الجمهورية العربية السورية
جامعة البعث
كلية الهندسة المعلوماتية
قسم هندسة البرمجيات

# أتمتة تطوير البرمجيات

# Automated Software Production


الدفعة الخامسة
2006 - 2007

إعداد :

إياد رياض حلواني

عبد المالك سمير الساعاتي

محمد أحمد الرفاعي

بإشراف الدكتور : عامر البوشي

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

﴿ وَقُلِ اعْمَلُوا فَسَيَرَى اللَّهُ عَمَلَكُمْ وَرَسُولُهُ وَالْمُؤْمِنُونَ وَسَتُرَدُّونَ

إِلَى عَالِمِ الْغَيْبِ وَالشَّهَادَةِ فَيُنَبِّئُكُم بِمَا كُنتُمْ تَعْمَلُونَ ﴾

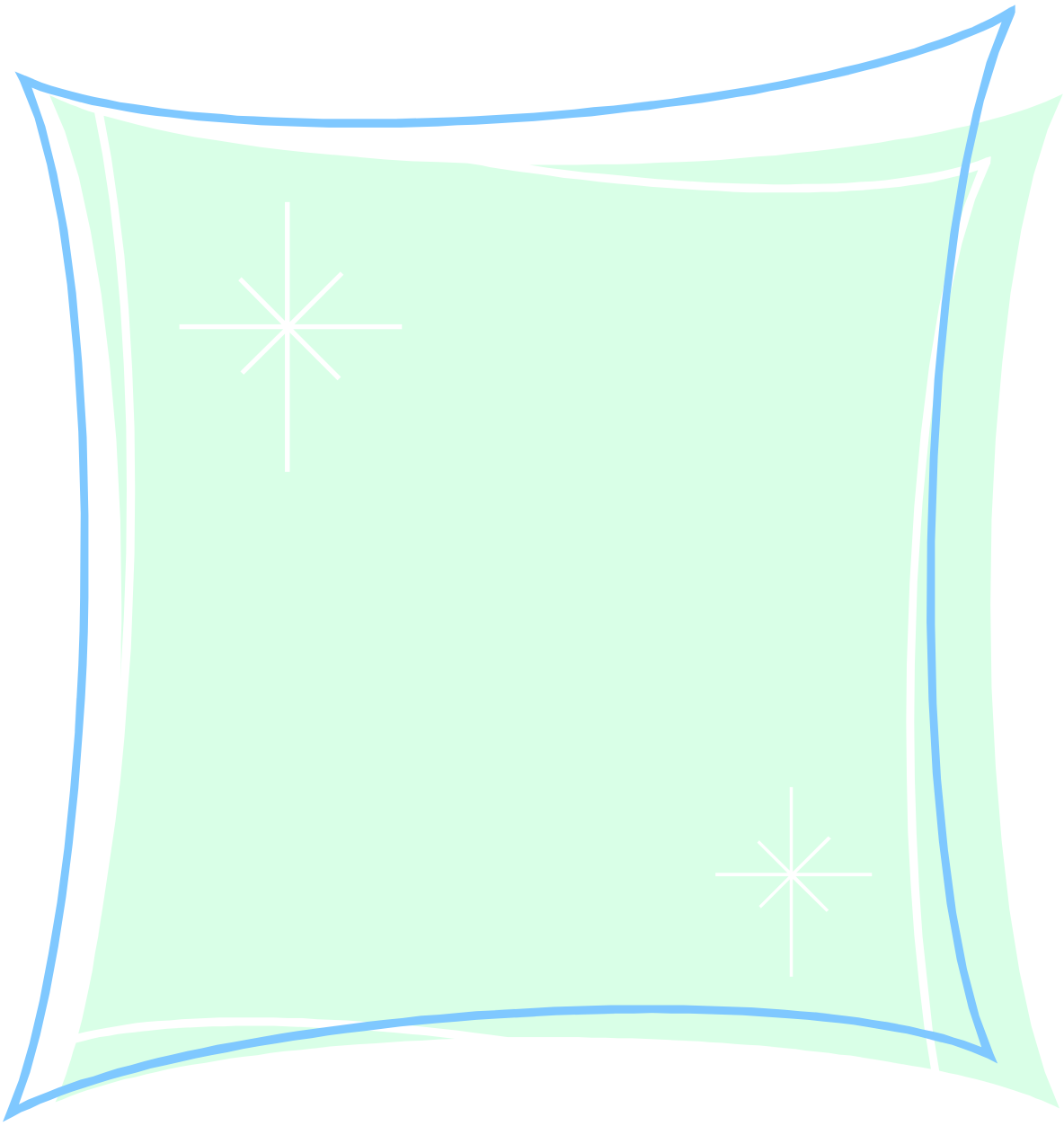
صدق الله العظيم

"إن الملائكة تضع أجنحتها لطالب العلم رضىً بما يطلب" .

