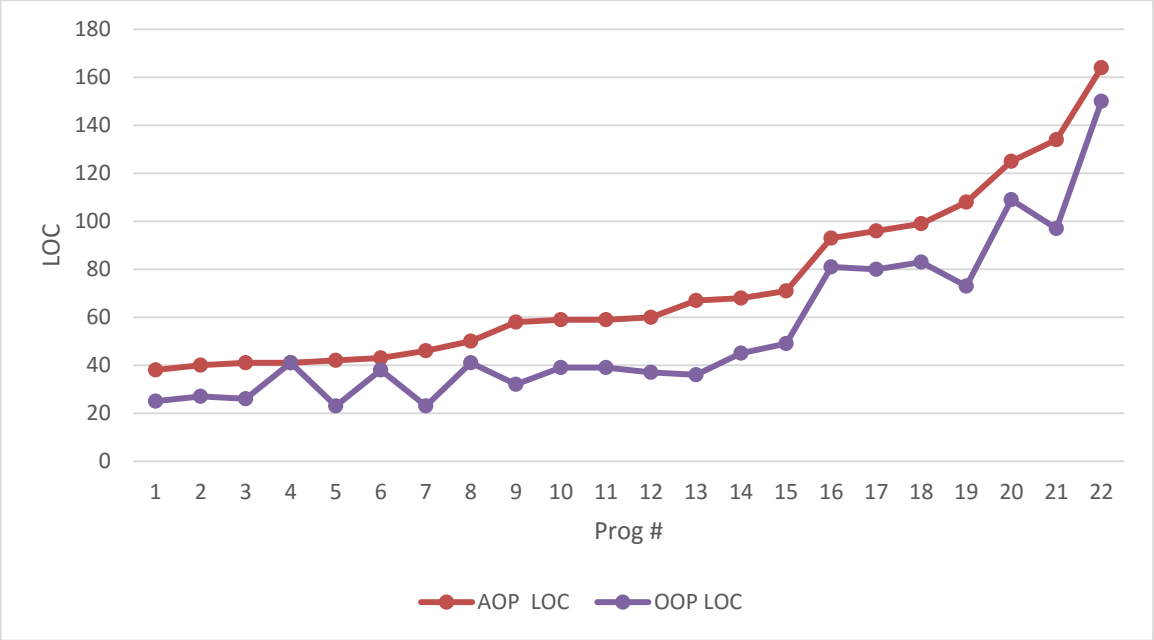
الشكل التالى 6-3 يمثل المقارنة بين قيمة مقياس أداء الصيانة للبرامج المكتوبة بأستخدام البرمجة المفاهيمية وتلك المكتوبة بأستخدام البرمجة الكائنية



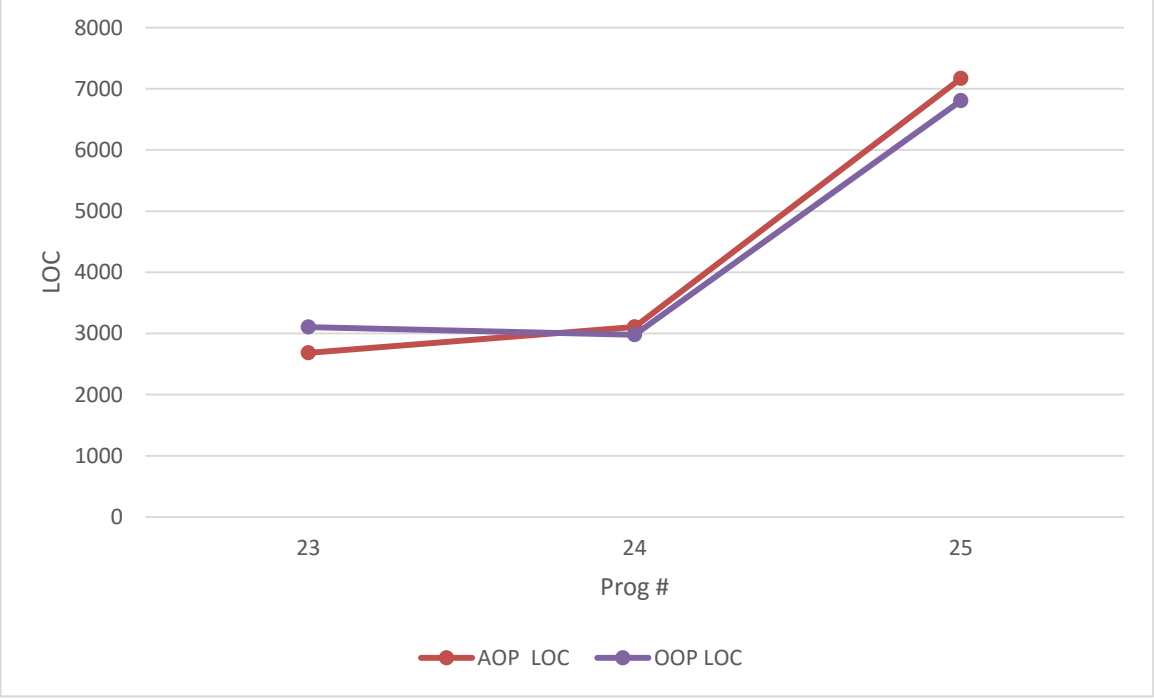
شكل 6-3 مقارنة مقياس أداء الصيانة للبرامج المطورة باستخدام البرمجة المفاهيمية والكائنية

بينما الشكل التالى 6-4 يمثل مقارنة بين حجم البرنامج LOC المكتوبة بأستخدام الطريقتين ما عدا البرامج 22 ، 23 و 24 نسبة لكبر حجمها والتى تظهر فى الشكل 6-5.

من الاشكال الثلاثة نلاحظ أنه بالنسبة لمعظم البرامج فأن أداء الصيانة أعلى للبرمجة الكائنية ما عدا البرامج 4 ، 6، 8، 16، 20 ، 23، 24، 25 . بالنسبة للبرامج الثلاثة الاخيرة نلاحظ أن عدد اسطر الكود المكتوبة فى البرنامجين تجاوزت الـ 2500 سطر كود، وبالتالى فأن قابلية الصيانة كانت أعلى بالنسبة للبرمجة المفاهيمية أما بالنسبة لبقية البرامج فنلاحظ ان عدد أسطر الكود للبرامج كان قليلا جدا ولم يتجاوز 200 سطر كود، والملاحظ أنه عندما تم تطويرها



شكل 6-4 مقارنة حجم البرامج بين الطريقتين ما عدا البرامج 22، 23 و 25

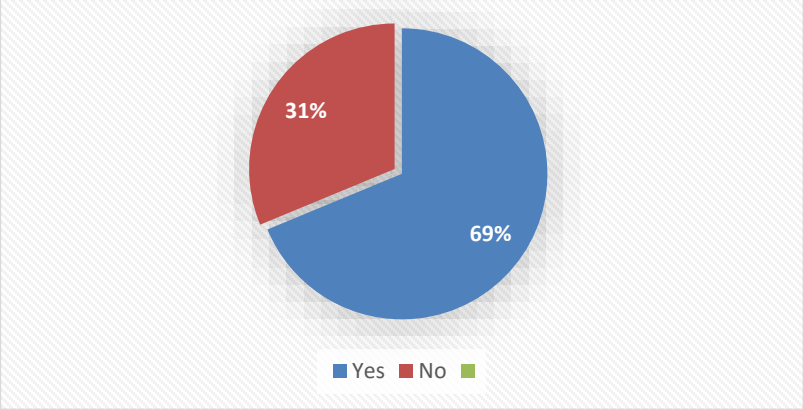


شكل 6-5 مقارنة حجم البرامج بين الطريقتين للبرامج 22، 23 و 25

بحيث ان الفارق بين عدد أسطر البرمجة المفاهيمية وعدد أسطر البرمجة الكائنية كان متقاربا فأنا قد حصلنا على نسبة صيانة أعلى بالنسبة للبرمجة المفاهيمية.

أما بالنسبة للاستبيان فانه أحتوى على سؤال مباشر عن قابلية الصيانة وهو:

فى النظام الذى قمت بتطويره ، أذا أردت أضافة متطلب جديد او التعديل على متطلب موجود ، هل من السهل القيام بذلك؟ أجاب 69% من المتطوعين بنعم بينما أجاب 31% من المتطوعين بلا. الشكل التالى 6-6 يوضح النسب



شكل 6-6 هل التعديل فى البرنامج سهل

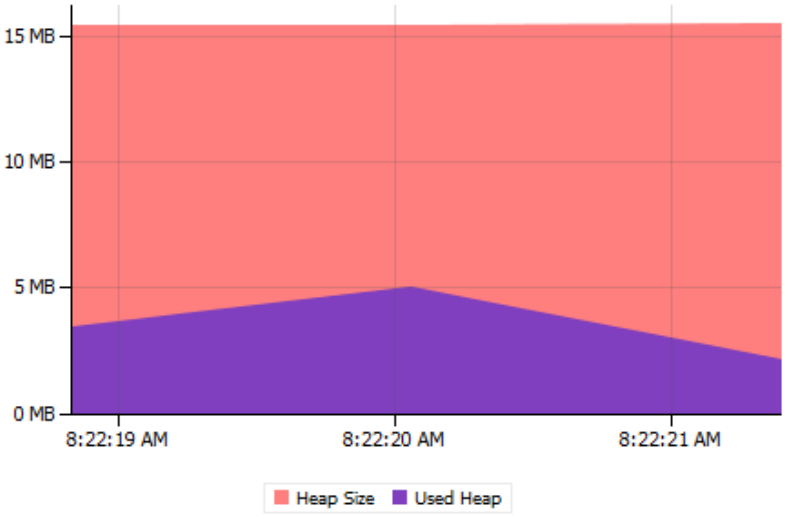
6.5 تأثير البرمجة المفاهيمية على الاداء Performance

الأداء كما تم ذكره يقصد به أستهلاك النظام للموارد المختلفة، ونسبة لمحدودية الموارد فى أجهزة الكمبيوتر فأن البرامج التى تستخدم هذه الموارد ينبغى أن تستهلكها بصورة مثالية، تمت

مقارنة أستهلاك الموارد للبرامج المكتوبة بأستخدام البرمجة المفاهيمية والبرمجة الكائنية لأختبار تأثير عملية النسيج Weaving على أداء البرنامج.

تم أختبار تأثير البرامج على الذاكرة من خلال مقارنة الحجم الكلى Heap Memeory الذى تم تعينة لماكينة الجافا الافتراضية JVM مع حجم الذاكرة المستخدم من قبل البرنامج فى فترة زمنية مقدارها ثلاثة ثوانى.

عندما يبدأ برنامج الجافا بتشغيل آلة الجافا الافتراضية Java Virtual Machine (JVM) فأنه يحصل علي بعض الذاكرة من نظام التشغيل يستخدمها لجميع اغراضه ، جزء من هذه الذاكرة تسمي (java heap memory) تقع عموما في اسفل مساحة عنوان الذاكرة وتزداد مساحتها صعوداً كلما انشأنا كائن جديد باستخدام مشغل جديد (operator) او اي وسيلة اخري ، الكائن الذي تم انشاؤه يتم تخصيص جزء له من ال(heap) وعندما يموت الكائن تعود مساحته التي كان يشغلها الي ذاكرة ال(heap) مرة اخري. الشكل التالى 6-8 يمثل التغير فى حجم الذاكرة Heap الحجم الكلى والحجم المستخد من قبل برنامج قيد التنفيذ فى فترة ثلاثة ثوانى.

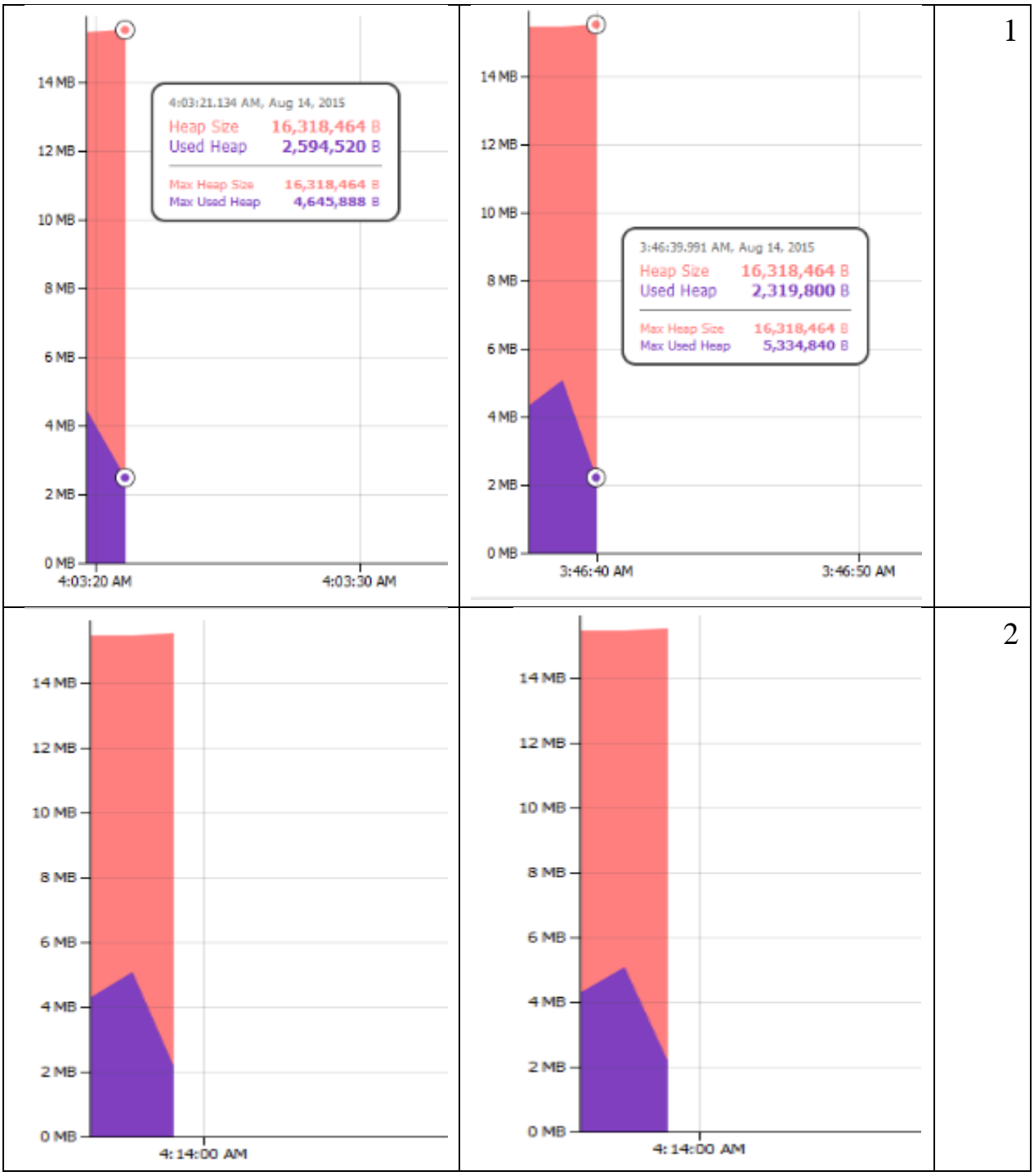


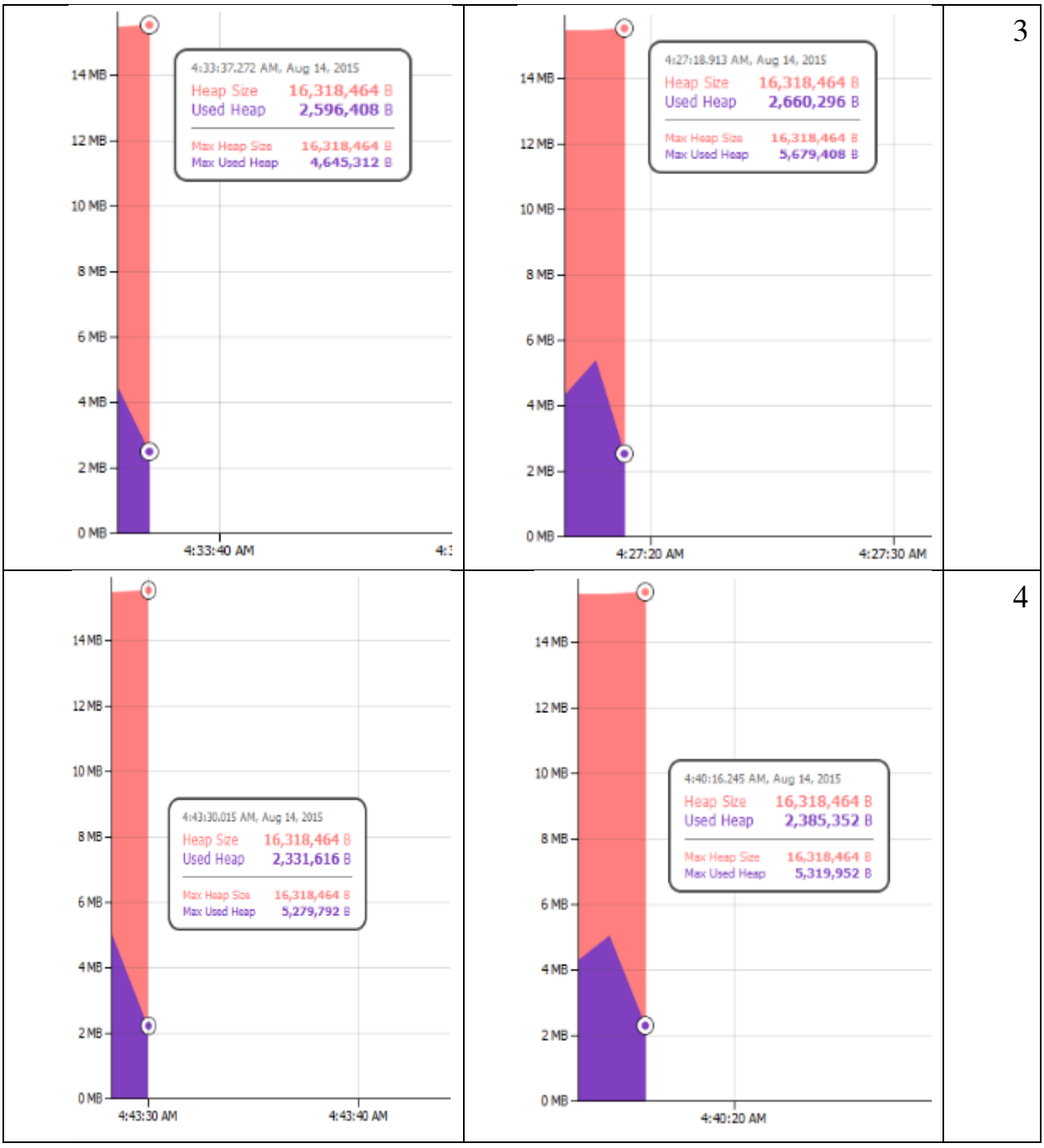
شكل 6-7 التغير فى حجم الذاكرة الكلى والمستخدم

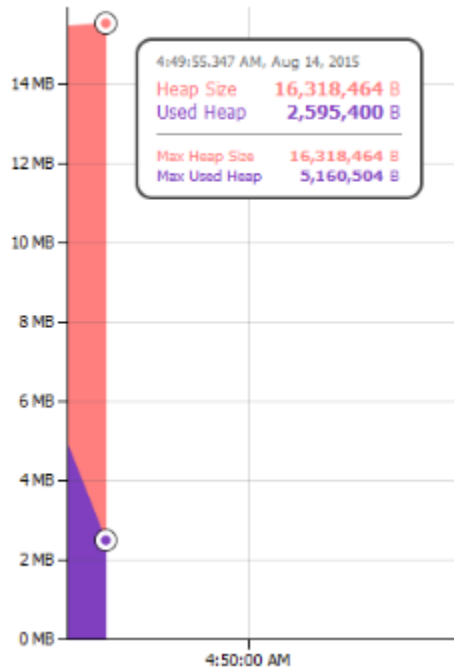
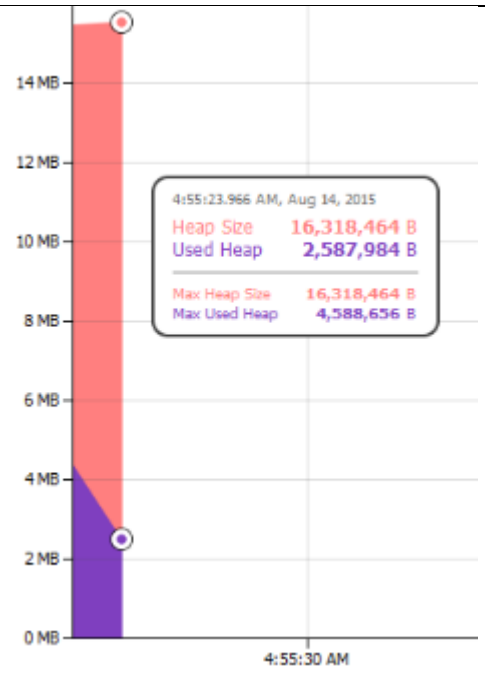
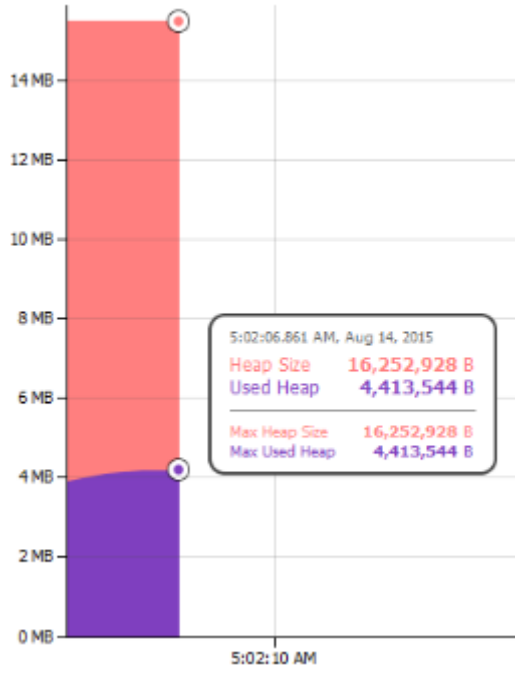
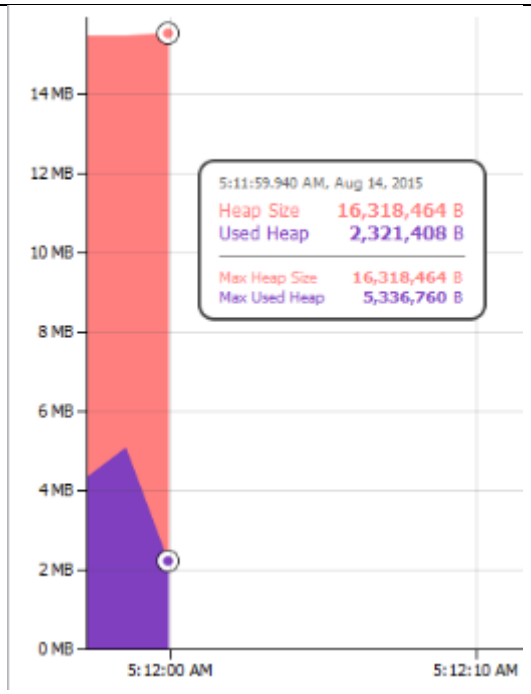
.

