رأسملة تصميم النظم باستخدام الهندسة المقادة بالنماذج

ملخص البحث

تمتاز النماذج "Models" بمستوى تجريد عالي مما يسمح بتبسيط فهم و معالجة النظم. تتزايد أهمية النماذج في عالم المعلوماتية مع تزايد عدد التطبيقات و حاجة المهندسين لحفظ تصاميمهم إما لصيانة هذه التطبيقات لاحقاً أو لإعادة إنتاج هذه التطبيقات باستخدام تقانات جديدة أو لدمج هذه التطبيقات في نظم أكثر تعقيداً.

في هذا البحث سنعرض كيف نستخدم الهندسة المقادة بالنماذج لتصميم إجرائيات لتطوير البرمجيات من خلال تنظيم عملية التطوير في عدة مراحل حسب الاهتمامات التي تفرضها طبيعة المسألة و دعم كل مرحلة بأدوات متخصصة و من ثم تحويل و دمج نماذج خرج كل مرحلة للوصول إلى المنتج البرمجي النهائي. تمثل النماذج في عملنا المركز الأول و تتوسط مختلف مراحل إجرائية التطوير و تعمل كغراء فيما بينها. كما سنوضح كيف يمكننا إجراء عمليات على النماذج و كيف يمكننا استخدام النماذج نفسها لتوليد أدوات نمذجة متخصصة بكل مرحلة من التطوير و كيف يمكننا استخدام هذه الأدوات لدمج النماذج آلياً أو يدوياً.

الكلمات المفتاحيّة :

الهندسة المُقادة بالنماذج, النماذج، النماذج المترفعة (Meta-models)، تحويلات النماذج.

Capitalization of System Design using Model Driven Engineering

ABSTRACT

Models are abstractions of systems. High levels of abstraction allow easier understanding and handling of systems. Models became a necessity in the world of computer science because the number of applications explodes and engineers seek to preserve knowledge related to these applications either to reproduce them using new technologies or to integrate them in more complex systems.

In this paper, we will present how to use Model Driven Engineering to design software development process. We organize the development process in several phases according to the problem nature. We support each phase by dedicated tools. After that, we transform and merge the output models from each phase until we obtain the final software product.

Models represent the principal key in our work. They work as intermediate between the different process phases. Also, we will explain how to use models to produce modeling tools dedicated to each development phase and how to use these tools to merge the models automatically or manually.

Keywords :

Model Driven Engineering (MDE), Models, Meta-models, Model Transformations.

1. مقدمة :

تعددت لغات البرمجة والتقنيات المستخدمة لتطوير البرمجيات. مما جعلنا نضطر لإعادة كتابة الأنظمة القديمة باستخدام تقنيات جديدة، مثلاً لعرض خدمات نظام ما على الويب أو لدمج النظام القديم مع أنظمة أخرى مكتوبة بتقنيات جديدة مختلفة. ظهرت العديد منالتقنيات و المنهجيات التي تدعم مفهوم إعادة الاستخدام بحجم حبيبي متغاير مثل functions, objects, componentsو لكنها لم ترقى لرأسملة تعريف النظم بشكل مستقل تماماً عن التقنيات. ندرك جيداً بأنه لا يوجد تقنية مثالية و لن يوجد. فالتقنيات الجديدة تحاول معالجة مشاكل التقنيات القديمة و لها مشاكلها أيضاً و التي سنحاول أيضاً حلها في التقنيات الجديدة و التي لم يتوقف تطويرها و لن يتوقف.يمكننا بدقة أكبر تحديد المشكلة وهي بأن منطق عمل النظام مخلوط و بشدة بالكود التقني. كما أننا نستطيع تصور الحل بأنه لا بد من الفصل بين منطق عمل النظام و الكود التقني مما يسمح لنا بإلقاء الكود التقني و الحفاظ على منطق العمل و إعادة استخدامه بعدة سياقات و (تقنيات).

نخصص الفصل الرابع من هذا البحث للدراسة المرجعية حيث نقدم للنمذجة و نشرح مبادئ الـ Meta-Modelingو آليات استثمار النماذج و تحويلاتها و الخطوط العريضة للهندسة المقادة بالنماذج (Model Driven Engineering) MDE.نطبق MDEفي الفصل الخامسعلى عملية تطوير تطبيقات الويب كدراسة حالة "إنتاج مقاد بالنماذج لتطبيقات الويب" و نبين أهمية استخدام النماذج و الفائدة من الفصل بين نمذجة محتوى الصفحات و نمذجة سلوك التطبيق بكامله. بنفس الوقت نبين كيفية تنظيم استخدام النماذج و الـ Meta-models و تحويلات النماذج لتحقيق إطار عمل لتوليد تطبيقات الويب آلياً و بشكل مستقل عن تقنيات التنفيذ والأجهزة.نعرض في الفصل السادس للحالة الدراسية الثانية "تقييس التحكم بالدارات" حيث نفصل بين نمذجة قدرات دارات التحكم و نمذجة سلوك الدارة في تطبيق ما (تسلسل آوامر هذه الدارة) و نشرح أهمية النماذج و كيفية توليد الأدوات التي تسمح بتعريفها و كيفية تفاعلها فيما بينها

ضمن إجرائية التطوير. نلخص في الفصل السابع ما أوردناه في هذا البحث و نعرض الأفاق المستقبلية لهذا العمل.

2. هدف البحث :

في هذا البحث، نركز على استخدام تقنيات الـ meta-modeling لتحقيق رأسملة تعريف النظم و نشرح كيف يمكن استخدام هذه التقنيات لرأسملة معرفة تطوير النظم.تقوم فكرتنا الرئيسة على الفصل بين الاهتماماتاللازمة لتطوير النظام و تبسيطها و تجريدهاحيث نقوم بتقسيم إجرائية تطوير البرمجيات في مجال ما إلى عدة مراحل. كل مرحلة توافق لـ meta-model و مدعومة بأداة نمذجة رسومية. تسمح كل أداة نمذجةبتعريف نموذج مجرد عن اهتمام في النظام. نولد هذه الأداة انطلاقاً من الـ meta-model الخاص بهذه المرحلة. علماً بأننا طورنا سابقاً [5][4] إطار عمل لتوليد أدوات نمذجة رسومية انطلاقاً من meta-modelsو بأسلوب مقاد بالنماذج و كانت من أوائل الإجرائيات المقادة بالنماذج و حددنا فيها الأسس و القواعد الضرورية لتشكيل إجرائية تطوير مقادة بالنماذج.كما نحاول في هذا البحث التمييز بين دور مهندس البرمجيات في تصميم إجرائية التطوير و مراحلها و بين دور مطوري البرمجيات في استخدام أدوات الإجرائية المقدمة لهم لتصميم و تحقيق تطبيقاتهم دون الحاجة لتأهيل تقني بل يجري التركيز على تدريبهم مهنياً فقط على نمذجة متطلبات أعمالهم.

3. مواد وطرائق البحث :

نطور حالتين لتوضيح خطوات بناء إجرائية تطوير برمجيات باستخدام تقنيات و مفاهيم الـ Meta-modeling و كذلك نستخدم هذه التقنيات لأتمتة توليد التطبيقات و نبين أهمية إعادة استخدام نماذج التطبيق لإعادة توليده نحو العديد من التقنيات و كذلك أهمية إعادة استخدام و تعديل أجزاء من هيكلية الإجرائية الهندسية.

تخص الحالة الدراسية الأولى تصميم و تطوير مواقع ويب ديناميكية بأسلوب مستقل عن الأجهزة و لغات البرمجة. سنبين في هذه الدراسة كيف نستخدم النماذج للفصل بين منطق الخدمة (التفاعل بين صفحات الموقع) و بين توصيف محتوى

صفحات الموقع. سنحدد أولاً ما هي المعلومات اللازمة لوصف محتويات الصفحات (Page meta-model) و كذلك المعلومات اللازمة لوصف التفاعل فيما بين هذه الصفحات (Control meta-model).ومن ثم سنصمم إطار عمل مقاد بالنماذج نحدد فيه عدد مراحل التطوير و ترتيبها و نماذج الخرج لكل مرحلة وعمليات التحويل والدمج حتى الوصول للمنتج النهائي (موقع الويب الديناميكي). في النهاية سنستخدم تقنيات الـ Meta-Modeling و الأدوات التي طورتها في أعمال سابقة [5] [4] لدعم إطار العمل هذا بأداتي نمذجة خاصتين : تسمح الأداة الأولى لمطوري مواقع الويب بتصميم محتوى صفحاتهم بينما تسمح الأداة الثانية بتحميل مختلف نماذج صفحات الموقع المعرفة سابقاً و تعريف سلوك الموقع ككل و بمكان واحد و على مستوى تجريد عالي من خلال تعريف كيفية تفاعل صفحات الموقع فيما بينها.