معلم البشرية سيدنا محمد صلى الله عليه وسلم

"من علم وعمل وعلّم، فذاك يدعى عظيماً في ملكوت السماء" .

سيدنا عيسى بن مريم عليه الصلاة السلام



# الإهداء:

الحمد لله رب العالمين ، و أفضل الصلاة و أتم التسليم على سيدنا محمد و على آله و أصحابه أجمعين .

هكذا تمضي الأيام مسرعة، تاركة العمر يرفرف في بساتين الذكريات، و نقف لنسترجع ماض عشناه مع أساتذتنا الأجلاء الذين جادوا علينا بنور العلم و المعرفة، فلهم جزيل الثناء و الشكر .

ولن يغيب عن ناظرنا أولئك الذين شاركونا لحظات الترح و الفرح ووقفوا معنا يداً بيد لتحقيق ما نصبوا إليه، لذا نسأل الله العلي القدير أن يمدهم بموفور الصحة و العافية .

اللهم علمنا ما ينفعنا و انفعنا بما علمتنا، إنك على ما تشاء قدير .

**و الله من وراء القصد**

# لمســـة وفـــــاء

إلى الأســاتذة الأجـلاء، والـسادة الأفاضـل والجنـود المجهـولين، الـذين سـاهموا ـفي خـروج هـذه الأطروحـة للنور، ولـم يبخلوا بعلـم ولا بجهد ولا بـوقت، ونخص بالذكـر:

**الدكتور المهندس عامر البوشي**

الذي تفضل بالإشراف على هذا العمل المتواضع.

وإلى جميـع العـاملين ـفي كليـة الهندسـة المعلوماتيـة، جزاكـم الله عنا كل خير.

محمد – إيـاد – عبد المالك

# فهرس

الفصل صفر

# نقطـــة الانطـــــلاق:

## المقدمة

توجد طريقتان لبناء تصميم برمجي: الأولى هي بجعله بسيطاً بحيث يبدو من الواضح عدم وجود علل، والثانية هي بجعله معقداً جداً بحيث لا يكون من الواضح وجود علل!!.

-س.آ.ر هاور-

في السنوات الأخيرة، أصبح المنهج غرضي التوجه (OOT Object Oriented Technology) شائعاً لبناء أي نظام برمجي.

يقوم فريق العمل عادة باستخدام عدد من المنهجيات و الأدوات المساعدة لأتمتة بعض المهام خلال مراحل التطوير. لكن عملية التحليل ظلت من المهام الصعبة و الحساسة نظراً لأن المراحل اللاحقة سوف تعتمد عليها، و نظراً لأن معظم مدخلات هذه المرحلة تقع تحت ضوء اللغات الطبيعية.

إن الأدوات المساعدة (CASE Tools) (هندسة البرمجيات بمعونة الحاسوب) توفر مساعدة كبيرة في توثيق نتائج التحليل و التصميم، ويمكن أن تكشف عن التناقض بين هاتين المرحلتين إن وُجد، بيد أن هذه الأدوات لا تساهم في المرحلة الأولى و الصعبة من عملية التحليل و التي تشمل اكتشاف الصفوف و الصفات والعلاقات و التي تعد الأساس في حل المشكلة.

و للمساهمة في هذا الجزء من عملية التحليل لابد من إيجاد أداة تكون قادرة على التعامل مع لغات البشر أي لغة مهندسي البرمجيات و بالتالي تساعد على فهم مجال المشكلة التي نريد حلها.

إن أدوات تحليل اللغات الطبيعية (NLP) تعتبر النهج الواعد الذي يمكن أن يساعد مهندسي البرمجيات في تحليل وثائق المتطلبات الخاصة بتطوير البرمجيات.

ولما كانت اللغة العربية هي اللغة الأم لأكثر من ثلاثمئة مليون إنسان، وهي من أوائل اللغات المحكية على مستوى العالم، كان لابد من إيجاد أداة لتحليل وثائق المتطلبات المكتوبة باللغة العربية، لتساهم في تطوير إنتاج البرمجيات وزيادة كفاءتها وفعاليتها، لتصبح على مستوى المنافسة مع البرمجيات العالمية.