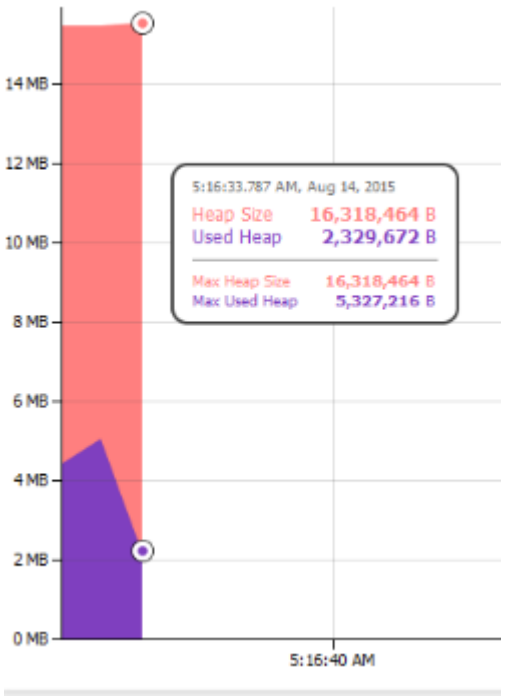
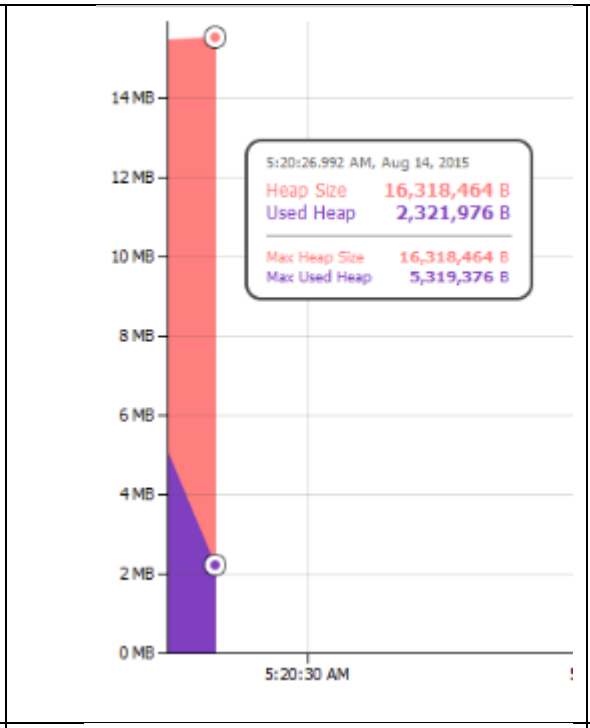
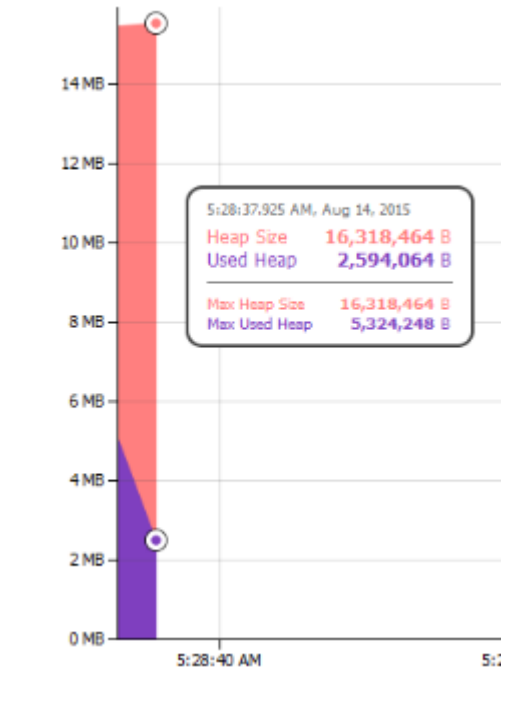
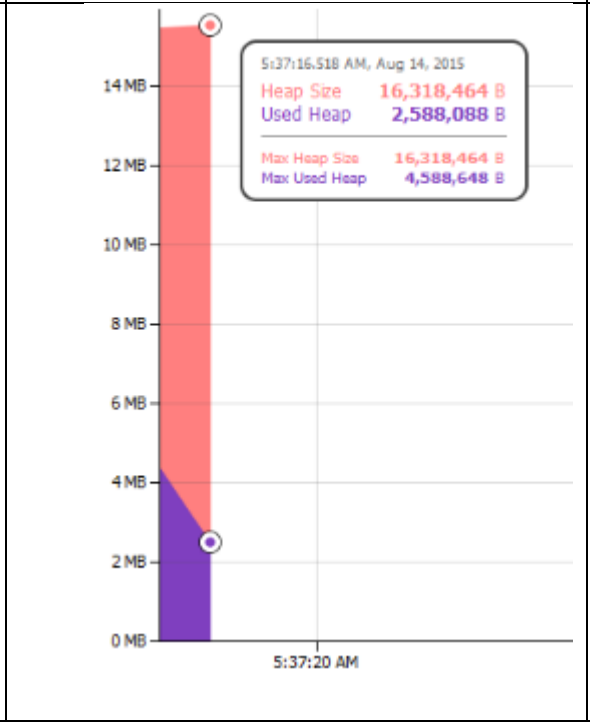
الجدول التالى يمثل الاشكال البيانية لقياس الاداء للبرامج المكتوبة بأستخدام البرمجة المفاهيمية والبرمجة الكائنية، تم قراءة التغير فى الذاكرة Heap فى فترة زمنية ثلاثة ثوانى

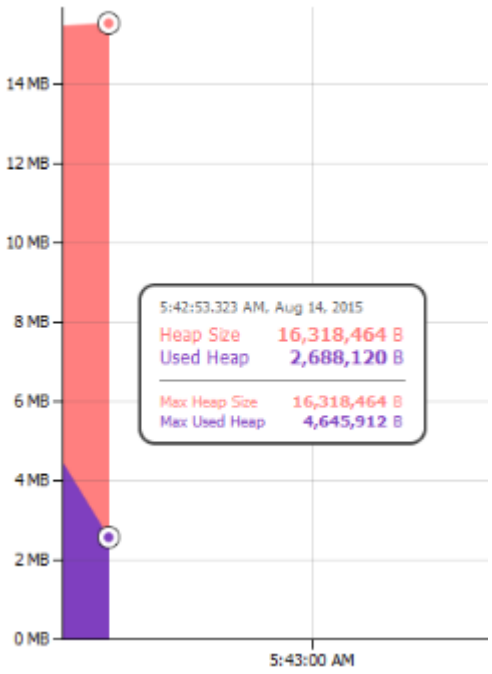
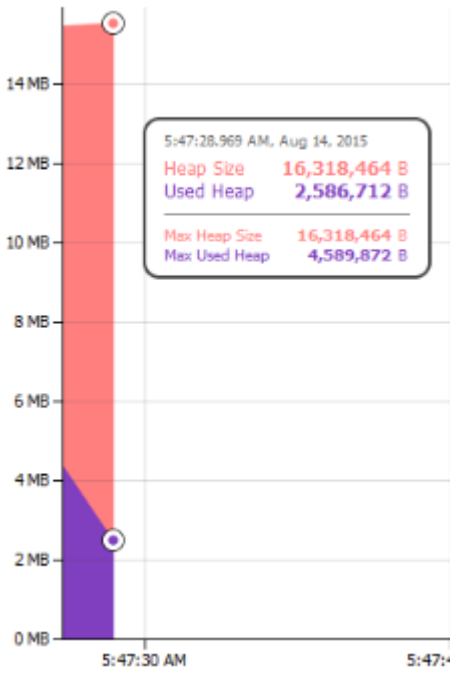
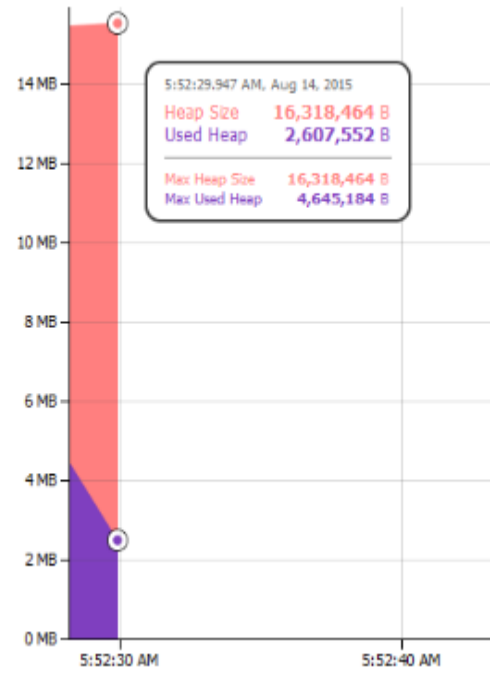
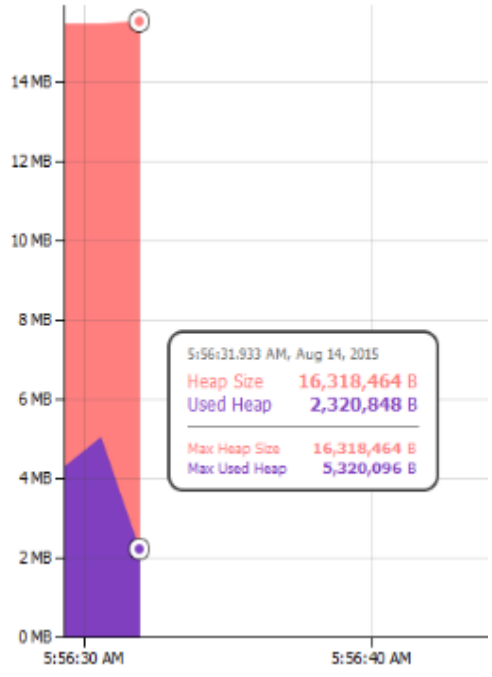
رقم البرنامج	OOP	AOP

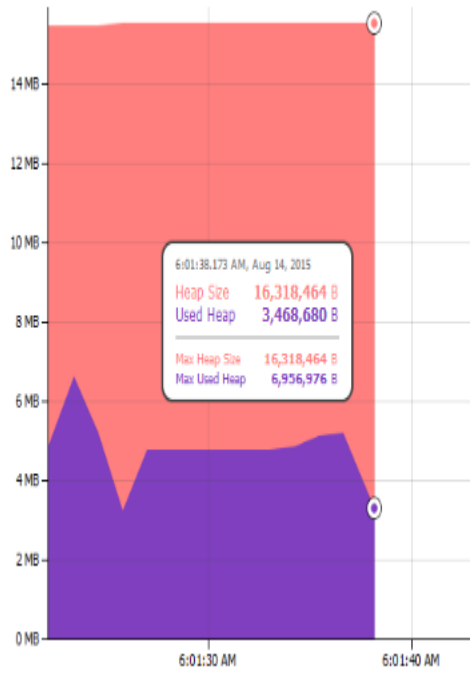
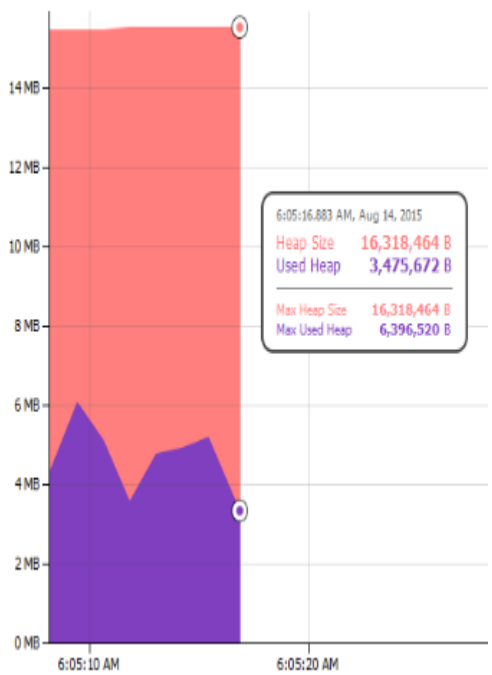
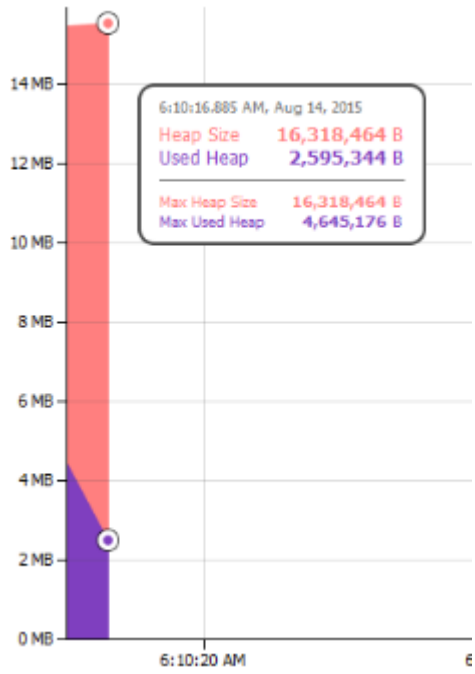
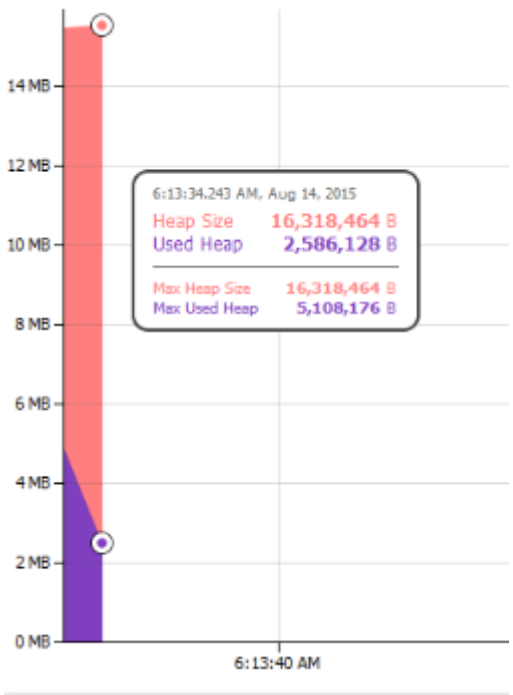

14 MB
12 MB
10 MB
8 MB
6 MB
4 MB
2 MB
0 MB
4:03:21.134 AM, Aug 14, 2015
Heap Size 16,318,464 B
Used Heap 2,594,520 B
Max Heap Size 16,318,464 B
Max Used Heap 4,645,888 B
4:03:20 AM
4:03:30 AM
3:46:39.991 AM, Aug 14, 2015
Heap Size 16,318,464 B
Used Heap 2,319,800 B
Max Heap Size 16,318,464 B
Max Used Heap 5,334,840 B
3:46:40 AM
3:46:50 AM
4:14:00 AM
4:14:00 AM
1
2

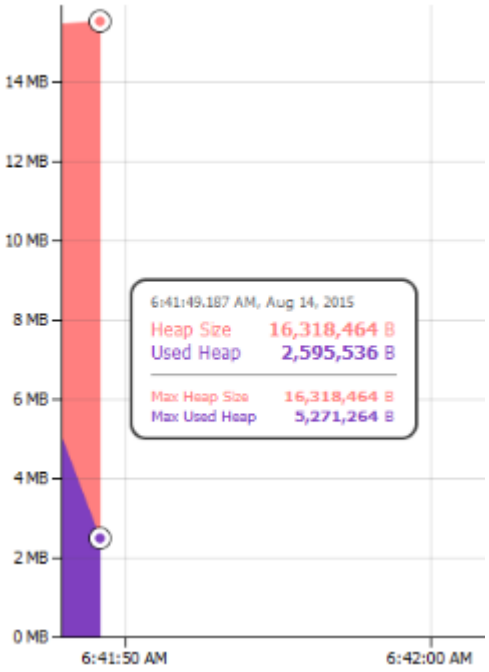
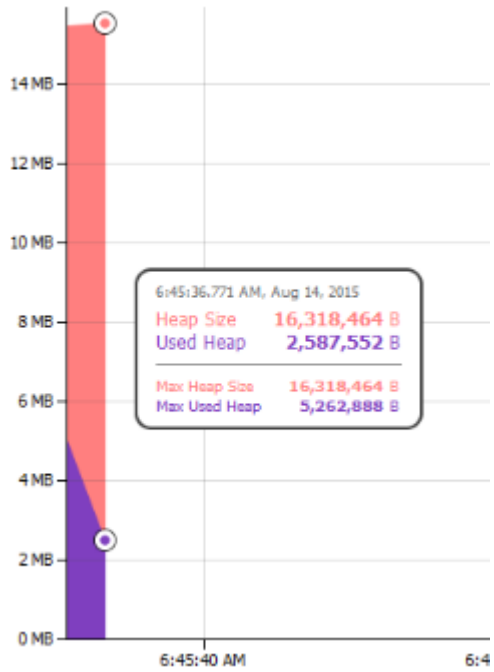
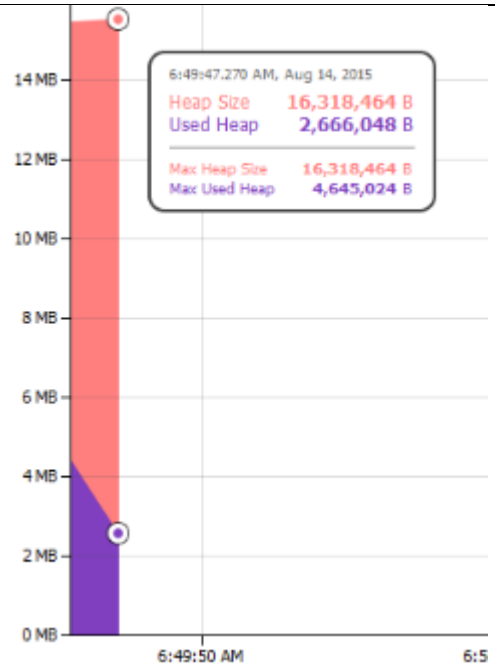
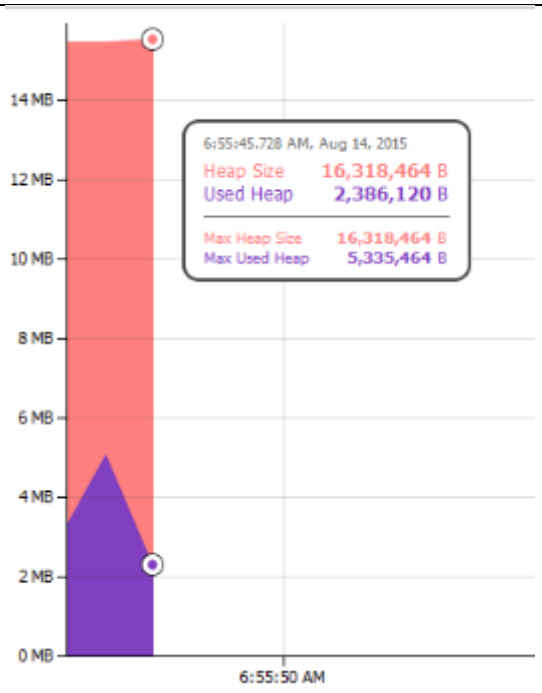

4:33:37.272 AM, Aug 14, 2015
Heap Size 16,318,464 B
Used Heap 2,596,408 B
Max Heap Size 16,318,464 B
Max Used Heap 4,645,312 B
14 MB
12 MB
10 MB
8 MB
6 MB
4 MB
2 MB
0 MB
4:33:40 AM
4:27:18.913 AM, Aug 14, 2015
Heap Size 16,318,464 B
Used Heap 2,660,296 B
Max Heap Size 16,318,464 B
Max Used Heap 5,679,408 B
4:27:20 AM
4:27:30 AM
3
4:43:30.015 AM, Aug 14, 2015
Heap Size 16,318,464 B
Used Heap 2,331,616 B
Max Heap Size 16,318,464 B
Max Used Heap 5,279,792 B
4:43:30 AM
4:43:40 AM
4:40:16.245 AM, Aug 14, 2015
Heap Size 16,318,464 B
Used Heap 2,385,352 B
Max Heap Size 16,318,464 B
Max Used Heap 5,319,952 B
4:40:20 AM
4

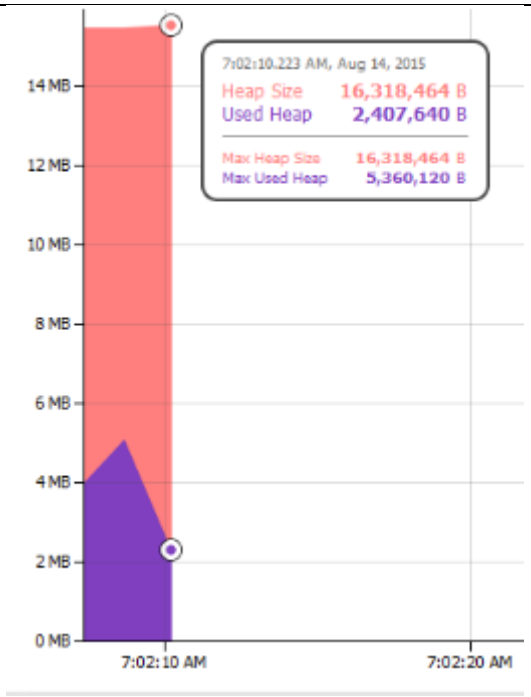
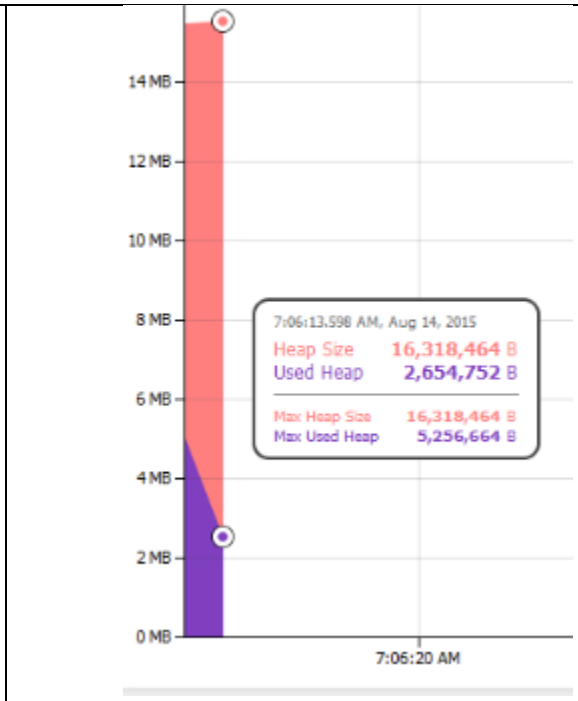
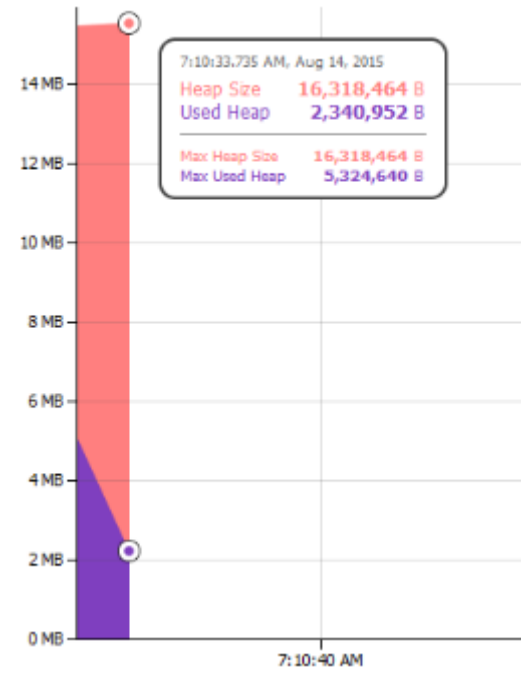
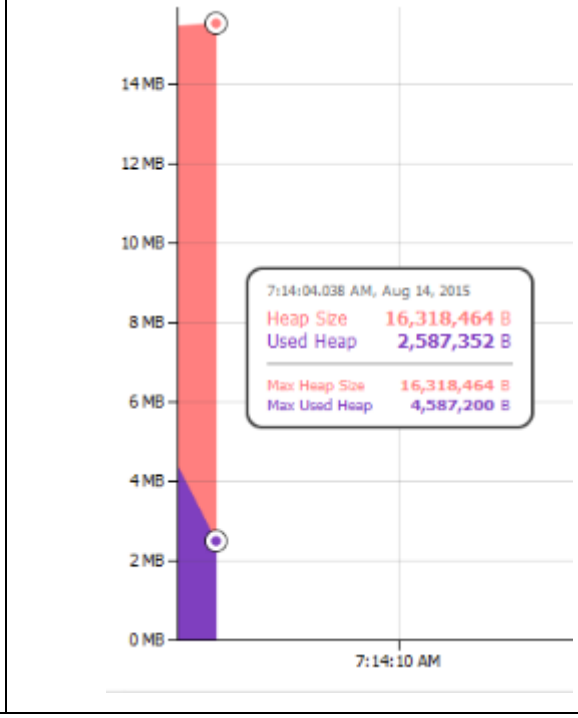
 4:49:55.347 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,595,400 B Max Heap Size 16,318,464 B Max Used Heap 5,160,504 B 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 4:50:00 AM	 4:55:23.966 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,587,984 B Max Heap Size 16,318,464 B Max Used Heap 4,588,656 B 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 4:55:30 AM	5
 5:02:06.861 AM, Aug 14, 2015 Heap Size 16,252,928 B Used Heap 4,413,544 B Max Heap Size 16,252,928 B Max Used Heap 4,413,544 B 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 5:02:10 AM	 5:11:59.940 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,321,408 B Max Heap Size 16,318,464 B Max Used Heap 5,336,760 B 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 5:12:00 AM 5:12:10 AM	6

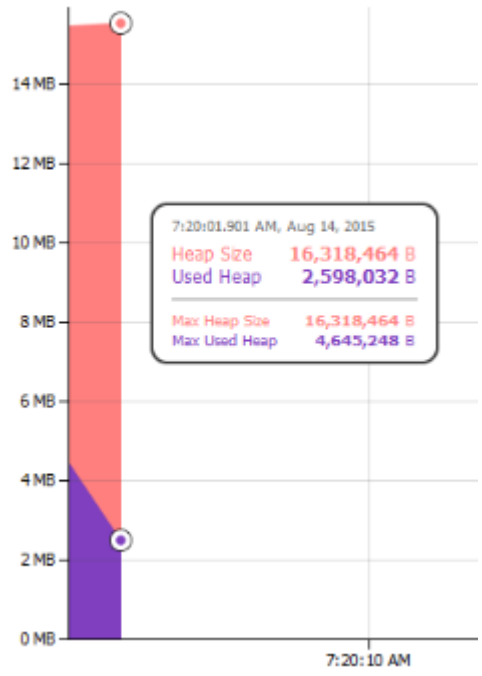
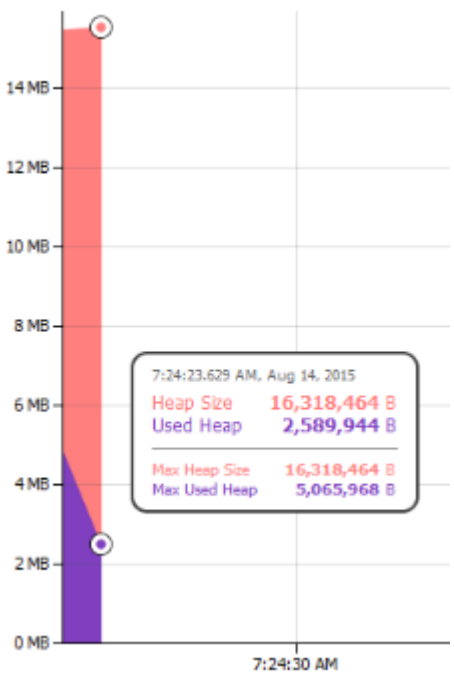
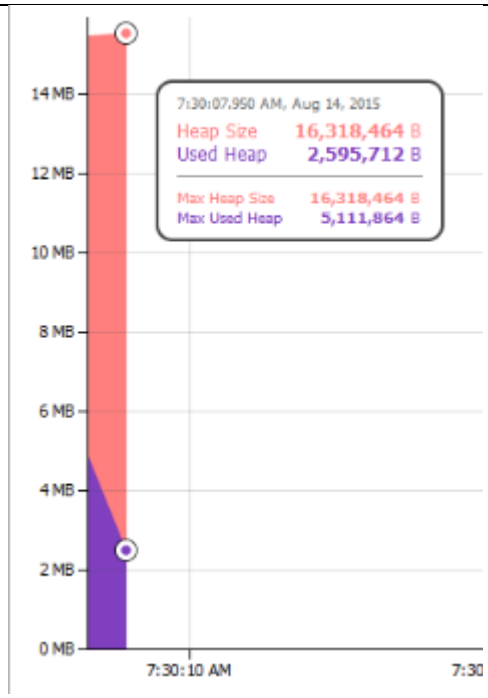
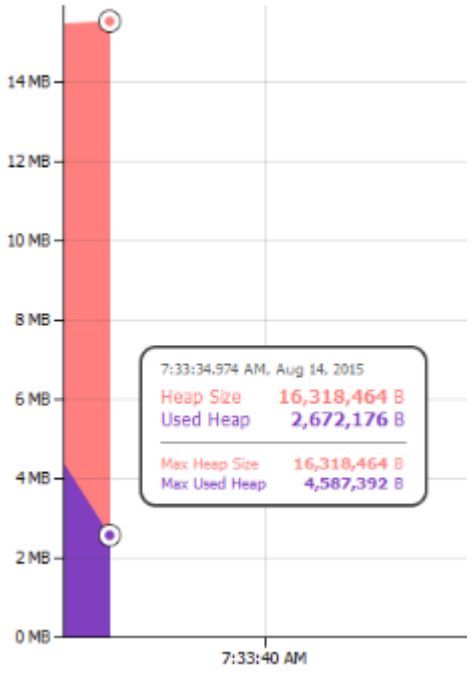
 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 5:16:40 AM 5:16:33.787 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,329,672 B Max Heap Size 16,318,464 B Max Used Heap 5,327,216 B	 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 5:20:30 AM 5:20:26.992 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,321,976 B Max Heap Size 16,318,464 B Max Used Heap 5,319,376 B	7
 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 5:28:40 AM 5:28:37.925 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,594,064 B Max Heap Size 16,318,464 B Max Used Heap 5,324,248 B	 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 5:37:20 AM 5:37:16.518 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,588,088 B Max Heap Size 16,318,464 B Max Used Heap 4,588,648 B	8