نؤكد في الحالة الدراسية الثانية على نفس الخطوات لهندسة إجرائية تطوير بالاعتماد على النماذج و لكن ضمن سياق آخر و هو تقييس العمل ضمن مجال معين من التطبيقات و أهميته و نوضح كيفية استخدام النماذج لتحقيق هذا الهدف.

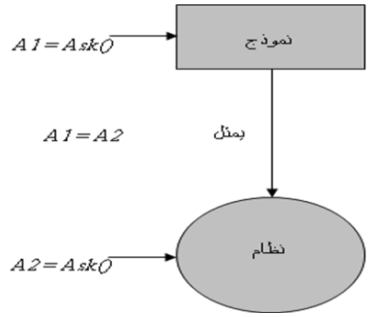
4. الدراسية المرجعية :

تشكل النماذج العنصر الأساسي في عملنا حيث أننا نحتفظ بها لإعادة استخدامها في تطوير تطبيقات جديدة فهي تسمح إذاً برأسملة تعريف النظم. كما أنها تشكل الصمغ الذي يربط أدوات النمذجةفي إجرائية التطوير بعضها ببعض حيث أن خرج أداة في مرحلة ما هو نموذج و سيكون الدخل لأداة أخرى في المرحلة التالية حيث يتم إضافة التفاصيل عليه و دمجه مع باقي اهتمامات النظام حتى الوصول إلى كود النظام. و كما سنرى بعد قليل بأن عمليات تحويل النماذج و دمجها هي بحد ذاتها نماذج و بالتالي نصل لرأسملةمعرفة تطوير النظم لأن معرفة التطوير هنا تتمثل بالانتقال من مستوى لآخر في إجرائية التطوير و تتم عبر تحويلات النماذج و التي هي بحد ذاتها نماذج صريحة و واضحة و ليست مخفية ضمن كود لأداة بل على العكس يتم توليد هذه الأدوات من النماذج.

1.4.النمذجة و النمذجة المترفعة"Meta-modeling" :

نمذجة نظام ما هي وصف للخصائص التي تهمنا من هذا النظام بهدف تسهيل معالجة ما يهمنا من هذا النظام.بالتالي فإن النموذج يجب أن يجيب على بعض الأسئلة التي تهمنا بدلاً من النظام بحيث تكون أجوبة النموذج مطابقة لأجوبة النظام.الشكل (1) يبين علاقة النموذج بالنظام.يختلف معنى كلمة "نموذج" بحسب مجالات نشاط الحياة الإنسانية [الفن، صناعة الملابس، هندسة البرمجيات .. إلخ]. بالمقابل فإن كلمة "نموذج" لها معاني مشتركة بين كل هذه المجالات:

- النموذج هو تجريد لشيء ما في العالم الحقيقي.
- النموذج يختلف عن الشيء الذي يمثله. في تفاصيله، في حجمه، إلخ.
- النموذج يمكن أن يستعمل لإنتاج شيء ما في العالم الحقيقي.



الشكل (1) : علاقة النموذج بالنظام

يجب تعريف النماذج باستخدام مباشر للمفاهيم الخاصة بمجال التطبيقات حيث نعمل، بدلاً من استخدام مفاهيم عامة مثل UMLلأنه سنقوم بلي معانيها لتناسب مجالتطبيقاتنا كما لا يمكن استثمارها مباشرة في تطوير تطبيقاتنا. حيث أن المفاهيم الأساسية في مجال تطوير تطبيقات الويب هي الصفحات و الانتقال بينها و ليس الصفوف و علاقاتها و كذلك في مجال نظم الزمن الحقيقي فإن المفاهيم الأساسية هي المهام و التزامن بينها. كما يجب عند تعريف النماذج التقيد بعدد من القواعد المعرفة بشكل جيد (Meta-model) لتحاشي الأخطاء في تفسير النماذج خصوصاً عند استخدام هذه النماذج لتوليد النظم آلياً.

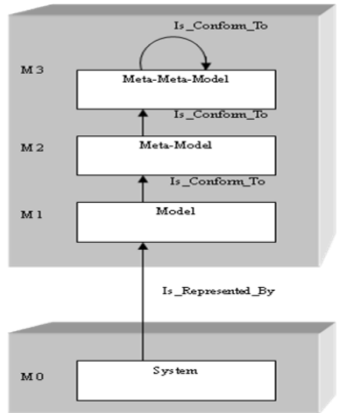
2.4.تقنيات الـ Meta-modeling

تقنيات الـ Meta-Modeling تسمح بتعريف مجموعة المفاهيم اللازمة لنمذجة الأنظمة أو ما يدعى بالـ Meta-models. الـ Meta-Model يعرف لغرض دقيق حيث يقدم مفاهيم خاصة بمجال معين من التطبيقات.

تبنت الـ OMG (Object Management Group) الـ MOF (Meta-Object-Facility) كوسيلة قياسية من أجل الـ "Meta-Modeling". الـ "MOF" هو معمارية من أربعة مستويات M0, M1, M2, M3. في كل مستوى نعرف المعلومات التي تصف المستوى الأدنى.يمكننا أن نقول بأن معمارية الـ "MOF" مرتبة من 3+1 مستوى. المستوى M0 هو النظام الحقيقيو هو ممثل بالـModel المعرفعندالمستوى M1. نموذج المستوى M1 يكون موافقاً لـ "Meta-Model" المعرف لدى المستوى M2 أي أن عناصر هذا النموذج هم أفراد من مفاهيم المستوى الأعلى و تخضع للعلاقات المعرفة بين مفاهيم المستوى الأعلى. الـ Meta-Model يوافق الـ "Meta-Meta-Model" عند المستوى M3 بينما الـ Meta-Meta-Model يوافق نفسه كعلاقة انعكاسية. الشكل(2) يبين تنظيم الـ MOF في معمارية من 3+1 مستوى.علاقة الموافقة "Is_Conform_To" بين مستوى و مستواه الـ "Meta" تتواجد بين المستويات الثلاثة العليا M1, M2, M3 من معمارية الـ "MOF". هذه العلاقة تعني بأن عناصر مستوىً هم أفراد من عناصر المستوى الأعلى. الشكل (3) يشرح استعمال معمارية الـ MOF من خلال أمثلة و فيما يلي نقدم تعريفاً لمستويات الـ MOF مستعينين بهذه الأمثلة.

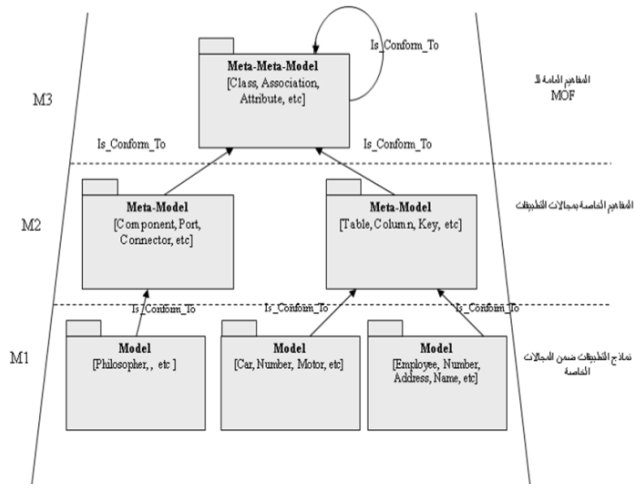
المستوى M2(Meta-Model) : يمثل الـ Meta-Model نوع لمجموعة من النماذج و يحدد كيفية استخدام هذه النماذج ضمن سياق معين. فمثلاً نعرف مفاهيم Table, Column, Key و علاقاتها ضمن Meta-Model خاص لنمذجةنظم قواعد البيانات. يمكننا بالاستناد إلى القواعد و العلاقات المعرفة في هذا الـ Meta-Modelأتمتة العمليات المطبقة على نماذج البيانات الموافقة مثل التوليد الآلي للكود بلغة الـ SQLابتداءاً من هذه النماذج أو إجراء عمليات أمثلة optimization على هذه

النماذج و حتى عمليات محاكاة و اختبارات. في Meta-Model آخر خاص بنمذجة التطبيقات المعتمدة على المكونات البرمجية (Component Based Applications). نعرف مفاهيم مثل Component, Port, Connection اللازمة لنمذجة هذه التطبيقات. عند هذا المستوى نجد عدة Meta-Models كالـ Meta-Model الخاص بالـ UML و الـ Meta-Model الخاص بتطبيقات المكونات البرمجية و الـ Meta-Model الخاص بقواعد المعطيات. كما أنه من الممكن تعريف Meta-Model بهدف تقييس النمذجة و التصميم ضمن المجال الذي من أجله تم تعريف هذا الـ Meta-model.



