تحسين جودة إجراءات هندسة البرمجيات المُقادة بالنماذج

د. باسم قصيبة * م. لبنى منصور **

كلية الهندسة المعلوماتيّة – جامعة دمشق

*. قسم هندسة البرمجيات، كلية الهندسة المعلوماتية، جامعة دمشق.

**. طالبة دراسات عليا (ماجستير) قسم هندسة البرمجيات، كلية الهندسة المعلوماتية، جامعة دمشق.

ملخص البحث

يزداد الاهتمام بجودة إجرائيات تطوير البرمجيات مما يجعل من السهل صيانة وتحديث هذه الإجرائيات لأنّ التغيير لابدّ منه.

إنّ الهندسة المُقادة بالنماذج هي منهجيّة لبناء إجرائيات تطوير للبرمجيات، فقد أثبتت نجاحاً باهراً حيث توسّعت أفقيّاً في كل مجالات الهندسة المعلوماتيّة.

تكاد تكون دراسات قياس جودة الإجراءات المبنية بالاعتماد على الهندسة المُقادة بالنماذج معدومة ولذلك فإننا سنقوم بهذه الدراسة بالتركيز على كيفية قياس جودة الإجرائيات المبنية باستخدام الهندسة المُقادة بالنماذج وتقديم آليات وخوارزميات لتحسين جودتها.

الكلمات المفتاحيّة :

عيوب النماذج، إعادة التصحيح، النماذج المُترفّعة، التقسيم، الدراسات التجريبيّة، الحجم الكبير، ضعف الترابط.

The Improvement of the Quality of Model Driven Engineering Procedures of Software

PhD. Bassem KOSAYBA *, Eng. Lubna MANSOUR **

Faculty of Information Technology Engineering

*. Dept. of Software Engineering, Faculty of Information Technology, Damascus University.

**. Master student at Dept. of Software Engineering, Faculty of Information Technology, Damascus University.

ABSTRACT

The importance of the quality of development procedures of software increases that makes maintenance and update of these procedures easier, because the change is inevitable.

Model Driven Engineering (MDE) is an approach for building procedures of developed software which proved a large success through wide expansion in all information technology fields.

There are few studies for measuring the quality of MDE processes of software. So, we focus in this paper on how to measure the quality of these processes and we present algorithms to improve their quality.

Keywords :

Model Smells, Refactoring, Meta-Models, Partitioning, Empirical Studies, Large Size, Less Cohesion, Large Size.

١. مقدمة :

يُعتبر موضوع جودة البرمجيات من المواضيع الهامة بسبب تعقيده وتغلغله في الأنظمة البرمجيّة، علاوةً على ذلك فالاتجاه الحالي من التطوير والصيانة يتطلب قياس الجودة مع الكثير من التفاصيل.إنّ الجودة العالية للبرمجيّة تسمح بالتخفيف من كلفة التصحيح، والاختبار، وإعادة استخدام المنتج، وناهيك عن أنّ الجودةتتأثر بشكل عكسي مع وجود العيوب التصميميّة [1].

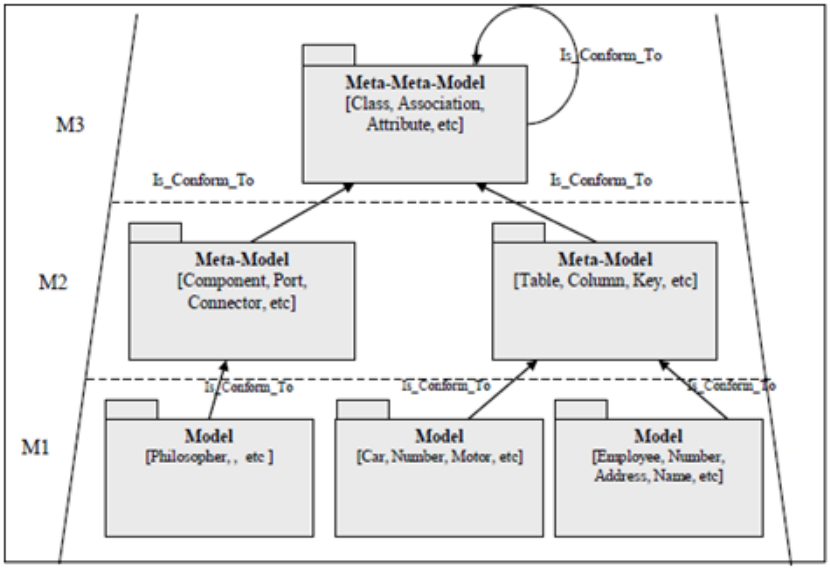
فالعيوب هي مؤشر إلى وجود مشكلة ما في مكان محدد من الرماز أو التصميم [2]، تؤدي إلى تراجع في جودة المنتج البرمجي وأدائه، وهدر الوقت لتعديله، وإعادة استخدامه، وفهمه [4],[3]. وإنّ الكشف المبكر عن العيوب وتصحيحها سيفيد في تحسين جودة إجراءات التطوير والصيانة.

يمكن استخدام النماذج المترفعة (Meta-Models) التي تقود إجرائيّة الهندسة المقادة بالنماذج ((Model Driven Engineering (MDE) في توليد أدوات نمذجة، تسمح لنا هذه الأدوات بتعريف النماذج (M1) في إحدى مراحل الإجرائية. ترتبط جودة البرمجيّة ارتباطاً وثيقاً بجودة إجرائية التطوير. و تتأثر جودة إجرائية التطوير بجودة الأدوات التي تدعم مختلف مراحلها. فكلما كانت الأدوات أعلى جودة، فإنّ إجرائيّة التطوير التي تستخدم هذه الأدوات تصبح أعلى جودة [5].

٢. أهميّة الهندسة المُقادة بالنماذج :

إنّ الهندسة المُقادة بالنماذج تقوم على فكرة استخدام النماذج وتحويلاتها لتنظيم فعاليّة نظام ما، فهي تسمح بتوصيف منهجيّة لتعريف المشكلة وكيفية الذهاب إلى حلها، إذ تقسم فعاليّة تطوير البرمجيات إلى عدة مستويات من التجريد [9]، ويتم نمذجة كل جانب بشكل مستقل عن الآخر عن طريق النماذج المُترفّعة [10]، فهي تحقق الفصل بين اهتمامات النظام، وتزيد من إمكانية إعادة استخدام النماذج.علاوةً على ذلك إمكانية توليد أداة آليّاً بحيث ندعم النمذجة في كل اهتمام [11]. هذا وتختلف النماذج في كل مستوى عن الآخر، بدقتها المهنية أو التقنية، والانتقال بين هذه المستويات يتم عن طريق تحويلات النماذج [12].

وتتألف النمذجة المترفّعة من عدة سويات تتدرج وفق مخطط هرمي، تعمل كل سوية على نمذجة وتوليد السوية التي تليها. تبدأ بسوية البيانات (M0) التي تتولد من سوية النماذج (M1) والتي بدورها تتولد من سوية النماذج المترفعة (M2) انتهاءً بسوية النماذج المترفعة المترفعة (M3) التي تعمل على توليد نفسها.



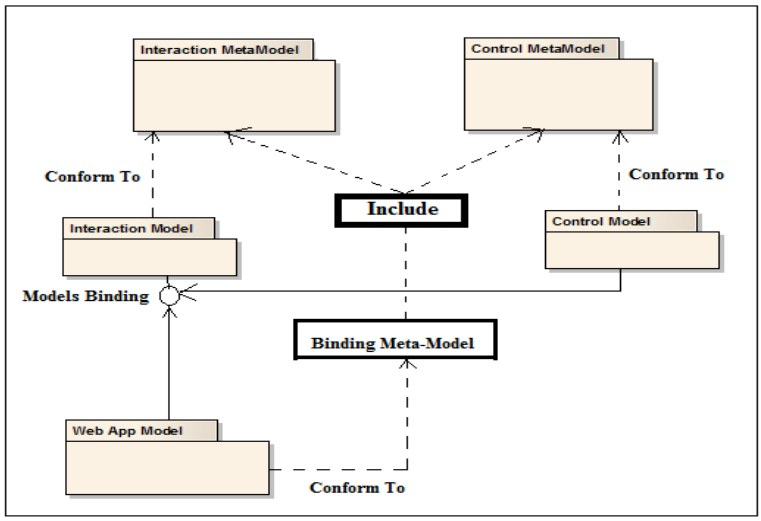
الشكل (١) : هرمية النماذج المترفعة Meta-Modeling .

يمكن استخدام النماذج المترفّعة، التي تقود إجرائية الهندسة المُقادة بالنماذج، لتوليد أدوات نمذجة ضمن كل اهتمام [9]. وبالتالي، العيوب في سوية النماذج المترفعة تسبب عيوباً في أدوات النمذجة المولدة منها والتي بدورها تسبب تراجعاً في جودة إجرائية التطوير التي تستخدم هذه الأدوات. مما يجعلنا نستنتج بأنه كلما كانت النماذج المترفّعة ذات جودة، كلما كانت جودة الأدوات الناتجة أعلى وأمكن الارتقاء بإجرائية تطوير الأدوات نحو أداء أسهل وأبسط و أكثر وضوحاً. هذا الاستنتاج مبني على حقيقة كون النماذج المترفعة هي اللبنة الأساسية التي تعتمد عليها الهندسة المقادة بالنماذج لبناء وتقسيم الإطار النظري لإجرائية تطوير معينة. كما تدعم مباشرة أدوات النمذجة المولدة من هذه النماذج المترفعة المراحل المختلفة للإطار العملي لإجرائية التطوير.

مثال :

يمكننا تصميم إجرائية لتطوير تطبيقات الويب بخطوتين أساسيتين : نقوم بالمرحلة الأولى بتصميم صفحات التطبيق بينما نقوم بالمرحلة الثانية بتحديد طريقة و ترتيب الذهاب (الملاحة) بين صفحات التطبيق.