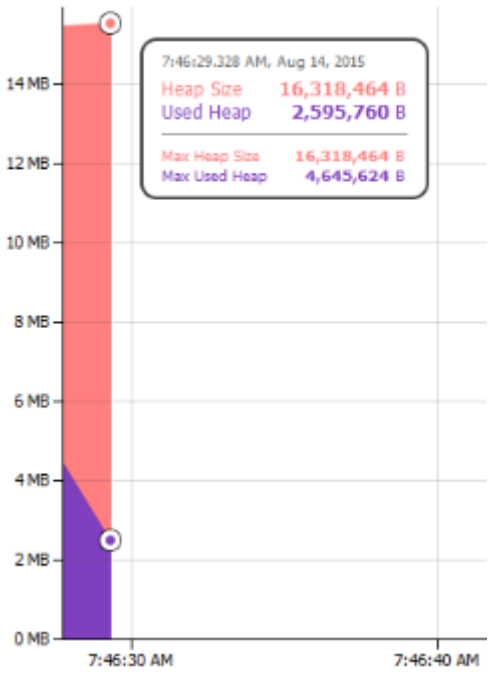
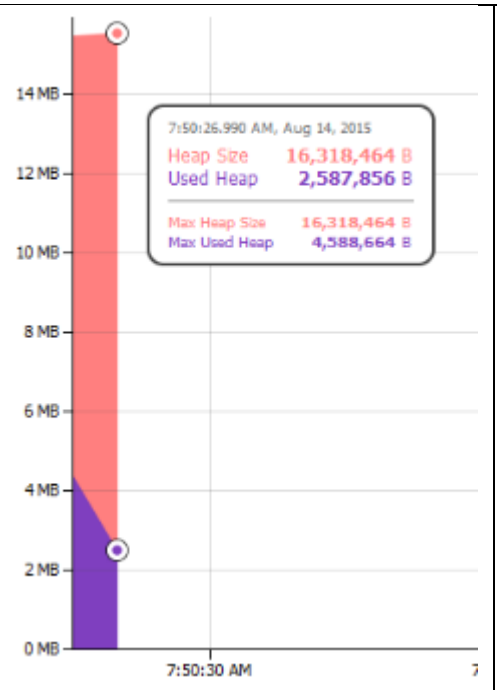
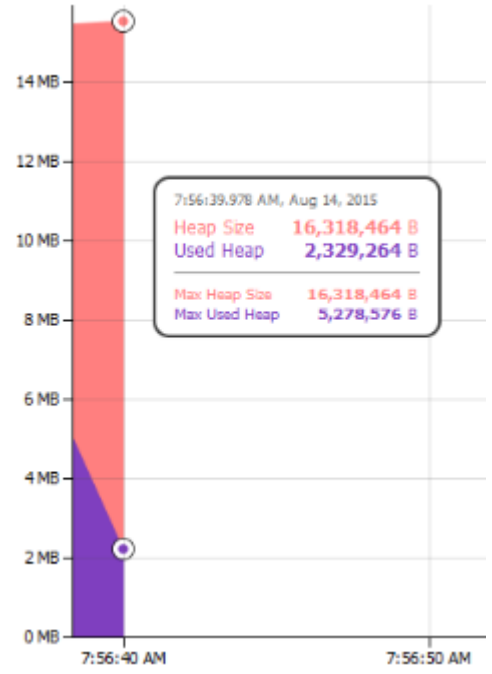
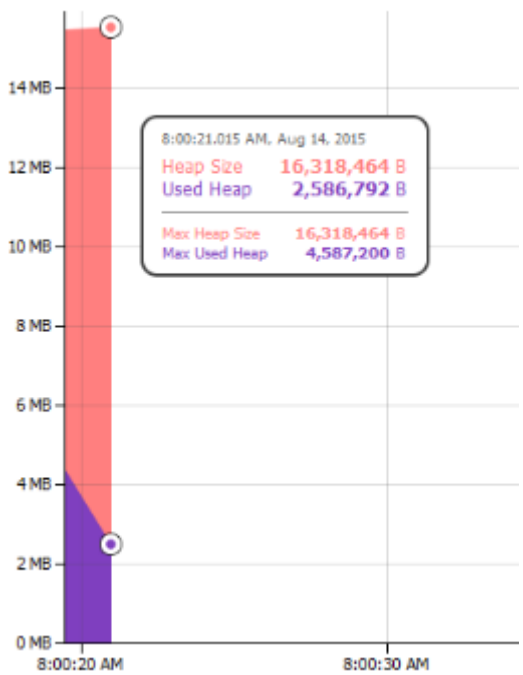
 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 5:43:00 AM 5:42:53.323 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,688,120 B Max Heap Size 16,318,464 B Max Used Heap 4,645,912 B	 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 5:47:30 AM 5:47:28.969 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,586,712 B Max Heap Size 16,318,464 B Max Used Heap 4,589,872 B	9
 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 5:52:30 AM 5:52:40 AM 5:52:29.947 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,607,552 B Max Heap Size 16,318,464 B Max Used Heap 4,645,184 B	 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 5:56:30 AM 5:56:40 AM 5:56:31.933 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,320,848 B Max Heap Size 16,318,464 B Max Used Heap 5,320,096 B	10

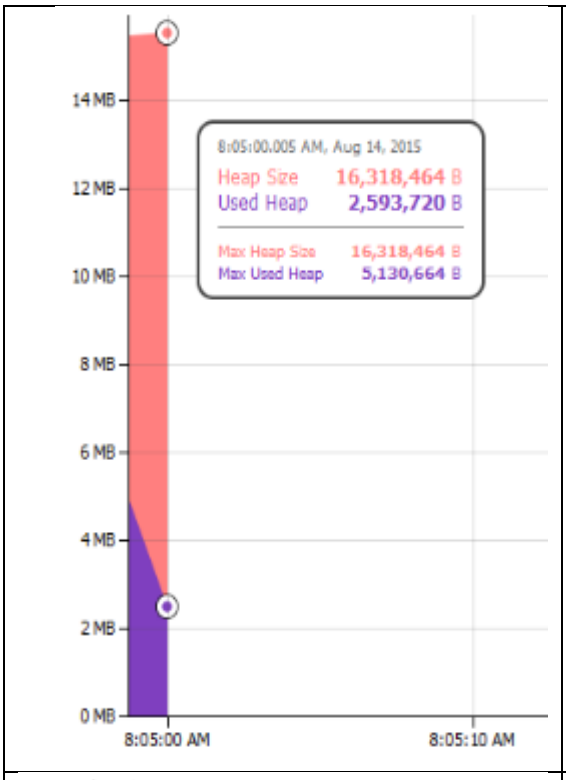
 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 6:01:38.173 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 3,468,680 B Max Heap Size 16,318,464 B Max Used Heap 6,956,976 B 6:01:30 AM 6:01:40 AM	 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 6:05:16.883 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 3,475,672 B Max Heap Size 16,318,464 B Max Used Heap 6,396,520 B 6:05:10 AM 6:05:20 AM	11
 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 6:10:16.885 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,595,344 B Max Heap Size 16,318,464 B Max Used Heap 4,645,176 B 6:10:20 AM	 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 6:13:34.243 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,586,128 B Max Heap Size 16,318,464 B Max Used Heap 5,108,176 B 6:13:40 AM	12

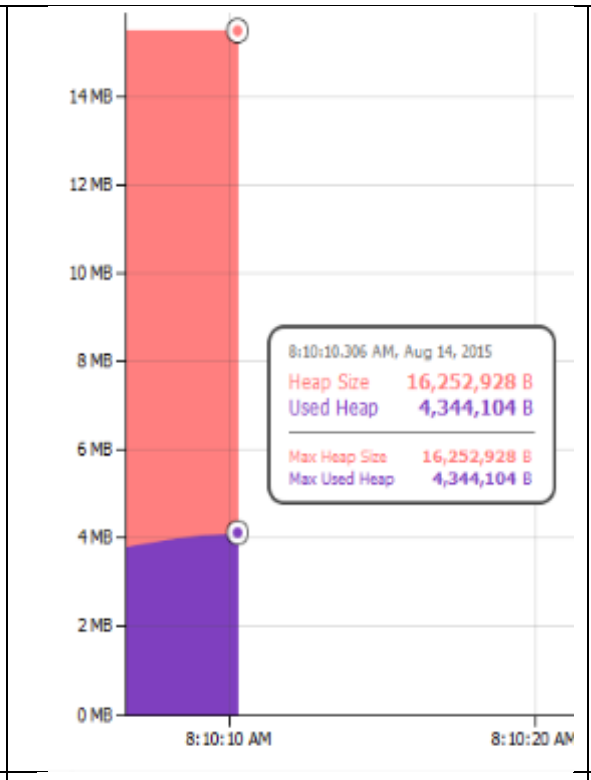
 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 6:41:50 AM 6:42:00 AM 6:41:49.187 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,595,536 B Max Heap Size 16,318,464 B Max Used Heap 5,271,264 B	 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 6:45:40 AM 6:4 6:45:36.771 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,587,552 B Max Heap Size 16,318,464 B Max Used Heap 5,262,888 B	13
 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 6:49:50 AM 6:5 6:49:47.270 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,666,048 B Max Heap Size 16,318,464 B Max Used Heap 4,645,024 B	 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 6:55:50 AM 6:55:45.728 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,386,120 B Max Heap Size 16,318,464 B Max Used Heap 5,335,464 B	14

 7:02:10.223 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,407,640 B Max Heap Size 16,318,464 B Max Used Heap 5,360,120 B 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 7:02:10 AM 7:02:20 AM	 7:06:13.598 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,654,752 B Max Heap Size 16,318,464 B Max Used Heap 5,256,664 B 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 7:06:20 AM	15
 7:10:33.735 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,340,952 B Max Heap Size 16,318,464 B Max Used Heap 5,324,640 B 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 7:10:40 AM	 7:14:04.038 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,587,352 B Max Heap Size 16,318,464 B Max Used Heap 4,587,200 B 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 7:14:10 AM	16

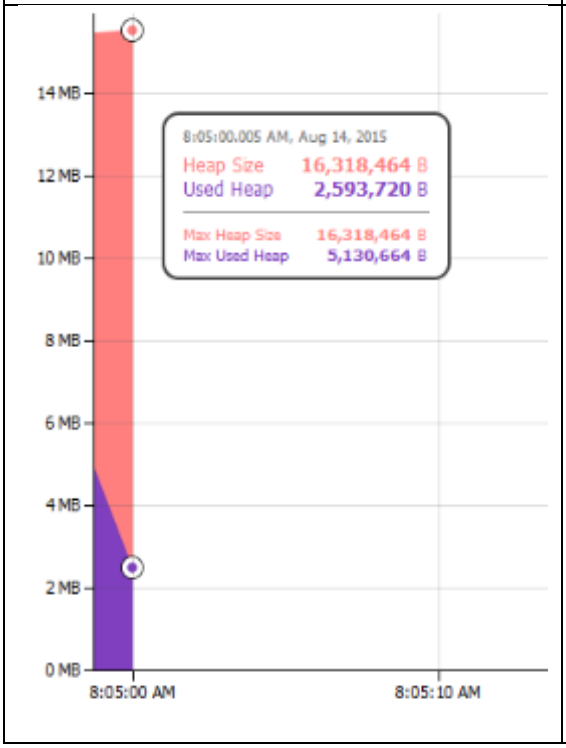
 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 7:20:01.901 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,598,032 B Max Heap Size 16,318,464 B Max Used Heap 4,645,248 B 7:20:10 AM	 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 7:24:23.629 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,589,944 B Max Heap Size 16,318,464 B Max Used Heap 5,065,968 B 7:24:30 AM	17
 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 7:30:07.950 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,595,712 B Max Heap Size 16,318,464 B Max Used Heap 5,111,864 B 7:30:10 AM 7:30	 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 7:33:34.974 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,672,176 B Max Heap Size 16,318,464 B Max Used Heap 4,587,392 B 7:33:40 AM	18

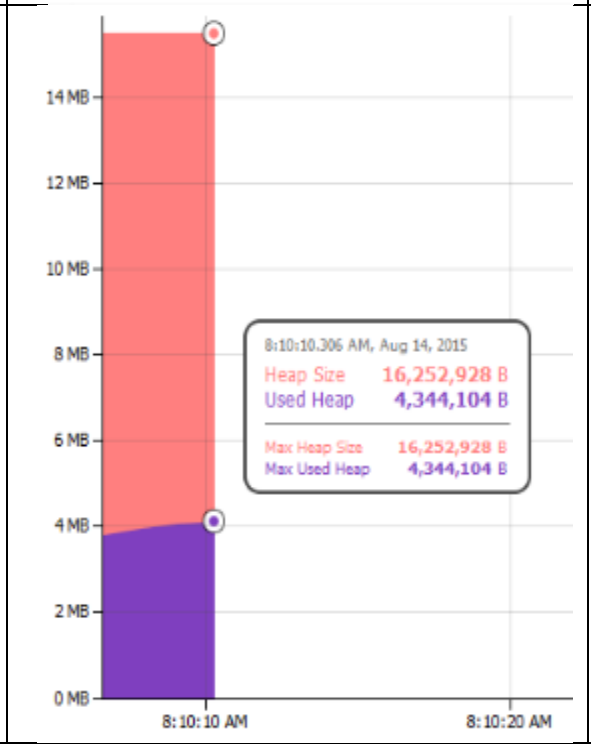
 7:46:29.328 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,595,760 B Max Heap Size 16,318,464 B Max Used Heap 4,645,624 B 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 7:46:30 AM 7:46:40 AM	 7:50:26.990 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,587,856 B Max Heap Size 16,318,464 B Max Used Heap 4,588,664 B 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 7:50:30 AM	19
 7:56:39.978 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,329,264 B Max Heap Size 16,318,464 B Max Used Heap 5,278,576 B 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 7:56:40 AM 7:56:50 AM	 8:00:21.015 AM, Aug 14, 2015 Heap Size 16,318,464 B Used Heap 2,586,792 B Max Heap Size 16,318,464 B Max Used Heap 4,587,200 B 14 MB 12 MB 10 MB 8 MB 6 MB 4 MB 2 MB 0 MB 8:00:20 AM 8:00:30 AM	20


14 MB
12 MB
10 MB
8 MB
6 MB
4 MB
2 MB
0 MB
8:05:00 AM
8:05:10 AM
8:05:00.005 AM, Aug 14, 2015
Heap Size 16,318,464 B
Used Heap 2,593,720 B
Max Heap Size 16,318,464 B
Max Used Heap 5,130,664 B

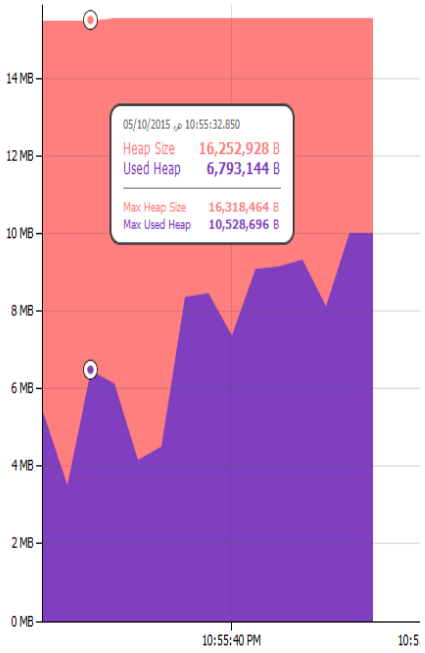

14 MB
12 MB
10 MB
8 MB
6 MB
4 MB
2 MB
0 MB
8:10:10 AM
8:10:20 AM
8:10:10.306 AM, Aug 14, 2015
Heap Size 16,252,928 B
Used Heap 4,344,104 B
Max Heap Size 16,252,928 B
Max Used Heap 4,344,104 B

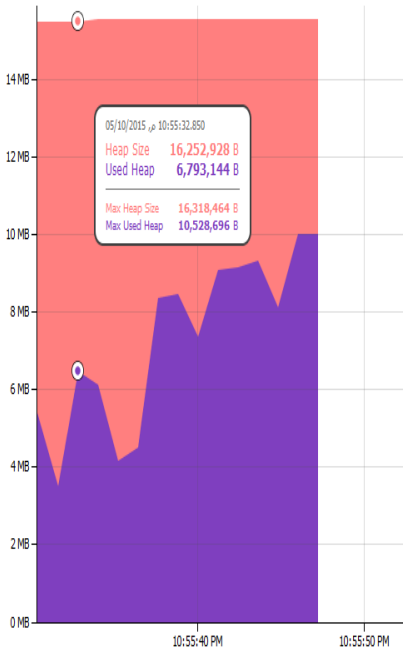
21


14 MB
12 MB
10 MB
8 MB
6 MB
4 MB
2 MB
0 MB
8:05:00 AM
8:05:10 AM
8:05:00.005 AM, Aug 14, 2015
Heap Size 16,318,464 B
Used Heap 2,593,720 B
Max Heap Size 16,318,464 B
Max Used Heap 5,130,664 B

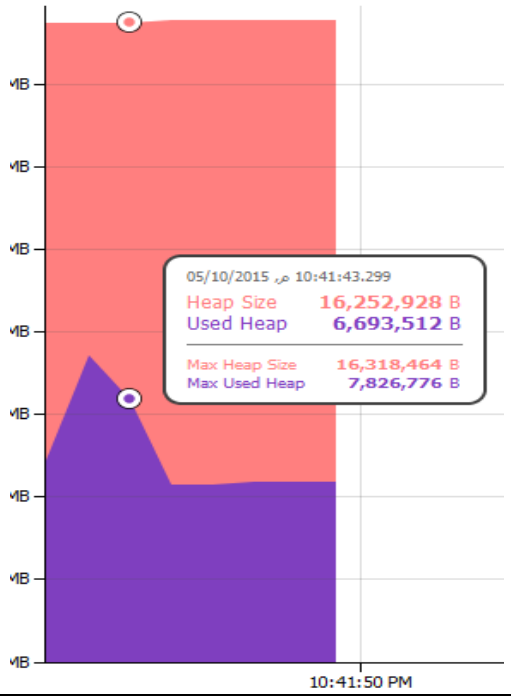

14 MB
12 MB
10 MB
8 MB
6 MB
4 MB
2 MB
0 MB
8:10:10 AM
8:10:20 AM
8:10:10.306 AM, Aug 14, 2015
Heap Size 16,252,928 B
Used Heap 4,344,104 B
Max Heap Size 16,252,928 B
Max Used Heap 4,344,104 B

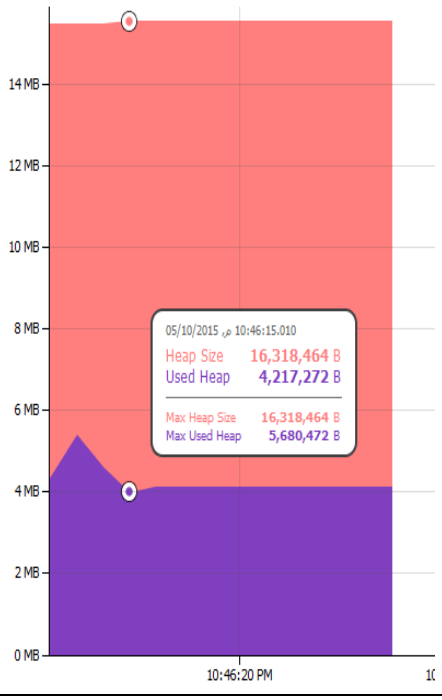
22


14 MB
12 MB
10 MB
8 MB
6 MB
4 MB
2 MB
0 MB
05/10/2015 م 10:55:32.850
Heap Size 16,252,928 B
Used Heap 6,793,144 B
Max Heap Size 16,318,464 B
Max Used Heap 10,528,696 B
10:55:40 PM
10:5

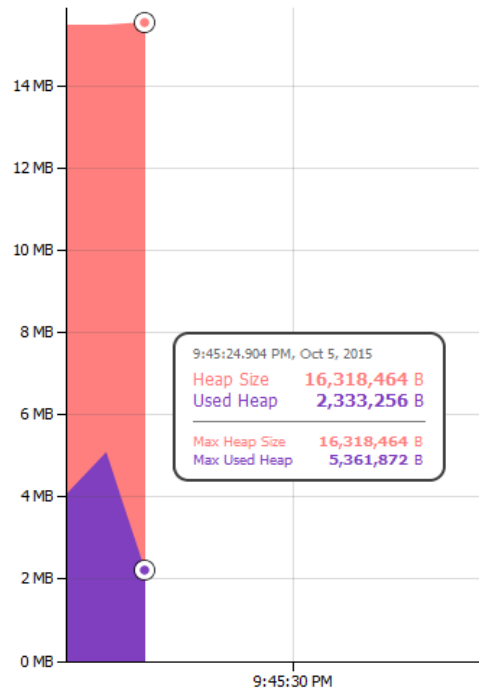
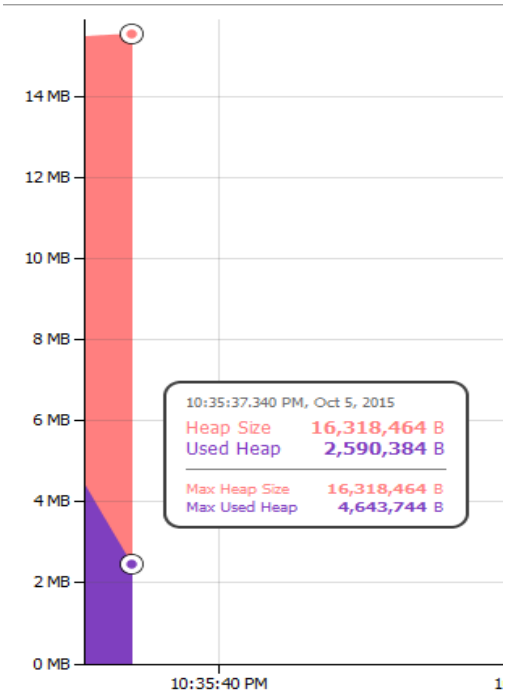

14 MB
12 MB
10 MB
8 MB
6 MB
4 MB
2 MB
0 MB
05/10/2015 م 10:55:32.850
Heap Size 16,252,928 B
Used Heap 6,793,144 B
Max Heap Size 16,318,464 B
Max Used Heap 10,528,696 B
10:55:40 PM
10:55:50 PM

23


MB
MB
MB
MB
MB
MB
MB
05/10/2015 م 10:41:43.299
Heap Size 16,252,928 B
Used Heap 6,693,512 B
Max Heap Size 16,318,464 B
Max Used Heap 7,826,776 B
10:41:50 PM

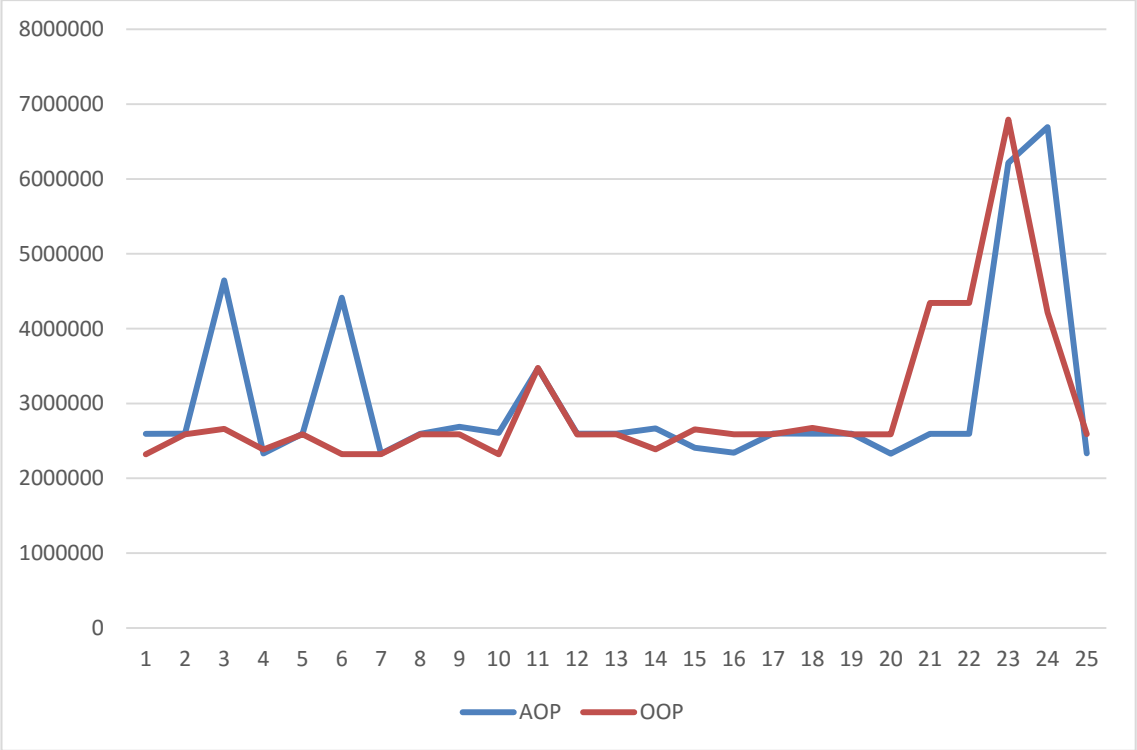

14 MB
12 MB
10 MB
8 MB
6 MB
4 MB
2 MB
0 MB
05/10/2015 م 10:46:15.010
Heap Size 16,318,464 B
Used Heap 4,217,272 B
Max Heap Size 16,318,464 B
Max Used Heap 5,680,472 B
10:46:20 PM

24

		25

جدول 6-4 مقارنة أستخدام الذاكرة للبرامج المكتوبة بالبرمجة المفاهيمية والبرمجة الكائنية

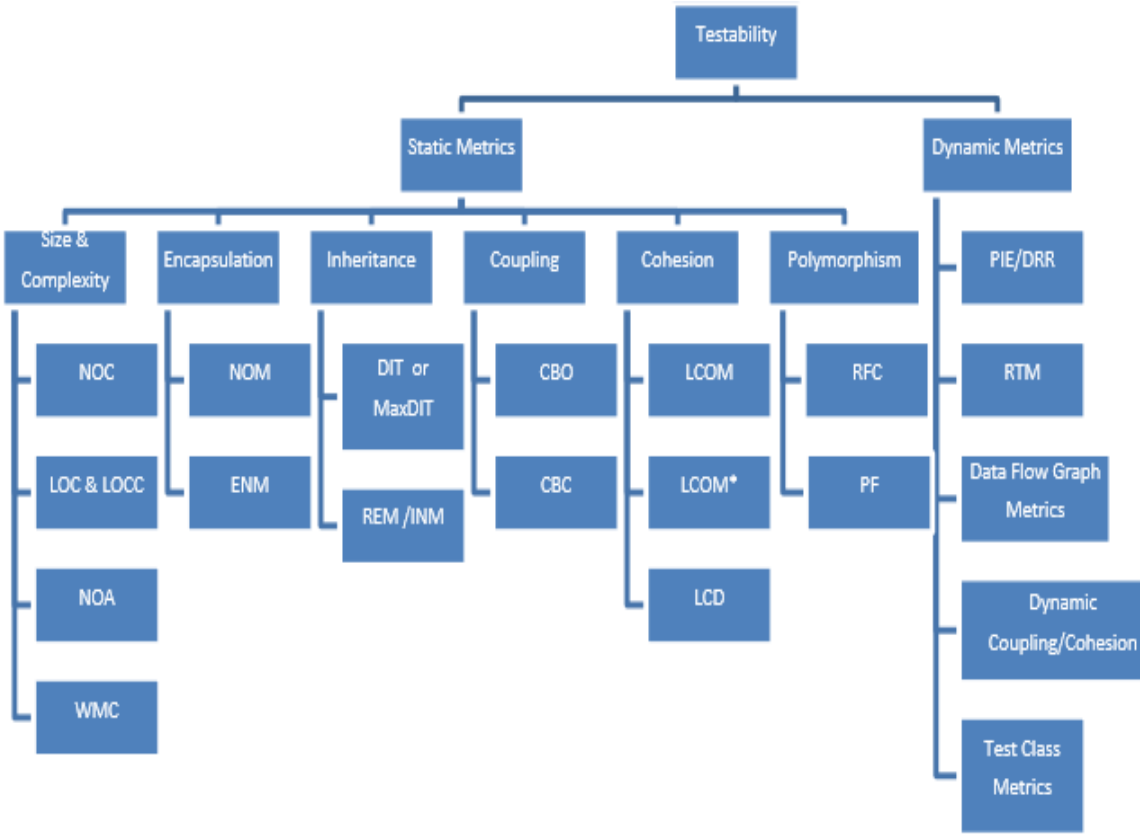
الشكل التالى 6-9 يوضح المقارنة بين أداء البرامج المكتوبة بأستخدام البرمجة المفاهيمية والبرمجة الكائنية، ونلاحظ من خلال الشكل ان أستخدام الذاكرة كان أكبر بالنسبة للبرامج المفاهيمية من البرامج الكائنية ، وبلغ متوسط أستخدام الذاكرة بالبايتات بالنسبة للبرامج الكائنية **2947356.48** بايت بينما بلغ المتوسط للبرامج المفاهيمية **3040793.88** بايت ، ولكن أيضا من الملاحظ أن أستخدام الذاكرة كان أقل بالنسبة للبرامج من 20 – 25 وهى البرامج التى تحتوى على أسطر كود أكبر من البرامج الاخرى ، وبالتالى فانه يمكننا القول أن الاداء بالنسبة للذاكرة يمكن ان يكون أفضل فى البرامج المفاهيمية كلما زاد عدد أسطر البرنامج.