الشكل (2) : معمارية الـ MOF من 3+1 مستوى

المستوى M3(Meta-Meta-Model): وبنفس الأسلوب فإننا نحتاج إلى مفاهيم لنمذجة الـ Meta-Models ولهذا عرفنا الـ MOF و مفاهيمه الـ Association, Attribute ... ,Class. لا يوجد إلا Meta-Meta-Model واحد في معمارية الـ MOF.



الشكل (3) : المستويات الثلاثة العليا للـ MOFمع Two Meta-Models عند المستوى M2

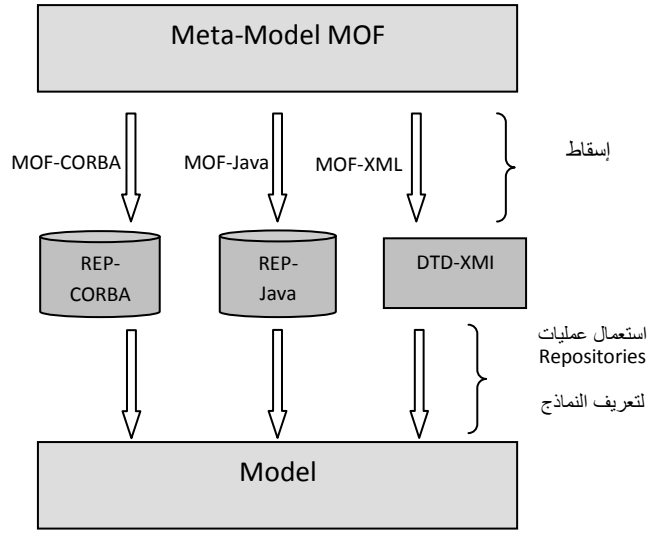
إن تعريف مختلف الـ "Meta-Models" عند مستوى أعلى وحيد يسهل اندماجهم فيما بعد و بالتالي مكاملة مجالات تطبيقاتهم.على سبيل المثال، نحن ننتج عدة أنواع من الخرائط لوصف الأراضي. بالمقابل، يتوجب علينا في بعض الأحيان استعمالعدة أنواع منالخرائط من أجل تحديد موقع غرض أو مورد محدد [11].

حتى لا تتعدد مستويات معمارية MOF إلى ما لا نهاية فقد تم تعريف مفاهيم الـ MOF باستخدام مفاهيم الـ MOF ذاتها عند المستوى M3. و لهذه العلاقة الانعكاسية أهمية خاصة سنتطرق إليها عندما نتعرف على إنتاج أدوات نمذجة خاصة بالـ MOF.

المستوى M1(Model) :عند هذا المستوى نستخدم أفراد مفاهيم مستوى الـ "Meta-Model" لوصف مسألة/نظام ضمن المجال الذي من أجله عرفنا هذا الـ "Meta-Model". نستطيع أن نرى عدة نماذج موافقة لنفس الـ "Meta-Model" حيث كل نموذج يناقش مشكلة ضمن المجال الذي من أجله هذا الـ "Meta-Model" قد تم تعريفه. على سبيل المثال، في الشكل (3) نستطيع أن نرى نموذجين موافقين للـ "Meta-Model" الخاص بقواعد المعطيات. الأول يعرف جدول الذي يصف معلومات مثل اسم و رقم و عنوان موظف. بينما الثاني يعرف ضمن جدول Car المعلومات المهمة مثل نوع المحرك و رقم السيارة إلخ.

3.4. إسقاطMeta-modeling إلى أدوات النمذجة

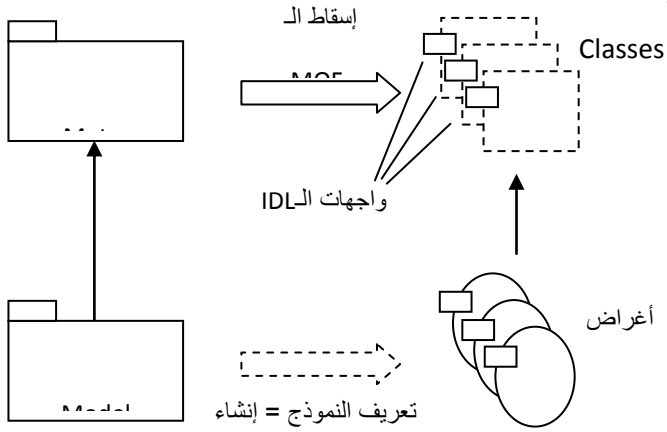
MOF يقترح مصطلح الـ Repository كوسيلة لمعالجة و تخزين النماذج بشكل فيزيائي. تخزين النماذج يمكن أن يتم ضمن قاعدة بيانات، ضمن ملف أو ضمن الذاكرة فقط [34].يمكن أن نعرف الـ Repository بمجموعة من العمليات التي تسمح معالجة نوع من النماذج. MOF يعرف فقط القواعد التي تسمح اعتباراً من Meta-Modelبتوليد واجهة لعمليات الـ Repository. وبما أن الـ MOF مستقل عن تقنيات التنفيذ فإنه يعرف عدة إسقاطات نحو التقنيات الأكثر أهمية CORBA, XML, Java كما هو موضح في الشكل (4).هذه الـ Repositories تقدم عمليات لتعريف و معالجة نماذج موافقة للـ Meta-model المولدة انطلاقاً منه.



الشكل (4) : إسقاطات الـ MOF نحو مختلف التقنيات

بما أن الـ MOF ذاتي التوصيف [موافق لنفسه] أي يمكننا اعتباره Meta-Model موافق لنفسه, فإننا نستطيع تطبيق الإسقاطات السابقة على الـ Meta-Meta-Model نفسه و بالتالي سنولد Repositories لمعالجة و تعريف Meta-Models دون أي تعديل على قواعد الإسقاط.

إسقاطات MOF نحو مختلف التقنيات : يعرفMOF قواعد الإسقاط التي تسمح بتوليد واجهات الـ IDL (Interface Definition Language) اعتباراً من Meta-Model. إن تنفيذ هذه الواجهات يسمح بالحصول على Repositories تسمح بتعريف النماذج على هيئة أغراض CORBA. الشكل (5) يوضح الإسقاط MOF-CORBA. Java تعرف الإسقاط JMI (Java Meta-data Interchange) [32]. هذا الإسقاط مماثل للإسقاط MOF-CORBA مع اختلاف واحد و هو أن الواجهات و الـ Repositories منفذة بلغة الـ Java و بالتالي فإن النماذج مخزنة على هيئة أغراض OMG.Java تبنى الإسقاط MOF-XML المسمى بـ XMI(XML Meta-data Interchange). MOF يهتم بالتقنية XML لدعم تبادل تعريفات الـ Meta-Models بين الأدوات. XMI تعرف مجموعة القواعد التي تسمح بتوليد الـ DTD لوثيقة XML اعتباراً من Meta-Model. هذه الـ DTD تعرف بنية نوع من ملفات الـ XML و التي تسمح بتبادل النماذج الموافقة لهذا الـ Meta-Model على هيئة ملفات XML موافقة للـ DTD المولدة.