لتحقيق هذه الإجرائية باستخدام الهندسة المقادة بالنماذج فإننا نعرف نموذجاً مترفعاً لتصميم صفحات التطبيق وآخر لتوصيف كيفية الملاحة بين صفحات التطبيق. نولد من النموذج المترفع الأول أداة تسمح بتصميم نماذج لصفحات التطبيق بينما نولد من الثاني أداة تسمح لنا بتحميل نماذج صفحات التطبيق وتحديد طريقة الملاحة بين هذه الصفحات. بعدئذ و من خلال مولد رماز يمكننا توليد التطبيق كاملاً وباستخدام عدة تقنيات (تطبيق ويب أو تطبيق للهاتف المحمول أو...).



الشكل (٢) : إجرائية مقادة بالنماذج لتطوير تطبيق الويب .

تتميز هذه الإجرائية بجعل صفحات التطبيق مستقلة عن بعضها البعض و بالتالي يمكن إعادة استخدامها في عدة تطبيقات أخرى، بالإضافة إلى تعريف التحكم بالموقع (كيفية الانتقال بين الصفحات) إذ يتم في مكان واحد بدلاً من أن يكون مبعثراً بين مختلف صفحات التطبيق، وبالتالي لا حاجة لتأهيل مستخدمي هذه الإجرائية بشكل تقني بغرض تطوير التطبيقات و إنما يتوجب عليهم فقط تحديد حاجاتهم من التطبيق (من حيث عدد صفحات الموقع و كيفية التنقل فيما بينها) بالإضافة إلى رأسملة منطق عمل

التطبيق و الاحتفاظ به كنموذج يمكن استخدامه لتوليد التطبيق بعدة تقنيات موجودة أو في المستقبل.

إنّ MDE توسّعت لتشمل جميع المجالات [8],[7],[6],[5]، فهي منهجيّة محورها الأساسي النماذج والعمليات على النماذج، غير أنّ الدراسات التي تهتم بقياس وتحسين جودة إجرائيات هذه المنهجيّة تكاد تكون معدومة.

٣. هدف البحث :

يهدف البحث إلى دراسة منهجيات وطرق لزيادة جودة الأدوات الداعمة للإجرائيات المطوّرة باستخدام الهندسة المُقادة بالنماذج، وتحسين الجودة بنظرنا يكون من خلال اكتشاف عيوب النماذج، وتصحيحها، قبل إنتاج الأدوات منها.

نستوحي أفكارنا من دراسة عيوب الرماز والتصميم لتحسين الجودة، وسننقل هذه الدراسات من مستوى الرماز إلى مستوى النماذج المترفّعة التي تقود الإجرائيّة المُقادة بالنماذج. وسنُعدّل خوارزميات تحسين التصميم البرمجي لنتمكّن من استخدامها في تحسين تصميم النماذج المترفعة وجعلها خالية من العيوب.

سنعمل على الاستفادة من الدراسة التجريبيّة والخبرة اليدويّة للمستخدمين في محاولة لنقل هذه الخبرة من المنحى اليدوي المحدود إلى المنحى الآلي الشامل. وسنسعى لتقديم معايير لقياس الجودة والتأكد من زيادتها.

٤. الدراسة المرجعيّة :

إنّ وجود عيوب الرماز والتصميم له تأثيرات حادة على جودة البرنامج، ولذلك فإنّ عملية اكتشافهم وتصحيحهم كانت محط اهتمام الباحثين والممارسين، وقد اقترحوا منهجيات مختلفة، لكشف هذه العيوب في البرنامج.

بدايةً، تمّ تعريف مجموعة من عيوب الرماز البالغ عددها 22 عيباً [13]، لها عدة خصائص تؤثر على النظم البرمجيّة بطرق مختلفة، وانطلاقاً من عيوب الرماز فإنّ مكوّن البرنامج الذي نعمل على قياسه يمكن أن يكون صفاً، أو تابعاً، أو نظاماً جزئياً. الأبحاث التي تبعت هذا العمل ساهمت في التعديل والتمديد على المجموعة البدائية من هذه العيوب [18] ,[17] ,[16] ,[15] ,[14] ,[2].

هذا وتمّ تأليف الكتاب الأول باسم "النماذج السيئة" [19]، وذلك في التطوير الغرضي التوجه، كانت مساهمته بأن غطت الإشكاليات الهيكليّة والرمازيّة وعملت على ضمان الجودة. وقد تمّ تعرّف 61 من الخصائص الإرشادية للغة غرضية التوجّه لتقييم الجودة البرمجيّة بشكل يدوي وتحسين التصميم والتحقيق [20]. إنّ من أبرز العيوب التي بحثت فيها الدراسات السابقة هي الرماز المكرر، قائمة الوسطاء الطويلة، صفوف البيانات (Data Class)، الصفوف الكسولة (Lazy Class)، الصفوف الكبيرة (God Class)، وصفوف التشابك (Brain Class).

إنّ أغلب هذه العيوب تدور حول مشكلة التعقيد العالي الناتج عن الحجم الكبير، ومشكلة التفكك الناتج عن الترابط الضعيف للصفوف الداخلية. هناك مجموعة من المنهجيات لاكتشاف وتحديد العيوب. أغلب هذه المنهجيات يدويةً [21]، أو تعتمد على قواعد محددة [24] ,[23] ,[22] ,[17] أو مقاييس لها علاقة بالحجم والترابط في سوية الصف. وفيما يلي نورد بعض المقاييس[25].

• TCC (Tight Class Cohesion)	• NMD (Number of Methods Declared)
• LCOM (Lack Of Cohesion Methods)	• NAD (Number of Attributes Declared)
• LOC (Lines Of method Code)	• NoDC (Number of Data Classes)

أمّا عمليات التصحيح لمثل هذه العيوب تنوعت ما بين استخراج عناصر من الصفوف إلى صفوف أخرى، وتقسيم الصف إلى عدة صفوف، أو استخراج صفوف جديدة [25]. غير أنّ عمليات التصحيح ليست هي الخيار الصحيح دائماً، ففي بعض الحالات قد تؤدي إلى تفكك النظام، لذا لابدّ من مراعاة عتبات محددة في حال تخطيها نتخذ قراراً بالتصحيح، وذلك حرصاً على عدم الطعن بمعايير الجودة الأخرى.

إنّ الكشف عن العيوب في إجرائيات الهندسة المُقادة بالنماذج يساعد في توفير الجهد والوقت بمراحل لاحقة. لذا سنوجّه العناية نحو النماذج المترفعة كونها اللبنة الأساسية في هذه الإجرائية ونستخدمها لتوليد أدوات النمذجة التي تدعم مختلف مراحل الإجرائية. إنّ النماذج المترفعة الجيدة لها قيود على الحجم وعدد المفاهيم التي تحويها. ونريد تصميمها بحيث تكون أكثر ترابطاً وتماسكاً [26],[25] مما يجعل الأدوات المولدة منها أسهل للاستخدام والتعلم وبالتالي تصبح الإجرائية بعموميتها أسهل للفهم والصيانة، وهذا المبدأ يتفق مع مبادئ منهجيّة هندسة البرمجيات التي تعمل بمبدأ "فرّق تسد"، إذ يتم

تقسيم فضاء العمل إلى مجموعة من الرزم، بحيث تحوي كل رزمة على العناصر المتشابهة والأكثر ترابطاً، مما يسمح بتنظيم العمل ورفع جودته. يمكننا تلخيص عملنا، بإيجاد أفضل أسلوب لتقسيم الإجرائية إلى مراحل. مستفيدين من الدراسات السابقة لعيوب الرماز والتصميم لتشمل النماذج المترفّعة التي تلعب الدور الأساسي في قيادة الإجراءات البرمجية.

بعبارة أخرى: إن الحجم الكبير للنموذج المترفع يعني بأن الأداة المولدة منه تحوي العديد من المفاهيم والتي تنتمي لعدة مجالات مما يجعل من الصعب السيطرة عليها وتعلمها. وبالتالي يجب تقسيم النموذج المترفّع كبير الحجم وضعيف الترابط ما بين مفاهيمه إلى عدد من النماذج المترفّعة الجزئية، أصغر حجماً، وأعلى تماسكاً، وأكثر تخصصاً. كما وجدنا في مثال "إجرائية تطوير تطبيقات الويب". سنضع منهجيّة لتقسيم النماذج المترفّعة إلى عدة نماذج مترفّعة جزئية بالاعتماد على بيانات تجريبيّة، كما حصل في دراسات مشابهة، حيث اعتمدت بيانات تجريبيّة [8],[7],[6],[5] بشكل أساسي لتعميم استنتاجاتها.

٥. مواد وطرائق البحث :

تمت الدراسة التجريبية على حوالي ١٠٠ طالباً في مرحلة ما قبل التخرج واستمرت الدراسة لمدة ستة أشهر. تم التخطيط للدراسة التجريبية كما يلي:

- يقوم الطلاب بتقسيم عدة نماذج مترفعة (ومنها النموذج المترفع لمعايير الجودة البرمجيّة CMMI والنموذج المترفع للغة نمذجة تطبيقات الويب WebML، نموذج لغة القيود الموجّهة OCL) بما يحقق تجميع للمفاهيم المترابطة والمنطقية مع بعضها البعض ضمن مجموعات جزئية، والذي يضمن بدوره سهولة الاستخدام، والتعلم وفهم لإجرائية التطوير وأدواتها المساعدة. توضّح الأشكال في الملحق/أ/ عينة عن طريقة التقسيم التي تمّ اعتمادها من قبل أغلب الطلاب؛
- يقوم الطلاب بتوليد أدوات النمذجة الداعمة لإجرائيات التطوير (باستخدام الإطار GMF [27]) من النماذج المترفعة قبل التقسيم و بعده؛

- يقارن الطلاب بين إجرائيات التطوير الناتجة عن النماذج المترفعة قبل و بعد التقسيم حسب سهولة استخدام الأدوات، والوقت اللازم لتعريف نموذج التطبيق باستخدام إجرائية التطوير، وسهولة الفهم العام لإجرائية التطوير؛
- قمنا باعتماد نتائج التقسيم واقتراح خوارزمية تستفيد من هذه البيانات التجريبيّة، بهدف أتمتة الطريقة اليدوية. وهو ما سنوضحه في الفصل التالي؛
- أعدنا اختبار هذه الخوارزمية على نماذج معياريّة عالميّة هماPADL [28] وهو نموذج مترفع للغة توصيف السويات التجريديّة والنموذج Henshin [4] وهو جزء من نموذج مترفع لقواعد التحويل المنمذجة. وقام الطلاب بالمقارنة بين إجرائيات التطوير قبل وبعد التقسيم حسب: الزمن المطلوب لتعريف النماذج، ومقدار استعمال الملفات والوثائق التوضيحيّة لإمكانية فهم طريقة عمل الأدوات، وعدد الضغطات الوسطي اللازمة لإنشاء عنصر من النموذج. إنّ مقارنة القيم الناتجة لهذه العوامل أكدت زيادة الجودة بعد تطبيق منهجيّة التقسيم. الملحق /ب/ يحوي على عينة من إجابات الطلاب.