شكل 6-8 مقارنة الأداء للبرامج المفاهيمية والكائنية

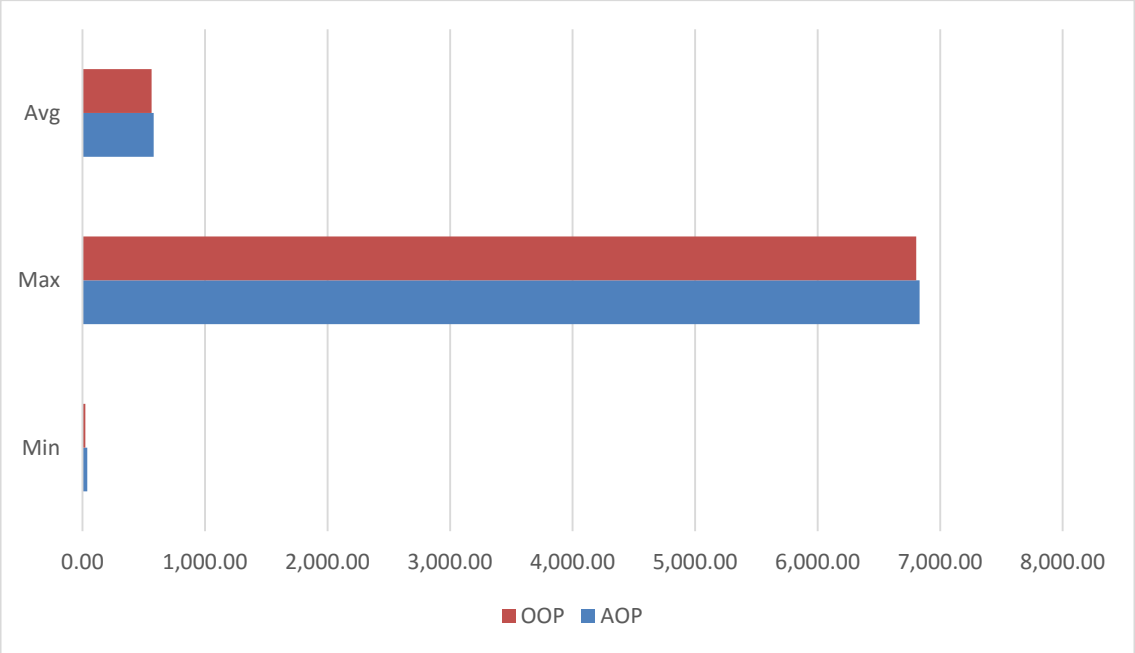
6.6 تأثير البرمجة المفاهيمية على قابلية الأختبار Testability

قابلية الاختبار من الخصائص الخارجية للبرمجيات والتى لا يمكن ان يتم قياسها بصورة مباشرة ، ولكن يمكن قياسها من خلال بعض الخصائص الداخلية، بالاضافة الى بعض القياسات الديناميكية والتى يتم قياسها بعد تنفيذ الكود، الشكل التالى 10-9 يوضح أنواع القياسات المختلفة التى يمكن ان تؤثر على قابلية الصيانة [54]



شكل 6-9 مقاييس قابلية الأختبار

من حيث عدد أسطر الكود وبالرجوع الى الشكل 6-3 نجد أن الفرق فى عدد أسطر الكود صغير للغاية ونجد أن متوسط عدد أسطر الكود للبرامج المفاهيمية 580.4 سطر بينما المتوسط لبرامج المنهجية الكائنية كان 563.3 . الشكل التالى 6-11 يوضح المقارنة بين المتوسط والقيمة الاعلى والادنى لاسطر الكود بأستخدام المنهجيتين



شكل 6-10 مقارنة قياسات عدد اسطر الكود فى المنهجيتين

بالنسبة للمقاييس الاخرى التى تؤثر على قابلية الأختبار نجد ان المنهجيتين قد حصلتا على نفس القيم بالنسبة للمقاييس التالية:

1- LCC – Loose Class Coupling نجد ان المنهجيتين قد حصلتا فى المتوسط على القيمة 2.667

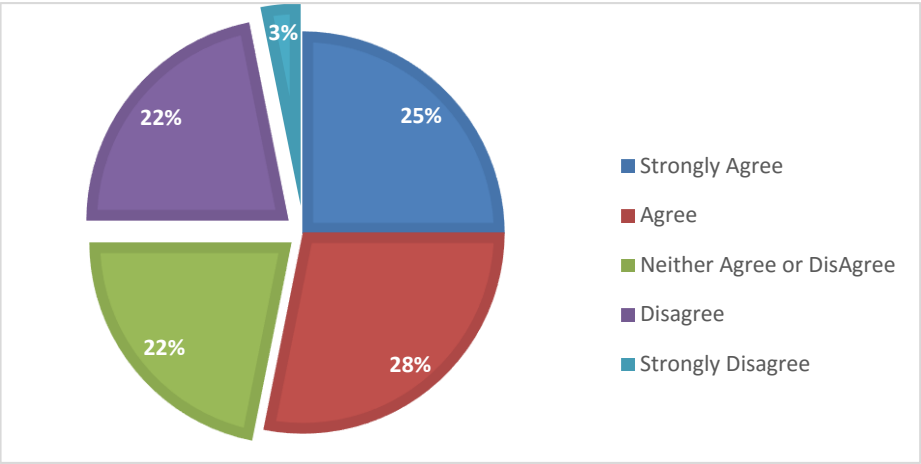
2- WMC –Weighted Methode Count والذى هو عبارة عن مجموع تعقيد الدوال فى الفصيل المحدد، نجد أيضا ان المنهجيتين قد حصلتا فى المتوسط على القيمة 9.

3- Lack Of Cohesion in Methods –LCOM1 المنهجيتين حصلتا فى المتوسط على 11 بينما للمقياس LCOM2 كان المتوسط 7 وبلغ للمقياسين LCOM3 و LCOM4 3.

4- Tight Class Coupling – TCC حصلت المنهجيتان على القيمة 2.6667.

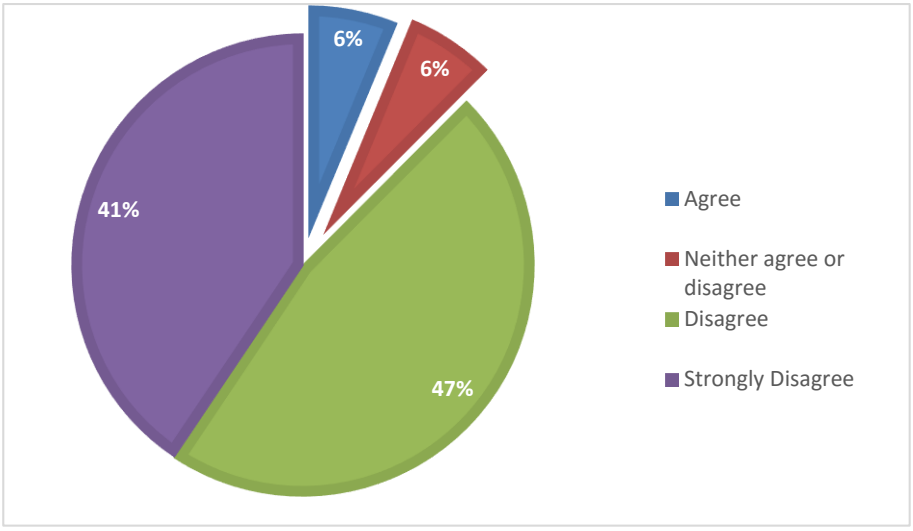
من الواضح من خلال القياسات الثابتة Static Metrics على الكود انه لا يوجد فرق كبير من حيث سهولة الأختبار، ولكن كما هو معروف فأن مراحل الأختبار تبدأ بأختبار المكونات Componenet Testing والذى من خلاله يتم أختبار صحة كل مكون على حده قبل ان يتم ربطه مع المكونات الاخرى، وهنا تبرز مشكلة أختبار الاسبكت Aspect أذ أنه كمكون برمجى لا يمكن ان يتم أختباره بصورة مستقلة لانه لا يتم تنفيذه الا بعد ان يقوم الناسج Weaver بتوزيع الكود على المكونات الاخرى، وهو ما قد يولد أخطاء تكون غير مرئية أثناء تطوير المكونات المختلفة وتظهر بعد ان يحدث تفاعل بينها.

أحتوى الاستبيان على سؤال عن مدى سهولة أختبار البرامج المكتوبة بالبرمجة المفاهيمية مقارنة مع البرامج المكتوبة بأستخدام المنهجية الكائنية. السؤال هو: البرامج المكتوبة بالمنهجية المفاهيمية صعبة الاختبار Aspect Oriented Programs are Difficult to Test ، 28% من العينة متفقون مع العبارة أعلاه بينما 25% يوافقون بشدة و 22% محايدين بينما فى المقابل 22% لا يوافقون و3% فقط يعارضون بشدة. الشكل التالى 6- 12 يوضح نسب الاجابة على سؤال صعوبة أختبار برامج البرمجة المفاهيمية



شكل 6-11 البرامج المفاهيمية صعبة الأختبار

بينما كانت الأجابة على السؤال : البرامج المفاهيمية من الصعب معالجة الأخطاء فيها Aspect Oriented Programs are difficult to debug كالاتى: فقط 6% يوافقون و 6% محايدون بينما 47% لا يوافقون و 41% يعارضون بشدة العبارة أعلاه ، الشكل 6-13 يوضح توزيع النسب

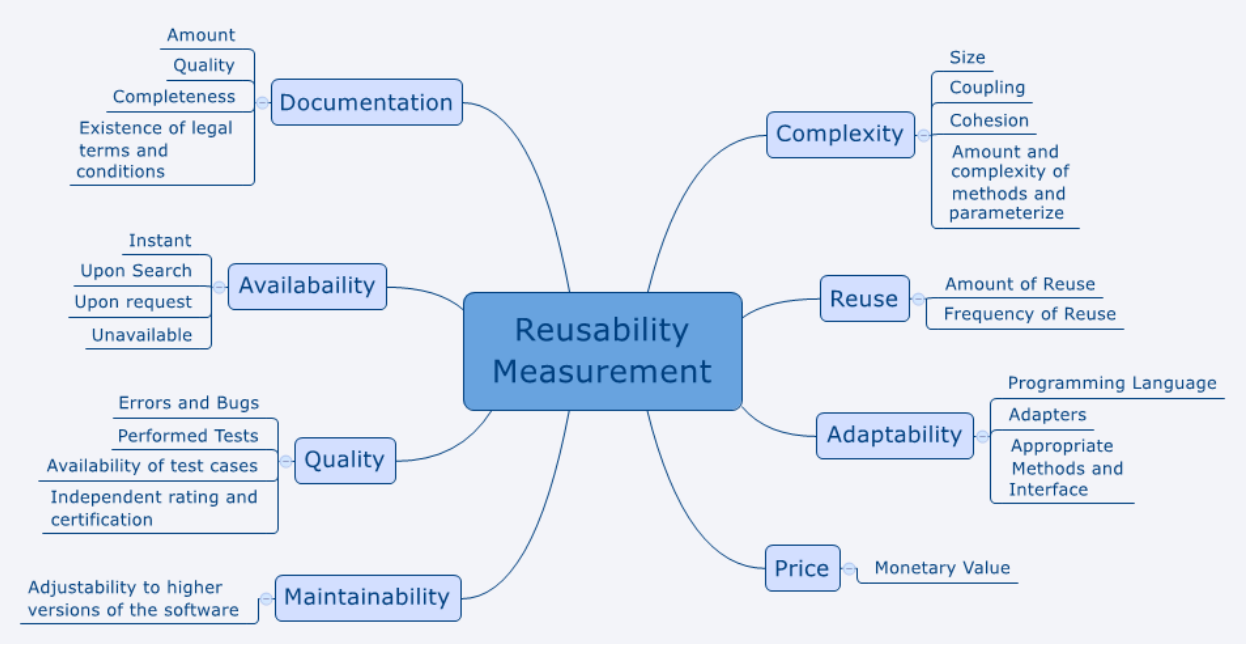


شكل 6-12 البرامج المفاهيمية صعبة فى أكتشاف الاخطاء ومعالجتها

6.7 تأثير البرمجة المفاهيمية على قابلية أعادة الأستخدام Reusability

خاصية أعادة الأستخدام تشير الى أمكانية أعادة أستخدام المكون البرمجى داخل البرنامج الواحد أو أعادة أستخدامه فى برامج أخرى، أعادة أستخدام الاجزاء البرمجية يزيد من صحة البرمجيات ويقلل تكلفة الاختبار ومعالجة الأخطاء.

قابلية أعادة الأستخدام من الخصائص الخارجية والتى لا يمكن قياسها بصورة مباشرة والتى تتاثر وتاثر على عده خصائص أخرى للجودة، الشكل التالى 6-14 يوضح العلاقة بين قابلية الأختبار والخصائص الاخرى والقياسات المختلفة لها [59]

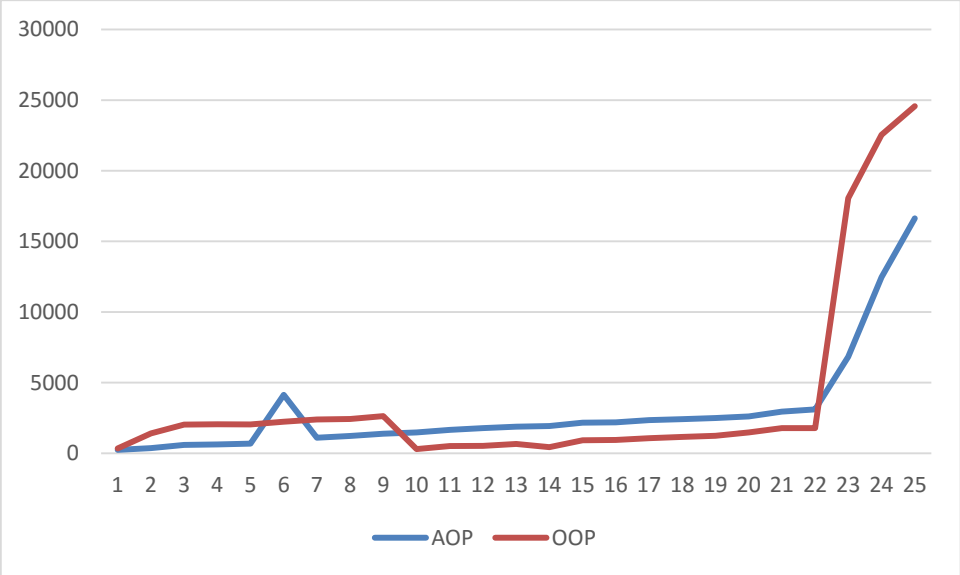


شكل 6-13 العوامل والمقاييس التى تؤثر على قابلية أعادة الأستخدام

من الشكل 6-14 نجد ان القياسات الوحيدة التى يمكن أجرائها بصورة داخلية على البرنامج هى تلك القياسات المتعلقة بالتعقيد Complexity والتى تنحصر فى مقاييس

Cohesion و Coupling والحجم Size ، ومما سبق ذكره فى قابلية الأختبار فأنه من خلال القياسات المباشرة لمقاييس Coupling و Cohesion فأنه لا يوجد فرق بين الطريقتين، بالنسبة للحجم بالرجوع الى الشكل 6-4 والشكل 6-5 نجد أن حجم البرنامج (عدد أسطر الكود) يقل فى البرمجة المفاهيمية عن البرمجة الكائنية كلما زاد عدد اسطر البرنامج الكلى.

أحد المقاييس المستخدمة لقياس التعقيد والتى تؤثر على قابلية أعادة الأستخدام مقياس هولستيد Halstead [56]. الشكل التالى 6-15 يبين المقارنة بين البرامج المفاهيمية والبرامج الكائنية بأستخدام مقياس الكمية Volume.



شكل 6-14 مقارنة البرامج المفاهيمية والكائنية بالنسبة لمقياس الكمية لهولستيد

من الشكل السابق نجد أن البرمجة المفاهيمية تعتبر أقل تعقيدا من البرمجة الكائنية وعلى الرغم من تفوق البرمجة الكائنية فى بعض البرامج الا انه عند قراءة المحصلة الاحصائية للمقاييس

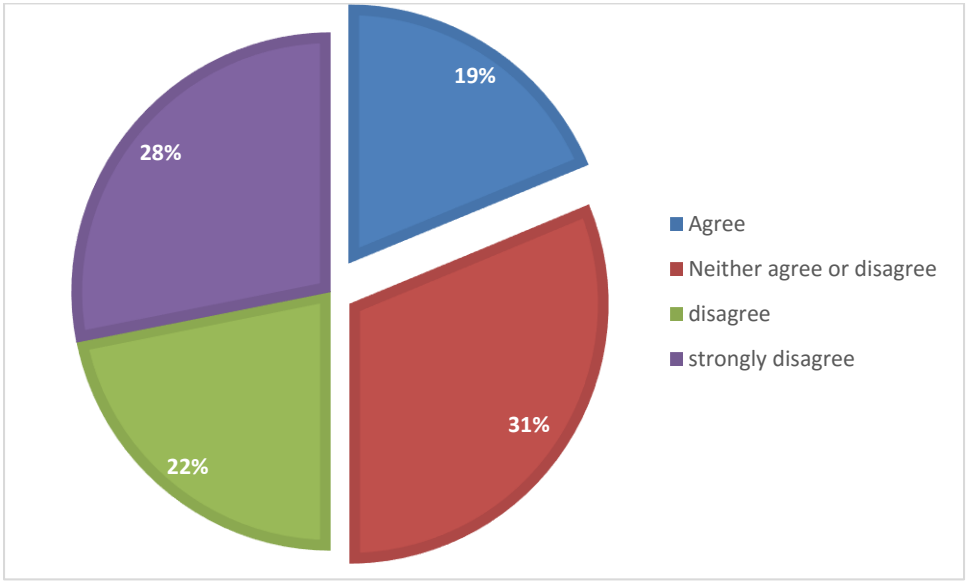
نجد أن أصغر قيمة بالنسبة للبرمجة المفاهيمية بلغت **240.81** بينما كانت للبرمجة الكائنية **303.58** وأعلى قيمة كانت للبرمجة المفاهيمية **16627.92** وللبرمجة الكائنية **24574.31** وبلغ المتوسط للبرمجة المفاهيمية **3012.78** بينما بلغ للبرمجة الكائنية **3822.62** . نلاحظ من القيم أعلاه أن البرمجة المفاهيمية أقل تعقيدا، وعليه يمكن القول ان قابلية أعادة الاستخدام أعلى فى البرمجة المفاهيمية.

6.8 تأثير البرمجة المفاهيمية على سهولة الفهم Understandability

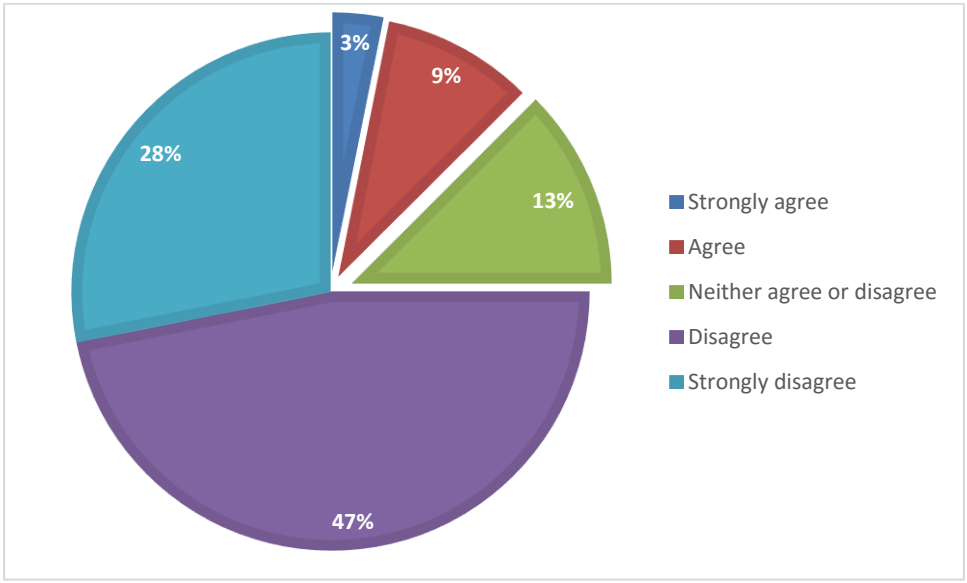
سهولة فهم البرنامج أو الكود من خصائص الجودة المهمة التى تؤثر على قابلية الصيانة ، قابلية أعادة الاستخدام ، بالأضافة الى تأثيرها على تكلفة البرنامج. يحتاج المبرمجون لفهم الكود حتى يتمكنوا من أجراء أى تعديل عليه او أعادة أستخدامه، وفهم الكود يعتمد على التعقيد الأدراكى Cognitive Complexity [61]. ولقياس التعقيد الأدراكى أحتوى الاستبيان على السؤال : البرمجة المفاهيمية تنتج برامج معقدة وكانت نتائج الاجابة على السؤال كما هو موضح بالشكل 6-16. نلاحظ من الشكل أن 50% من المتطوعين يعارضون أن البرمجة المفاهيمية تنتج برامج معقدة بينما فقط 19% من المتطوعين يرون ان البرمجة المفاهيمية تنتج برامج معقدة بينما 31% من المتطوعين كان رايهم محايدا.

أيضا أحتوى الأستبيان على السؤال: صيغة البرامج المفاهيمية مثل نقاط القطع من الصعب فهمها. وكانت نتيجة السؤال كما موضح بالشكل 6-17، وكما هو واضح فى الشكل

فأن 75% من المتطوعين لا يوافقون على ان صيغة البرامج المفاهيمية صعبة الفهم بينما فقط 12% يرون صعوبة فهم صيغة البرامج المفاهيمية بينما 13% كانو محايدين.



شكل 6-15 البرمجة المفاهيمية تنتج برامج معقدة



شكل 6-16 صيغة البرامج المفاهيمية صعبة الفهم

العنوان:	تأثير البرمجة ذات التوجه المفاهيمى على جودة البرمجيات
المؤلف الرئيسي:	عثمان، أحمد سيد أحمد علي
مؤلفين آخرين:	حاج على، عوض(مشرف)
التاريخ الميلادي:	2015
موقع:	الخرطوم
الصفحات:	1 - 240
رقم MD:	830496
نوع المحتوى:	رسائل جامعية
اللغة:	Arabic
الدرجة العلمية:	رسالة دكتوراه
الجامعة:	جامعة النيلين
الكلية:	كلية الدراسات العليا
الدولة:	السودان
قواعد المعلومات:	Dissertations
مواضيع:	هندسة البرمجيات، برمجة المفاهيمية، جودة البرمجيات، التحليل البرمجى
رابط:	https://search.mandumah.com/Record/830496

الفصل السابع

النتائج والتوصيات (RESULTS &

RECOMMENDATIONS)

7.1 النتائج

من خلال الدراسة وبمقارنة البرمجة الكائنية والبرمجة المفاهيمية من حيث التأثير على جودة البرمجيات، خلصت الدراسة الى الاتى:

- إن تأثير البرمجة المفاهيمية على قابلية الصيانة بصورة عامة يعتمد على حجم البرنامج (عدد أسطر الكود) ويعتمد على الطريقة التى تم بها تنفيذ البرمجة المفاهيمية. عندما كان حجم البرامج صغيرا بحيث أن عدد أسطر الكود أقل من 2000 نجد ان البرمجة المفاهيمية أنتجت برامج أقل قابلية للصيانة من البرمجة الكائنية بأستثناء الحالات التى كان فيها حجم برنامج البرمجة المفاهيمية مساويا او قريبا للبرنامج المقابل له والمطور بإستخدام المنهجية الكائنية. أما بالنسبة للبرامج الكبيرة والتى فاق عدد أسطر الكود فيها الـ 2000 سطر، نجد أن البرمجة المفاهيمية أنتجت برامج أكثر قابلية للصيانة من البرامج الكائنية.
- من خلال القياسات الثابتة التى تؤثر على قابلية الاختبار لا يوجد فرق يذكر فى القيم بين المنهجيتين إذ تم الحصول على نفس القيم بالنسبة للمنهجيتين ولكن توجد مشكلة فى إختبار الاسبكت Aspect كوحدة برمجية مستقلة وذلك لانه لا يمكن أن يتم تنفيذه بصورة مستقلة ومن خلال الاستبيان إتضح أن أكثر من 53% من المتطوعين يرون أن هنالك صعوبة في أختبار البرامج المفاهيمية.