سوف نقوم في هذه الأطروحة بتقديم بحث حول بناء أداة تقوم بتحليل الوثائق البرمجية المكتوبة باللغة العربية وتكوين مخططات الـ (UML) للنظام المدروس ومن ثم توليد البنية البرمجية الهيكلية للنظام، وصولاً إلى نظام برمجي متكامل.

إن هذا البحث مقسم على النحو التالي:

الفصل الأول: ويركز على التعريف بالأدوات المساعدة في هندسة البرمجيات.

الفصل الثاني: ويعرف بتقنية معالجة اللغة العربية، والتي هي المرحلة الأساس في تحليل الوثائق العربية.

الفصل الثالث: ويعرف باستخدام معالجة اللغات الطبيعية في هندسة البرمجيات.

الفصل الرابع: يناقش كيفية توليد الشيفرة المصدر (Code Generator).

الفصل الخامس: تقديم النظام المقترح.

الفصل السادس: يناقش التطلعات المستقبلية.

الفصل الأول

# هندسة البرمجيات بمعونة الحاسب

# COMPUTER AIDED SOFTWARE ENGINEERING

محتويات الفصل:

سنبحث في هذا الفصل:

- مراحل تطوير البرمجية.
- الأدوات المساعدة في هندسة البرمجيات (CASE Tools).
- تصنيف الأدوات المساعدة، أهدافها ومساوئها.
- الحاجة إلى أداة تدعم اللغة العربية.

من وجهة نظر المبرمج، المستخدم هو جهـــاز محيطـــي يكتب عندما ينفذ طلب القراءة.

-بيتر ويليامز-

**مقدمة:**

إن البرمجيات التي تعتمد منهجية التحليل والتصميم غرضي التوجه الـــ (OOA/D) (Object Oriented Analysis/Design) تعتمد في آلية بنائها على طريقة فصل الطبقات حيث يتم تجميع الصفوف المترابطة وظيفياً في طبقات منفصلة و يتم التفاعل بين هذه الطبقات من خلال إنشاء غرض من الصف الموجود في الطبقة الأدنى داخل صف من الطبقة الأعلى.

## 1-1- مراحل تطوير البرمجية:

و بشكل عام فإن الطبقات الأساسية في أغلب التطبيقات غرضية التوجه هي ثلاث طبقات :

1 – طبقة الوصول إلى مصدر البيانات (DB Layer).
2- طبقة منطق العمل (Application Logic layer).
3- طبقة التفاعل مع المستخدم (GUI Layer) .

يمكن أن يتم إضافة طبقات بحسب الحاجة أو إزالة طبقات وتبعاً لذلك تـــم تـــصنيف النماذج إلى نماذج (3Tier) وهي النماذج التي تحوي الطبقات الثلاث الأساسية كمـــا يمكـــن توسيع هذا النموذج ليشمل أربع أو خمس طبقات (مثل إضافة طبقة حماية Security) فيسمى النموذج في هذه الحالة بنموذج (NTier).

تتألف عملية تطوير أو صناعة البرمجية من مراحل محددة تنفـــذ تباعـــاً و يمكـــن تصنيف هذه المراحل كالتالي:

- **جمع المتطلبات:**

ويتم في هذه المرحلة جمع المعلومات حول المشكلة المراد تـــصنيع البرمجيـــة لهـــا وتوثيق هذه المعلومات. و يتم كتابة هذه المتطلبات على شكل وثيقة تدعى بوثيقة المتطلبات و تشكل الإجابة عن السؤالين التاليين الحجر الأساس في هذه المرحلة:

1- من هم مستخدمو النظام.
2- ما هي الوظائف التي على كل مستخدم تأديتها داخل النظام.

- **التحليل (Analysis):**

هنا يتم تحليل الوثيقة التي تم الحصول عليها من المرحلة السابقة من خلال[1]:

1- تعيين حالات الاستخدام Use Cases.
2- استخراج صفوف المجال الرئيسية.
3- تحديد العلاقات بين الصفوف و هرميات الوراثة بينها .
4- تعيين الخصائص (Attributes) والعمليات (Operations) المعرفة في كل صف.