الشكل (5) CORBA

4.4. استثمار النماذج

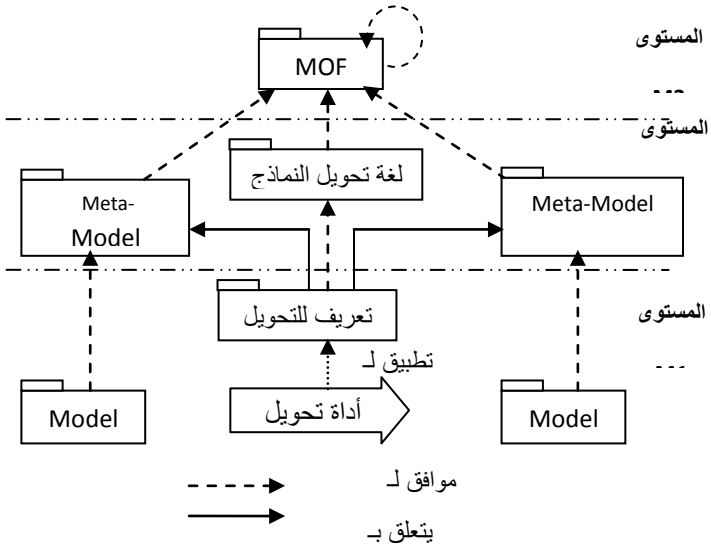
يجري البحث حالياً في هندسة البرمجيات فيأتمتة تطوير الأنظمة اعتباراً من النماذج التي تصفها.النماذج تسمح بفهم و معالجة تجريد للنظام حسب وجهة نظر

معينة. يهدف تطبيق عمليات الدمج و التحويل على هذه النماذج للانتقال من مستويات مجردة للنظام إلى مستويات أقل تجريداً أو حتى كود النظام. فيما يلي سنقدم تحويلات النماذج و المنهجيةMDE التي تستخدم النماذج لتنظيم تطوير البرمجيات.

تحويلات النماذج (Model Transformations) :من الممكن امتلاك عدة نماذج تخص نفس النظام. يمكن لهذه النماذج أن تصف أجزاءً مختلفةً للنظام أو يمكن لهذه النماذج أن تصف النظام حسب عدة وجهات نظر أو على عدة مستويات تجريد مختلفة. بما أن هذه النماذج تناقش نفس النظام فإنه يتوجب علينا في لحظة معينة الانتقال من نموذج مجرد إلى نموذج أكثر تفصيلاً (من الناحية التقنية أو المنطقية). كما يتوجب أيضاً علينا دمج النماذج التي تصف عدة جوانب من النظام في نموذج واحد. الشكل التالي يعبر عن لغة تحويل النماذج المعرفة من قبل الـ OMG [19].

يبين الشكل (6) حقيقة مهمة و هي بأن تحويل النماذج بين نوعين من النماذج هو نموذج بحد ذاته في المستوى M1. مما يسمح بحفظ معرفة تطوير النظم لأن خطوات الانتقال في مراحل إجرائية التطوير يتم بتحويل النماذج و الذي هو نموذج قائم بذاته.

الهندسة المقادة بالنماذج (Model Driven Engineering) MDE: تمثل منهجية واعدة في هندسة البرمجيات يضع النماذج في مركز تطوير البرمجيات. فكرتها الرئيسية هي استخدام النماذج و تحويلاتها لتنظيم فعالية تطوير نظام ما [06] [07] [08][09].



الشكل (6) :نموذج تحويل النماذج من MOF

MDE يسمح بتوصيف منهجية لتعريف المشكلة و كيفية الذهاب إلى حلها فيقسم فعالية تطوير البرمجيات على عدة مستويات من التجريد. النماذج في مستوى معين تختلف عن النماذج في مستوى آخر بدقتهم التقنية أو المهنية (Business). الانتقال بين هذه المستويات يتم عبر تحويلات النماذج. MDE يشجع على تحديد و فصل الاهتمامات المختلفة لنظام مما يسمح بنمذجة كل اهتمام بشكل مستقل عن الآخر و من ثم نعرف دمج توصيفات النظام على مرحلة أو أكثر.MDE يسمح برأسملة التوصيفات البدائية للنظام و رأسملة معرفة تطوير (Know–How) هذا النظام. معرفة تطوير النظام تُعرف ضمن تحويلات النماذج. هذا يسمح بامتلاك وثائق لكيفية الانتقال بين مراحل إجرائية تطوير النظام.كما يمكن لفعالية تطوير البرمجيات MDE أن تتكيف بسرعة تلبيةً لتغيير في شروط تحقيق النظام حيث يمكن بسهولة حصر أجزاء الفعالية الواجب تغييرها بينما يمكن إعادة استخدام الأجزاء الأخرى غير المصابة.MDEتمثل ارتقاء ضمن مجال تطوير البرمجيات حيث تم رفع مستوى التطوير من التجميع المباشر للأغراض (Objects) نحو تحويلات النماذج. هذه التحويلات مقادة بالنماذج المعرفة في مستوى تجريد أعلى مما يسمح بأتمتة عملية تجميع الأغراض [02].عند تعريف فعالية MDE فإنه يتوجب علينا الإجابة على الأسئلة التالية [18] :

- كم مستوى تجريد يتواجد في فعالية تطوير البرمجيات؟
- ما هي المفردات المجردة للاستخدام في كل مستوى تجريد؟
- ما هي المعلومات الإضافية للدمج في مستوى التجريد الأدنى؟
- كيف نولد الشيفرة؟
- كيف نتوثق من نموذج بالمقارنة مع نماذج المستوى الأعلى؟

5. دراسة حالة 1 "إنتاج مقاد بالنماذج لتطبيقات الويب" :

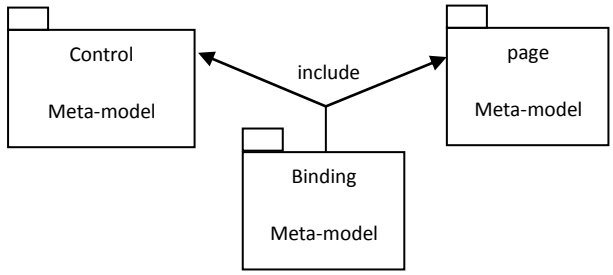
برمجة مواقع الويب الديناميكية هي عبارة عن form-based service حيث نرسل صفحة html إلى متصفح المستخدم الرسومي. صفحة الـ html تقدم معلومات للمستخدم و مجموعة من الحقول لطلب معلومات من المستخدم كالاسم و كلمة السر. بعد استقبال هذه المعلومات من المستخدم، الخدمة تعالج المعلومات المدخلة من قبل مستخدم الموقع و من ثم ترسل صفحة html أخرى و هكذا.نقطة الضعف في هذا النوع من التطبيقات هو الخلط ما بين شيفرة الخدمة (business logic) و كود الواجهات الرسومية الموجهة نحو المستخدم (html pages).برمجة مواقع الويب الديناميكية بهذا النحو تماثل إلى حد بعيد البرمجة باستخدام اللغات غيرالمهيكلة حيث استخدمنا تعليمة GOTO. يمكننا تشبيه تعليمة GOTO في اللغات غيرالمهيكلة بذكر عنوان صفحة الويب اللاحقة ضمن صفحة ويب سابقة.إن فصل التحكم في البرنامج عن محتوى صفحات الويب يسمح لنا بإعادة استخدام الصفحات ضمن تطبيقات أخرى. كما أنه يسهل عملية تنقيح البرنامج و يساعد على فهم السلوك العام للبرنامج من خلال قراءة لكود الخدمة دون الغوص في تفاصيل واجهات المستخدم.

1.5. تصميم إطار عمل مقاد بالنماذج لتطوير تطبيقات الويب

نعرض فيما يلي تصميمنا لإطار عمل مقاد بالنماذج لتوليد تطبيقات الويب آلياً. نستخدم في إطار العمل هذا نماذج مجردة لوصف الخدمة (تسلسل الصفحات، اتخاذ

القرارات حسب دخل المستخدم) و نماذج مجردة لوصف واجهات المستخدم (ما يجب إظهاره للمستخدم و ما يجب التقاطه منه). ميزة إضافية لهذا الإطار هو أن توصيف التطبيق مستقل عن تقنية التنفيذ و عن الأجهزة حيث يمكننا فيما بعد تحقيق الخدمة كتطبيق ويب عادي أو باستخدام تقنية الـ VoiceXML و من أجل أجهزة الهاتف المحمول مثلاً. كما أن مستخدمو هذا الإطار لا يحتاجون لتأهيل تقني خاص بكتابة صفحات التطبيق و برمجة الانتقال بينها و إنما فقط يكفي فهمهم بأن تطبيق الويب هو مجموعة من الصفحات التي يمكننا الانتقال فيما بينها و تحديد هذه الصفحات الخاصة بعملهم و سياسة الانتقال بينها.