- تأثير البرمجة المفاهيمية على الأداء وبالتحديد على أستخدام الذاكرة نجد أنه تأثير سلبي بالنسبة للبرامج الصغيرة والتى تحتوى على أسطر كود أقل من 150 سطر أما بالنسبة للبرامج الكبيرة والتى أحتوت على أكثر من 150 سطر نجد أن البرامجة المفاهيمية أنتجت برامج أكثر كفاءة فى أستخدام الذاكرة من البرامج الكائنية.
- تؤثر البرمجة المفاهيمية تأثيرا إيجابيا على قابلية إعادة الإستخدام مقارنة بالبرمجة الكائنية المنحى.
- تأثير البرمجة المفاهيمية على قابلية الفهم understandability هو تأثير إيجابي، اغلبية المتطوعون يرون أن البرمجة المفاهيمية تنتج برامج سهلة الفهم.

7.2 التوصيات

يوصى الباحث بالاتى:

- دراسة تأثير البرمجة المفاهيمية على الخصائص الاخرى لجودة البرمجيات.
- أستخدام عينه تمثل مجتمع المهنين فى تقييم تاثير البرمجة المفاهيمية على الجودة
- تطبيق الدراسة على عينة أحصائية أكبر مما يؤدى الى نتائج أكثر دقة.
- أستخدام برامج أكبر تحتوى على الاف الاسطر من الكود فى عملية تطبيق القياسات.
- قياس جودة البرامجيات المطورة بأســـتخدام البرمجة المفاهيمية خلال فترة زمنية طويلة من الاســـتخدام الفعلى لها مما يمكن من تطبيبق بعد المقاييس التى تعتمد على زمن التشـــغيل الفعلى للبرنامج
- دراســـة تأثير البرمجة المفاهيمية على موارد الجهاز الاخرى غير الذاكرة كتأثيرها مثلا على المعالج.

العنوان:	تأثير البرمجة ذات التوجه المفاهيمى على جودة البرمجيات
المؤلف الرئيسي:	عثمان، أحمد سيد أحمد علي
مؤلفين آخرين:	حاج على، عوض(مشرف)
التاريخ الميلادي:	2015
موقع:	الخرطوم
الصفحات:	1 - 240
رقم MD:	830496
نوع المحتوى:	رسائل جامعية
اللغة:	Arabic
الدرجة العلمية:	رسالة دكتوراه
الجامعة:	جامعة النيلين
الكلية:	كلية الدراسات العليا
الدولة:	السودان
قواعد المعلومات:	Dissertations
مواضيع:	هندسة البرمجيات، برمجة المفاهيمية، جودة البرمجيات، التحليل البرمجى
رابط:	https://search.mandumah.com/Record/830496

ملخص البحث

على مر الزمن أهتم الباحثون حول العالم بتطوير العديد من المنهجيات التى تستخدم فى تطوير البرمجيات والتى تحسن من تقسيم البرنامج , ومما لا شك فيه أن طريقة تقسيم الكود الى وحدات بناءا على المنهجية المتبعة فى التطوير له تأثير كبير على جودة البرمجيات، ، ولعل أهم منهجية أحدثت تحولا كبيرا فى تطوير البرمجيات هى البرمجة الكائنية، ولكن على الرغم من الفوائد الكبيرة التى حققتها البرمجة الكائنية الان انه مع كبر حجم البرامج وزيادة تعقيدها ظهر تحدى الكود الموزع والكود المتداخل أو ما يعرف بتداخل الاهتمامات والتى جاءت البرمجة ذات التوجه المفاهيمي لتفصل الأهتمامات عن بعضها البعض.

تناولت الدراسة تأثير البرمجة المفاهيمية على خصائص جودة البرمجيات قابلية الصيانة، الاداء، قابلية الأختبار، قابلية أعادة الأستخدام وسهولة الفهم عن طريق مقارنتها مع البرمجة الكائنية، وذلك من خلال تحليل أسئلة أستبيان قام بملئها 32 متطوع عن البرمجة المفاهيمية بالاضافة الى أجراء قياسات على كود 25 برنامجا طور بأستخدام البرمجة المفاهيمية ومقارنتها مع 25 برنامجا تم تطويره بأستخدام المنهجية الكائنية .

خلصت الدراسة الى تفاوت تأثير البرمجة المفاهيمية على خصائص جودة البرمجيات فبينما كان التأثير أيجابيا بصورة نسبية على قابلية الصيانة ، الأداء ، سهولة الفهم، وقابلية أعادة الأستخدام كان سلبيا على خاصية قابلية الأختبار.

Abstract

No doubt the modularity and organization of the code based on the development methodology used to develop software is a key parameter that affects the quality of the software. Over time، researchers around the globe worked to develop new methodologies for developing software that can enhance the software modularity. No doubt that the most important methodology which changed the software development is OOP. Despite the benefits that OOP brings, new challenges appeared due to the increased size and complexity of software. These new challenges are tangled and scattered code or what is known as separation of concerns. AOP concerned with enhancing the software modularity by separating the concerns.

This research studied the impact of AOP on the following software quality characteristics: Maintainability, Performance, Testability, Reusability, and Understandability by analyzing the results of a questionnaire that has been filled by 32 volunteers, and by applying statics software metrics on the code of 25 programs that developed using AOP compared with 25 programs developed using OOP.

At the end of this research it was concluded that the impact of AOP on some features are positive to some extent like Maintainability, Understandability , Reusability and Performance while the impact was negative on testability.

العنوان:	تأثير البرمجة ذات التوجه المفاهيمى على جودة البرمجيات
المؤلف الرئيسي:	عثمان، أحمد سيد أحمد علي
مؤلفين آخرين:	حاج على، عوض(مشرف)
التاريخ الميلادي:	2015
موقع:	الخرطوم
الصفحات:	1 - 240
رقم MD:	830496
نوع المحتوى:	رسائل جامعية
اللغة:	Arabic
الدرجة العلمية:	رسالة دكتوراه
الجامعة:	جامعة النيلين
الكلية:	كلية الدراسات العليا
الدولة:	السودان
قواعد المعلومات:	Dissertations
مواضيع:	هندسة البرمجيات، برمجة المفاهيمية، جودة البرمجيات، التحليل البرمجى
رابط:	https://search.mandumah.com/Record/830496

الفهرس

فهرس الأشكال

فهرس الجداول

قاموس المصطلحات (GLOSSARY)

م	المصطلح باللغة الأنجليزية	المصطلح باللغة العربية
1	Aspect Oriented Programming	البرمجة ذات التوجه المفاهيمى
2	Point-Cut	نقاط القطع
3	Join-Point	نقاط الربط
4	Weaver	الناسج / الحائك
5	Weaving Process	عملية الحياكة / عملية النسج
6	Functionality	الوظيفية
7	Suitability	الملائمة
8	Accuracy	الدقة
9	Interoperatability	امكانية التشغيل المتداخل
10	Compliance	ألأمتثال
11	Efficiency	الفعالية
12	Time Behaviuor	السلوك الزمنى
13	Resource Behaviuor	السلوك فى أستهلاك الموارد
14	Reliablity	الموثوقية
15	Maturity	النضج
16	Recoverability	قابلية الأسترداد
17	Fault Tolerance	التسامح مع الأخطاء
18	Attractiveness	الجاذبية

العنوان:	تأثير البرمجة ذات التوجه المفاهيمى على جودة البرمجيات
المؤلف الرئيسي:	عثمان، أحمد سيد أحمد علي
مؤلفين آخرين:	حاج على، عوض(مشرف)
التاريخ الميلادي:	2015
موقع:	الخرطوم
الصفحات:	1 - 240
رقم MD:	830496
نوع المحتوى:	رسائل جامعية
اللغة:	Arabic
الدرجة العلمية:	رسالة دكتوراه
الجامعة:	جامعة النيلين
الكلية:	كلية الدراسات العليا
الدولة:	السودان
قواعد المعلومات:	Dissertations
مواضيع:	هندسة البرمجيات، برمجة المفاهيمية، جودة البرمجيات، التحليل البرمجى
رابط:	https://search.mandumah.com/Record/830496

جامعة النيلين

كلية الدراسات العليا

بحث مقدم لنيل درجة الدكتوراة فى هندسة البرمجيات بعنوان:

تأثير البرمجة ذات النوجه المفاهيمى على جودة البرمجيات

Aspect Oriented Programming Impact on Software Quality

إعداد الطالب :

أحمد سيداحمد على عثمان

إشراف:

بروفيسور: عوض حاج على

أكتوبر 2015

الآيـــة

قال تعالى فى محكم تنزيله:

"وعلم آدم الأسماء كلها ثم عرضهم على الملائكة فقال أنبئونى بأسماء هؤلاء إن كنتم صادقين* قالوا سبحانك لا علم لنا إلا ما علمتنا إنك أنت العليم الحكيم* قال يا آدم أنبئهم بأسمائهم فلما أنبأهم بأسمائهم قال ألم أقل لكم أنى أعلم غيب السموات والأرض وأعلم ما تبدون وما كنتم تكتمون"

سورة البقرة الايات(31-33)

الأهداء

الى من كانوا سببا فى وجودى ... ومن كانوا وراء ما وصلت اليه الان... من قدموا لى الدعم

خلال كل مراحل حياتى... من كانوا بقربى فى الاوقات العصيبة... وكذلك الجميلة... شكرا

على كل الحب والحنان الذى منحتمونى أياه **والدى العزيزين (سيداحمد وأسماء)**.....

الى **زوجتى العزيزة** التى وقفت معى وصبرت على وأنا أجلس الساعات الطوال أمام الحاسوب

لأنجاز هذا البحث....

الى **بناتى الحبيبات** رهف ، رغد ، ريان، ورنا الذين اعطوا حياتى طعما اخر.....

الى **أساتذتى الاجلاء** ... الشموع التى تحترق لتضئ لنا الطريق...

الى **زملائى وزميلاتى** خلال مراحل الدراسية ...

الى **كل المخلصين** من أبناء هذا الوطن الذين يعملون جاهدين من أجل رفعته...

شكر وعرفان

قال تعالي :﴿فاذكروني اذكركم واشكروا لي ولا تكفرون﴾ صدق الله العظيم

فالحمد لله بدءاً وإنتهاءً.. والحمد لله الذي وفقنا .. وهدانا .. وجعلنا بنعمته.. حملة للواء العلم.. وافراداً من مجتمعٍ له بصمةً واضحةً في الارتقاء بوطنه ..

والحمد لله الذي قيض لنا من الناس أخيرهم.. ومن الأساتذة أجلهم.. فهانت الصعاب .. ولانت كل العقبات..فالشكر وأسمي آيات التقدير نسوقها لكم مزدانة بإقترانها بإسمكم.. سيادة **البروفيسور /عوض حاج علي**.. الأب الروحي لكلية علوم الحاسوب وتقانة المعلومات بجامعة النيلين..

والشكر من بعده موصول لكلية علوم الحاسوب وتقانة والمعلومات أساتذة وموظفين وطلابا ،،،، واخص بالشكر قسم هندسة البرمجيات أساتذة وطلابا ... والشكر أخيرا لكل من ساهم في ان يرى هذا البحث النور.

ملخص البحث

على مر الزمن أهتم الباحثون حول العالم بتطوير العديد من المنهجيات التى تستخدم فى تطوير البرمجيات والتى تحسن من تقسيم البرنامج , ومما لا شك فيه أن طريقة تقسيم الكود الى وحدات بناءا على المنهجية المتبعة فى التطوير له تأثير كبير على جودة البرمجيات، ، ولعل أهم منهجية أحدثت تحولا كبيرا فى تطوير البرمجيات هى البرمجة الكائنية، ولكن على الرغم من الفوائد الكبيرة التى حققتها البرمجة الكائنية الان انه مع كبر حجم البرامج وزيادة تعقيدها ظهر تحدى الكود الموزع والكود المتداخل أو ما يعرف بتداخل الاهتمامات والتى جاءت البرمجة ذات التوجه المفاهيمي لتفصل الأهتمامات عن بعضها البعض.

تناولت الدراسة تأثير البرمجة المفاهيمية على خصائص جودة البرمجيات قابلية الصيانة، الاداء، قابلية الأختبار، قابلية أعادة الأستخدام وسهولة الفهم عن طريق مقارنتها مع البرمجة الكائنية، وذلك من خلال تحليل أسئلة أستبيان قام بملئها 32 متطوع عن البرمجة المفاهيمية بالاضافة الى أجراء قياسات على كود 25 برنامجا طور بأستخدام البرمجة المفاهيمية ومقارنتها مع 25 برنامجا تم تطويره بأستخدام المنهجية الكائنية .

خلصت الدراسة الى تفاوت تأثير البرمجة المفاهيمية على خصائص جودة البرمجيات فبينما كان التأثير أيجابيا بصورة نسبية على قابلية الصيانة ، الأداء ، سهولة الفهم، وقابلية أعادة الأستخدام كان سلبيا على خاصية قابلية الأختبار.

Abstract

No doubt the modularity and organization of the code based on the development methodology used to develop software is a key parameter that affects the quality of the software. Over time، researchers around the globe worked to develop new methodologies for developing software that can enhance the software modularity. No doubt that the most important methodology which changed the software development is OOP. Despite the benefits that OOP brings, new challenges appeared due to the increased size and complexity of software. These new challenges are tangled and scattered code or what is known as separation of concerns. AOP concerned with enhancing the software modularity by separating the concerns.

This research studied the impact of AOP on the following software quality characteristics: Maintainability, Performance, Testability, Reusability, and Understandability by analyzing the results of a questionnaire that has been filled by 32 volunteers, and by applying statics software metrics on the code of 25 programs that developed using AOP compared with 25 programs developed using OOP.

At the end of this research it was concluded that the impact of AOP on some features are positive to some extent like Maintainability, Understandability , Reusability and Performance while the impact was negative on testability.

الفهرس

فهرس الأشكال

فهرس الجداول

الفصل الأول (الأطار العام)

1.1 مقدمة

البرمجة Programming هى عملية كتابة مجموعة من الشفرات [الأكواد – Codes] بأستخدام احدى لغات البرمجة Programing Languages ومن ثم أختبار الشفرة البرمجية وتصحيح الاخطاء ان وجدت ، وكل شفرة او برنامج أو نظام حاسوبى يتم كتابته للقيام بأداء مهمة محددة.

تطورت عملية البرمجة وكتابة الشفرات البرمجية عبر الزمن تطورا كبيرا ففى الجيل الأول من لغات البرمجة كانت تتم عملية البرمجة بأستخدام الرمزين 0 و 1، وكانت تسمى لغة الالة Machine Language وهى اللغة الوحيدة التي يفهمها كمبيوتر ويقوم بتحويلها الى أشارات كهربية . أذ أن الـ1 يعنى وجود التيار الكهربائى والـ0 يعنى عدم وجوده، وبالتالى نجد أن عملية البرمجة كانت تتطلب معرفة دقيقة وتفصيلية بمعمارية الجهاز الذى سيقوم بتنفيذ البرنامج.

وجاء بعد ذلك الجيل الثاني من اللغات والذى عرف بأسم لغة التجميع Assembly Language ، لغة التجميع تحول تسلسل الـ 0 والـ1 الى كلمات يفهمها الانسان مثل Add وعند كتابة برنامج عن طريق هذه اللغة فأن المجمع Assembler يقوم بتحويل هذه العبارات البسيطة الى ما يقابلها من سلاسل الـ 0 والـ1 وبالتالى الى حد ما أصبحت عملية البرمجة أسهل وأقرب الى الانسان من الالة مقارنة بلغات الجيل الاول، ولكن ما زالت هنالك حوجة الى معرفة البنية الهيكلية للجهاز الذى سيقوم بتنفيذ البرنامج.

بعد ذلك جاء الجيل الثالث وهو ما عرف بلغات المستوى العالى High Level HLL،Languages والتي أصبحت تحتوى على جمل وتعابير أقرب الى الأنسان. وحتى يستطيع الكمبيوتر فهم وتنفيذ هذه البرامج فأن هذه اللغات احتوت على مترجم Compiler يقوم بتحويل هذه الجمل والتعابير الى لغة الالة او الى لغة التجميع. وكما نعلم فأن جميع البرامج يجب فى النهاية أن تترجم الى لغة الالة حتى تستطيع البنية الالكترونية لجهاز الكمبيوتر تنفيذها. هنالك العديد من لغات برمجة المستوى العالى بدءا من لغات البرمجة الهيكلية Structural Language واللغات الوظيفية Functional Language الى اللغات الكائنية Object Oriented Language . فى هذا الجيل أصبحت عملية البرمجة تركز أكثر على المشكلة موضوع الحل عوضا عن التركيز على تفاصيل الالة وأهم خطوة فى هذه الاتجاه كانت التركيز على الكائنات المكونة للنظام وتحديد سلوكها مما أتاح للمبرمجين ان ينتجو برامج اكثر تعقيدا واكبر حجما وبجودة أعلى. ولكن مع ظهور البرامج الكبيرة والمعقدة والتى يمكن أن تحتوى على ملايين الأسطر من الكود والتى تعمل على أنظمة حساسة بدأت تظهر بعض المشاكل فى البرمجة الكائنية وأهمها كان الكود الموزع والمتداخل والذى لا يمكننا الجزم بتبعيتة لكائن محدد من كائنات النظام. فعلى سبيل المثال عندما نتحدث عن الامن Security أو عن التتبع Logging فلا يمكننا ان نتحدث عنهما كوحدة مستقلة بل نجد أنهما يتوزعان داخل جميع الكائنات المكونة للنظام. وعليه فأن عملية برمجة الانظمة المعقدة والكبيرة أصبحت تواجه بعض المشاكل أذ تتطلب أن يكون المبرمج ملما بجميع التفاصيل المتعلقة بالوظائف الاساسية للنظام Functional Requirement زائدا الوظائف غير الوظيفية Non-Functional Requirement مثل الأمن والتتبع والتعامل مع

الاخطاء Error Handling هذا بالاضافة الى المشاكل التى أصبحت تواجهنا عند أجراء الصيانة والتعقيد الذى أصبح موجودا فى البرامج جراء صيانتها لعدة مرات.

ظهرت تقنية البرمجة المفاهيمية Aspect- Oriented Software Development – AOSD – لتعالج مشاكل الكود المتداخل والموزع المرتبطة بالبرمجة الكائنية المنحى، كما تساعد على فصل الاهتمامات المختلفة عن بعضها البعض، كما تقوم بفصل المتطلبات الوظيفية Functional Requirements عن المتطلبات غير الوظيفية Non-Functional Requirements بصورة يمكن بها تتبع كل وظيفة أبتداءا من مرحلة المتطلبات وحتى مرحلة التنفيذ.

1.2 مشكلة الدراسة

مشكلة الدراسة تكمن فى أيجاد الطريقة الأمثل لتطوير البرامج وكتابة الكود, أذ أنه ومنذ ظهور الحاسبات وبرامجها وحتى يومنا هذا ما زال السؤال الشاغل للعاملين فى المجال ماهى الطريقة الامثل لتصميم البرامج وتطويرها والتى يمكن أن تنتج برامج ذات جودة عالية.

1.3 أهداف الدراسة

تهدف الدراسة الى دراسة تاثير البرمجة المفاهيمية Aspect- Oriented Software Development على خصائص جودة البرمجيات (قابلية الصيانة ، الأداء، قابلية أعادة الأستخدام، وقابلية الفهم) ومقارنة هذا التاثير مع البرمجة الكائنية المنحى Object – Oriented Programming.

1.4 أهمية الدراسة

لا شك ان البرمجيات غزت جميع مناحى الحياة ، فاصبحت هى الاساس فى ادارة جميع الاعمال سواءا التجارية او الصناعية او التعليمية او اى مجال اخر. وانتاج برمجيات بجودة عالية هو التحدى الاساسى الذى يواجه هندسة البرمجيات، بالاضافة الى أهمية أنتاج برمجيات تتوفر فيها خاصية الصيانة مستقبلا نسبة لاهمية أستمرارية البرمجيات بالعمل لفترات طويلة مع أمكانية أستيعاب المتغيرات التى يمكن أن تحدث فى أى بيئة عمل نتيجة للتطور المستمر فى الاعمال. تنبع أهمية الدراسة من مساعدة المبرمجين ومهندسي البرمجيات على أختيار المنهجية الافضل لتقسيم النظام والتى تمكنهم من أنتاج برمجيات بجودة عالية

1.5 منهجية الدراسة

لأختبار تأثير البرمجة المفاهيمية على خصائص جودة البرمجيات ستنتهج الدراسة المنهج التحليلى Imperical Study، وذلك من خلال تطبيق المنهجية الكائنية والمنهجية المفاهيمية على مجموعة من البرامج المختلفة ومن ثم مقارنة البرامج مع بعضها البعض بالأضافة الى تحليل نتائج أستبيان قام بملئه مطورو تلك البرامج.

1.6 هيكل البحث

تتكون الدراسة من سبعة فصول حيث يتكون الفصل الاول(هيكل البحث) من مقدمة البحث، ومشكلة الدراسة، واهمية الدراسة، وخطة الدراسة. اما الفصل الثاني (الدراسات السابقة) فيتحدث عن عدد من الدراسات السابقة التي تم اجرائها خلال السنوات الماضية حيث يعرض

كل دراسة النتائج التى توصلت اليها كل دراسة . اما الفصل الثالث (البرمجة المفاهيمية التوجه) فيتكون من مقدمة تاريخية عن البرمجة وتطورها حتى الان ، ومفهوم البرمجة المفاهيمية والمشاكل التى قامت بحلها هذه الطريقة ، بالاضافة الى اللغات البرمجية التى من خلالها يمكن تطبيق البرمجة المفاهيمية. الفصل الرابع (جودة البرمجيات) فيتكون تعريف الجودة من وجهات النظر المختلفة ، كما تناول بعض نماذج جودة البرمجيات بالاضافة الى القياسات المستخدمة فى قياس الجودة ، أما الفصل الخامس (تحليل وتصميم النظام) فيتكون من تحليل وتصميم النظام الذى قام بتطويره الباحث باستخدام البرمجة الكائنية والبرمجة المفاهيمية . أما الفصل السادس (النتائج الأحصائية) فيتعرض الى تحليل الاستبيانات التى قام بملئها المتطوعون بالاضافة الى نتائج تطبيق قياسات جودة البرمجيات على الأنظمة التى قاموا بتطويرها ومقارنة النتائج بين البرمجة المفاهيمية والكائنية . أما الفصل السابع (الخاتمة والتوصيات) فتناول خاتمة البحث، وتوصيات الباحث، بالإضافة الى المصادر والمراجع، واخيرا الملاحق.

الفصل الثانى (الدراسات السابقة –

(PREVIUOS STUDIES

2.1 مقدمة

أنصب اهتمام الباحثين فى مجال هندسة البرمجيات منذ ظهورها والى يومنا هذا نحو الأهتمام بمعالجة التحديات التى تواجه هندسة البرمجيات. ووفقا للكاتب [Ian Summervil] [2] فأن هذه التحديات تنقسم الى ثلاثه تحديات رئيسية هى:

1- تسليم البرمجيات فى الزمن المحدد [Delivery]

2- أنتاج برمجيات يمكن الاعتماد عليها (الجودة) [Trust]

3- تنوع بيئات التشغيل [Heterogeneity]

وكنتيجة لهذه البحوث ظهرت مجموعة من التظريات Methods والادوات Tools والتقنيات Techniuques فى مجال هندسة البرمجيات والتى سعت بطريقة أو أخرى لمعالجة هذه التحديات، ومن المجالات التى نالت حظا وافرا من الباحثين هى كيفية تقسيم البرنامج Modularity بطريقة تمكن من سهولة كتابته وبالتالى تقليل زمن التطوير كما تزيد من مقروئية الكود – Code Readability وتؤثر أيجابا على امكانية الصيانة Maintainabiliy فى المستقبل، ومن اخر هذه التقنيات فى مجال البرمجة هى البرمجة المفاهيمية والتى بعد ظهورها لاقت اهتماما كبيرا من الباحثين والمطورين وذلك لما تحمله من ميزات واعدة يمكن ان تساهم بصورة ايجابية فى تطوير البرمجيات، ونسبة لأن البرمجة المفاهيمية تعتبر من التقنيات الجديدة والتى لم تبلغ مرحلة النضج بعد، فنجد ان مجتمع البرمجة المفاهيمية أنصب أهتمامه على تطوير هذه الطريقة أكثر من أهتمامه بدراسة التأثيرات التى يمكن أن تنتج عن أستخدامها، ولذلك نلاحظ انه توجد دراسات قليلة [17]

أهتمت بدراسة التأثيرات المحتملة من أستخدام البرمجة المفاهيمية على جوانب تطوير البرمجيات المختلفة مقارنة بتلك الدراسات التى أهتمت بتطوير الطريقة نفسها، فى هذا الفصل سنتناول بعضا من الدراسات التى أهتمت بدراسة تأثير البرمجة المفاهيمية على جودة البرمجيات.

2.2 الدراسات السابقة

2.2.1 البرمجة المفاهيمية وجودة البرمجيات

Rpoger Alexander and James Bieman، 2004، Aspect Oriented Technology ans Software Quality [15]

ذكرت الدراسة ان البرمجة المفاهيمية من الطرق الجديدة فى البرمجة والتى لاقت اهتماما من الباحثين ومجتمع المطورين، وهى عبارة عن طريقة لتطوير البرمجيات تهتم بفصل الاهتمامات عن بعضها البعض، وعلى الرغم من الفوائد المرتبطة بهذه الطريقة والمتمثلة فى تقليل اسطر الكود المكتوبة والتقسيم الجيد للبرنامج الا انها كاى تقنية اخرى تقابل هذه الميزات تكلفة، من خلال الدراسة تعرض الباحثان الى التكلفة المرتبطة بهذه الطريقة وذلك لاظهار الجانب السلبي فى هذه الطريقة

عدد من الباحثين والمطورين قاموا بدراسات عن الفوائد المرتبطة باستخدام البرمجة المفاهيمية، ووجد الباحثان عدد قليل جدا من البحوث عن التكلفة المرتبطة بالبرمجة المفاهيمية، وكأى تقنية جديدة فان الفوائد تكون واعدة، ولكن اى تقنية جديدة تجلب ايضا تكلفة اضافية مرتبطة بها، وأذا ما تم اعتماد البرمجة المفاهيمية فان ذلك سيكون له تاثير كبير على هندسة البرمجيات

العنوان:	تأثير البرمجة ذات التوجه المفاهيمى على جودة البرمجيات
المؤلف الرئيسي:	عثمان، أحمد سيد أحمد علي
مؤلفين آخرين:	حاج على، عوض(مشرف)
التاريخ الميلادي:	2015
موقع:	الخرطوم
الصفحات:	1 - 240
رقم MD:	830496
نوع المحتوى:	رسائل جامعية
اللغة:	Arabic
الدرجة العلمية:	رسالة دكتوراه
الجامعة:	جامعة النيلين
الكلية:	كلية الدراسات العليا
الدولة:	السودان
قواعد المعلومات:	Dissertations
مواضيع:	هندسة البرمجيات، برمجة المفاهيمية، جودة البرمجيات، التحليل البرمجى
رابط:	https://search.mandumah.com/Record/830496

المراجع

(REFERENCES)

قائمة المراجع

[1] القران الكريم

[2] Sommerville، Ian. Software Engineering. 8th ed. Harlow، England: Addison-Wesley، 2006. PDF.