و بذلك نرى أنه تم الحصول على مخطط مبدئي لحل هذه المشكلة يدعى بمخطط الصفوف (Class Diagrams) إلا أننا نلاحظ أن هذا المخططات التي حصلنا عليها لا يمكن في حال نقلها إلى تحقيقات برمجية أن تشكل برمجية قابلة للعمل إذ أن هذه المخططات لا تحوي إلا على الصفوف الرئيسية لحل المشكلة (Application Logic) وبالتالي تأتي مهمة المرحلة التالية ألا وهي التصميم (Design).

- **التصميم (Design):**

يتم في هذه المرحلة إضافة الصفوف التي نحتاجها -للوصول إلى برمجية متكاملة قابلة للتحقيق- إلى مخططات الصفوف الأساسية التي تم الحصول عليها في مرحلة التحليل و يمكن أن تكون هذه الصفوف عبارة عن صفوف الواجهات (GUI Layer) أو صفوف التعامل مع قاعدة البيانات (DB Layer).

- **التحقيق (Implementation):**

و يتم هنا الانتقال من المخططات التي وصلنا إليها في المرحلة السابقة إلى شيفرة محققة بلغة معينة وهنا يجب أن يجب استخدام لغة لكتابة التحقيق البرمجي تكون داعمة للنموذج غرضي التوجه.

تجدر الإشارة هنا وفي عجالة إلى أن جميع المراحل عدا مرحلة التحقيق عامة أي لا ترتبط بلغة معينة إلا أنه يوجد بعض الحالات التي يجب فيها مسبقاً معرفة اللغة التي سيتم كتابة التحقيق بها و ذلك بسبب خصائص بعض اللغات (مثل Java التي لا تقبل الوراثة من أكثر من صف).

[1] النمذجة المرئية في التحليل والتصميم غرضي التوجه. أ.بلال الإسماعيل.

يمكن أن يتم إضافة مراحل أخرى مثل مرحلة التأكد من خلو البرمجية من الأخطـــاء المنطقية (Testing) و مرحلة فحص أداء البرمجية.

يمكن أن يتم تنفيذ هذه الخطوات بدورة واحدة أو بعدة دورات أو التركيز على أنشطة محددة أساسية (Higher Use Cases) في البرمجية ليتم تنفيذ هذه الأنشطة ثم يتم اســتكمال باقي الأنشطة الثانوية (Lower Use Cases) في مراحل تالية وبالتالي فإن عملية صــناعة البرمجية تتم تبعاً للإستراتيجية المعتمدة في عملية الصناعة هذه.

## 1-2- الحاجة إلى الأدوات المساعدة في عملية تطوير البرمجية :

في العرض السابق لخطوات صناعة البرمجية يمكن أن نلمس التعقيد والرتابة في هذه المراحل المتلاحقة ومن هنا فإن الحاجة ماسة إلى إيجاد أدوات تسهل هذه العملية.

في ضوء هذه المعطيات بدا جلياً أن المنهجية المعتمدة للقيام بتطوير البرمجيات بحاجة ملحة إلى أتمتة، خاصةً في ظل الخطوات الروتينية التكرارية، و بالتالي كان لابد من القيــام بأتمتة توليد شيفرة طبقة الصفوف (Code Generation) و أتمتة إنشاء هيكل قاعدة البيانات (DDL Script Generation) و توليــد الــصفوف الخاصــة بالواجهــات المرئيــة (GUI Generation).

وهنا بدأت الأدوات المساعدة في عملية صناعة البرمجيات تتوالى في الظهور.

## 1-3- تعريف الأدوات المساعدة في هندسة البرمجيات (Computer Aided Software Engineering CASE):

و تعرف بـــ CASE tools، وهي مجموعة من البرامج و الأدوات المستخدمة خلال أي مرحلة من مراحل حياة تطوير النظام (Software Life Cycle)، والتــي تــساعد فــي تطوير وصيانة البرمجيات، و تشمل الأدوات المــساعدة خــلال مرحلــة التحليــل (Data Modeling Tools) و التـــــصميم (UML Tools) وتوليـــــد الـــــشيفرة (Code Generation tools) أدوات الاختبار (Testing tools) ... .