استخدمنا في هذا الإطار الـ meta-models للفصل بين مجالين أساسين : مجال توصيف محتويات صفحات الموقع ومجال منطق الخدمة. Page Meta-model يضم المفاهيم التي تسمح بتوصيف عناصر صفحات الموقع و صفاتها بينما Control Meta-model يعرف مفاهيم تسمح بتطبيق عمليات مثل (if, while, GetField, SetLabel, SetField) بين صفحات الموقع لتحديد منطق عمل التطبيق (تسلسل التفاعل بين صفحاته). يبين الشكل (7) الأساس النظري لإطار العمل و الـ meta-models المستخدمة فيه. Page Meta-model يسمح بتعريف نماذج لصفحات الموقع. Control Meta-model يسمح بتعريف كيفية تفاعل صفحات الموقع فيما بينها. الفصل باستخدام هذين الـ meta-models يسمح بتقسيم الأدوار بين مستخدمي الإطار بين مصمم لصفحات الموقع و بين مصمم لسلوك الملاحة بين صفحات الموقع. كما يسمح بإعادة استخدام نفس نماذج الصفحات في العديد من تطبيقات الويب.



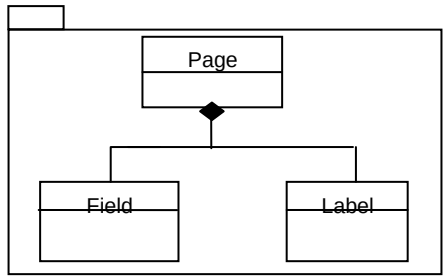
الشكل (7) : الـ meta-models الخاصة بإطار العمل

لتحديد كيفية يمكن تفاعل نماذج الصفحات و التحكم فإننا نعرف الـ Binding Meta-model و الذي يضم مفاهيم كلا الـ meta-models السابقين بالإضافة لعلاقات جديدة فيما بينها.

2.5. التحقيق العملي لإطار عمل تطوير تطبيقات الويب

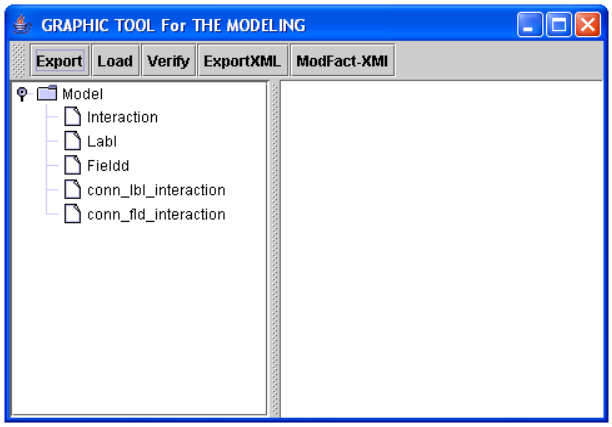
إن دعم إطار العمل هذا بأدوات نمذجة ضروري ليتمكن مستخدمي الإطار من تصميم نماذج تطبيقاتهم. نرى بأن مصممي الصفحات يحتاجون أداة تسمح لهم بتحديد محتوى صفحاتهم من عناصر و أشكال. كما نرى بأن مطور الموقع يحتاج لأداة تسمح له بتحميل نماذج صفحات الموقع و تعريف التفاعل (الملاحة) فيما بينها جميعاً وفي مكان واحد.في الفقرة 2-3 من هذا البحث وضحنا بأنه يمكننا انطلاقاً من meta-model التوليد الآلي لـ repository يسمح بتخزين و تعريف نماذج موافقة للـ meta-model الأصل.

في أعمالنا السابقة[4][3] قدمنا إطار عمل يسمح بتوليد أداة نمذجة رسومية انطلاقاً من MOF Meta-model. تقدم هذه الأدوات عمليات لإنشاء أفراد من مفاهيم هذه الـ Meta-models المولدة و لمعالجة هذه الأفراد وفقاً للعلاقات المعرفة بين مفاهيم هذا الـ Meta-models.



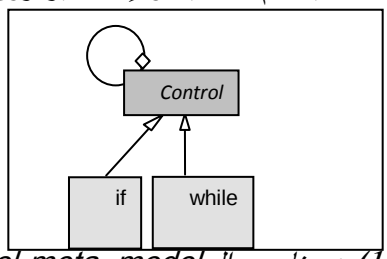
الشكل (8) : مفاهيم الـ Page Meta-model

الشكل (8) يبين Page meta-model و الذي يعرف العناصر التي يمكن لمصمم صفحات الموقع إنشاء أفراد من هذه العناصر لتعريف نموذج صفحة نحدد فيه كيفية التفاعل مع مستخدمي الموقع. أفراد العنصر "Field" تسمح بالحصول على معلومات من مستخدم الموقع بينما الأفراد من العنصر "Label" لتوجيه رسائل و توجيهات لمستخدمي الموقع.



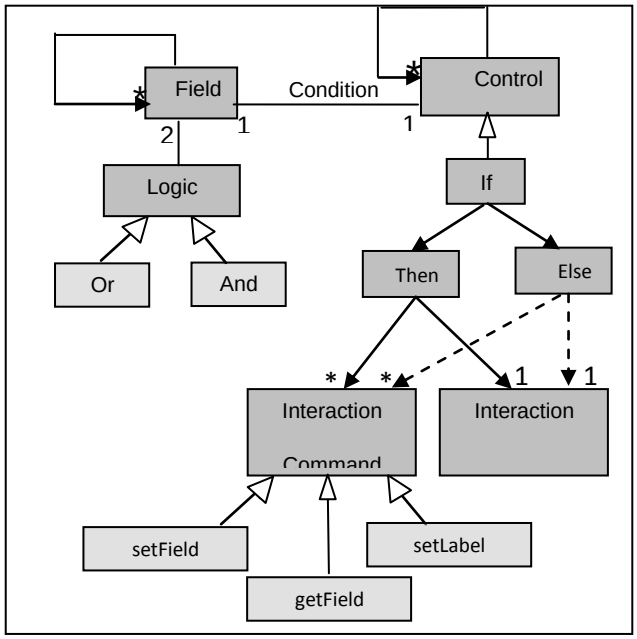
الشكل (9) : أداة رسومية لنمذجة صفحات المواقع

إن هذا الـ meta-model بسيط. يمكننا تطوير هذا الـ meta-model ليشمل أنواع مختلفة من عناصر الإدخال و الإخراج و لكننا نفضل بقاؤه بسيطاً كوننا نركز على التنظيم العام لإطار العمل و كيفية دمج النماذج خطوة خطوة حتى بناء النموذج الكلي للنظام. كما أننا نركز على توزيع و فصل أدوار الأشخاص المشتركين بتصميم تطبيق الويب و دعمهم بأدوات النمذجة المخصصة حسب دورهم و مهمتهم في تصميم النظام.انطلاقاً من هذا الـ meta-model ولدنا آلياً الأداة الرسومية الموضحة بالشكل(9) و التي تدعم تصميم نماذج صفحات التطبيق. هذه الأداة تم توليدها باستخدام أعمالنا السابقة [12] [9].نماذج الصفحات المعرفة باستخدام الأداة المبينة بالشكل (9) يمكن تصديرها كملفات XML لإعادة استخدامهافي العديد من التطبيقات بشكل مستقل عن بيئات التنفيذ. كما أن مصممو هذه الصفحات لا يحتاجون لمعرفة تقنية XML أو حتى التقنية المستخدمة لبناء تطبيقاتهم النهائية و إنما سيركزون بشكل منطقي فقط على تصميم محتوى صفحاتهم.



الشكل (10) : عناصر الـ Control meta-model

الشكل (10) يبين جزءاً من مفاهيم Control Meta-model(if, while, etc.) و التي يحتاجها مصمم التطبيق لتحديد تسلسل التفاعل مع المستخدم.هنا لم نقم بتوليد أداة نمذجة انطلاقاً من هذا الـ meta-model لأنه لا أهمية لنموذج تحكم مستقل عن صفحات التطبيق.الشكل (11) يوضح الجزء الأهم من إطار عملنا و هو الـ Binding meta-model حيث نعرف المفاهيم و العلاقات اللازمة للربط بين كل من عناصر الـ Control meta-model و عناصر الـ Page meta-model.