٦. الخوارزميّة المُقترحة :

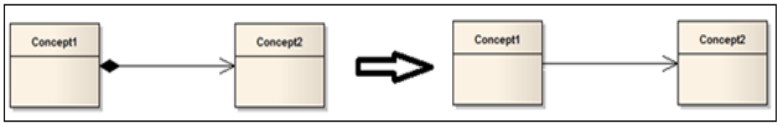
توضّح الأشكال في الملحق /أ/ عيّنة عن طريقة التقسيم التي اعتمدها أغلب الطلاب، ولأتمتة عملية التقسيم كان لابدّ من إيجاد بنية موحدة لتمثيل النموذج المترفع ليصبح دخلاً مقبولاً لخوارزميتنا. وجدنا أن البنية الشجرية بنية مناسبة، لتبسيط العمل والحسابات العددية. وبالتالي نستطيع إسقاط عناصر النموذج المترفع على عناصر الشجرة من خلال التقابل التالي:

نمثل مفاهيم النموذج بالعقد الشجريّة، والعلاقات بين هذه المفاهيم بالروابط الشجرية بين العقد. وفيما يلي نبيّن خوارزميّة التحويل المُقترحة إلى البنيّة الشجريّة ونبيّن أهم العقد.

أ- خوارزميّة التحويل إلى البنية الشجريّة :

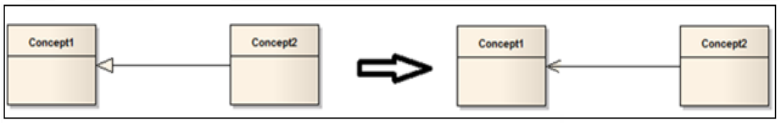
تعاملنا خلال النماذج المترفّعة السابقة مع ٣ أنواع من الروابط.

١- علاقة التركيب ونحوّلها إلى سهم موجّه فقط من الكل إلى الجزء.



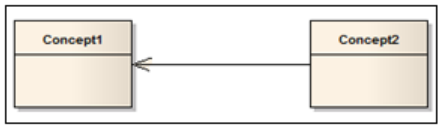
الشكل (3) : تحويل علاقة التركيب .

٢- علاقة الوراثة: ونحوّلها إلى سهم فقط من الابن باتجاه الأب.



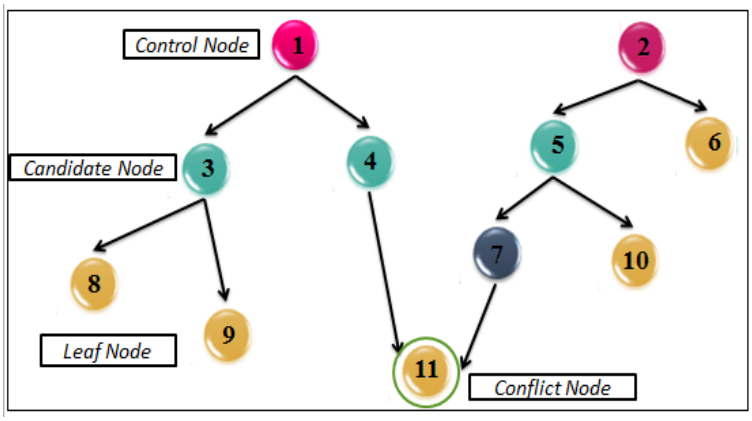
الشكل (4) : تحويل علاقة الوراثة .

٣- علاقة التجميع: ونبقيها على حالها حسب اتجاه السهم.



الشكل (5) : علاقة التجميع .

لم تتوسّع الدراسة لتشمل العلاقات المكررة والعلاقات العكسيّة أو حتى الأعداد على طرفي العلاقات والتي تحدد عدد المفاهيم المرتبطة (واحد لواحد) أو (واحد لكثير)، فالهدف معرفة اتجاه العلاقة فقط. لذا قمنا بإهمالها، وحصرنا التركيز على العلاقات الثلاثة السابقة. وفيما يلي نبيّن المخطط العام الشجري الناتج بعد تطبيق خوارزمية التحويل.



الشكل (6) : البنية الشجريّة .

ب. أنواع العقد الموجودة في البنية الشجريّة :

نميز بين الأنواع التالية من عقد البنية الشجرية:

- **العقدة الورقة Leaf Node:** هي العقدة التي لا يصدر عنها أي رابط، هي فقط تعتبر مستقبل للروابط، كحال العقد (6, 8, 9, 10, 11) في الشكل السابق.
- **العقدة التحكمية Control Node:** هي العقدة الجذر في المخطط الذي يمسك بكل العقد ويمكن أن يحوي المخطط على أكثر من عقدة تحكميّة، كحال العقد (1,2) في الشكل السابق.
- **العقدة المرشحة Candidate Node:** هي العقدة التي ترتبط برابط مباشر مع العقدة التحكمية، ففي حال التقسيم، هي مرشّحة لأن تصبح عقدة تحكمية للنموذج الجزئي الناتج، كحال العقد (3, 4, 5, 6) في الشكل السابق.
- **العقدة المُتنازع عليها Conflict Node:** هي العقد التي ترتبط مع عقد تحكمية بشكل مباشر أو غير مباشر، فعند التقسيم تغدو العقدة محط تنازع مابين عدة نماذج جزئية، كحال العقدة (11) في الشكل السابق.

وفيما يلي نقترح خوارزمية للتقسيم بالاعتماد على المفاهيم السابقة.

ج. المفاهيم العامة المُقترحة :

نقدم في هذا القسم عدد من العوامل والمفاهيم لتقسيم النماذج المترفعة. حصلنا على هذه القواعد من خلال تحليل الدراسات التجريبيّة السابقة.

- **مفهوم الرقم المحسوب للعقدة (Calculated Number for Node (*CNN*)):**

نقترح للتعبير عن حجم النموذج وأهمية العقدة مفهوم الرقم المحسوب (***CNN***) فهو يشير إلى سوية العقد وعدد الوصلات الخارجة منها باتجاه السويات الدنيا وصولاً إلى سوية الأوراق. ونقترح التابع اللغرتمي كمعامل تخميد.

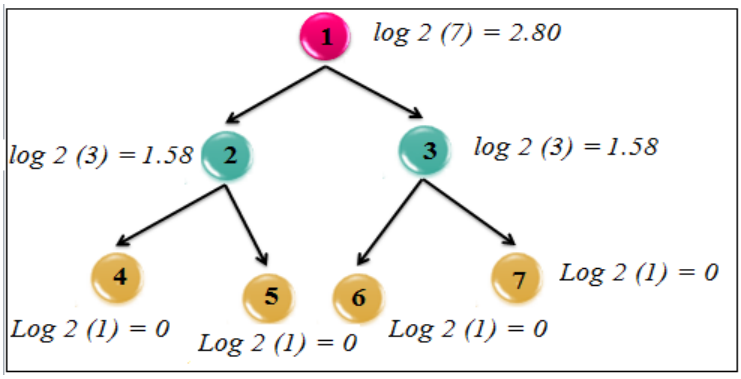
$$CNN = Log\ 2\ (Number\ of\ Output\ Links\ to\ low\ levels + 1)$$

نعمد إلى إضافة العدد (١) إلى عدد الوصلات الخارجة من العقدة وذلك بهدف تجنب القيم اللانهائيّة.

إن السوية الدنيا للعقدة الورقة معدومة، لذا ***CNN*** لها يُحتسَب كالتالي:

$$CNN\ (Leaf\ Node) = Log\ 2\ (0 + 1) = log\ 2\ (1) = 0$$

وفيما يلي نُقدم مثالاً حول حساب ***CNN*** للعقد.



الشكل (7) : مثال حول حساب *CNN* .

- **مفهوم وزن العقدة (Weight Number for Node (*WN*))**

نقترح هذا المفهوم للتعبير عن حجم النموذج والترابط ما بين عقدة محددة وإحدى العقد التحكميّة في النماذج المترفعة الجزئيّة، فكلما كانت ***WN*** أصغر،

كلما كانت العقدة أكثر تلاحماً وارتباطاً بالعقدة التحكميّة وأصعب من ناحية الفصل والتقسيم.وبالتالي نحتاج إلى عدة مفاهيم في هذه المرحلة.

- مفهوم **الترابط** ما بين عقدة محددة والعقدة التحكميّة في أحد النماذج الجزئيّة، فعندما يكون لدينا حالة تنازع مابين عقدتين تحكميتين، لابدّ من تحديد أيهما أضعف ترابطاً مع العقدة المدروسة. إنّ الترابط الضعيف واحد من أهم العوامل المسببة للعيوب، لذا لابدّ من تجنبه وذلك باقتطاع الرابط من جهته.
- مفهوم **حجم وأهمية العقدة التحكميّة** في النماذج المترفّعة الجزئيّة، يُعتبر الحجم الكبير واحداً من أهم العوامل المسببة للعيوب، لذا لابدّ من تجنبه أيضاً.
- معامل **التخميد** وهو يساعد على جعل القيم أصغر ويساعدنا على وضع عتبة للتقسيم.