**أصناف الأدوات المساعدة:[2]**

- **Upper Case :**

وهي الأدوات التي تركز على مرحلة التحليل و التصميم خلال دورة حياة البرمجية، مثال :(diagramming tools, report and form generators, analysis tools)

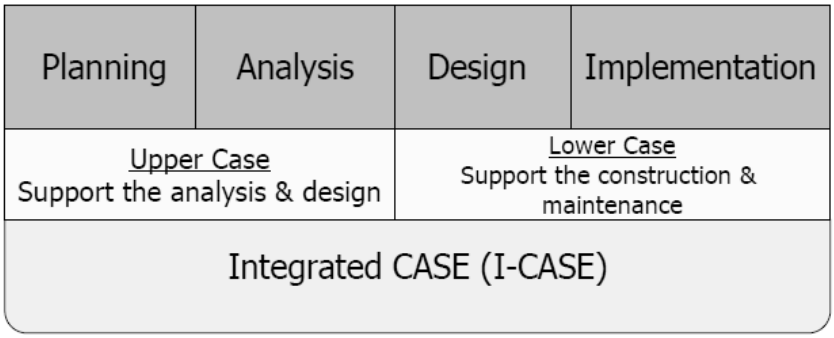
[2] http://en.wikipedia.org/wiki/Computer/aided_software_engineering

- **Lower Case :**

وهي عموما تشير إلى مولدات الكود و أدوات الاختبار مثل: (database schema generation, program generation, testing)

- **I-CASE :**

و هي الأدوات التي تدعم جميع المراحل السابقة، أي هي الأدوات التي ترافق المطور من مرحلة التحليل انتهاء بمرحلة الاختبار.




**الشكل (1-1) تصنيف الأدوات المساعدة**

### 1-3-1- بعض الأدوات المساعدة القياسية:

- أدوات رسم مخططات التحليل والتصميم (Analysis and design diagrams).
- أدوات توليد الشيفرة (Source code generation).
- مستودع إدارة البيانات (Data management repository).

## 1-4- أهداف الأداوت المساعدة في تطوير البرمجيات (CASE Tool: Goals)[3]

- تحسين جودة البرمجية (Software Quality).
- زيادة سرعة عملية التصنيع (Development speed).
- معيارية طريقة تصنيع البرمجية (Standardization).
- تسهيل صيانة البرمجية بسبب المعيارية في التصنيع (Maintenance).
- تسهيل إعادة تصنيع البرمجية في حال تغيير بعض المتطلبات (Reverse Engineering).
- الاعتماد على نماذج معيارية لحل مشاكل معروفة(Patterns).

[3] Modern Systems Analysis and Design (Third Edition)

- دعم إعادة الاستخدام (Reusability).
- تسهيل إنتاج توثيق البرمجية (Documentation).
- تسهيل إدارة المشروع (Management).
- تسهيل و تسريع عملية الفحص (Testing).

## 1-5- الصعوبات والمساوئ (CASE Tools : Difficulties)

- الكلفة العالية للأدوات المساعدة (Cost).
- التقييد في عملية التصنيع حيث تعتمد استراتيجية محددة لا تعطي حرية كبيرة للمطور.
- تحتاج إلى فترة تدريب ليتمكن المطور من التآلف معها (Training Cost).

بعد هذا العرض السريع للـ (CASE Tools) نلاحظ أن هذه الأدوات تساعد في جميع المراحل المذكورة سابقاً في عملية تطوير البرمجية ما عدا الخطوة الأولى و التي تمثل الانتقال من وثيقة المتطلبات إلى المخطط التصوري الأولي لحل المشكلة (Conceptual Model) وهنا بدأ التفكير في إنجاز هذه الخطوة التي يمثل دخلها وثيقة مكتوبة بلغة محكية و يكون الخرج لهذه الخطوة هو مخطط تصوري إلا أن هذه الأداة يجب أن تكتب لكل لغة من لغات العالم لأن وثيقة الدخل المكتوبة في كل لغة تختلف عن وثيقة الدخل المكتوبة بلغة أخرى تبعاً لاختلاف طبيعة اللغات و مدى تعقيدها.

لقد ظهرت بعض المحاولات التي تغطي هذه المرحلة سواءً للغة الإنكليزية أو الفرنسية أو حتى الألمانية، ولكن هذه المحاولات ليست إلا دراسات أكاديمية مازالت في مراحلها البدائية، ولم تظهر أداة متكاملة تبدأ من عملية تحليل المتطلبات وتنتهي بتطبيق قابل للعمل.

وبما أن القطاع المعلوماتي العربي بدأ في التطور، كان لابد من وجود أداة مساعدة لعمل مهندسي البرمجيات، تدعم اللغة العربية وتساعد في الوصول إلى تطبيقات ترقى إلى مستوى التطبيقات المطورة على مستوى العالم من حيث الأداء والجودة وسرعة التنفيذ.