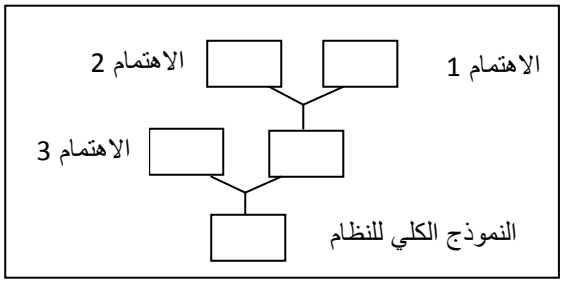
الشكل (11) : Binding Meta-model

قادنا مبدأ الفصل بين الاهتمامات إلى تصميم عدة meta-models و التي سمحت لنا بنمذجة كل اهتمام على حدا. مما مكننا من إعادة استخدام نماذج النظام في أنظمة أخرى و تقسيم الأدوار بين المشاركين في إجرائية التطوير. لكن يجب تجميع هذه النماذج في مكان ما و بالتالي لا بد من تعريف قواعد تحدد كيف لنماذج اهتمامات النظام أن تترابط فيما بينها و لهذا عرفنا الـ Binding meta-model.

لا بد أن ننوه بأنه ليس من الضرورة أن يتم تعريف قواعد الترابط دفعة واحدة في meta-modelوحيد و إنما يمكننا القيام بذلك تدريجياً بحيث ندمج كل 2 meta-

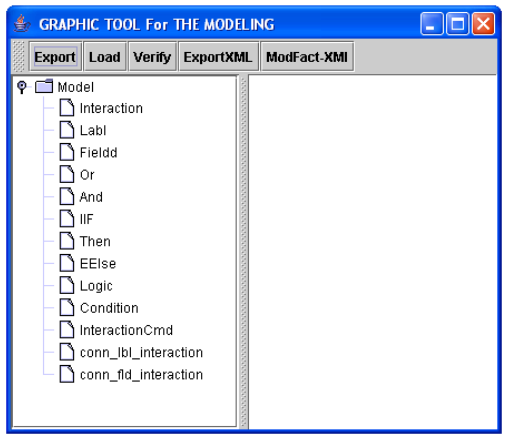
models مع بعضهم البعض و من ثم ندمج النتيجة مع meta–model آخر و هكذا. و بالتالي نشكل شجرة Y كما يبينه الشكل (12).

إن طبيعة الأنظمة المراد تطويرها و الإجرائية التي نحددها لتطوير هذه الأنظمة تحدد شكل الشجرة Y و فروعها.



الشكل (12) : دمج اهتمامات النظام بأسلوب Y

نولد انطلاقاً من الـBinding Meta–model المبين في الشكل (11) أداة نمذجة رسومية موجهةلمبرمج/مطور تطبيق الويب كما هو موضح بالشكل (13). تسمح هذه الأداة لمطور التطبيق بتحميل نماذج صفحات الموقع المعرفة مسبقاً و تعريف تسلسل التفاعل المرغوب بين صفحات الموقع و مستخدميه. يمكننا اعتبار أداة النمذجة السابقة كأداة لدمج نماذج صفحات الموقع بنماذج التحكم. كما أن النماذج المعرفة باستخدام هذه الأداة تمثل صفحات الموقع و كيفية تفاعلها مع المستخدم. هذه النماذج موافقة لـ Binding Meta–model و لقد طورنا برنامجاً يولد تطبيقات ويب و تطبيقات للهواتف المحمولة من هذه النماذج.



الشكل (13) : أداة رسومية لنمذجة التفاعل بين صفحات الموقع و مستخدميه

إن مولد الكود يعتمد على المفاهيم و العلاقات المعرفة في الـ Binding meta-model لأن نموذج أي موقع نبنيه باستخدام إطار العمل هذا هو موافق لهذا الـ meta-model و بالتالي ستنطبق عليه العمليات المعرفة الخاصة بمولد الكود.كما أننا اختبرنا إطار العمل هذا و مولد الكود لبناء عدد من مواقع الويب. هذه المواقع تعمل دون أي مشكلة.

3.5. مناقشة النتائج :

إحدى أهم ميزات إجرائيات التطوير المقادة بالنماذج هي سهولة التعديل وذلك لأنها مجزأة إلى مراحل مترابطة بالنماذج حيث أن دخل كل مرحلة هو نموذج خرج لمرحلة سابقة أو أكثر.عندما نحتاج للتعديل على مرحلة ما يكفي أن نعدل الـ meta-model الخاص بهذه المرحلة و توليد أداة نمذجة جديدة خاصة بهذه المرحلة. كما أن إضافة مرحلة جديدة بالكامل لا تشكل مشكلة حيث نقوم بتعريف الـ meta-model الخاص بهذه المرحلة و علاقته مع الـ meta-models للمراحل السابقة و اللاحقة (دمج أم تحويل). نستثمر حالياً نتائج إطار العمل السابق لدعم هندسة الويب WebML الموصفة من W3C. WebML يقسم إجرائية التطوير إلى مراحل:

- تعريف مخطط كيانات (معطيات) الموقع و ذلك من خلال تحليل متطلبات الموقع.
- تعريف صفحات الموقع و حقول هذه الصفحات.
- كما يتم الربط بين حقول الصفحات و صفات كيانات الموقع و تحديد اتجاه انتقال المعطيات بين كيانات الموقع و صفحاته.
- تحديد المظهر الرسومي لصفحات الموقع و حقوله.
- تعريف مخطط التصفح بين صفحات الموقع و شروط الانتقال فيما بينها.

حيث نقوم بتعريف Entity M2 لمخطط الكيانات و نولد منه أداة نمذجة تسمح لمحلل النظام بناء نموذج لكيانات الموقع. نعرف أيضاً Page M2 خاص بصفحات الموقع و نولد منه أداة نمذجة تسمح لمصمم صفحات الموقع بتحديد ما هي حقول صفحاته.نحناج لبناء Binding M2 لدمج الـ meta-models السابقين و ذلك لتوليد أداة نمذجة تسمح بتحميل الصفحات و نموذج الكيانات و الربط بين حقول الصفحات و صفات كيانات الموقع.في النهاية نبني Navigation M2 و نولد منه أداة نمذجة تسمح لمصمم الموقع بتحديد صفحة البداية و شروط الانتقال من صفحة لأخرى بالموقع.

6. دراسة حالة 2 "تقييس التحكم على شبكة الانترنيت":

تسمح الـ meta-models بتقييس مجال معين من التطبيقات[3]. سنعرض هنا مسألة أخرى لتوضيح أهمية توحيد المفاهيم في مجال معين و الفوائد التي يمكن جنيها لو استخدمنا الـ meta-models لتقييس مجال تطبيقي ما.

نلاحظ بسهولة أن المبرمج في مجال التحكم يتوجب عليه معرفة قدرات الدارات التي يستخدمها لحل مسألته. بينما مصمم هذه الدارات هو المسؤول عن توصيف قدرات الدارات و ليس من الضروري أن يكون هو المستخدم النهائي لهذه الدارات. لذلك فإننا نقترح فصلاً بين مجال توصيف قدرات الدارات (أوامر و حالات الدارة) و مجال برمجة الدارات (تسلسل الأوامر المطبقة على الدارة).

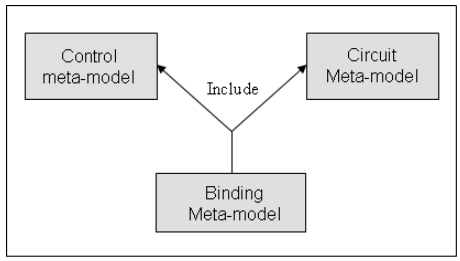
1.6. تصميم إطار عمل لتقييس التحكم

سنعتمد على الـ meta-models لتصميم إطار عمل مجهز بأداتي نمذجة رسومية. الأولى تسمح لمصمم الدارة من تعريف نموذج يصف فيه قدرات داراته و شروط الاستخدام بينما الأداة الثانية تستطيع التعامل مع أكثر من دارة من مصانع مختلفة و تسمح لمبرمج الدارة بوصف تسلسل الأوامر الذي يرغب به ضمن شروط استخدام هذه الدارات.