تتكوّن الخوارزمية المُقترحة من عدة معاملات تعتمد على المفاهيم السابقة:

- عدد الروابط ما بين العقدة المدروسة والعقدة التحكميّة في النموذج المترفّع الجزئي ← وهو يمثّل الترابط.
- العدد المحسوب ***CNN*** للعقدة التحكميّة ← وهو يمثّل الحجم والأهميّة.
- Log 2 .← هو يمثّل معامل التخميد.

وبالتالي الشكل النهائي للقانون كما يلي:

$$Weight\ of\ Node\ (WN) = Log\ 2\ (Number\ of\ Links\ between\ this\ Node\ and\ Control\ Node) + CNN\ (Control\ Node)$$

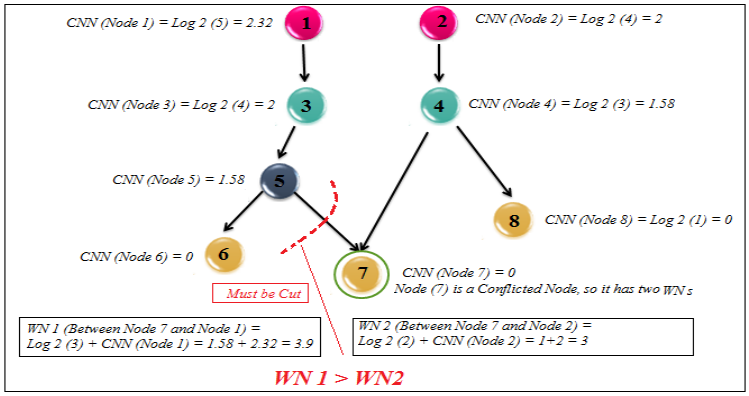
وزن العقدة المرشّحة (التي يفصلها عن العقدة التحكميّة رابط واحد) هو كالتالي:

$$Weight\ CAndidate\ Node\ (WCAN) = Log\ 2\ (1) + CNN\ (Control\ Node) = CNN\ (Control\ Node)$$

إذ لكل عقدة مُتنازع عليها أكثر من وزن واحد ***(WN)***، وذلك بسبب وجود أكثر من عقدة تحكميّة تتنازع على العقدة المذكورة. لذا كان لابدّ من حساب أوزان هذه العقدة بالنسبة لكل واحدة من العقد التحكميّة. فكلما كان هذا الوزن

أصغر، كلما كانت العقدة أقرب للنموذج المترفّع الجزئي وصعبة الانفصال عنه.

وفيما يلي نقدم مثالاً يوضح حساب الأوزان ***WN***.



الشكل (8) : حساب الأوزان *WN* لعقدة التنازع .

فعندما نقسّم النموذج المترفّع إلى عدة نماذج جزئيّة، فإنّ العقد التحكميّة تتنازع على العقد المشتركة كالعقدة رقم (7) في الشكل السابق، فإنّه يتوجب علينا حساب الأوزان لهذه العقدة مع كل العقد التحكميّة المتنازعة عليها. يظهر في الشكل السابق عقدتان تحكميتان (1, 2) تتنازعان على هذه العقدة. إنّ الوزن الأعلى يدل على حجم أكبر ومسافة أكبر وبالتالي ترابطاً أضعف، لذا يتوجب علينا اقتطاع الرابط من جهة الوزن الأعلى.

د. آلية عمل الخوارزمية المقترحة :

نسرد في ما يلي الخطوات الرئيسة للخوارزمية المقترحة :

- إذ كان العدد المحسوب للعقدة التحكميّة ***CNN*** (Controller Node) >= 4 عندها يكون النموذج كبيراً ويستوجب التقسيم إلى نماذج جزئية أصغر بحيث تكون أكثر ترابطاً، وإنّ العتبة 4 تعني أن لدينا 16 مفهوماً فما فوق في نفس النموذج المترفّع لذا وجب التقسيم.
- العقد المرشحة ذات الوزن المحسوب الأعلى ***CNN*** لها أولوية باختيار عملية التقسيم.

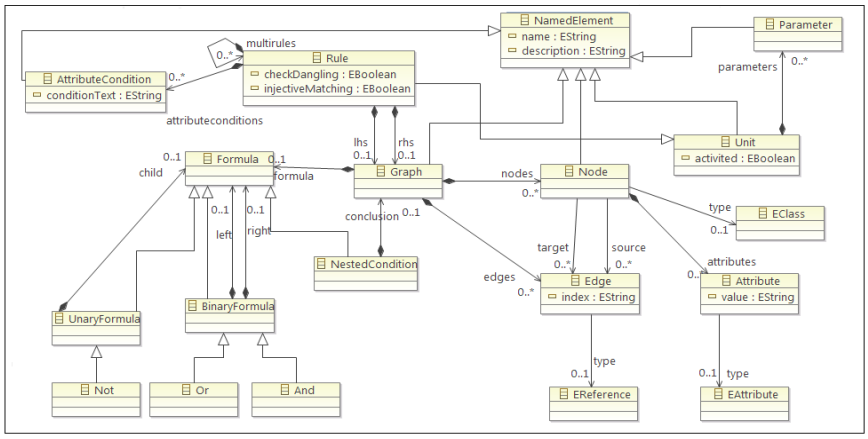
- في حال وجود عقد مرشحة ترتبط ببعضها عن طريق علاقات معينة، سيكون لدينا الخيارين التاليين:
  - إذا كانت العقد ذات أعداد محسوبة كبيرة: عندها سيتم إلغاء العلاقات فيما بينها وجعل كل منها عقدة تحكمية مستقلة بذاتها وجذراً لنماذج جزئيّة لاحقة.
  - وإلّا سيتم الإبقاء على العلاقات وبالتالي سيكون لدينا عقدة تحكمية تنتمي إليها باقي العقد المرشحة ولن تعود عقداً تحكميّة.
- نبدأ باختبار العقد واحدة تلو الأخرى:
  - في حال تساوي وزن العقدة بالنسبة لعقدتين تحكميتين فإن أولوية التقسيم تعود للمستخدم.
  - يتم إعادة حساب الأوزان والأعداد المحسوبة للعقد مباشرة بعد كل مرة من الاقتطاع.
- في حال كان لدينا أكثر من عقدة تحكم، فإننا نحرص على مايلي:
  - إذا كانت العقدة تحوي رابطاً وحيداً وجميع العقد الأعلى منها سوية لا تحوي إلا رابطاً وحيداً، عندها لا نقتطعها، كي نحول دون تفكك النموذج.
  - العتبة الخاصة لنبدأ الاقتطاع إذا كان ***CNN*** للعقدة الجذر أكبر أو يساوي اللغرتم الثنائي لعدد مفاهيم النموذج المترفع / ٣. أي :

CNN (Control Node) > = Log2 (Number of Concepts / 3)

٧. النتائج ومناقشتها :

قمنا بتطبيق منهجيّة التقسيم المُقترحة على مجموعة من النماذج المترفعة المعيارية المُعتمدة عالمياً، وعمدنا إلى إيضاح الخطوات التفصيلية لبيان آلية عمل المنهجيّة من خلال الحالات الدراسيّة التالية.

حالة دراسية ١: نختبر خوارزميتنا على جزء من نموذج مترفع لقواعد التحويل المنمذجة من قبل نموذج Henshin[4]، وهو نموذج مترفع لأداة تهدف إلى النمذجة وتحقق انتقالات لغة النمذجة وإعادة تشكيل البيان في Eclipse[27].

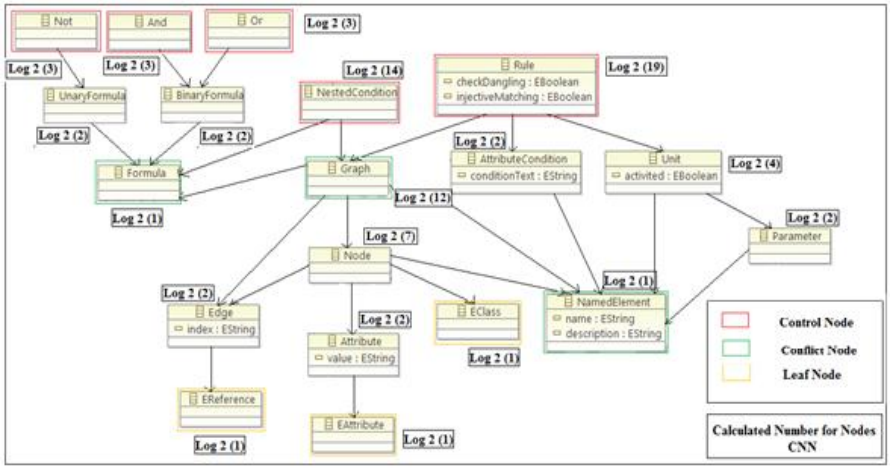


الشكل (9) : جزء من النموذج المترفع *Henshin* للقواعد.

لدينا عدة خطوات لخوارزميّة التقسيم، وهي مشروحة كما يلي:

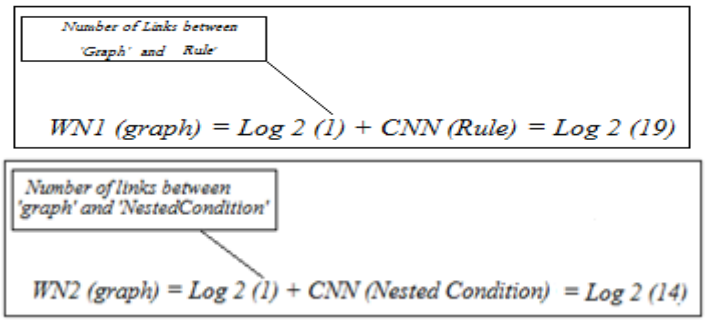
١- التحويل إلى البنية الشجريّة، وذلك حسب القواعد الموضحة سابقاً.