إن الأداة (Arabic CASE Tool) التي قمنا بتطويرها تندرج تحت الأدوات المتكاملة (Integrated Case Tools) فهي تتألف بشكل مجمل من ثلاثة أقسام رئيسة:

1- محلل النص العربي: الذي يقوم بتحليل وثيقة المتطلبات وينتج عن هذا التحليل قائمة بالصفوف المقترحة والعلاقات فيما بينها والتي ستشكل البنية الأساسية للتطبيق.

2- أداة الرسم (UML Diagram): وتقوم بعرض مخطط الصفوف الناتج عن مرحلة تحليل النص ويتاح هنا للمطور تعديل ما يريد بإضافة صفوف أو حذفها أو تعديل العلاقات بينها فهي تشكل بيئة متكاملة لمخطط الصفوف.

3- مولد الشيفرة: وهو المرحلة النهائية في عملية إنتاج البرمجية وفيها يتم توليد التحقيق (implement) وتوليد الشيفرة النهائية التي تمثل التطبيق وهذه الشيفرة مكتوبة بلغة (java).

من أهم ميزات مولد الشيفرة هذا هو أنه يمكن المطور من توليد نوعين من الشيفرة إما شيفرة لتطبيق محلي يعمل على جهاز واحد (locally) أو شيفرة لتطبيق موزع على عدة أجهزة (Distributed) وهذا ما سيتم تفصيله في القسم العملي بإذن الله.

الفصل الثاني

# تحليل اللغات الطبيعية

## تحليل اللغة العربية

# NATURAL LANGUAGE PROCESSING (NLP)

## ARABIC LANGUAGE PROCESSING

محتويات الفصل:

ستتعرف في هذا الفصل على:

- طبيعة الكلمات في اللغة العربية.
- ماذا نعني بـــ (تحليل اللغة الطبيعية).
- كيفية تمثيل الكلمة بشكل يشمل جميع الكلمات العربية.
- بنية محلل النص والخوارزمية المتبعة.

اللغة العربية من أكثر لغات الأرض امتيازاً – وهذا الامتياز من وجهين: الأول من حيث ثروة معجمها، والثاني من حيث استيعاب آدابها.

–ادوراد فارديك–

إنه لا يعقل أن تحل اللغة الفرنسية، أو الانجليزية محل اللغة العربية. وإن شعباً له آداب غنية، منوعة، كالآداب العربية، ولغة مرنة، ذات مادة لا تكاد تفنى، لا يخون ماضيه، ولا ينبذ إرثاً ورثه، بعد قرون طويلة عن آبائه وأجداده.

–ريتشارد كريتفيل–

اللغة العربية: إنها لغة المستقبل.

–الكاتب جول فيرن–*

## مقدمة:

ما يزال تعامل الحاسب مع اللغة العربية قاصراً مقارنة باللغات الأخرى خصوصاً الإنكليزية سواء على مستوى النص من حيث الشكل والتخزين ومشاكل الترميز والمعالجة أو من ناحية التعامل مع المعنى الذي يتضمنه النص.

وبما أن اللغة العربية بخلاف اللغات الأوربية لغة اشتقاقية أي أن هناك نظاماً صارماً يربط بناء الكلمات العربية بالعلاقات المعنوية والمنطقية لهذه الكلمات، فإن هذا يبرر السعي إلى آلية مختلفة تستفيد من بنية اللغة العربية الاشتقاقية وسد الحاجات الملحة إلى معالجة نقص التشكيل والمعنى للغة العربية.

## 2–1– طبيعة الكلمات في اللغة العربية:

الكلمة في اللغة العربية تقسم إلى ثلاثة أقسام رئيسة: اسم، فعل ، حرف .. كما قال ابن مالك في ألفيته الشهيرة التي وضع فيها قواعد اللغة العربية ونظمها في ألف بيت وابتدأها:

كلامُنا لفظٌ مفيدٌ كَاسْتَقِم، و اسمٌ، وفعلٌ، ثمّ حرفٌ – الكَلِمْ

هذه الكلمة أيضاً ينظر إليها على أنها إما جذر أو كلمة مشتقة ، وأيضاً ينظر إليها من ناحية موقعها من الجملة .. فالفعل ماض و مضارع وأمر، والاسم فاعل أو مفعول أو مصدر أو اسم مجرور.. إلى آخره.