[3] Pressman، Roger S. Software Engineering: A Practitioner's Approach. 6th ed. New York: McGraw-Hill، 2008. PDF.

[4] Filman، Robert E. Aspect-oriented Software Development. Harlow: Addison-Wesley، 2005. PDF.

[5] Jacobson، Ivar، and Pan-Wei Ng. Aspect-oriented Software Development with Use Cases. Upper Saddle River، NJ: Addison-Wesley، 2005. PDF.

[6] Tian، Jeff. Software Quality Engineering: Testing، Quality Assurance، and Quantifiable Improvement. Hoboken، NJ: Wiley، 2005. PDF.

[7] Schulmeyer، G. Gordon.، and James I. McManus. Handbook of Software Quality Assurance. 4th ed. New York: Van Nostrand Reinhold، 2006. PDF.

[8] Laird، Linda M.، and M. Carol Brennan. Software Measurement and Estimation: A Practical Approach. Hoboken، NJ: John Wiley & Sons، 2006. PDF.

[9] Clarke، Siobhán، and Elisa Baniassad. Aspect-oriented Analysis and Design: The Theme Approach. Upper Saddle River، NJ: Addison-Wesley، 2005. PDF.

[10] Gradecki، Joe، and Nicholas Lesiecki. Mastering AspectJ: Aspect-oriented Programming in Java. Indianapolis، IN: Wiley، 2008. PDF.

[11] O'Regan، Gerard. A Practical Approach to Software Quality. New York: Springer، 2002. PDF.

[12] Kiczales، Gregor، John Lamping، Anurag Mendhekar، Chris Maeda، Cristina Videira Lopes، Jean-Marc Loingtier، and John Irwin. Aspect-Oriented Programming. ، 2002. PDF.

[13] Fernando Asteasuain، Bernardo Contreras، Elsa Est´evez، Pablo R. Fillottrani. Evaluation of UML Extensions for Aspect Oriented Design

[14] Shiu Lun Tsang، Siobhán Clarke، Elisa Baniassad. An Evaluation of Aspect-Oriented Programming for Java-based Real-time Systems Development.

[15] Khalid Aljasser ،Peter Schachte. toward Reusable and Maintainable Aspect Oriented Programs. ParaAJ.

[16] Alexander، Roger، and James Bieman. "Editorial: Aspect-Oriented Technology and Software Quality." Software Quality Journal 12.2 (2004): 93-97. Print.

[17] Alexander، Roger، and James Bieman. "Editorial: Aspect-Oriented Technology and Software Quality." Software Quality Journal 12.2 (2004): 93-97. Print.

[18] Andrew Matthews. Producing High-Qulity software with Aspect Oriented Programming. PostSharp (2011). PDF.

[19] Bartsch، Marc، and Rachel Harrison. "An exploratory study of the effect of aspect-oriented programming on maintainability." Software Quality Journal 16.1 (2008): 23-44.

[20] Bradley، Jeremy T. "An examination of aspect-oriented programming in industry." Colorado State University، Colorado، USA (2003).

[21] Tourwé، Tom، Johan Brichau، and Kris Gybels. "On the existence of the AOSD-evolution paradox." SPLAT: Software engineering Properties of Languages for Aspect Technologies (2003).

[22] *Add two numbers*. (2014، May 20). Retrieved from Daily Free Code: http://www.dailyfreecode.com/code/two-numbers-1758.aspx

[23] *Difference Between Compiler and Interpreter*. (2014، May 21). Retrieved from engineers garage: http://www.engineersgarage.com/contribution/difference-between-compiler-and-interpreter

[24] *Machine Language To Add Two Numbers*. (2014، May 21). Retrieved from Blueroot: http://www.blueroot.biz/boulderprep/code/Code%20Hierarchy%20-%20Add%202%20Numbers.htm

[25] *Wikipedia*. (2014، May 12). Retrieved from Wikipedia Website: http://en.wikipedia.org/wiki/History_of_software_engineering

[26] *تاريخ تطور البرمجيات*. (2014، May 20). Retrieved from Wikipedia: http://ar.wikipedia.org/wiki/%D8%AA%D8%A7%D8%B1%D9%8A%D8%AE_%D8%AA%D8%B7%D9%88%D8%B1_%D8%A7%D9%84%D8%A8%D8%B1%D9%85%D8%AC%D9%8A%D8%A7%D8%AA

[27] http://en.wikipedia.org/wiki/Aspect-oriented_software_development 5/12/2009

[28] http://en.wikipedia.org/wiki/Business_logic 5/12/2009

[29] http://en.wikipedia.org/wiki/Concern_(computer_science) 5/12/2009

[30] http://msdn.microsoft.com/en-us/library/aa288717(VS.71).aspx 13/12/2009

[31] http://inventors.about.com/od/sstartinventions/a/software.htm 21\1\2010

[32] http://en.wikipedia.org/wiki/Software_quality 22\1\2010

[33] http://it.toolbox.com/wiki/index.php/Software_Quality_Metrics 28\2\2010

[34] Chibani، Meriem، Brahim Belattar، and Abdelhabib Bourouis. "Practical benefits of aspect-oriented programming paradigm in discrete event simulation." Modelling and Simulation in Engineering 2014 (2014): 47.

[35] Jose، Mrs Roby. "A Theoretical Framework for the Maintainability Model of Aspect Oriented Systems." Procedia Computer Science 62 (2015): 505-512.

[36] Chaudhary، Ruchi، and Ram Chatterjee. "Reusability in AOSD-The aptness، assessment and analysis." Optimization، Reliabilty، and Information Technology (ICROIT)، 2014 International Conference on. IEEE، 2014.

[37] Azuma، Motoei. "Applying iso/iec 9126-1 quality model to quality requirements engineering on critical software." Proceedings of the 3rd IEEE Int. Workshop on Requirements for High Assurance Systems (RHAS). 2004.

[38] ISO، ISQS. "ISO/IEC 25010". 2011." Systems and software engineering---Systems and software Quality Requirements and Evaluation (SQuaRE)---System and software quality models (2011).

[39] Suman، Manoj Wadhwa. "A Comparative Study of Software Quality Models." (IJCSIT) International Journal of Computer Science and Information Technologies ، V 5(4). (2014).

[40] Ahmad، Sheikh Fahad، Mohd Rizwan Beg، and Mohd Haleem. "A Comparative Study of Software Quality Models." International Journal of Science، Engineering and Technology Research 2.1 (2013): pp-172.

[41] Elkhidir، Fakhreldin، and Husni Al-Muhtaseb. "لـــغة النمـذجة الموحدة العربية." Communications of the ACS 4.1 (2011).

[42] Jacobson، Ivar. "Use cases and aspects-working seamlessly together." Journal of Object Technology 2.4 (2003): 7-28.

[43] Mendhekar، Anurag، Gregor Kiczales، and John Lamping. RG: A case-study for aspect-oriented programming. Vol. 9710044. Technical Report SPL97-009، 1997.

[44] Araújo، João، et al. "Aspect-oriented requirements with UML." Workshop on Aspect-oriented Modeling with UML. Vol. 7. 2002.

[45] Mohamed، Ahmed Yakout A.، Abd El Fatah A. Hegazy، and Ahmed R. Dawood. "Aspect Oriented Software Development vs. other Techniques (Structured Approach and Object Oriented Approach)." Computer and Information Science 3.3 (2010): p256.

[46] Cheaito، R.، et al. "Defining and Measuring Maintainability." (1995).

[47] Berander، Patrik، Lars-Ola Damm، Jeanette Eriksson، Tony Gorschek، Kennet Henningsson، Per Jönsson، Simon Kågström et al. "Software quality attributes and trade-offs." Blekinge Institute of Technology (2005).

[48] Coleman، Don، Dan Ash، Bruce Lowther، and Paul Oman. "Using metrics to evaluate software system maintainability." Computer 27، no. 8 (1994): 44-49.

[49] Heitlager، Ilja، Tobias Kuipers، and Joost Visser. "A practical model for measuring maintainability." In Quality of Information and Communications Technology، 2007. QUATIC 2007. 6th International Conference on the، pp. 30-39. IEEE، 2007.

[50] Kukreja، N. (2015، February 3). Measuring Software Maintainability. Retrieved September 22، 2015، from https://quandarypeak.com/2015/02/measuring-software-maintainability/

[51] V. Deursen، Arie. "Think Twice Before Using the "Maintainability Index"" Arie Van Deursen (Software Engineering in Theory and Practice). 29 Aug. 2014. Web. 23 Sept. 2015. <http://avandeursen.com/2014/08/29/think-twice-before-using-the-maintainability-index/>.

[52] Yu، Sheng، and Shijie Zhou. "A survey on metric of software complexity." Information Management and Engineering (ICIME)، 2010 The 2nd IEEE International Conference on. IEEE، 2010.

[53] Suri، Pushpa R.، and Harsha Singhani. "Testability Assessment of Object Oriented Software Using Static Metric Model and Analytic Hierarchy Process." International Journal of Computer Science Issues (IJCSI) 12.3 (2015): 76.

[54] Voas، Jeffrey M.، Keith W. Miller، and Jeffery E. Payne. "An empirical comparison of a dynamic software testability metric to static cyclomatic complexity." Proceedings of the 18th Annual SE Workshop، NASA-Goddard SE Laboratory Series Report. 1993.

[55] Singhani، H.، and P. R. Suri. "Object Oriented SoftwareTestability (OOSTe) Metrics Assessment Framework." Int. J. Adv. Res. Comput. Sci. Softw. Eng 5.4 (2015): 1096-1106.

[56] Poulin، Jeffrey S. "Measuring software reusability." Software Reuse: Advances in Software Reusability، 1994. Proceedings.، Third International Conference on. IEEE، 1994.

[57] ¹G، N. K.، and DRSK SRIVATSA. "ANALYSIS AND MEASURES OF SOFTWARE REUSABILITY." (2009).

[58] Gui، Gui، and Paul D. Scott. "Measuring software component reusability by coupling and cohesion metrics." Journal of computers 4.9 (2009): 797-805.

[59] Hristov، Danail، Oliver Hummel، Mahmudul Huq، and Werner Janjic. "Structuring Software Reusability Metrics for Component-Based Software Development." The Seventh International Conference on Software Engineering Advances (2012).

[60] Emanuel، Andi Wahju Rahardjo، et al. "Statistical Analysis on Software Metrics Affecting Modularity in Open Source Software." International Journal of Computer Science & Information Technology (IJCSIT) 3.3 (2011): 105-118.

[61] Srinivasulu، D. "Evaluation of Software Understandability Using Software Metrics." (2012).

العنوان:	تأثير البرمجة ذات التوجه المفاهيمى على جودة البرمجيات
المؤلف الرئيسي:	عثمان، أحمد سيد أحمد علي
مؤلفين آخرين:	حاج على، عوض(مشرف)
التاريخ الميلادي:	2015
موقع:	الخرطوم
الصفحات:	1 - 240
رقم MD:	830496
نوع المحتوى:	رسائل جامعية
اللغة:	Arabic
الدرجة العلمية:	رسالة دكتوراه
الجامعة:	جامعة النيلين
الكلية:	كلية الدراسات العليا
الدولة:	السودان
قواعد المعلومات:	Dissertations
مواضيع:	هندسة البرمجيات، برمجة المفاهيمية، جودة البرمجيات، التحليل البرمجى
رابط:	https://search.mandumah.com/Record/830496

الملاحق

(APPENDIXES)

الاستبيان

Questionnaire

Please answer all question based on your experience with OOP and AOP and based on the program you developed with each method

- ***Your name (optional)***

……

Gender*:*

☐ *Male* ☐ *female*

Age*:*

☐ *20-25* ☐ *25-30* ☐ *>30*

1. ***From your study، is aspect-oriented technology a good idea?***

   ☐ *YES* ☐ *NO*

2. ***The core idea of AOP، separation of concerns***

   ☐ *YES* ☐ *NO* ☐ *NOT SURE*

3. ***Which of The following cross-cutting concerns are the most commonly found in business applications(in your opinion):***

   ☐ *Logging*

   ☐ *Security Auditing*

   ☐ *Transactions*

   ☐ *Multithreading*

   ☐ *User Interface*

4. ***From the study of The Effect of Cross-Cutting Concerns (answer the following)***

   a- Business logic code becomes entangled with cross-cutting code

   ☐ *YES* ☐ *NO* ☐ *NOT SURE*

   b- Business logic code is tightly coupled to cross-cutting concerns.

   ☐ *YES* ☐ *NO* ☐ *NOT SURE*

5. ***What are the differences between OOP and AOP(in your opinion) ?***

……

……

……

……

6. ***If you have barriers to the adoption of aspect oriented programming answer the following:***
   1. *Strongly agree*
   2. *Agree*
   3. *Neither agree or disagree*
   4. *Disagree*
   5. *Strongly disagree*

7. ***Using the scale of 1-5 shown above، please select the extent to which you believe the following are barriers to the adoption of aspect oriented programming:***

   a. AOP is an immature technology.

   Your answer:

   ……..

   b. Not many interesting aspects outside of logging or tracing.

   Your answer:

   ……

   c. Not many meaningful examples to demonstrate the advantages of AOP.

   Your answer:

   ……..

   d. AOP is not an improvement over object oriented programming.

   Your answer:

   ……..

   e. Aspect oriented programs are very complicated.

   Your answer:

   ……

   f. Aspect oriented programs are difficult to test.

   Your answer:

   ……

g. Aspect oriented programs are difficult to maintain.
Your answer:

…….

h. Aspect oriented programs are difficult to debug.
Your answer:

……..

i. Aspect oriented programs perform slowly.
Your answer:

……

j. The syntax of aspect oriented programs (e.g. join points) isdifficult to understand.
Your answer:

……

k. It is difficult to understand the execution pattern of an aspect oriented application.
Your answer:

……..

l. AOP is not supported in my chosen development language.
Your answer:

……..

m. AOP is not supported in my chosen software development environment/ IDE.
Your answer:

……..

n. AOP tools are immature.
Your answer:

……..

o. I do not have enough time to learn AOP.
Your answer:

……..

p. High costs of adopting AOP (training/processes etc)

Your answer:

……..

q. No established AOP software development methodology.

Your answer:

……..

r. Application design is more difficult in AOP than in OO or procedural development.

Your answer:

……..

s. Please list below any other barriers which have not been identified above.

……

…….

8. Aspect-Oriented techniques can improve the readability of an implementation.

☐ *YES* ☐ *NO*

9. Aspect oriented programming decreases an application line of code

☐ *YES* ☐ *NO*

10. Aspect oriented programming decreases the number of errors in the program

☐ *YES* ☐ *NO*

11. aspect oriented programming make software system easy to use to the customer

☐ *YES* ☐ *NO*

12. Are the required function available in the aspect software?

☐ *YES* ☐ *NO*

13. Aspect software system can contain defect?

☐ *YES* ☐ *NO*

14. Is aspect oriented easy to fix bugs?

☐ *YES* ☐ *NO*

15. Is aspect oriented allow the programmer to add new functionality easier?

☐ *YES* ☐ *NO*

16. Can you use aspect code in your system in other system that seem to it easy.

☐ *YES* ☐ *NO*

17. Is aspect code is harder to find error on it

☐ ☐

YES *NO*

18. ***Is aspect codes is easy to fix bugs***

☐ *YES* ☐ *NO*

19. ***Do you think aspect oriented machine independent?***

☐ *YES* ☐ *NO*

20. ***In your system if you want to delete or add any requirement in the aspect code:***

A- Is it easy to do so?

☐ *YES* ☐ *NO*

B- Is this make a lot of errors?

☐ *YES* ☐ *NO*

21. ***The concept of a modularized crosscutting concern، or aspect، is not an easily tested unit of code، unlike classes or procedures.***

☐ *YES* ☐ *NO*

22. ***Aspect-oriented modeling (AOM) should be built upon a conceptual framework that we refer to as the aspect model***

☐ *YES* ☐ *NO* ☐ *NOT SURE*

23. ***Aspect Oriented Programming (AOP) aims to ease maintenance and promote reuse of software components by separating core concerns from crosscutting concerns***

☐ *YES* ☐ *NO* ☐ *NOT SURE*

24. ***AOP affect a system's size:***

☐ *YES* ☐ *NO* ☐ *NOT SURE*

25. ***Programming with AspectJ uses both objects and aspects to separate concerns.***

☐ *YES* ☐ *NO* ☐ *NOT SURE*

26. ***Metrics are an important technique in quantifying software and software development characteristics***

☐ *YES* ☐ *NO*

27. ***AOP can be used to implement security requirements in an application***

☐ *YES* ☐ *NO* ☐ *NOT SURE*

28. ***Aspect-Oriented techniques can improve the understandability of an implementation***

☐ *YES* ☐ *NO* ☐ *NOT SURE*

29. ***Can you use AOP and AspectJ today on real projects?***

☐ *YES* ☐ *NO*

30. Aspect oriented programming make software system easy to understand to the customer

☐ *YES* ☐ *NO*

31. AOP can reduce the amount of time required to develop the product

☐ *YES* ☐ *NO*

Inventory System Code (With Aspect).

1- FrmCr

```
package StockingAspect;
import java.sql.*;
import javax.swing.JOptionPane;
public class FrmCreditStock extends MyBase {
   Connection con;
   Statement stmt;
   ResultSet Temp،rs;
   int ID;
   boolean Updat;
   public FrmCreditStock() {
      initComponents();
      try{
         con = DriverManager.getConnection("jdbc:derby://localhost:1527/StokingDB"،
null، null);
         stmt = con.createStatement();
         rs = stmt.executeQuery("select * from APP.WRHMOVE");
      }catch(SQLException e){
          JOptionPane.showMessageDialog(null،e);
      }
      UserAccess();
      Updat = false;
      Enable(false);
      Updat = false;
      BtnEnable(false);
   }
   public void UserAccess(){
      try{
       Temp = stmt.executeQuery("select * from APP.USERTB where USRID= " + UsrID);
       String Str = "";
       if (Temp.next())
```

```
            Str = Temp.getString("USRWHR");
            if (Str.compareTo("All")==0){
              ResultSet Rs = stmt.executeQuery("select * from APP.WAREHOUSE") ;
              while (Rs.next())
                 CmbWareHouse.addItem(Rs.getString("WRHID"));
            }else
               CmbWareHouse.addItem(Str);
          }catch (SQLException e){
             JOptionPane.showMessageDialog(null, e);
          }
      }
      public void ClearForm() {
          TxtID.setText("");
          TxtDate.setText("");
          TxtName.setText("");
          TxtQty.setText("");
          TxtSource.setText("");
          BtnEnable(false);
          Updat=false;
      }
      void BtnEnable(boolean b){
          BtnDelete.setEnabled(b);
          BtnEdit.setEnabled(b);
          BtnSave.setEnabled(!b);
    }
      public void Enable(boolean b){
          TxtID.setEditable(b);
          TxtDate.setEditable(b);
          TxtName.setEnabled(b);
          TxtQty.setEditable(b);
          TxtSource.setEditable(b);
          BtnEnable(b);
       }
      @SuppressWarnings("unchecked")
      // <editor-fold defaultstate="collapsed" desc="Generated Code">
      private void initComponents() {
          jLabel2 = new javax.swing.JLabel();
          jLabel1 = new javax.swing.JLabel();
          jLabel3 = new javax.swing.JLabel();
          jLabel4 = new javax.swing.JLabel();
          jLabel5 = new javax.swing.JLabel();
          TxtSource = new javax.swing.JTextField();
```

```
TxtName = new javax.swing.JTextField();
TxtID = new javax.swing.JTextField();
TxtQty = new javax.swing.JTextField();
BtnNew = new javax.swing.JButton();
BtnSave = new javax.swing.JButton();
BtnEdit = new javax.swing.JButton();
BtnDelete = new javax.swing.JButton();
BtnExit = new javax.swing.JButton();
TxtDate = new javax.swing.JFormattedTextField();
jLabel6 = new javax.swing.JLabel();
CmbWareHouse = new javax.swing.JComboBox();
jLabel2.setText("Item No:");
jLabel1.setText("Transaction Date:");
jLabel3.setText("Item Name:");
jLabel4.setText("Qty:");
jLabel5.setText("Source:");
TxtName.setEditable(false);
TxtID.addFocusListener(new java.awt.event.FocusAdapter() {
   public void focusLost(java.awt.event.FocusEvent evt) {
      TxtIDFocusLost(evt);
   }
});
BtnNew.setText("New");
BtnNew.addMouseListener(new java.awt.event.MouseAdapter() {
   public void mouseClicked(java.awt.event.MouseEvent evt) {
      BtnNewMouseClicked(evt);
   }
});

BtnSave.setText("Save");
BtnSave.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      BtnSaveActionPerformed(evt);
   }
});

BtnEdit.setText("Edit");
BtnEdit.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      BtnEditActionPerformed(evt);
   }
});
```

```
        BtnDelete.setText("Delete");
        BtnDelete.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                BtnDeleteActionPerformed(evt);
            }
        });
        BtnExit.setForeground(new java.awt.Color(255, 0, 0));
        BtnExit.setText("Exit");
        BtnExit.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                BtnExitMouseClicked(evt);
            }
        });
        TxtDate.setFormatterFactory(new javax.swing.text.DefaultFormatterFactory(new
javax.swing.text.DateFormatter(new java.text.SimpleDateFormat("yyyy-MM-dd"))));
        jLabel6.setText("Warehouse No:");
        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jLabel6)
                    .addComponent(jLabel2)
                    .addComponent(jLabel1)
                    .addComponent(jLabel4)
                    .addComponent(jLabel5)
                    .addComponent(jLabel3))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
      .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                        .addComponent(TxtSource)
                        .addComponent(TxtName)
                        .addComponent(TxtQty)
                        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                            .addComponent(BtnNew)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
                    .addComponent(BtnSave)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(BtnEdit)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(BtnDelete)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(BtnExit))
                .addComponent(TxtID))
 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING‹
false)
                .addComponent(CmbWareHouse‹
javax.swing.GroupLayout.Alignment.LEADING‹ 0‹
javax.swing.GroupLayout.DEFAULT_SIZE‹ Short.MAX_VALUE)
                .addComponent(TxtDate‹ javax.swing.GroupLayout.Alignment.LEADING‹
javax.swing.GroupLayout.DEFAULT_SIZE‹ 97‹ Short.MAX_VALUE)))
            .addContainerGap(20‹ Short.MAX_VALUE))
      );
      layout.setVerticalGroup(
         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
         .addGroup(layout.createSequentialGroup()
            .addGap(43‹ 43‹ 43)
   .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
               .addComponent(jLabel6)
               .addComponent(CmbWareHouse‹ javax.swing.GroupLayout.PREFERRED_SIZE‹
javax.swing.GroupLayout.DEFAULT_SIZE‹ javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
               .addComponent(jLabel1)
               .addComponent(TxtDate‹ javax.swing.GroupLayout.PREFERRED_SIZE‹
javax.swing.GroupLayout.DEFAULT_SIZE‹ javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
               .addComponent(jLabel2)
               .addComponent(TxtID‹ javax.swing.GroupLayout.PREFERRED_SIZE‹
javax.swing.GroupLayout.DEFAULT_SIZE‹ javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
               .addComponent(jLabel3)
               .addComponent(TxtName‹ javax.swing.GroupLayout.PREFERRED_SIZE‹
javax.swing.GroupLayout.DEFAULT_SIZE‹ javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
```

```
                .addComponent(jLabel4)
                .addComponent(TxtQty، javax.swing.GroupLayout.PREFERRED_SIZE،
javax.swing.GroupLayout.DEFAULT_SIZE، javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel5)
                .addComponent(TxtSource، javax.swing.GroupLayout.PREFERRED_SIZE،
javax.swing.GroupLayout.DEFAULT_SIZE، javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(26، 26، 26)
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(BtnExit)
                .addComponent(BtnDelete)
                .addComponent(BtnEdit)
                .addComponent(BtnSave)
                .addComponent(BtnNew))
            .addContainerGap(30، Short.MAX_VALUE))
        );
        pack();
    }// </editor-fold>
    private void BtnNewMouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        Enable(true);
        ClearForm();
}
    private void BtnSaveActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        if (TxtID.getText().compareTo("")!=0 && TxtID.getText().compareTo("")!= 0 ){
            try{
                if (Updat){
                    stmt.executeUpdate("delete  from APP.WRHMOVE where MOVID = " + ID);
                    Updat = false;
                }
                Temp = stmt.executeQuery("select * From APP.WRHMOVE ");
                ID = Temp.getFetchSize() + 1 ;
                String Str ="INSERT INTO APP.WRHMOVE (MOVID، ITMNO، QUANTITYIN،
WRHNO، QUANTITYOUT، MOVDATE) VALUES          (" + ID + "، " + TxtID.getText() +"، " +
TxtQty.getText() + "، " + CmbWareHouse.getSelectedItem() + "، 0، '" +  TxtDate.getText()
+ "')" ;
                stmt.executeUpdate(Str);
                Enable(false);
                BtnEnable(true);
                JOptionPane.showMessageDialog(null،"Save seccesufully");
```

```
            }catch (SQLException e){
               JOptionPane.showMessageDialog(null‹e);
            }
         }else
            JOptionPane.showMessageDialog(null‹ "Error!! Please check the input data");
}
   private void BtnEditActionPerformed(java.awt.event.ActionEvent evt) {
      // TODO add your handling code here:
      Enable(true);
      ID = Integer.parseInt(TxtID.getText());
      Updat = true;
      BtnEnable(false);
}
   private void BtnDeleteActionPerformed(java.awt.event.ActionEvent evt) {
      //TODO add your handling code here:
      try{
         if (JOptionPane.showInternalConfirmDialog(null‹ "If you Delete this record‹ you
cann't return it back. Are you sure you want to delete this record?"‹
"Delete"‹JOptionPane.YES_NO_OPTION )== 0) {
            ID = Integer.parseInt(TxtID.getText());
            stmt.executeUpdate("delete  from APP.WAREHOUSE where WHRID = " + ID);
            ID =-1;
            Enable(true);
            ClearForm();
            BtnEnable(false);
         }
      }catch (SQLException e){
         JOptionPane.showMessageDialog(null‹e);
      }}
   private void BtnExitMouseClicked(java.awt.event.MouseEvent evt) {
      // TODO add your handling code here:
      this.setVisible(false);
      this.dispose();
}
   private void TxtIDFocusLost(java.awt.event.FocusEvent evt) {
      // TODO add your handling code here:
      if (IsInteger(TxtID.getText()))
      {
       try{
        ResultSet Rs = stmt.executeQuery("select * from APP.ITEM where ITMNO=" +
TxtID.getText());
        if (Rs.next())
```

```
            TxtName.setText(Rs.getString("ITMNAME"));
        else{
            JOptionPane.showMessageDialog(null,"The Item number you entered is not
correct. Please check it and try agan");
            TxtID.setText("");
        }

       }catch (SQLException e){
          JOptionPane.showMessageDialog(null,e);
       } } else
         JOptionPane.showMessageDialog(null, "you must entered integer value in this
field! please check the input and try again");
    }
    public static void main(String args[]) {
       java.awt.EventQueue.invokeLater(new Runnable() {
          public void run() {
             new FrmCreditStock().setVisible(true);
          }
       }
       );
    }
    public static javax.swing.JComboBox CmbWareHouse;
    public static javax.swing.JFormattedTextField TxtDate;
    public static javax.swing.JTextField TxtID;
    public static javax.swing.JTextField TxtName;
    public static javax.swing.JTextField TxtQty;
    public static javax.swing.JTextField TxtSource;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    // End of variables declaration
}
```