٢- حساب الأعداد المحسوبة لكافة العقد.

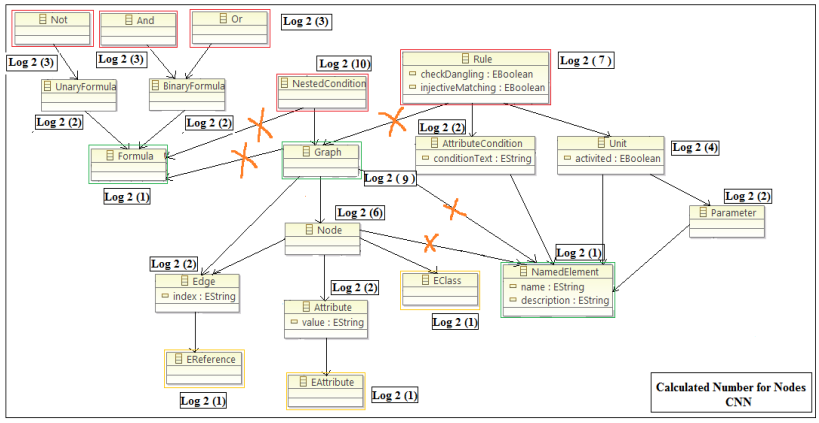


الشكل (10) : البنية الشجريّة لنموذج *Henshin* والأعداد المحسوبة *CNN* للعقد

٣- نبدأ من العقدة ذات العدد المحسوب الأكبر وهي Rule، نبحث عن العقد المُتنازع عليها نجد العقدة المرشّحة Graph، هي محط تنازع ما بين العقدتان التحكميتان Rule، Nested Condition. ويتم حساب وزن هذه العقدة ***WN*** بالنسبة لكليهما.



٤- نقارن القيم فنجد ***WN*** 2 > ***WN*** 1 وبالتالي نقتطع من عند القيمة الأكبر أي نقطع الرابط ما بين Graph وRule ونعيد حساب ***CNN*** من جديد للعقد. تتعدل القيم فنبدأ بالبحث عن عقد التنازع من جديد والاقتطاع من جهة الرابط ذي الوزن الأعلى، نستمر بالعملية حتى نصل إلى حد العتبة، عندها نتوقف عن الاختبار والاقتطاع.

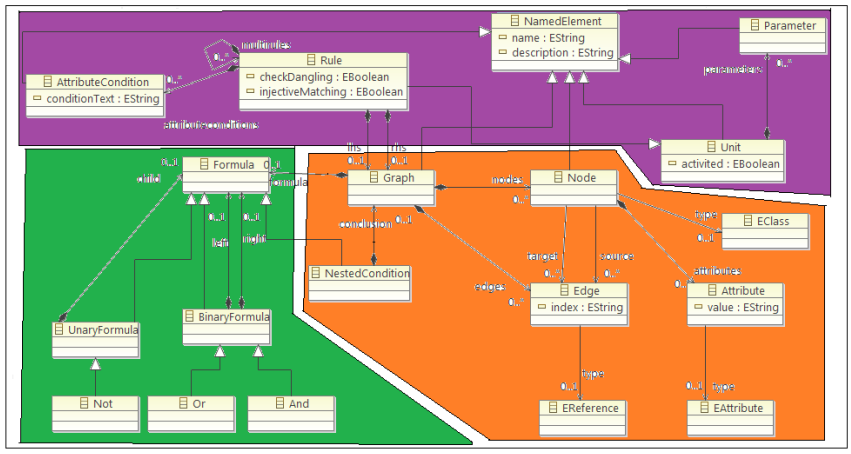


الشكل (11) : خطوات الاقتطاع التفصيليّة للمنهجيّة المُقترحة .

هناك العديد من العقد التحكمية (Not, And, Or) التي تتنازع مع بعضها على عقد أخرى. لكننا لا نقتطع هذه العقد كونها تملك رابطاً وحيداً مع عقد النموذج وذلك كي لا تغدو عقداً حرّة، وبالتالي لنحول دون تجزء وتفكك النموذج.

وهكذا تمّ تقسيم النموذج المترفع إلى ٣ نماذج مترفّعة جزئيّة:

- **الأول** يختص بعناصر البيان من عقد ووصلات وأنواع لهذه العقد؛
- **الثاني** يختص بالصيغ المنطقيّة الأحادية والثنائيّة؛
- **الثالث** يختص بالقواعد والوحدات والمتحولات.

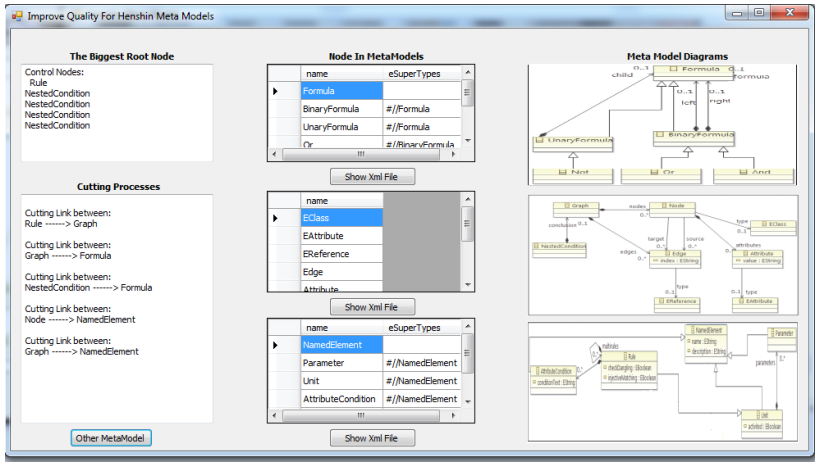


الشكل (12) : النموذج المترفع *Henshin* بعد التقسيم وقبل التحويل إلى البنية الشجريّة.

إنّ عملية التقسيم حققت:

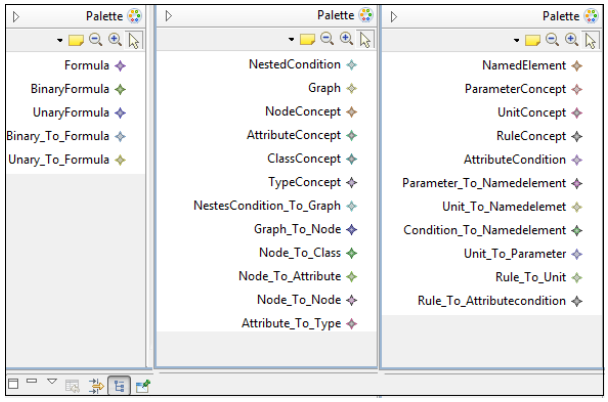
- منطقيّة عالية في جمع المفاهيم المتشابهة والمترابطة ضمن نموذج مترفع جزئي واحد؛
- سهولة ومرونة في استعمال أدوات النمذجة الناتجة عنها؛
- وضوح في إطار العمل المعتمد على هذه الأدوات؛
- أدوات مستقلة عن بعضها البعض، وأكثر تخصيصاً وبالتالي يمكن إعادة استخدامها في عدة تطبيقات أخرى.

هذا ما أثبتته الدراسة التجريبيّة التي تمت حوالي ١٠٠ طالباً، من خلال الممارسة العمليّة لاستخدام الأدوات الناتجة عن النموذج المترفع Henshin قبل وبعد التقسيم، وبدت تقييمات الطلاب بهذا الشأن واضحة وعادت بالنتائج الإيجابية عن طريق الاستبيانات التي تمّ طرحها عليهم، وفي فقرة التقييم نوضح الإحصائيات والمخططات البيانيّة والتي نراها البرهان القوي على تحسّن الجودة. الملحق /ب/ يبيّن الاستبيان المطروح بخصوص تقييم الخوارزميّة المُقترحة.



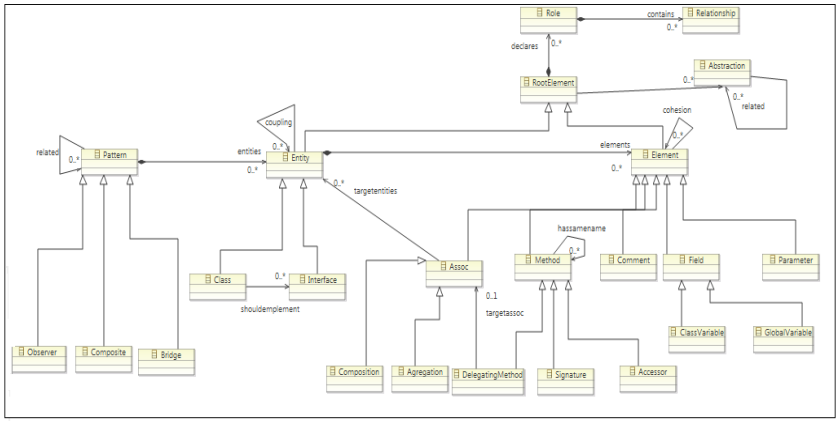
الشكل (13) : الواجهة الخاصة بتطبيقنا الذي يقوم بالتقسيم عند اختباره على نموذج *Henshin*

على حين يتم توليد الأدوات المستقلة من النماذج المترفّعة الجزئية باستخدام إطار العمل GMF. حيث أن GMF هي إطار عمل مقدم من IBM و الذي يولد آلياً أداة نمذجة من نموذج مترفع.



الشكل (14) : الأدوات الناتجة عن النماذج المترفعة الجزئية من *Henshin*.

حالة دراسية ٢: نختبر أيضاً خوارزميتنا على نموذج مترفع للغة توصيف السويات التجريديّة والنماذج PADL[28].