* موقع صوت العربية: http://www.alarabiyah.ws/showpost.php?postid=28

نحصل على الكلمات المشتقة من الجذور بإضافة مجموعة من الأحرف التي تسمى أحرف الزيادة –المجموعة في كلمة "سألتمونيها"– مع مراعاة وزن الكلمة الذي يجب أن يكون مطابقاً لأحد الأوزان المعروفة في اللغة.

مثلاً كلمة "كاتب" مشتقة من الجذر "كتب" بإضافة حرف " ا " الألف، وهي تحمل معنى الشخص الذي قام بفعل الكتابة، أي أنها مطابقة للوزن "فاعل".

إذاً اللغة العربية لها قواعد محددة وواضحة لاشتقاق مجموعة من الكلمات من جذر واحد، وجميع الكلمات المشتقة من جميع الجذور تتطابق مع مجموعة محددة من الأوزان.

## 2-2- المحلل النصي:

إن الغرض الأساسي من المحلل النصي الذي نحتاجه في دراستنا وفي تطبيقنا هو الحصول على أبسط كلمة لها معنى من كلمة مزيدة.

مثلاً كلمة "المهندسون" هي كلمة مركبة، نحصل منها على أبسط كلمة هي كلمة "مهندس"، والأحرف الباقية "ال" للتعريف ، و "ون" للجمع هي أحرف زائدة.

إذاً الكلمات انطلاقاً من هذا التقسيم هي إما كلمات بسيطة، أو كلمات مركبة والتي هي في النهاية كلمات بسيطة ألحقت بها أحرف زائدة.

### 2-2-1- تمثيل الكلمة العربية:

نستطيع الآن تمثيل أي كلمة في اللغة العربية على الشكل التالي:

[بادئة_1 | بادئة_2] + كلمة بسيطة + [لاحقة_1] + [لاحقة_2] + [لاحقة_3] *

حيث أنّ القوسين يدلان –كما نعلم– على حالتين مقبولتين: إما وجود ما بينهما أو عدم وجوده، والرمز "|" يدل على الاختيار أي إما بادئة_1 أو بادئة_2، وليست الاثنتان معاً.

إذاً أصبح عملنا الآن هو الحصول على الكلمة البسيطة بحذف الأحرف الزائدة من الكلمة المركبة اعتماداً على التمثيل السابق، لكن لسوء الحظ الأمر ليس بهذه السهولة!

* سنقوم بوضع جدول لاحقاً يبين ما نعنيه بكل رمز من الرموز بين الأقواس ونوضح سبب تقسيمها.

**بعض الصعوبات التقنية لهذا التصور:**

قد يبدو الأمر على أنه مجرد مقارنة بداية الكلمة مع مجموعة من الأحرف التي تمثل الأحرف الزائدة التي يمكن أن تأتي في بداية الكلمة وحذفها، ومقارنة نهايتها كذلك مع مجموعة من الأحرف الزائدة التي يمكن أن تأتي في نهاية الكلمة وحذفها، والحصول على الكلمة البسيطة.

أمثلة على هذه الصعوبات:

1. الأحرف المشددة .. مثل كلمة "هممت" الكلمة البسيطة منها هي "همّ" .. لكن بحذف الحرف اللاحق المزيد "ت" نحصل على الكلمة "همم".

2. قد تكون الكلمة البسيطة قد تغيرت بعد إضافة الأحرف الزائدة لها .. بالتالي حذف هذه الأحرف يستلزم تعديل الكلمة الناتجة للحصول على الكلمة البسيطة. مثلاً الفعل "يقول" عند حذف حرف الياء من أولها نحصل على الفعل "قول" بينما الكلمة البسيطة المستنتجة منها يجب أن تكون "قال" فيجب إبدال الواو ألفاً.

3. بعض الكلمات تحتاج لإضافة حرف على الكلمة الناتجة من عملية الحذف . مثل كلمة "محامون" يعد حذف حرفي "ون" من آخرها نحصل على كلمة "محام" يجب إضافة حرف الياء إلى نهايتها لنحصل على الكلمة البسيطة الصحيحة "محامي".

## 2-3- منهجية العمل:

إن بعض أنظمة معالجة اللغات الطبيعية ( Natural Language Processing) –وهنا نتحدث عن التي تهتم باللغة الإنكليزية بسبب غياب الدراسات حول اللغة العربية– تعتمد في عملها على تخزين جميع كلمات اللغة في قاموس (lexicon).