2- FrmDepitStock Class:

```
package StockingAspect;
import java.sql.*;
import javax.swing.JOptionPane;
```

```
public class FrmDepitStock extends MyBase {
   Connection con;
   ResultSet rs ، Temp;
   Statement stmt;
   boolean Updat;
   int ID;
    public FrmDepitStock() {
      initComponents();
      try{
         con = DriverManager.getConnection("jdbc:derby://localhost:1527/StokingDB"،
null، null);
         stmt = con.createStatement();
         rs = stmt.executeQuery("select * from APP.USERTB");
      }catch(SQLException e){
          JOptionPane.showMessageDialog(null،e);
      }
      UserAccess();
      Updat = false;
      Enable(false);
      Updat = false;
      BtnEnable(false);
   }
    public void UserAccess(){
      try{
       Temp = stmt.executeQuery("select * from APP.USERTB where USRID= " + UsrID);
       String Str = "";
       if (Temp.next())
       Str = Temp.getString("USRWHR");
       if (Str.compareTo("All")==0){
         ResultSet Rs = stmt.executeQuery("select * from APP.WAREHOUSE") ;
         while (Rs.next())
            CmbWareHouse.addItem(Rs.getString("WRHID"));
       }else
          CmbWareHouse.addItem(Str);

      }catch (SQLException e){
         JOptionPane.showMessageDialog(null، e);
      }   }
    public void ClearForm() {
      TxtID.setText("");
      TxtDate.setText("");
      TxtName.setText("");
```

```
        TxtQty.setText("");
        TxtSource.setText("");
        BtnEnable(false);
        Updat=false;
    }
void BtnEnable(boolean b){
        BtnDelete.setEnabled(b);
        BtnEdit.setEnabled(b);
        BtnSave.setEnabled(!b);
}
   public void Enable(boolean b){
        TxtID.setEditable(b);
        TxtDate.setEditable(b);
        TxtName.setEnabled(b);
        TxtQty.setEditable(b);
        TxtSource.setEditable(b);
        BtnEnable(b);
      }
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        jLabel6 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel1 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        TxtSource = new javax.swing.JTextField();
        TxtName = new javax.swing.JTextField();
        TxtQty = new javax.swing.JTextField();
        BtnNew = new javax.swing.JButton();
        BtnSave = new javax.swing.JButton();
        BtnEdit = new javax.swing.JButton();
        BtnDelete = new javax.swing.JButton();
        BtnExit = new javax.swing.JButton();
        TxtID = new javax.swing.JTextField();
        CmbWareHouse = new javax.swing.JComboBox();
        TxtDate = new javax.swing.JFormattedTextField();
        jLabel6.setText("Warehouse No:");
        jLabel2.setText("Item No:");
        jLabel1.setText("Transaction Date:");
        jLabel4.setText("Qty:");
```

```
        jLabel5.setText("Source:");
        jLabel3.setText("Item Name:");
        TxtName.setEditable(false);
        BtnNew.setText("New");
        BtnNew.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                BtnNewMouseClicked(evt);
            }  });
        BtnSave.setText("Save");
        BtnSave.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                BtnSaveActionPerformed(evt);
            } });
        BtnEdit.setText("Edit");
        BtnEdit.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                BtnEditActionPerformed(evt);
            } });
        BtnDelete.setText("Delete");
        BtnDelete.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                BtnDeleteActionPerformed(evt);
            } });
        BtnExit.setForeground(new java.awt.Color(255، 0، 0));
        BtnExit.setText("Exit");
        BtnExit.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                BtnExitMouseClicked(evt);
            } });
        TxtID.addFocusListener(new java.awt.event.FocusAdapter() {
            public void focusLost(java.awt.event.FocusEvent evt) {
                TxtIDFocusLost(evt);
            } });
        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jLabel6)
                    .addComponent(jLabel2)
```

```
                .addComponent(jLabel1)
                .addComponent(jLabel4)
                .addComponent(jLabel5)
                .addComponent(jLabel3))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING‹
false)
                    .addComponent(TxtSource)
                    .addComponent(TxtName)
                    .addComponent(TxtQty)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING‹
layout.createSequentialGroup()
                        .addComponent(BtnNew)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(BtnSave)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(BtnEdit)
  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(BtnDelete)
   .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(BtnExit))
                    .addComponent(TxtID))
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING‹
false)
                    .addComponent(CmbWareHouse‹
javax.swing.GroupLayout.Alignment.LEADING‹ 0‹
javax.swing.GroupLayout.DEFAULT_SIZE‹ Short.MAX_VALUE)
                    .addComponent(TxtDate‹ javax.swing.GroupLayout.Alignment.LEADING‹
javax.swing.GroupLayout.DEFAULT_SIZE‹ 97‹ Short.MAX_VALUE)))
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE‹ Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(43‹ 43‹ 43)
 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel6)
                    .addComponent(CmbWareHouse‹ javax.swing.GroupLayout.PREFERRED_SIZE‹
javax.swing.GroupLayout.DEFAULT_SIZE‹ javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel1)
                .addComponent(TxtDate‹ javax.swing.GroupLayout.PREFERRED_SIZE‹
javax.swing.GroupLayout.DEFAULT_SIZE‹ javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel2)
                .addComponent(TxtID‹ javax.swing.GroupLayout.PREFERRED_SIZE‹
javax.swing.GroupLayout.DEFAULT_SIZE‹ javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel3)
                .addComponent(TxtName‹ javax.swing.GroupLayout.PREFERRED_SIZE‹
javax.swing.GroupLayout.DEFAULT_SIZE‹ javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel4)
                .addComponent(TxtQty‹ javax.swing.GroupLayout.PREFERRED_SIZE‹
javax.swing.GroupLayout.DEFAULT_SIZE‹ javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel5)
                .addComponent(TxtSource‹ javax.swing.GroupLayout.PREFERRED_SIZE‹
javax.swing.GroupLayout.DEFAULT_SIZE‹ javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(26‹ 26‹ 26)
  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(BtnExit)
                .addComponent(BtnDelete)
                .addComponent(BtnEdit)
                .addComponent(BtnSave)
                .addComponent(BtnNew))
            .addContainerGap(22‹ Short.MAX_VALUE))
        );
        pack();
    }// </editor-fold>
    private void BtnNewMouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        Enable(true);
        ClearForm();
}
    private void BtnSaveActionPerformed(java.awt.event.ActionEvent evt) {
```

```
        // TODO add your handling code here:
        if (TxtID.getText().compareTo("")!=0 && TxtID.getText().compareTo("")!= 0 ){
            try{
                if (Updat){
                    stmt.executeUpdate("delete  from APP.WRHMOVE where MOVID = " + ID);
                    Updat = false;
                }
                Temp = stmt.executeQuery("select * From APP.WRHMOVE ");
                ID = Temp.getFetchSize() + 1 ;
                String Str ="INSERT INTO APP.WRHMOVE (MOVID‹ ITMNO‹ QUANTITYIN‹
WRHNO‹ QUANTITYOUT‹ MOVDATE) VALUES          (" + ID + "‹ " + TxtID.getText() +"‹ 0‹ "
+ CmbWareHouse.getSelectedItem() + "‹ " + TxtQty.getText() + "‹ '" +  TxtDate.getText() +
"')" ;
                stmt.executeUpdate(Str);
                Enable(false);
                BtnEnable(true);
                JOptionPane.showMessageDialog(null‹"Save seccesufully");
            }catch (SQLException e){
                JOptionPane.showMessageDialog(null‹e);
            }
        }else
            JOptionPane.showMessageDialog(null‹ "Error!! Please check the input data");
}
    private void BtnEditActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Enable(true);
        ID = Integer.parseInt(TxtID.getText());
        Updat = true;
        BtnEnable(false);
}
    private void BtnDeleteActionPerformed(java.awt.event.ActionEvent evt) {
        //TODO add your handling code here:
        try{
            if (JOptionPane.showInternalConfirmDialog(null‹ "If you Delete this record‹ you
cann't return it back. Are you sure you want to delete this record?"‹
"Delete"‹JOptionPane.YES_NO_OPTION )== 0) {
                ID = Integer.parseInt(TxtID.getText());
                stmt.executeUpdate("delete  from APP.WAREHOUSE where WHRID = " + ID);
                ID =-1;
                Enable(true);
                ClearForm();
                BtnEnable(false);
```

```
            }
        }catch (SQLException e){
            JOptionPane.showMessageDialog(null‹e);
        }
}
    private void BtnExitMouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        this.setVisible(false);
        this.dispose();
}
    private void TxtIDFocusLost(java.awt.event.FocusEvent evt) {
        // TODO add your handling code here:
        if (IsInteger(TxtID.getText())) {
            try{
                ResultSet Rs = stmt.executeQuery("select * from APP.ITEM where ITMNO=" +
TxtID.getText());
                if (Rs.next())
                    TxtName.setText(Rs.getString("ITMNAME"));
                else{
                    JOptionPane.showMessageDialog(null‹"The Item number you entered is not
correct. Please check it and try agan");
                    TxtID.setText("");
                }

            }catch (SQLException e){
                JOptionPane.showMessageDialog(null‹e);
            } } else
                JOptionPane.showMessageDialog(null‹ "you must entered integer value in this
field! please check the input and try again");
    }
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new FrmDepitStock().setVisible(true);
            } } );
    }
    public static javax.swing.JComboBox CmbWareHouse;
    public static javax.swing.JFormattedTextField TxtDate;
    public static javax.swing.JTextField TxtID;
    public static javax.swing.JTextField TxtName;
    public static javax.swing.JTextField TxtQty;
    public static javax.swing.JTextField TxtSource;
```

```
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    // End of variables declaration
}
```

3- FrmItem Class:

```
package StockingAspect;
import javax.swing.*;
import java.sql.*;
public class FrmItem extends MyBase {
    Connection con;
    Statement stmt;
    ResultSet rs ، Temp ;
    public  int ID;
    boolean updat;
      public FrmItem() {
       initComponents();
        try{
        con = DriverManager.getConnection("jdbc:derby://localhost:1527/StokingDB"، null،
null);
        stmt = con.createStatement();
        rs = stmt.executeQuery("select * from APP.ITEM");
     }catch(SQLException e){
          System.err.println(e);
           JOptionPane.showMessageDialog(null،e);
        }
        Enable(false);
        BtnEnable(false);
        updat = false;
    }
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        jDialog1 = new javax.swing.JDialog();
        jFrame1 = new javax.swing.JFrame();
        BtnExit = new javax.swing.JButton();
        BtnDelete = new javax.swing.JButton();
        BtnEdit = new javax.swing.JButton();
```

```
        BtnSave = new javax.swing.JButton();
        BtnNew = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        TxtID = new javax.swing.JTextField();
        TxtName = new javax.swing.JTextField();
        TxtOrigin = new javax.swing.JTextField();
        TxtCapacity = new javax.swing.JTextField();
        TxtNotes = new javax.swing.JTextField();
        javax.swing.GroupLayout jDialog1Layout = new
javax.swing.GroupLayout(jDialog1.getContentPane());
        jDialog1.getContentPane().setLayout(jDialog1Layout);
        jDialog1Layout.setHorizontalGroup(
            jDialog1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 400, Short.MAX_VALUE)
        );
        jDialog1Layout.setVerticalGroup(
            jDialog1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 300, Short.MAX_VALUE)
        );
        javax.swing.GroupLayout jFrame1Layout = new
javax.swing.GroupLayout(jFrame1.getContentPane());
        jFrame1.getContentPane().setLayout(jFrame1Layout);
        jFrame1Layout.setHorizontalGroup(
            jFrame1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 400, Short.MAX_VALUE)
        );
        jFrame1Layout.setVerticalGroup(
            jFrame1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 300, Short.MAX_VALUE)
        );
        BtnExit.setForeground(new java.awt.Color(255, 0, 0));
        BtnExit.setText("Exit");
        BtnExit.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                BtnExitMouseClicked(evt);
            } });
        BtnDelete.setText("Delete");
        BtnDelete.addActionListener(new java.awt.event.ActionListener() {
```

```
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            BtnDeleteActionPerformed(evt);
        }  });
        BtnEdit.setText("Edit");
        BtnEdit.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                BtnEditActionPerformed(evt);
            } });
        BtnSave.setText("Save");
        BtnSave.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                BtnSaveActionPerformed(evt);
            } });
        BtnNew.setText("New");
        BtnNew.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                BtnNewMouseClicked(evt);
            }  });
        jLabel1.setText("Item ID:");
        jLabel2.setText("Item Name:");
        jLabel3.setText("Item Origin:");
        jLabel4.setText("Capacity:");
        jLabel5.setText("Notes:");
        TxtID.addFocusListener(new java.awt.event.FocusAdapter() {
            public void focusLost(java.awt.event.FocusEvent evt) {
                TxtIDFocusLost(evt);
            }   });
        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
      .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jLabel5)
                    .addComponent(jLabel3)
                    .addComponent(jLabel2)
                    .addComponent(jLabel1)
                    .addComponent(jLabel4))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
                .addComponent(TxtID، javax.swing.GroupLayout.PREFERRED_SIZE، 96،
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(TxtCapacity، javax.swing.GroupLayout.PREFERRED_SIZE، 96،
javax.swing.GroupLayout.PREFERRED_SIZE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING،
false)
                    .addComponent(TxtNotes، javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(TxtOrigin، javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(TxtName، javax.swing.GroupLayout.Alignment.LEADING،
javax.swing.GroupLayout.DEFAULT_SIZE، 228، Short.MAX_VALUE)))
            .addContainerGap(24، Short.MAX_VALUE))
          .addGroup(javax.swing.GroupLayout.Alignment.TRAILING،
layout.createSequentialGroup()
            .addContainerGap(14، Short.MAX_VALUE)
            .addComponent(BtnNew)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(BtnSave)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(BtnEdit)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(BtnDelete)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(BtnExit)
            .addContainerGap()));
        layout.setVerticalGroup(
          layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
          .addGroup(layout.createSequentialGroup()
            .addGap(48، 48، 48)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel1)
                .addComponent(TxtID، javax.swing.GroupLayout.PREFERRED_SIZE،
javax.swing.GroupLayout.DEFAULT_SIZE، javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel2)
                .addComponent(TxtName، javax.swing.GroupLayout.PREFERRED_SIZE،
javax.swing.GroupLayout.DEFAULT_SIZE، javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel3)
```

```
                .addComponent(TxtOrigin، javax.swing.GroupLayout.PREFERRED_SIZE،
javax.swing.GroupLayout.DEFAULT_SIZE، javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel4)
                .addComponent(TxtCapacity، javax.swing.GroupLayout.PREFERRED_SIZE،
javax.swing.GroupLayout.DEFAULT_SIZE، javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel5)
                .addComponent(TxtNotes، javax.swing.GroupLayout.PREFERRED_SIZE،
javax.swing.GroupLayout.DEFAULT_SIZE، javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18، 18، 18)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(BtnExit)
                .addComponent(BtnDelete)
                .addComponent(BtnEdit)
                .addComponent(BtnSave)
                .addComponent(BtnNew))
            .addContainerGap(31، Short.MAX_VALUE)));
        getAccessibleContext().setAccessibleParent(this);
        pack();
    }// </editor-fold>
    private void BtnNewMouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        Enable(true);
        ClearForm();    }
    private void BtnExitMouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
       // JOptionPane.showMessageDialog(null،ID);
        this.setVisible(false);
        this.dispose();    }
    private void TxtIDFocusLost(java.awt.event.FocusEvent evt) {
        // TODO add your handling code here:
        try{
         if (TxtID.getText().compareTo("")!= 0){
         Temp = stmt.executeQuery("select * from APP.ITEM where ITMNO = " +
TxtID.getText());
         if (Temp.next()){
            TxtID.setText(Temp.getString("ITMNO")) ;
            TxtName.setText(Temp.getString("ITMNAME"));
            TxtCapacity.setText(Temp.getString("ITMPROPERTY1"));
```

```
            TxtOrigin.setText(Temp.getString("ITMPROPERTY2"));
            TxtNotes.setText(Temp.getString("ITMPROPERTY3"));
            Enable(false);
            BtnEnable(true);     } }
          }catch (SQLException e){
            JOptionPane.showMessageDialog(null‹e);   } }
      private void BtnEditActionPerformed(java.awt.event.ActionEvent evt) {
           Enable(true);
          ID = Integer.parseInt(TxtID.getText());
          updat = true;
          BtnEnable(false);    }
      private void BtnDeleteActionPerformed(java.awt.event.ActionEvent evt) {
            try{
           if (JOptionPane.showInternalConfirmDialog(null‹ "If you Delete this record‹ you
cann't return it back. Are you sure you want to delete this record?"‹
"Delete"‹JOptionPane.YES_NO_OPTION )== 0) {
           ID = Integer.parseInt(TxtID.getText());
           stmt.executeUpdate("delete  from APP.ITEM where ITMNO = " + ID);
           ID =-1;
           Enable(true);
           ClearForm();
           BtnEnable(false); }
          }catch (SQLException e){
           JOptionPane.showMessageDialog(null‹e);}}
      private void BtnSaveActionPerformed(java.awt.event.ActionEvent evt) {
          // TODO add your handling code here:
          if (TxtID.getText().compareTo("")!=0 && TxtName.getText().compareTo("")!= 0 &&
IsInteger(TxtID.getText())){
            try{              if (updat){
                  stmt.executeUpdate("delete  from APP.ITEM where ITMNO = " + ID);
                  updat = false;
               }
               ID = Integer.parseInt(TxtID.getText());
               stmt.executeUpdate("INSERT INTO APP.ITEM (ITMNO‹ ITMNAME‹
ITMPROPERTY1‹ ITMPROPERTY2‹ ITMPROPERTY3‹ ITMPROPERTY4) VALUES (" + ID +"‹ '" +
TxtName.getText() + "'‹ '" + TxtCapacity.getText() + "'‹ '" + TxtOrigin.getText() + "'‹ '" +
TxtNotes.getText() + "'‹ ')");
               Enable(false);
               BtnEnable(true);
               JOptionPane.showMessageDialog(null‹"Save seccesufully");
            }catch (SQLException e){
               JOptionPane.showMessageDialog(null‹e); }
```

```
      }else  JOptionPane.showMessageDialog(null، "Error!! Please check the input data");}
   public void ClearForm() {
   TxtCapacity.setText("");
   TxtID.setText("");
   TxtName.setText("");
   TxtNotes.setText("");
   TxtOrigin.setText("");
   BtnEnable(false);
   updat=false; }
void BtnEnable(boolean b){
   BtnDelete.setEnabled(b);
   BtnEdit.setEnabled(b);
   BtnSave.setEnabled(!b);
   // sec();}
  public void Enable(boolean b){
      TxtCapacity.setEnabled(b);
      TxtID.setEditable(b);
      TxtName.setEnabled(b);
      TxtNotes.setEnabled(b);
      TxtOrigin.setEnabled(b);    }
   public static void main(String args[]) {
      java.awt.EventQueue.invokeLater(new Runnable() {
         public void run() {
            new FrmItem().setVisible(true);  } });    }
   public static javax.swing.JTextField TxtCapacity;
   public static javax.swing.JTextField TxtID;
   public static javax.swing.JTextField TxtName;
   public static javax.swing.JTextField TxtNotes;
   public static javax.swing.JTextField TxtOrigin;
   private javax.swing.JDialog jDialog1;
   private javax.swing.JFrame jFrame1;
   private javax.swing.JLabel jLabel1;
   private javax.swing.JLabel jLabel2;
   private javax.swing.JLabel jLabel3;
   private javax.swing.JLabel jLabel4;
   private javax.swing.JLabel jLabel5;
  }
```

4- FrmMain Class:

```
package StockingAspect;
public class FrmMain extends javax.swing.JFrame {
```

```
    public FrmMain() {
        initComponents();
    }
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        jDesktopPane1 = new javax.swing.JDesktopPane();
        jMenuBar1 = new javax.swing.JMenuBar();
        jMenu1 = new javax.swing.JMenu();
        jMenuItem1 = new javax.swing.JMenuItem();
        jMenuItem2 = new javax.swing.JMenuItem();
        jMenuItem3 = new javax.swing.JMenuItem();
        jMenu2 = new javax.swing.JMenu();
        jMenuItem4 = new javax.swing.JMenuItem();
        jMenuItem5 = new javax.swing.JMenuItem();
        jMenu3 = new javax.swing.JMenu();
        jMenuItem6 = new javax.swing.JMenuItem();
        jMenu4 = new javax.swing.JMenu();
        jMenuItem7 = new javax.swing.JMenuItem();
        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        jMenu1.setText("Control");
jMenuItem1.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent
.VK_A, java.awt.event.InputEvent.CTRL_MASK));
        jMenuItem1.setText("Add New Item");
        jMenuItem1.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                jMenuItem1MouseClicked(evt);          }     });
        jMenuItem1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem1ActionPerformed(evt);       }    });
        jMenu1.add(jMenuItem1);
jMenuItem2.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent
.VK_W, java.awt.event.InputEvent.CTRL_MASK));
        jMenuItem2.setText("Add WareHouse");
        jMenuItem2.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                jMenuItem2MouseClicked(evt);     }    });
        jMenuItem2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem2ActionPerformed(evt);       }    });
        jMenu1.add(jMenuItem2);
```

```
jMenuItem3.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent
.VK_U‹ java.awt.event.InputEvent.CTRL_MASK));
        jMenuItem3.setText("Control the Users");
        jMenuItem3.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                jMenuItem3MouseClicked(evt);      }   });
        jMenuItem3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem3ActionPerformed(evt);   }  });
        jMenu1.add(jMenuItem3);
        jMenuBar1.add(jMenu1);
        jMenu2.setText("Stoking");
jMenuItem4.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent
.VK_D‹ java.awt.event.InputEvent.CTRL_MASK));
        jMenuItem4.setText("Depit the stock");
        jMenuItem4.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                jMenuItem4MouseClicked(evt);          }     });
        jMenuItem4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem4ActionPerformed(evt);          }       });
        jMenu2.add(jMenuItem4);
jMenuItem5.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent
.VK_C‹ java.awt.event.InputEvent.CTRL_MASK));
        jMenuItem5.setText("Credit the stock");
        jMenuItem5.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                jMenuItem5MouseClicked(evt);          }       });
        jMenuItem5.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem5ActionPerformed(evt);          }       });
        jMenu2.add(jMenuItem5);
        jMenuBar1.add(jMenu2);
        jMenu3.setText("Reports");
jMenuItem6.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent
.VK_Q‹ java.awt.event.InputEvent.SHIFT_MASK));
        jMenuItem6.setText("Item Quantity");
        jMenu3.add(jMenuItem6);
        jMenuBar1.add(jMenu3);
        jMenu4.setText("Help");
jMenuItem7.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent
.VK_B‹ java.awt.event.InputEvent.CTRL_MASK));
```

```
        jMenuItem7.setText("About");
        jMenu4.add(jMenuItem7);
        jMenuBar1.add(jMenu4);
        setJMenuBar(jMenuBar1);
        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jDesktopPane1‹ javax.swing.GroupLayout.DEFAULT_SIZE‹ 788‹
Short.MAX_VALUE)       );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jDesktopPane1‹ javax.swing.GroupLayout.DEFAULT_SIZE‹ 589‹
Short.MAX_VALUE)    );
        pack();
    }// </editor-fold>
    private void jMenuItem1MouseClicked(java.awt.event.MouseEvent evt) {    }
    private void jMenuItem2MouseClicked(java.awt.event.MouseEvent evt) {     }
    private void jMenuItem3MouseClicked(java.awt.event.MouseEvent evt) {     }
    private void jMenuItem4MouseClicked(java.awt.event.MouseEvent evt) {       }
    private void jMenuItem5MouseClicked(java.awt.event.MouseEvent evt) {       }
    private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
        FrmItem frm = new FrmItem();
        jDesktopPane1.add(frm);
        frm.setVisible(true);
        frm.setTitle("Adding New Item");    }
    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
        FrmWarehouse frm = new FrmWarehouse();
        jDesktopPane1.add(frm);
        frm.setVisible(true);
        frm.setTitle("Adding New Warehouse");   }
    private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
        FrmUsers  frm  = new FrmUsers();
        jDesktopPane1.add(frm);
        frm.setVisible(true);
        frm.setTitle("Users Control");    }
    private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {
         FrmDepitStock frm = new FrmDepitStock();
         jDesktopPane1.add(frm);
        frm.setVisible(true);
        frm.setTitle("Depit the stock");    }
    private void jMenuItem5ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
        FrmCreditStock frm = new FrmCreditStock();
        jDesktopPane1.add(frm);
        frm.setVisible(true);
        frm.setTitle("Credit the Stock");    }
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new FrmMain().setVisible(true);          }       });    }
    private javax.swing.JDesktopPane jDesktopPane1;
    public static javax.swing.JMenu jMenu1;
    private javax.swing.JMenu jMenu2;
    private javax.swing.JMenu jMenu3;
    private javax.swing.JMenu jMenu4;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JMenuItem jMenuItem1;
    private javax.swing.JMenuItem jMenuItem2;
    private javax.swing.JMenuItem jMenuItem3;
    private javax.swing.JMenuItem jMenuItem4;
    private javax.swing.JMenuItem jMenuItem5;
    private javax.swing.JMenuItem jMenuItem6;
    private javax.swing.JMenuItem jMenuItem7;
 }
```