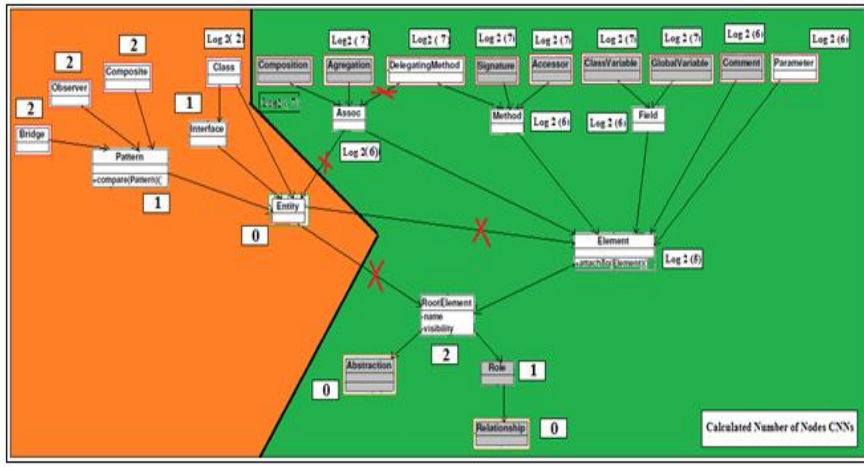
الشكل (15) : النموذج المترفع *PADL* .

ينتج عن التقسيم نموذجان مترفعان جزئيان:

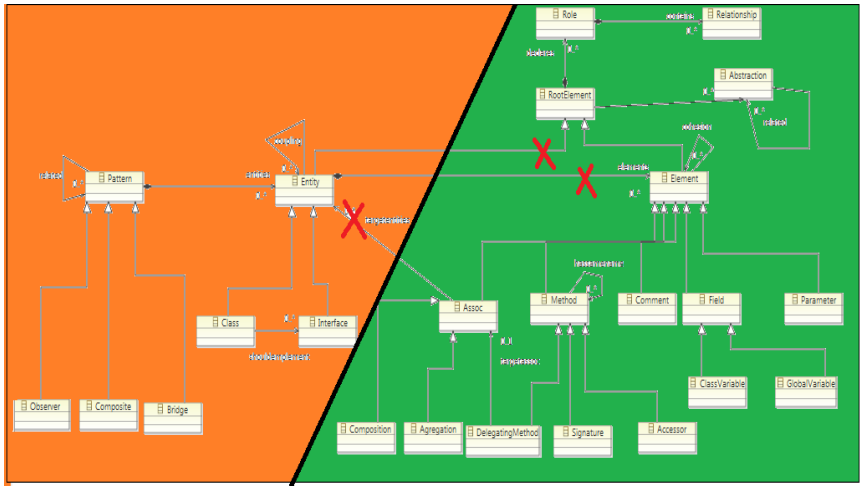
- **الأول** يختص بالعناصر الغرضية التوجّه من توابع وعناصر ومتحولات والعلاقات فيما بينها؛
- **الثاني** يختص بالنماذج التصميميّة.

نلاحظ أنّ النماذج الجزئية الناتجة:

- أكثر تخصصاً وأسهل استخداماً،
- إمكانية توليد الهيكليّة العامة للنظام من خلال الجزء الذي يختص بالنماذج التصميميّة، ثم توليد محتوى التوابع والصفوف المكوّنة له.

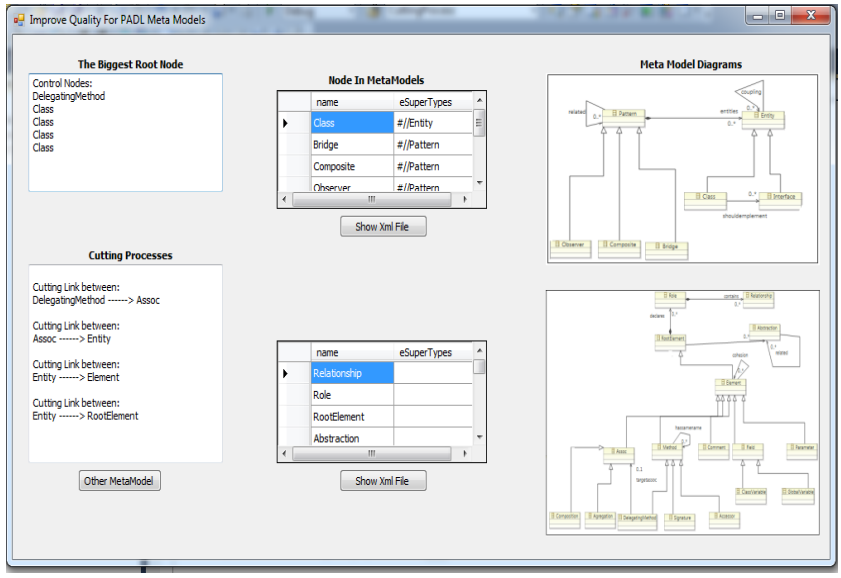


الشكل (16) : البنية الشجرية للنموذج المترفع *PADL* بعد التقسيم .



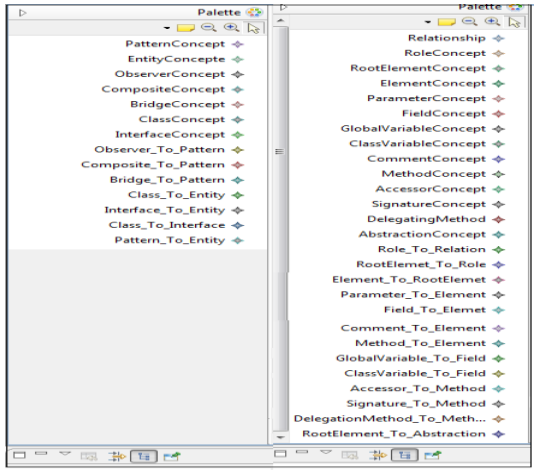
الشكل (17) : النموذج المترفع *PADL* بعد التقسيم قبل التحويل إلى البنية الشجريّة.

كحال نموذج Henshin المترفع، تمّ طرح نموذج PADL المترفع قبل وبعد التقسيم في الدراسة التجريبيّة وكانت تقييمات الطلاب إيجابيّة تمّ توضيحها في فقرة التقييم.



الشكل (18) : الواجهة الخاصة بتطبيقنا الذي يقوم بالتقسيم عند اختباره على نموذج *PADL*.

أما الأدوات المستقلة التي تمّ توليدها عن النماذج المترفّعة الجزئية باستخدام إطار العمل GMF مبيّنة كما يلي.



الشكل (19) : الأدوات الناتجة عن النماذج المترفع الجزئية من *PADL* .

٨. معايير الاختبار والتقييم :

إن جوهر خوارزميتنا الحالية تهدف إلى تقسيم النماذج المترفعة الكبيرة الحجم، ضعيفة الترابط إلى عدة نماذج مترفعة جزئيّة مترابطة المفاهيم، صغيرة الحجم، يسهل السيطرة عليها. وبالتالي تنتج أدواتها أسهل استخداماً وإجرائية أسهل فهماً وأكثر وضوحاً. حتى نستطيع اختبار جودة خوارزميتنا في تحسين جودة إجرائيات تطوير البرمجيات المبنية بالاعتماد على الهندسة المُقادة بالنماذج، قمنا بالمقارنة بين جودة إجرائيات تطوير البرمجيات في نفس المجال قبل استخدام خوارزميتنا وبعدها.

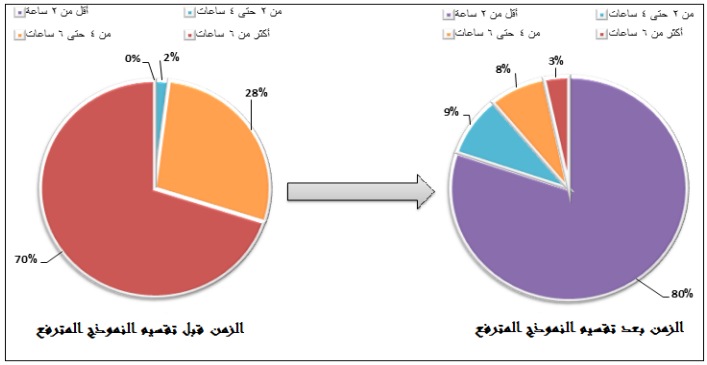
إنّ المعايير التي سنعتمدها للمقارنة بين جودة هذه الإجرائيات هي التالية:

- **سهولة الاستخدام:** وتُقاس من خلال عدد الضغطات الوسطي المطلوبة لإنشاء عنصر باستخدام الأدوات الناتجة عن النماذج المترفعة.
- **الوضوح وقابلية الفهم:** وتُقاس من خلال الزمن المطلوب لفهم واستعمال كافة أدوات الإجرائية لتطوير ونمذجة البرمجيات. ومن خلال عدد مرات طلب الوثائق والملفات المُساعدة التي تسهّل عملية الفهم.
- **إمكانية إعادة الاستخدام:** فعملية التقسيم سمحت بمزيد من التخصيص في أدوات إطار العمل وبالتالي تشكل كل من هذه الأدوات الجزئية وحدات مستقلة يمكن إعادة استخدامها مجدداً عدة مرات.

- **القوة أو الشدة:** وتقاس بعدد الانهيارات أثناء الاستخدام.

وتتطابق هذه المعايير مع معايير الجودة العالمية للبرمجيّة [30],[29] ISO.