الشكل (14) يبين الـ meta-models التي تقود إطار عملنا. إن الـ meta-model الأول يوصف مجال الدارات (حالات و آوامر) بينما الـ meta-model الثاني يوصف تعليمات التحكم (if, while, ..) الممكن تطبيقها لوصف تسلسل أوامر الدارات و الـ meta-model الثالث ينسج علاقات بين مفاهيم الـ meta-modes السابقين.

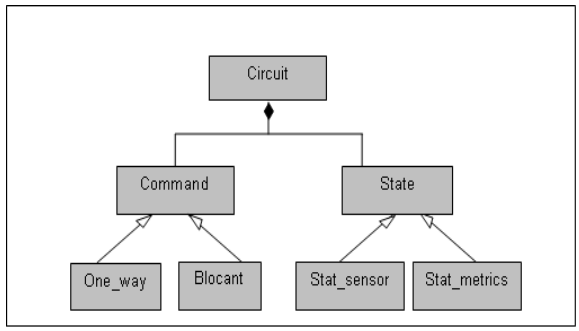
2.6. التحقيق العملي لإطار عمل تقييس التحكم

يمكننا لتحقيق أدوات إطار العمل استخدام الأدوات المشروحة في أعمالنا السابقة [5][4] و التي تسمح لنا بتوليد أدوات نمذجة رسومية لتعريف نماذج موافقة للـ meta-model الذي من أجله و انطلاقاً منه ولدنا أداة النمذجة.



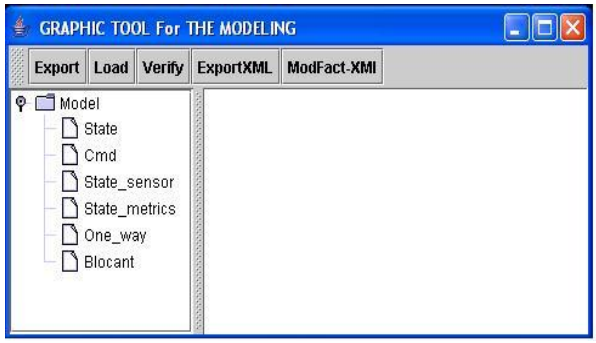
الشكل (14) : الـ meta-models الخاصة بإطار العمل

سنقدم فيما يلي تفاصيل الـ meta-models الثلاث لإطار عملنا و أداتي النمذجة الرسومية المولدة. كما سنركز على كيفية بناء أطر عمل مؤلفة من عدة أدوات و كيف نفصل بين أدوار مستخدمي هذه الأدوات من خلال تجريد مفاهيم المجال و تقسيمها في عدة meta-models. الشكل (15) يبين الـ meta-model الأول و الذي يقدم مفاهيم الحالات و أوامر الدارة التي تسمح لمصمم الدارة بتعريف الأوامر و الحالات الخاصة بدارته.



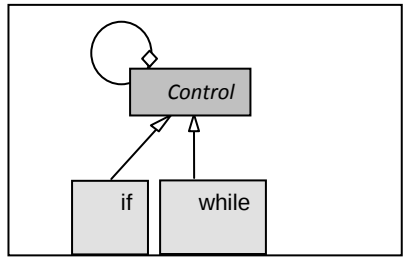
الشكل (15) : meta-model خاص بتوصيف الدارات

نولد آلياً انطلاقاً من الـ meta-model المبين في الشكل (15) و بمعونة أدوات [5][4] الأداة الرسومية المبينة بالشكل (16). تسمح هذه الأداةالرسومية لمصنع الدارات بتوصيف قدرات داراته بشكل قياسي على شكل نموذج لكل دارة. إن هذا النموذج سيتم استخدامه فيما بعد من قبل مبرمج الدارة.



الشكل (16) : أداة لتعريف نماذج توصف قدرات الدارات

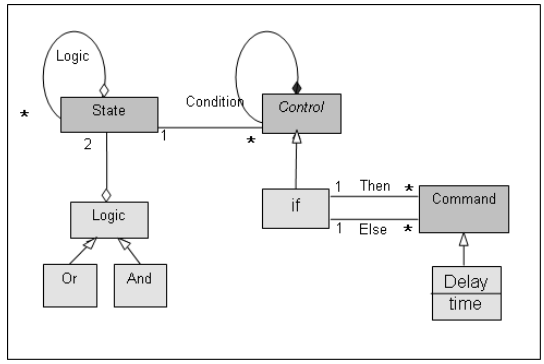
الشكل (17) يبين مفاهيم الـ meta-modelالخاص بالتحكم و التي تسمح لمبرمج الدارة بوصف ترتيب أوامر الدارة و يسمح له بتحديد استجابة دارته للحالات العديدة التي يمكن أن تقع بها. كما تسمح للمبرمج بتكرار العديد من الأوامر لعدة مرات حتى الخروج من حالة ما.لم نقم بتوليد أداة رسومية انطلاقاً من هذا الـ meta-model لأنه لا معنى لتعريف نموذج من أفراد هذه التعليمات بشكل مستقل عن أوامر و حالات الدارة.



الشكل (17) : meta-model خاص بالتحكم

إن نموذج لبرنامج دارة ما هو إلا ربط بين عناصر من مفاهيم الدارة (الأوامر مثلاً) و بين تعليمات تحكم لتنظيم قدرات الدارة لحل مشكلة ما. لذلك نعرفالعلاقات و المفاهيم اللازمة لتوصيف عمليات الربط هذه في الـ Binding meta-model.

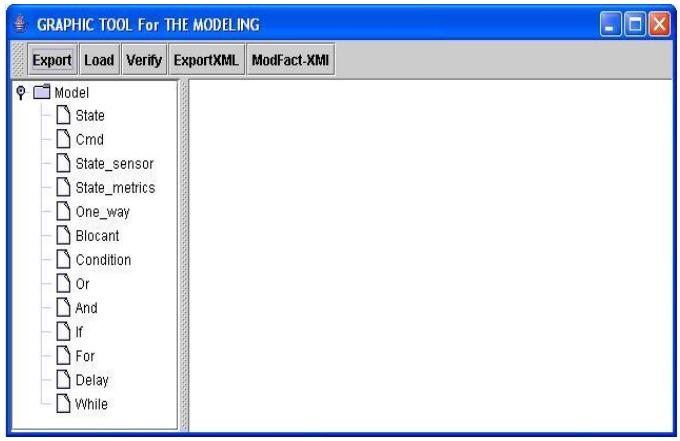
الشكل (18) يبين جزءاً من الـ Binding meta-model حيث يمكن تعريف شروط مركبة من خلال الربط بين حالات الدارة مع معاملات OR و AND. كما أننا عرفنا العلاقتين Then, Else بين المفهوم if من مجال التحكم و المفهوم Command من مجال الدارة مما يسمح لنا باختيار مجموعة من الأوامر لدى تحقق شرط معين و مجموعة أخرى من الأوامر عند عدم تحقق هذا الشرط.



الشكل(18) مفاهيم و علاقات الـ meta-modelBinding

نولد الأداة الرسومية الموضحة بالشكل (19) انطلاقاً من الـ Binding Meta-model كما سبق ووضحنا. هذه الأداة تمكن مبرمج الداراة من تحميل نموذج لدارة ما و من ثم بناء تسلسل خاص لأوامر الدارة المحملة لحل مسألة معينة. لاحظ بأن هذه الأداة

الرسومية تدعم تحويل نماذج لدمج نموذج التحكم مع نموذج الدارة و تسمح للمستخدم بدرجات الحرية اللازمة مما يمكنه من حل العديد من المشاكل.إن النماذج المعرفة باستخدام أداة الشكل (19) تمثل برامج تحكم خاصة بالدارات المحملة. يمكننا إرسال هذه النماذج بصيغة XML إلى حاسب بعيد يرتبط بالدارةذات النموذج المحمل في أداتنا. يوجد على الحاسب البعيد آلة افتراضية ستقرأ النموذج و ترسل الإشارات المناسبة للدارة المرتبطة بالحاسب.إن الآلة الافتراضية هي مفسر XML يكتب استناداً لمفاهيم و علاقات الـ Binding meta-mode و بالتالي فهو قادر على تفسير أي نموذج (برنامج تحكم بدارة) موافق لهذا الـ meta-model و من ثم إرسال الإشارات المناسبة للدارة المادية.