5- FrmUsers Class

```
package StockingAspect;
import java.sql.*;
import javax.swing.JOptionPane;
public class FrmUsers extends MyBase {
    Statement stmt;
    Connection con;
    ResultSet Temp‹rs;
    int ID;
    boolean Updat;
    String Priv;
    public FrmUsers() {
        initComponents();
        try{  con = DriverManager.getConnection("jdbc:derby://localhost:1527/StokingDB"‹
null‹ null);
         stmt = con.createStatement();
         rs = stmt.executeQuery("select * from APP.USERTB");
     }catch(SQLException e){
           JOptionPane.showMessageDialog(null‹e);       }
```

```
        Enable(false);
        Updat = false;
        BtnEnable(false);
        try{
         Temp = stmt.executeQuery("select * from APP.WAREHOUSE");
         while (Temp.next()){
            CmbWHR.addItem(Temp.getString("WRHID"));        }
        }catch (SQLException e){
           JOptionPane.showMessageDialog(null, e);       }    }
        public void ClearForm() {
        TxtID.setText("");
        TxtName.setText("");
        TxtPass.setText("");
        TxtAccessName.setText("");
        ChkDelete.setSelected(false);
        ChkInsert.setSelected(false);
        ChkModify.setSelected(false);
        BtnEnable(false);
        Updat=false;    }
void BtnEnable(boolean b){
        BtnDelete.setEnabled(b);
        BtnEdit.setEnabled(b);
        BtnSave.setEnabled(!b);}
   public void Enable(boolean b){
        TxtID.setEditable(b);
        TxtName.setEnabled(b);
        TxtAccessName.setEnabled(b);
        TxtPass.setEnabled(b);
        CmbWHR.setEnabled(b);
        ChkDelete.setEnabled(b);
        ChkInsert.setEnabled(b);
        ChkModify.setEnabled(b);    }
     // </editor-fold>
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        jLabel3 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel1 = new javax.swing.JLabel();
        TxtID = new javax.swing.JTextField();
        TxtAccessName = new javax.swing.JTextField();
        TxtName = new javax.swing.JTextField();
```

```
BtnNew = new javax.swing.JButton();
BtnSave = new javax.swing.JButton();
BtnEdit = new javax.swing.JButton();
BtnDelete = new javax.swing.JButton();
BtnExit = new javax.swing.JButton();
TxtPass = new javax.swing.JTextField();
jLabel4 = new javax.swing.JLabel();
jPanel1 = new javax.swing.JPanel();
ChkInsert = new javax.swing.JCheckBox();
ChkDelete = new javax.swing.JCheckBox();
ChkModify = new javax.swing.JCheckBox();
jLabel5 = new javax.swing.JLabel();
CmbWHR = new javax.swing.JComboBox();
jLabel3.setText("User Access Name:");
jLabel2.setText("User Name:");
jLabel1.setText("User ID:");
TxtID.addFocusListener(new java.awt.event.FocusAdapter() {
   public void focusLost(java.awt.event.FocusEvent evt) {
      TxtIDFocusLost(evt);          }       });
BtnNew.setText("New");
BtnNew.addMouseListener(new java.awt.event.MouseAdapter() {
   public void mouseClicked(java.awt.event.MouseEvent evt) {
      BtnNewMouseClicked(evt);          }       });
BtnSave.setText("Save");
BtnSave.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      BtnSaveActionPerformed(evt);          }       });
BtnEdit.setText("Edit");
BtnEdit.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      BtnEditActionPerformed(evt);          }       });
BtnDelete.setText("Delete");
BtnDelete.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      BtnDeleteActionPerformed(evt);          }       });
BtnExit.setForeground(new java.awt.Color(255، 0، 0));
BtnExit.setText("Exit");
BtnExit.addMouseListener(new java.awt.event.MouseAdapter() {
   public void mouseClicked(java.awt.event.MouseEvent evt) {
      BtnExitMouseClicked(evt);          }    });
jLabel4.setText("User Password:");
jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder("User Privillige"));
```

```
        ChkInsert.setText("Insert");
        ChkDelete.setText("Delete");
        ChkModify.setText("Modify");
        jLabel5.setText("WareHouse:");
        CmbWHR.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "All" }));

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILI
NG)
                    .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
jPanel1Layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jLabel5)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(CmbWHR, 0, 192, Short.MAX_VALUE))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addComponent(ChkInsert)
                        .addGap(46, 46, 46)
                        .addComponent(ChkDelete)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
51, Short.MAX_VALUE)
                        .addComponent(ChkModify)))
                .addGap(52, 52, 52))      );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASEL
INE)
                    .addComponent(ChkInsert)
                    .addComponent(ChkModify)
                    .addComponent(ChkDelete))
                .addGap(18, 18, 18)
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASEL
INE)
                    .addComponent(jLabel5)
                    .addComponent(CmbWHR, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE، Short.MAX_VALUE))
        );
        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jLabel3)
                    .addComponent(jLabel2)
                    .addComponent(jLabel1)
                    .addComponent(jLabel4))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING،
false)
                    .addComponent(TxtID، javax.swing.GroupLayout.PREFERRED_SIZE، 96،
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(TxtAccessName)
                    .addComponent(TxtName، javax.swing.GroupLayout.DEFAULT_SIZE، 228،
Short.MAX_VALUE)
                    .addComponent(TxtPass))
                .addContainerGap(81، Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING،
layout.createSequentialGroup()
                .addContainerGap(71، Short.MAX_VALUE)
                .addComponent(BtnNew)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(BtnSave)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(BtnEdit)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(BtnDelete)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(BtnExit)
                .addGap(45، 45، 45))
            .addGroup(layout.createSequentialGroup()
                .addGap(51، 51، 51)
                .addComponent(jPanel1، javax.swing.GroupLayout.PREFERRED_SIZE،
javax.swing.GroupLayout.DEFAULT_SIZE، javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(30، Short.MAX_VALUE))
        );
```

```
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(30، 30، 30)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(TxtID، javax.swing.GroupLayout.PREFERRED_SIZE،
javax.swing.GroupLayout.DEFAULT_SIZE، javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(TxtName، javax.swing.GroupLayout.PREFERRED_SIZE،
javax.swing.GroupLayout.DEFAULT_SIZE، javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel3)
                    .addComponent(TxtAccessName، javax.swing.GroupLayout.PREFERRED_SIZE،
javax.swing.GroupLayout.DEFAULT_SIZE، javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel4)
                    .addComponent(TxtPass، javax.swing.GroupLayout.PREFERRED_SIZE،
javax.swing.GroupLayout.DEFAULT_SIZE، javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(24، 24، 24)
                .addComponent(jPanel1، javax.swing.GroupLayout.PREFERRED_SIZE،
javax.swing.GroupLayout.DEFAULT_SIZE، javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(36، 36، 36)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(BtnExit)
                    .addComponent(BtnDelete)
                    .addComponent(BtnEdit)
                    .addComponent(BtnSave)
                    .addComponent(BtnNew))
                .addGap(23، 23، 23))  );
        pack();
    }// </editor-fold>
    private void TxtIDFocusLost(java.awt.event.FocusEvent evt) {
        // TODO add your handling code here:
        try{
            if (TxtID.getText().compareTo("")!= 0){
                Temp = stmt.executeQuery("select * from APP.USERTB where USRID= " +
TxtID.getText());
```

```
            if (Temp.next()){
               TxtID.setText(Temp.getString("USRID")) ;
               TxtName.setText(Temp.getString("USRNAME"));
               TxtAccessName.setText(Temp.getString("USRACCESNAME"));
               TxtPass.setText(Temp.getString("USRPASS"));
               Priv= Temp.getString("USRPRIVILLIGE");
               CmbWHR.setSelectedItem(Temp.getString("USRWHR"));
               String Pr[] = Priv.split(";");
               int x = Pr.length;
               for (int i = 0 ; i<x ; i++){
                  if (Pr[i].compareTo("Ins")==0)
                     ChkInsert.setSelected(true);
                  if (Pr[i].compareTo("Mod")==0)
                    ChkModify.setSelected(true);
                  if (Pr[i].compareTo("Del")==0)
                    ChkDelete.setSelected(true);                    }
               Enable(false);
               BtnEnable(true);             }          }
         }catch (SQLException e){
            JOptionPane.showMessageDialog(null‹e);       }}
      private void BtnNewMouseClicked(java.awt.event.MouseEvent evt) {
         // TODO add your handling code here:
         Enable(true);
         ClearForm();}
      private void BtnSaveActionPerformed(java.awt.event.ActionEvent evt) {
         if (TxtID.getText().compareTo("")!=0 && TxtName.getText().compareTo("")!= 0 &&
IsInteger(TxtID.getText())){
            try{
               if (Updat){
                  stmt.executeUpdate("delete  from APP.USERTB where USRID = " + ID);
                  Updat = false;             }
               Priv ="";
               if (ChkDelete.isSelected())
                  Priv += "Del;";
               if (ChkInsert.isSelected())
                  Priv += "Ins;" ;
               if (ChkModify.isSelected())
                  Priv += "Mod;";
               ID = Integer.parseInt(TxtID.getText());
               stmt.executeUpdate("INSERT INTO APP.USERTB (USRID‹ USRNAME‹ USRPASS‹
USRACCESNAME‹ USRPRIVILLIGE‹USRWHR) VALUES (" + ID +"‹ '" + TxtName.getText()+ "'‹
```

```
'" + TxtPass.getText() + "'‹ '" + TxtAccessName.getText() + "'‹ '" + Priv +"'‹'" +
CmbWHR.getSelectedItem() + "')");
            Enable(false);
            BtnEnable(true);
            JOptionPane.showMessageDialog(null‹"Save seccesufully");
          }catch (SQLException e){
            JOptionPane.showMessageDialog(null‹e);          }
       }else
          JOptionPane.showMessageDialog(null‹ "Error!! Please check the input data");}
   private void BtnEditActionPerformed(java.awt.event.ActionEvent evt) {
      Enable(true);
      ID = Integer.parseInt(TxtID.getText());
      Updat = true;
      BtnEnable(false);}
   private void BtnDeleteActionPerformed(java.awt.event.ActionEvent evt) {
            try{
         if (JOptionPane.showInternalConfirmDialog(null‹ "If you Delete this record‹ you
cann't return it back. Are you sure you want to delete this record?"‹
"Delete"‹JOptionPane.YES_NO_OPTION )== 0) {
            ID = Integer.parseInt(TxtID.getText());
            stmt.executeUpdate("delete  from APP.USERTB where USRID = " + ID);
            ID =-1;
            Enable(true);
            ClearForm();
            BtnEnable(false);
         }   }catch (SQLException e){
         JOptionPane.showMessageDialog(null‹e);       }}
   private void BtnExitMouseClicked(java.awt.event.MouseEvent evt) {
       this.setVisible(false);
      this.dispose();}
   public static void main(String args[]) {
      java.awt.EventQueue.invokeLater(new Runnable() {
         public void run() {
            new FrmUsers().setVisible(true);     }  });   }
   public static javax.swing.JCheckBox ChkDelete;
   public static javax.swing.JCheckBox ChkInsert;
   public static javax.swing.JCheckBox ChkModify;
   public static javax.swing.JComboBox CmbWHR;
   public static javax.swing.JTextField TxtAccessName;
   public static javax.swing.JTextField TxtID;
   public static javax.swing.JTextField TxtName;
   public static javax.swing.JTextField TxtPass;
```

```
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JPanel jPanel1;
   }
```

6- FrmWarehouse Class:

```
package StockingAspect;
import java.sql.*;
import javax.swing.JOptionPane;
public class FrmWarehouse extends MyBase {
    Connection con;
    Statement stmt;
    ResultSet rs ، Temp ;
    int ID;
    boolean updat;
       public FrmWarehouse() {
        initComponents();
        try{
         con = DriverManager.getConnection("jdbc:derby://localhost:1527/StokingDB"، null،
null);
         stmt = con.createStatement();
         rs = stmt.executeQuery("select * from APP.USERTB");
     }catch(SQLException e){
           JOptionPane.showMessageDialog(null،e);     }
        Enable(false);
        BtnEnable(false);
        updat = false;    }
    public void ClearForm() {
    TxtID.setText("");
    TxtName.setText("");
    TxtOrigin.setText("");
    BtnEnable(false);
    updat=false;    }
void BtnEnable(boolean b){
    BtnDelete.setEnabled(b);
    BtnEdit.setEnabled(b);
```

```
 BtnSave.setEnabled(!b);}
public void Enable(boolean b){
    TxtID.setEditable(b);
    TxtName.setEnabled(b);
    TxtOrigin.setEnabled(b);    }
 @SuppressWarnings("unchecked")
 // <editor-fold defaultstate="collapsed" desc="Generated Code">
 private void initComponents() {
    jLabel3 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jLabel1 = new javax.swing.JLabel();
    TxtID = new javax.swing.JTextField();
    TxtOrigin = new javax.swing.JTextField();
    TxtName = new javax.swing.JTextField();
    BtnNew = new javax.swing.JButton();
    BtnSave = new javax.swing.JButton();
    BtnEdit = new javax.swing.JButton();
    BtnDelete = new javax.swing.JButton();
    BtnExit = new javax.swing.JButton();
    jLabel3.setText("WareHouse Location:");
    jLabel2.setText("WareHouse Namse:");
    jLabel1.setText("WareHouse ID:");
    TxtID.addFocusListener(new java.awt.event.FocusAdapter() {
       public void focusLost(java.awt.event.FocusEvent evt) {
          TxtIDFocusLost(evt);          }       });
    BtnNew.setText("New");
    BtnNew.addMouseListener(new java.awt.event.MouseAdapter() {
       public void mouseClicked(java.awt.event.MouseEvent evt) {
          BtnNewMouseClicked(evt);          }       });
    BtnSave.setText("Save");
    BtnSave.addActionListener(new java.awt.event.ActionListener() {
       public void actionPerformed(java.awt.event.ActionEvent evt) {
          BtnSaveActionPerformed(evt);          }       });
    BtnEdit.setText("Edit");
    BtnEdit.addActionListener(new java.awt.event.ActionListener() {
       public void actionPerformed(java.awt.event.ActionEvent evt) {
          BtnEditActionPerformed(evt);          }       });
    BtnDelete.setText("Delete");
    BtnDelete.addActionListener(new java.awt.event.ActionListener() {
       public void actionPerformed(java.awt.event.ActionEvent evt) {
          BtnDeleteActionPerformed(evt);          }       });
    BtnExit.setForeground(new java.awt.Color(255، 0، 0));
```

```
        BtnExit.setText("Exit");
        BtnExit.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                BtnExitMouseClicked(evt);          }       });
        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
      .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
     .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                            .addComponent(jLabel3)
                            .addComponent(jLabel2)
                            .addComponent(jLabel1))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(TxtID‹ javax.swing.GroupLayout.PREFERRED_SIZE‹ 96‹
javax.swing.GroupLayout.PREFERRED_SIZE)
  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING‹
false)
                                .addComponent(TxtOrigin‹
javax.swing.GroupLayout.Alignment.LEADING)
                                .addComponent(TxtName‹
javax.swing.GroupLayout.Alignment.LEADING‹ javax.swing.GroupLayout.DEFAULT_SIZE‹
228‹ Short.MAX_VALUE))))
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(BtnNew)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(BtnSave)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(BtnEdit)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(BtnDelete)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(BtnExit)))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE‹ Short.MAX_VALUE))
);
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
```

```
            .addGap(48، 48، 48)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel1)
                .addComponent(TxtID، javax.swing.GroupLayout.PREFERRED_SIZE،
javax.swing.GroupLayout.DEFAULT_SIZE، javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel2)
                .addComponent(TxtName، javax.swing.GroupLayout.PREFERRED_SIZE،
javax.swing.GroupLayout.DEFAULT_SIZE، javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel3)
                .addComponent(TxtOrigin، javax.swing.GroupLayout.PREFERRED_SIZE،
javax.swing.GroupLayout.DEFAULT_SIZE، javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18، 18، 18)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(BtnExit)
                .addComponent(BtnDelete)
                .addComponent(BtnEdit)
                .addComponent(BtnSave)
                .addComponent(BtnNew))
            .addContainerGap(19، Short.MAX_VALUE))  );
        pack();
    }// </editor-fold>
    private void TxtIDFocusLost(java.awt.event.FocusEvent evt) {
        try{          if (TxtID.getText().compareTo("")!= 0){
              Temp = stmt.executeQuery("select * from APP.WAREHOUSE where WRHID= " +
TxtID.getText());
            if (Temp.next()){
                TxtID.setText(Temp.getString("WRHID")) ;
                TxtName.setText(Temp.getString("WRHNAME"));
                TxtOrigin.setText(Temp.getString("WRHLOC"));
                Enable(false);
                BtnEnable(true);    }         }
        }catch (SQLException e){
           JOptionPane.showMessageDialog(null،e);       }    }
    private void BtnNewMouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        Enable(true);
        ClearForm();}
```

```
    private void BtnSaveActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        if (TxtID.getText().compareTo("")!=0 && TxtName.getText().compareTo("")!= 0 &&
IsInteger(TxtID.getText())){
            try{              if (updat){
                  stmt.executeUpdate("delete  from APP.WAREHOUSE where WRHID = " + ID);
                  updat = false;              }
               ID = Integer.parseInt(TxtID.getText());
               stmt.executeUpdate("INSERT INTO APP.WAREHOUSE (WRHID‹ WRHNAME‹
WRHLOC) VALUES (" + ID + "‹ '" + TxtName.getText() + "'‹ '" + TxtOrigin.getText() + "')");
               Enable(false);
               BtnEnable(true);
               JOptionPane.showMessageDialog(null‹"Save seccesufully");
            }catch (SQLException e){
               JOptionPane.showMessageDialog(null‹e);          }
        }else
            JOptionPane.showMessageDialog(null‹ "Error!! Please check the input data");}
    private void BtnEditActionPerformed(java.awt.event.ActionEvent evt) {
              Enable(true);
        ID = Integer.parseInt(TxtID.getText());
        updat = true;
        BtnEnable(false);}
    private void BtnDeleteActionPerformed(java.awt.event.ActionEvent evt) {
        try{
            if (JOptionPane.showInternalConfirmDialog(null‹ "If you Delete this record‹ you
cann't return it back. Are you sure you want to delete this record?"‹
"Delete"‹JOptionPane.YES_NO_OPTION )== 0) {
               ID = Integer.parseInt(TxtID.getText());
               stmt.executeUpdate("delete  from APP.WAREHOUSE where WHRID = " + ID);
               ID =-1;
               Enable(true);
               ClearForm();
               BtnEnable(false);          }
        }catch (SQLException e){
            JOptionPane.showMessageDialog(null‹e);       }}
    private void BtnExitMouseClicked(java.awt.event.MouseEvent evt) {
         this.setVisible(false);
        this.dispose();}
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
               new FrmWarehouse().setVisible(true);          }       });    }
```

```
    public static javax.swing.JTextField TxtID;
    public static javax.swing.JTextField TxtName;
    public static javax.swing.JTextField TxtOrigin;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;   }
```

7- Login Class:

```
package StockingAspect;
public class Login extends javax.swing.JFrame {
    public Login() {
        initComponents();
        MainWin frm = new MainWin();
        jDesktopPane1.add(frm);
        frm.setTitle("Login Screen");
        frm.setVisible(true);    }
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        jDesktopPane1 = new javax.swing.JDesktopPane();
        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        jDesktopPane1.addComponentListener(new java.awt.event.ComponentAdapter() {
            public void componentHidden(java.awt.event.ComponentEvent evt) {
                jDesktopPane1ComponentHidden(evt);          }       });
        jDesktopPane1.addContainerListener(new java.awt.event.ContainerAdapter() {
            public void componentRemoved(java.awt.event.ContainerEvent evt) {
                jDesktopPane1ComponentRemoved(evt);          }       });
        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addComponent(jDesktopPane1، javax.swing.GroupLayout.DEFAULT_SIZE، 450،
Short.MAX_VALUE)
                .addContainerGap())       );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addComponent(jDesktopPane1، javax.swing.GroupLayout.DEFAULT_SIZE، 277،
Short.MAX_VALUE)
                .addContainerGap())       );
        pack();
```

```
   }// </editor-fold>
   private void jDesktopPane1ComponentHidden(java.awt.event.ComponentEvent evt) {
      this.dispose();    }
   private void jDesktopPane1ComponentRemoved(java.awt.event.ContainerEvent evt) {
      this.dispose();   }
   public static void main(String args[]) {
      java.awt.EventQueue.invokeLater(new Runnable() {
         public void run() {
            new Login().setVisible(true);          }       });    }
   private javax.swing.JDesktopPane jDesktopPane1;
}
```

8- MainWin Class

```
package StockingAspect;
import javax.swing.*;
import java.sql.*;
import java.util.Arrays;
public class MainWin extends MyBase {
   Connection con;
   Statement stmt;
   ResultSet rs ;
   public MainWin() {
      initComponents();
      //jDesktopPane1.add(this);    }
   @SuppressWarnings("unchecked")
   // <editor-fold defaultstate="collapsed" desc="Generated Code">
   private void initComponents() {
      bindingGroup = new org.jdesktop.beansbinding.BindingGroup();
      jDesktopPane1 = new javax.swing.JDesktopPane();
      TxtUser = new javax.swing.JTextField();
      TxtPassword = new javax.swing.JPasswordField();
      jLabel1 = new javax.swing.JLabel();
      jLabel2 = new javax.swing.JLabel();
      BtnOk = new javax.swing.JButton();
      BtnCancel = new javax.swing.JButton();
      setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
      setName("FrmLogin"); // NOI18N
      org.jdesktop.beansbinding.Binding binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBi
nding.UpdateStrategy.READ_WRITE، this،
```

```
org.jdesktop.beansbinding.ELProperty.create("Login Form")‹ this‹
org.jdesktop.beansbinding.BeanProperty.create("title"));
        bindingGroup.addBinding(binding);
        TxtPassword.setNextFocusableComponent(BtnOk);
        jLabel1.setText("User Name:");
        jLabel2.setText("Password:");
        BtnOk.setText("OK");
        BtnOk.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                BtnOkMouseClicked(evt);          }       });
        BtnCancel.setText("Cancel");
        BtnCancel.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                BtnCancelMouseClicked(evt);          }       });
        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING‹
layout.createSequentialGroup()
                .addContainerGap(81‹ Short.MAX_VALUE)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jLabel2)
                    .addComponent(jLabel1))
                .addGap(18‹ 18‹ 18)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING‹
false)
                    .addComponent(TxtPassword‹ javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(TxtUser‹ javax.swing.GroupLayout.Alignment.LEADING‹
javax.swing.GroupLayout.DEFAULT_SIZE‹ 183‹ Short.MAX_VALUE))
                .addGap(62‹ 62‹ 62))
            .addGroup(layout.createSequentialGroup()
                .addGap(177‹ 177‹ 177)
                .addComponent(BtnOk)
                .addGap(18‹ 18‹ 18)
                .addComponent(BtnCancel)
                .addContainerGap(93‹ Short.MAX_VALUE))       );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(106‹ 106‹ 106)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
```

```
                .addComponent(TxtUser، javax.swing.GroupLayout.PREFERRED_SIZE،
javax.swing.GroupLayout.DEFAULT_SIZE، javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel1))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(TxtPassword، javax.swing.GroupLayout.PREFERRED_SIZE،
javax.swing.GroupLayout.DEFAULT_SIZE، javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel2))
            .addGap(30، 30، 30)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(BtnCancel)
                .addComponent(BtnOk))
            .addContainerGap(15، Short.MAX_VALUE))       );
        bindingGroup.bind();
        pack();
    }// </editor-fold>
    private void BtnOkMouseClicked(java.awt.event.MouseEvent evt) {
        try{
         con = DriverManager.getConnection("jdbc:derby://localhost:1527/StokingDB"، null،
null);
         stmt = con.createStatement();
         rs = stmt.executeQuery("select * from APP.USERTB");
     }catch(SQLException e){
           System.err.println(e);
            JOptionPane.showMessageDialog(null،e);       }
        try{
           //String UsrName ;
           char UsrPass[]، pass[];
           boolean res = false;
        while (rs.next()) {
        UsrName = rs.getString("USRNAME");
        UsrID = rs.getInt("USRID");
        UsrPrivilige = rs.getString("USRPRIVILLIGE");
        UsrPass = rs.getString("USRPASS").toCharArray();
        pass =TxtPassword.getPassword();
        if (TxtUser.getText().compareTo(UsrName) == 0 && Arrays.equals(UsrPass، pass) ){
          res = true;
          break;    }       }
        if (res){
           FrmMain frm = new FrmMain();
           frm.setVisible(true);
          this.setVisible(false);
```

```
          this.dispose();
        }else
           JOptionPane.showMessageDialog(null، "The user name or the password is
incorrect");
        }catch(SQLException e){
           System.err.println(e);
           JOptionPane.showMessageDialog(null،e);       }    }
    private void BtnCancelMouseClicked(java.awt.event.MouseEvent evt) {
       this.setVisible(false);
       System.exit(0);    }
    public static void main(String args[]) {
       java.awt.EventQueue.invokeLater(new Runnable() {
          public void run() {
             new MainWin().setVisible(true);          }       });
       try{
       Class.forName("org.apache.derby.jdbc.ClientDriver");
       }catch(ClassNotFoundException e){
          System.out.println(e);
          JOptionPane.showMessageDialog(null، e);       }    }
    public static javax.swing.JButton BtnCancel;
    public static javax.swing.JButton BtnOk;
    public static javax.swing.JPasswordField TxtPassword;
    public static javax.swing.JTextField TxtUser;
    private javax.swing.JDesktopPane jDesktopPane1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private org.jdesktop.beansbinding.BindingGroup bindingGroup;}
```

9- Aspect (MyAspect):

```
package StockingAspect;
import javax.swing.JOptionPane;
public aspect MyAspectC {
    after():execution( * *.BtnEnable(..)){
         MyBase.sec();
         JOptionPane.showMessageDialog(null، this.getClass().getName());
     }
    after():initialization(*.new(..)) {
       MyBase.rset();
       JOptionPane.showMessageDialog(null، this.getClass().getName());
    }
```

```
      public static void MyBase.sec(){
       BtnDelete.setEnabled(false);
       BtnSave.setEnabled(false);
       BtnEdit.setEnabled(false);
       String Str[] = UsrPrivilige.split(";");
       int x = Str.length;
       for (int i =0 ; i<x ; i++){
         if (Str[i].compareTo("Ins")==0)
             BtnSave.setEnabled(true);
         if (Str[i].compareTo("Mod")==0)
            BtnEdit.setEnabled(true);
         if (Str[i].compareTo("Del")==0)
           BtnDelete.setEnabled(true);
       }
}
       public static void MyBase.rset(){
         BtnSave.setEnabled(false);
         BtnEdit.setEnabled(false);
         BtnDelete.setEnabled(false);
   }
}
```

10- MyBase Class

```
package StockingAspect;
public class MyBase extends javax.swing.JInternalFrame {
public static int UsrID;
   public static String UsrName;
   public static String UsrAccess;
   public static String UsrPrivilige;
   public static javax.swing.JButton BtnDelete;
   public static javax.swing.JButton BtnEdit;
   public static javax.swing.JButton BtnExit;
   public static javax.swing.JButton BtnNew;
   public static javax.swing.JButton BtnSave;
   public void GetPrivilige(){    }
   public boolean IsInteger(String s){
    try{
       Integer.parseInt(s);
       return true;
    }catch (Exception e){
       return false;    }   }}
```