قمنا بطرح مجموعة نماذج مترفعة على ١٠٠ طالباً قبل استخدام خوارزميّة التقسيم وبعدها. عمد الطلاب إلى تنفيذ هذه النماذج المترفعة وتوليد الأدوات الخاصة بهم بهدف تقييمها وتقييم الإجرائيّة التي تستعمل هذه الأدوات بطرق عملية وذلك قبل وبعد استخدام الخوارزميّة. زودنا الطلاب بملاحظاتهم من خلال عدة استبيانات تشرح النتائج. وكما أشرنا الملحق/ب/ يضم عينة من هذه الاستبيانات. إنّ التحسّن الجاري على صعيد **الزمن** اللازم للاستخدام والفهم قبل وبعد التقسيم مبيّن كما يلي.

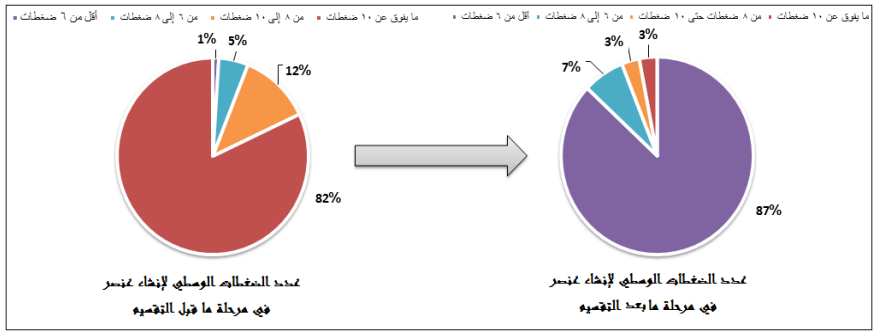


الشكل (20) : الأداء على الصعيد الزمني قبل وبعد التقسيم .

الجدول (1) التحسّن على المستوى الزمني قبل التقسيم وبعده.

قبل التقسيم				
عدد الساعات لتشكيل النموذج	أقل من ٢ ساعة	من ٢ إلى ٤ ساعة	من٤إلى ٦ ساعة	أكثر من ٦ ساعة
نسبة الطلاب	0%	2%	28%	70%
بعد التقسيم				
نسبة الطلاب	80%	9%	8%	3%

هذا وتنخفض عدد الضغطات الوسطي اللازمة لإنشاء مفهوم معيّن وبالتالي زيادة سهولة الاستخدام. مُبيّن كما يلي.

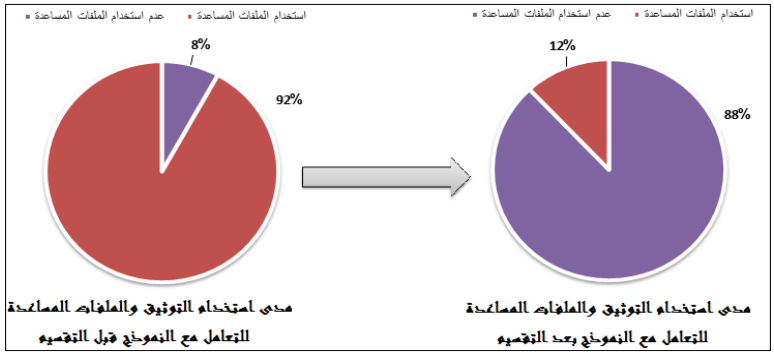


الشكل (21) : مدى سهولة الاستخدام من خلال عدد الضغطات قبل وبعد التقسيم .

الجدول (2) التحسّن الجاري على مستوى الاستخداميّة

قبل التقسيم				
عدد الضغطات الوسطي لإنشاء عنصر	أقل من ٦ ضغطات	من ٦ إلى ٨ ضغطات	من٨ إلى ١٠ ضغطات	أكثر من ١٠ ضغطات
نسبة الطلاب	1%	5%	12%	82%
بعد التقسيم				
نسبة الطلاب	87%	7%	3%	3%

على حين أنّ استخدام التوثيق والملفات المساعدة بعد التقسيم، سجلت تراجعاً ملحوظاً عن عددها قبل التقسيم، مُبيّن كما يلي.



الشكل (22) : مدى وضوح الأدوات من خلال استخدام التوثيق قبل وبعد التقسيم .

الجدول رقم (3) التحسّن الجاري على مستوى الوضوح

قبل التقسيم		
عدد مرات استخدام الملفات المساعدة	استخدام الملفات المساعدة	عدم استخدام الملفات المساعدة
نسبة الطلاب	8%	92%
بعد التقسيم		
نسبة الطلاب	12%	88%

في ضوء النتائج التي حصلنا عليها، نرى بوضوح أنّ خوارزميّة التقسيم المُقترحة حققت معايير الجودة بجدارة من سهولة الاستخدام، الوضوح، وقابليّة الفهم، بالإضافة إلى القوة أو الشدة إذ لم يحدث أي انهيار أو خلل خلال استخدام الإجرائيّة من قبل ١٠٠ طالباً، وإمكانية إعادة الاستخدام محققة كوننا جزئنا النموذج المترفع إلى أجزاء أصغر وبالتالي أمكن استخدامها في إجرائيات تطوير أخرى.

٩. الاستنتاجات والتوصيات :

قدمنا في هذه الورقة خوارزميّة لتحسين جودة إجراءات هندسة البرمجيات المُقادة بالنماذج من خلال فصل وتقسيم النماذج المترفعة الكبيرة الحجم وضعيفة الترابط إلى نماذج مترفعة جزئية أكثر ترابطاً وأصغر حجماً، وقد استوحينا أفكارنا من عيوب الصف الكبير (God Class) وعملنا على تمديد هذه العيوب من السوية المنخفضة باتجاه السوية العليا (سوية النماذج المترفعة) في هرمية الهندسة المُقادة بالنماذج، وقمنا بتجميع نتائج الدراسة التجريبيّة التي استمرت لمدة ستة أشهر فعملنا على وضع قواعد و ضوابط لعملية التقسيم بما يضمن تحسين جودة إجرائية التطوير كما أننا جمعنا هذه القواعد ضمن خوارزمية و تم تنفيذها و من ثم تم تقييمها من قبل ١٠٠ طالباً في مرحلة ما قبل التخرج، وحققت نتائج إيجابية في تحسين جودة إجرائيات التطوير.

تمكنّا بذلك من تعميم الدراسة التجريبيّة ونقلها من التقسيم اليدوي المحدود الذي اقترحه الطلاب إلى التقسيم الآلي الأكثر شمولاً.

يُعتبر هذا العمل رائداً من نوعه، كون أكثر الدراسات والأعمال بمجال الهندسة المُقادة بالنماذج ركزت على (التوسع العرضي) دون النظر نحو معايير و مقاييس جودة

إجراءاتها، لذا يُعتبر عملنا توسعاً شاقولياً رائداً من نوعه ويركز على قياس و وضع آليات وقواعد لتحسين جودة الإجراءات المقادة بالنماذج. ويبقى هناك الكثير مما يمكننا البحث فيه ضمن هذا التوجه مثل مناقشة أنواع الروابط الأخرى كعلاقات التجميع غير الموجّهة، والعلاقات الحلقيّة، ومناقشة أمور عدد الارتباطات ما بين المفاهيم مثل (واحد لواحد) أو (واحد لكثير) ليدخل في التوزين أيضاً، كما يمكن جعل التقسيم لا يعتمد المنحى الهيكلي المرتبط بالبنية فحسب، بل جعله يتعداه ليصل إلى المنحى المنطقي المُعتمد على المعنى، وذلك من خلال التحليل القواعدي لمعاني أسماء المفاهيم باستخدام الأنطولوجي، وبالتالي يمكن أن نصل إلى تقسيم أكثر دقة وأعلى منطقيّة. كما يمكن فتح آفاق باتجاه تطوير معايير أخرى لقياس الجودة والتأكد من زيادتها.

١٠. المراجع

[1] Brown, W. J., Malveau, R. C., Brown, W. H. , McCormick III, H. W., Mowbray, T. J., 1998 – **Anti-Patterns: Refactoring Software, Architectures, and Projects in Crisis**. John Wiley and Sons, 1st edition.

[2] Moha, N., Gu´eh´eneuc, Y. G., Duchien, L., Le Meur, A. F., 2010 – **DECOR: A Method for the Specification and Detection of Code and Design Smells**, Montreal.

[3] Rahman, F., Bird, C., Devanbu, P., 2010 – **What is that Smell?**. MSR (72-81), Cape Town, South Africa.

[4] Tichy, M., Krause, C., Liebel, G., 2011 – **Detecting performance bad smells for Henshin model transformations**, University of Gothenburg, Sweden.

[5] Baker, P., Loh, Sh., Weil, F., 2005 – **Model-Driven Engineering in a Large Industrial Context**. Springer-Verlag, Berlin, Heidelberg.

[6] Hutchinson,J., Rouncefield, M.,Whittle,J., 2009 – **Model Driven Engineering Practices In Industry**, School of Computing and Communications in Lancaster University, UK.

[7] France, R., and Rumpe, B., 2007 – **Model driven development of complex software**.IEEE The Computer Society.

[8] Mohagheghi, P., Dehlen, V., 2008 – **Where is the Proof? – A Review of Experiences from Applying MDE in Industry**. Proc. 4th European Conference on Model Driven Architecture Foundations and Applications (ECMDA'08).

[9] KOSAYBA, B., 2006 – **A framework for Model Driven Production of Graphic ModelingTools**. IEEE ICCTA, Damascus, Syria.

[10] Saraiva, J., Silva, A., 2010 – **CMS-based Web-Application Development Using Model-Driven Languages**. ACM, Lisboa, Portugal.

[11] Misbhauddin, M., Alshayeb, M., 2012 – **Towards a Multi-view Approach to Model-driven Refactoring**. African Conference for Software Engineering and Applied Computing, King Fahd University of Petroleum and Minerals. Dhahran, Saudi Arabia.

[12] KOSAYBA, B., 2008 – **Towards Standard Control Protocol through Internet Using MDE Approach**.IEEE, Damascus, Syria.

[13] Fowler, M., Beck, K., Brant, J., Opdyke, W., Roberts, D., 1999 – **Refactoring: Improving the Design of Existing Code**. Addison Wesley.