الشكل (19) أداة النمذجة المولدة انطلاقاً من Binding M2

3.6. مناقشة النتائج :

اختبرنا إطار العمل هذا بعد بناء الآلة الافتراضية لتفسير النماذج الموافقة لـ Binding-M2. بنينا بالاختبار الأول نظام للتحكم بتشغيل مكيف حسب درجة حرارة المحيط. بينما بالاختبار الثاني، صممنا نظام لعكس اتجاه حركة مساحات السيارة عند الوصول إلى حساس في كلا الاتجاهين.

كانت الميزة الهامة الأولى هي تمكين أشخاص غير تقنيين لاستخدام أدواتنا للتحكم بتجهيزاتهم و تغيير أسلوب التحكم بهذه التجهيزات دون مساعدة خبير بالتحكم. يتوجب على هؤلاء الأشخاص فقط إدراك الأحداث التي يمكن للدارات التقاطها و ما هي الأعمال التي يمكننا من خلال هذه الدارات إطلاقها. بينما الأداة المبينة بالشكل (19) قدمت أسلوب رسومي بسيط لبرمجة سلوك نظام التحكم بالكامل.

إن الميزة الهامة الثانية هي تطوير الآلة الافتراضية و التي تقدم أسلوب موحد لتفسير نماذج التحكم (المعرفة باستخدام أداة الشكل 19). لم يكن هذا الأمر ليتحقق دون فرض قواعد يجب على مصنعي الدارات التقيد بها و قرن داراتهم المصنعة مع نموذج لتوصيف قدراتها و الذي سيجري تحميله بالأداة (19) و إضافة سياق التحكم عليه من خلال مطور نظام التحكم.

7. مناقشة النتائج والاستنتاجات والتوصيات :

الهندسة المقادة بالنماذج تسمح بتقسيم إجرائية تطوير البرمجيات على عدة مستويات من التجريد مما يساعدنا على الفصل بين الاهتمامات المختلفة و معالجة كل اهتمام بشكل مستقل عن الآخر. تتمايز النماذج ضمن إجرائية تطوير مقادة بالنماذج بدقتهم التقنية أو المهنية من مستوى تجريد لآخر ضمن هذه الإجرائية. تمثل النماذج المفتاح الرئيسي لرأسملة التوصيفات الأساسية للنظام. بينما تسمح تحويلات النماذج بالانتقال بين المستويات المختلفة لإجرائية التطوير و بالتالي رأسملة معرفة التطوير (Know–How) لنوع معين من الأنظمة. استخدمنا مصطلح رأسملة التوصيفات الأساسية للنظام لأن النماذج تساعد مهندسي النظم لحفظ تصاميمهم إما لصيانة هذه التطبيقات لاحقاً أو لإعادة إنتاج هذه التطبيقات باستخدام تقنيات جديدة و دمجها في نظم أكثر تعقيداً. كما أننا استخدمنا مصطلح رأسملة معرفة تطوير النظام لأن الانتقال من مستوى لآخر في إجرائية التطوير يتم عبر تحويلات النماذج و التي هي بحد ذاتها نماذج. إن كيفية الانتقال من نموذج إلى آخر في تحويلات النماذج صريحة و ليست

غامضة و ضمنية كما هي في كود الأنظمة بمعنى آخر ستقوم بهيكلة أفكارك ضمن تحويل للنماذج بدلاً من صياغتها مباشرة في كود برمجي. كما تتميز فعاليات الهندسة المقادة بالنماذج MDEبالتكيف لتغييرات في شروط تحقيق النظام حيث أن الآثار المترتبة عن هذه التغيرات يمكن حصرها بسهولة و بالتالي تعديلها و إعادة استخدام الأجزاء الأخرى الغير مصابة.

نتوسع حالياً أفقياً باستخدام الهندسة المقادة بالنماذج في العديد من المجالات. على سبيل المثال ، نستخدم حالياً الهندسة المقادة بالنماذج لتنفيذ جزء من إجرائية تطوير البرمجيات في الشركة MIS بشكل موافق لمعايير الـ CMMI Capability Maturity model) (Integrated[10]. و ذلك من خلال تعريف Requirement M2 يسمح بتعريف متطلبات المشروع. كما أننا عرفنا ChReq M2 و ذلك كي يسمح بتعريف التغييرات على المتطلبات المعرفة مسبقاً. ثم قمنا بتعريف PlanPrj M2 لتعريف مهام المشروع و كذلك Binding M2 لدمج الـ Meta-models السابقة مما يسمح بتوزيع المتطلبات على مهام المشروع و هكذا.

كما نتوسع أفقياً باستخدام الهندسة المقادة بالنماذج لتطوير تطبيقات الهاتف المحمول بشكل عابر لمنصات التطوير المختلفة مثل Android, IOS إلخ بأسلوب لا يترتب علينا فيه إعادة كتابة نفس التطبيق لعدة منصات تطوير. كما نعمل حالياً لتطوير إجرائيات مقادة بالنماذج لتطوير تطبيقات سحابية متكيفة و الأسلوب الأمني لطيف واسع من مستخدميها دون الحاجة لكتابة التطبيق من أجل كل نوع من المتطلبات الأمنية.

كما أننا نبحث شاقولياً بكيفية تحسين جودة الإجرائيات المبنية [1] [2] بمنهجية الهندسة المقادة بالنماذج من خلال تحديد العيوب [11] التي يمكن ارتكابها عند تعريف الـ meta-models و إيجاد خوارزميات لتصحيح هذه العيوب مما يحسن من جودة و استخدامية الأدوات الرسومية المولدة انطلاقاً من هذه الـ meta-models و يحسن من قابلية الفهم و الصيانة و إعادة الاستخدام لإجرائية التطوير بكامل مراحلها.و كتطبيق مباشر سنحاول تفادي مرحلة الفحص المفروضة على الشركات البرمجية لتجاوز مستويات النضوج في معيار CMMI. خطتنا تكمن في البرهان بأن الـ Meta-models

التي تقود إجرائية التطوير مطابقة تماماً لإرشادات معيار الـ CMMI و تتمتع بجودة تصميم مناسبة من خلال تطبيق خوارزميات كشف العيوب و التصحيح. و تنصب أعمالنا الحالية على استخلاص قواعد و معايير و قياسات خاصة بضمان جودة إجرائيات التطوير المقادة بالنماذج.

9. المراجع

[1] Thorsten Arendt, SieglindeKranz, Florian Mantz, NikolausRegnat, Gabriele Taentzer "Towards Syntactical Model Quality Assurance in Industrial Software Development: Process Definition and Tool Support"

[2] Thorsten Arendta, Florian Mantzb, Gabriele Taentzer, "EMF Refactor: Specification and Application of Model Refactorings within the Eclipse Modeling Framework"

[3] B. KOSAYBA + 4th year students : Raneem SALEH, Rain ALSALEH "Towards Standard "Control Protocol through Internet Using MDE Approach", IEEE ICTTA'08, Damascus - Syria, April 2008

[4] BassemKosayba, "A framework for Model Driven Production of Graphic Modeling Tools", IEEE ICCTA, Damascus, Syria, April 2006

[5] BassemKosayba, Raphael Marvie, Jean-Mark Geib, "Model Driven Production of Domain-Specific Modeling Tools", In 4th OOPSLA Workshop on Domain-Specific Modeling (DSM'04), Vancouver, Canada, October 2004.

[6] S. Kent, "Model Driven Engineering", Third International Conference on Integrated Formal Methods", 2002.

[7] Jean BEZIVIN. "On the Unification Power ofModels. Software and System Modeling", 4(2) :171–188, 2005. http://www.sciences.univ-nantes.fr/lina/atl/www/papers/OnTheUnificationPowerOfModels.pdf.

[8] Marten J. VAN SINDEREN Giancarlo GUIZZARDI, Luis Ferreira PIRES. "On the role of Domain Ontologies in the design of Domain-Specific Visual Modeling Languages". In The Second Workshop on Domain-Specific Visual Languages at OOPSLA, Seattle, WA, USA, November 2002.

[9] Jean BEZIVIN. “From Object Composition to Model Transformation with the MDA”. In TOOLS’USA, Volume IEEE TOOLS-39, Santa Barbara, August 2001

[10] Mary Beth Chrissis, Mike Konrad, Sandy Shrum, "CMMI®: Guidelines for Process Integration and Product Improvement", 2003, ISBN: 0-321-15496-7

[11] 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.