[14] Emden, E. V., Moonen, L., 2002 – **Java Quality Assurance by Detecting Code Smells**.WCRE, 97-108.

[15] Marinescu, R., 2001 – **Detecting Design Flaws via Metrics in Object Oriented Systems**. IEEE Computer Society.

[16] Marinescu, R., 2002 – **Measurement and Quality in Object-Oriented Design (PhD Thesis)**. Timisora: "Politehenica" University of Timisora.

[17] Marinescu. R., 2004 – **Detection strategies: Metrics-based rules for detecting design flaws**. Proceedings of the 20th International Conference on Software Maintenance, pages 350{359}. IEEE Computer Society Press.

[18] Lanza, M., Marinescu, R., 2006 – **Object-Oriented Metrics in Practice**, Springer.

[19] Webster, B. F., 1995 – **Pitfalls of Object Oriented Development**, M & T Books, 1st edition.

[20] Riel, A. J., 1996 – **Object-Oriented Design Heuristics**. Addison- Wesley.

[21] Travassos, G., Shull, F., Fredericks, M., Basili, V. R., 1999 – **Detecting defects in object-oriented designs: using reading techniques to increase software quality**. Proceedings of the 14th Conference on Object-Oriented Programming, Systems, Languages, and Applications, pages 47{56}. ACM Press.

[22] Alikacem, E. H., Sahraoui. H., 2006 – **Generic metric extraction framework.Proceedings of the 16th International Workshop on Software Measurement and MetrikKongress (IWSM/MetriKon)**, pages 383{390}.

[23] Moha, N., Gu'eh'eneuc, Y.-G., Meur, A.-F. L., Duchien, L., Tiberghien, A., 2009 – **From a domain analysis to the specification and detection of code and design smells.** Formal Aspects of Computing (FAC).

[24] Munro. M. J., 2005 – **Product metrics for automatic identification of bad smell design problems in java source-code**. Proceedings of the 11th International Software Metrics Symposium. IEEE Computer Society Press.

[25] Vaucher, S., Khomh, F., Moha, N., Gu´eh´eneuc, Y. G., 2010 – **Tracking Design Smells: Lessons from a Study of God Classes**, Ptidej Team Dept. de G´enieInformatiqueE´colePolytechnique de Montre´al, Canada.

[26] Arendt, Th., Kranz, S., Mantz, F., Regnat, N., Taentzer, G., 2011 – **Towards Syntactical Model Quality Assurance in Industrial Software Development: Process Definition and Tool Support**, Philipps-Universit¨atMarburg, Germany.

[27] Steinberg, D., Budinsky, F., Paternostro, M., Merks. E. (2009). **EMF: Eclipse Modeling Framework**. Addison-Wesley, 2.edition.

[28] Moha, N., Bouden, S., Gu'eh'eneuc, Y. G., 2007 – **Correction of High-Level Design Defects with Refactorings**, Department of Informatics and Operations Research University of Montreal, Quebec, Canada.

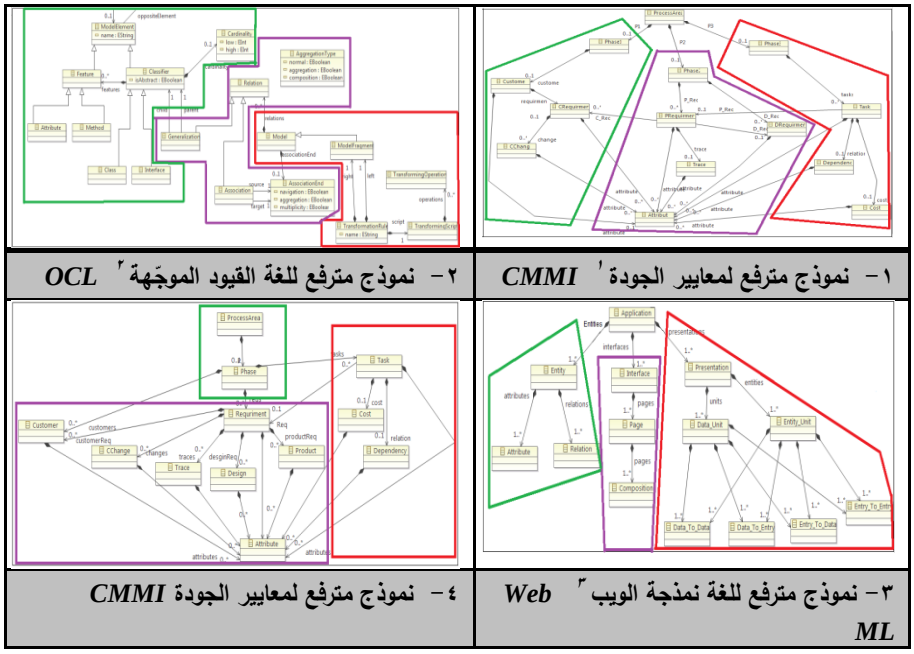
[29] Stelzer, D., Mellis,W., Herzwurm,G., 1997 – **A critical look at ISO 9000 for software quality management.Software Quality Journal**, Germany.

[30] Pressman, R., Graw Hill, Mc., 2001 – **Software Engineering — A Practitioner's Approach**. 1st edition.

١١. الملحقات

١.١١. الملحق / أ /

– قسّم النماذج المترفعة تقسيماً منطقياً بما يضمن سهولة الاستخدام للأدوات الناتجة.



١- نموذج مترفع لمعايير الجودة [1] *CMMI*

٢- نموذج مترفع للغة القيود الموجّهة [2] *OCL*

٣- نموذج مترفع للغة نمذجة الويب [3] *Web ML*

٤- نموذج مترفع لمعايير الجودة *CMMI*

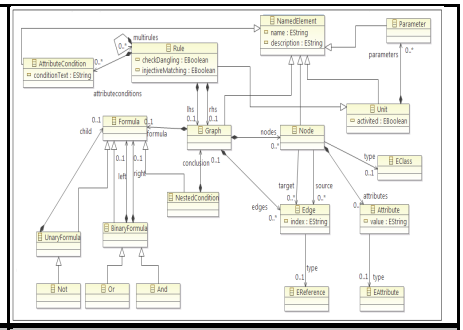
٢.١٠. الملحق / ب /

– قيّم كلاً من النماذج المترفّعة قبل التقسيم وبعده بالاعتماد على الممارسة العمليّة للأدوات الناتجة حسب المعايير التالية.

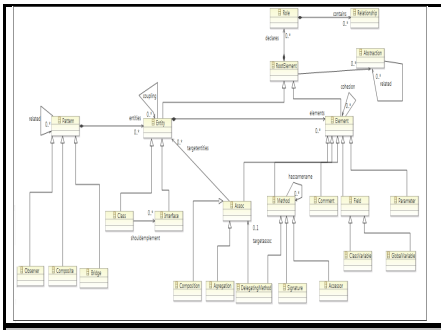
[1] *-Kulpa, P., Margaret,K. (2008). Interpreting The CMMI A Process Improvement Approach 2nd Edition.*

[2] *-Stolc, M., Polasek, I. (2010). A Visual Based Framework for the Model Refactoring Techniques, 8th IEEE International Symposium on Applied Machine Intelligence and Informatics, Slovakia.*

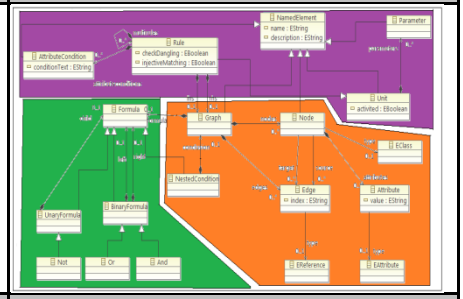
[3] *- Hennicker, R., Koch, N. (2002). Modeling the User Interface of Web Applications with UML, Germany.*



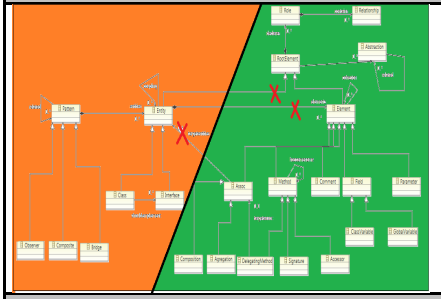
١- النموذج المترفع [4] *Henshin*



٢- النموذج المترفع [5] *PADL*



٣- النموذج المترفع *Henshin* بعد التقسيم بواسطة الخوارزميّة المُقترحة.



٤- النموذج المترفع *PADL* بعد التقسيم بواسطة الخوارزميّة المُقترحة.

معايير الاختبار	قبل التقسيم للنماذج المترفعة (١ + ٢)	بعد التقسيم للنماذج المترفعة (٣+٤)
عدد الساعات لاستخدام وإنشاء النماذج الخاصة (الفهم)	ما يفوق عن الست ساعات	أقل من ساعتين
عدد الضغطات وسطياً لإنشاء عنصر (الاستخدامية)	ما يزيد عن ١٠ ضغطات	أقل من ٦ ضغطات
استعمال الملف المساعد (الوضوح)	نعم تمّ استعمال الملفات المساعدة	النموذج واضح ولا داعي للملفات

- هل تعتقد أنّ جودة الأدوات قد ارتفعت بعد التقسيم الحاصل على النموذج المترفع بفعل خوارزميّة التقسيم المُقترحة؟ بالطبع زادت الجودة على عدة مناحي من حيث قابلية الفهم والتوسع والمرونة والوضوح والاستخداميّة.

[4]- *Tichy, M., Krause, C., Liebel, G. (2011). Detecting performance bad smells for Henshin model transformations, University of Gothenburg, Sweden.*

[5]- *Moha, N., Bouden, S., Gu'eh'eneuc, Y. G. (2007). Correction of High-Level Design Defects with Refactorings, Department of Informatics and Operations Research University of Montreal, Quebec, Canada.*