إن هذا الحل يعتبر حلاً غير عملي أبداً ، فالكثير من الكلمات تشتق من أصلها البسيط بإضافة حرف "s" مثلاً عند الحديث عن الكلمات التي تدل على الجمع في اللغة الإنكليزية ويقابلها حرفا "ون" في اللغة العربية مثلاً ، فهنا سيكون من غير العملي أن نقوم بتخزين آلاف الكلمات بدون فائدة.

وبعضها الآخر يعتمد على إيجاد الجذر الأولي للكلمة فمثلاً في الكلمات العربية تعود إلى الفعل الثلاثي أو الرباعي أو الخماسي الذي تتشكل منه الكلمة، وهذه الجذور تكون مخزنة

ضمن قاعدة بيانات، فهي بهذه الحالة اختصرت عدد الكلمات المخزنة لكن بالتالي هذا سيؤدي إلى تعقيد كبير في خوارزمية استخلاص الجذر.

والخوارزميات الباقية استخدمت ما يسمى بالتجذير الخفيف (Light Stemming) وهو فعال جداً، يعتمد على معلومات إحصائية حول أكثر الأحرف المزيدة تكراراً ، تقوم الخوارزمية على حذف هذه الأحرف من كلمات النص، وتصل إلى الكلمات البسيطة[4].

إن هذه الخوارزميات تعتبر أسرع الخوارزميات في الحصول على الكلمة الأصلية، حيث أنها لا تعتمد إطلاقاً على أي قاموس، لكن من الواضح أنها بهذه الحالة ستفتقد إلى الدقة، وفي دراستنا هذه نحتاج إلى الدقة للوصول إلى نسبة مقبولة من الجودة في التطبيق.

اعتمدنا في دراستنا على ما يسمى بالـ (corpus)، هذا الـ(corpus) بسيط، خاص بالكلمات العربية، وهو عبارة عن قاعدة بيانات تحوي الكلمات البسيطة فقط ، أي أنه يهتم فقط بالجزء الأساسي من الكلام، أي الكلمات التي لا يمكن اشتقاقها، مثلاً توجد فيه جميع الكلمات التي تمثل جمع التكسير بينما لا يوجد فيه أي جمع مذكر أو مؤنث سالم، يحوي حوالي مئة وستين ألف كلمة بسيطة.

في هذه الحالة سيكون العمل أكثر فعالية عند البحث في هذا القاموس، حيث تم التخلص من آلاف الكلمات التي لا فائدة من تخزينها.

كذلك ستكون الدقة جيدة جداً فلن نصل إلى كلمات لا أصل لها في اللغة العربية، بسبب التجذير غير المؤصّل.

إذاً نستطيع الآن تقديم تفصيل متكامل لخوارزمية تحليل النص العربي المستخدمة في الوصول للكلمات البسيطة.

## 2-4- خوارزمية تحليل النص:

### 2-4-1- أحرف الزيادة في الكلمات العربية:

كنا قد ذكرنا سابقاً أن الكلمة البسيطة تنتج من الكلمة المركبة بحذف أحرف الزيادة، وذكرنا أن الأحرف الزائدة مجموعة في كلمة "سألتمونيها".

[4] - Light Stemming and Co-occurrence Analysis: Larkey, Ballesteros, Connel , 2003

بتفصيل أكثر، تقسم الأحرف الزائدة إلى أحرف زائدة تلحق في بداية الكلمة (بوادئ) ، وأحرف زائدة تلحق بنهاية الكلمة (لواحق).

**بوادئ الكلمات:**

البوادئ تقسم إلى قسمين : يوادئ تظهر في بداية الأسماء ، وأخرى تظهر في بداية الأفعال.

غالباً هذه البوادئ تفيد في تغير معنى الكلمة، فمثلاً عند إضافتها للأفعال تغير من زمن الفعل، مثل "يفعل" و "سيفعل" و "فعل".

الجدول التالي يوضح هذه البوادئ بقسميها:

**(جدول2-1 البوادئ المضافة للكلمات العربية)**

| بوادئ الأسماء | بوادئ الأفعال |
|---|---|
| فال | فسأ – سأ |
| فـ | فسن – سن |
| كـ | فست – ست |
| لل | فسي – سي |
| ال | ت – ي – ن – أ |
| بال | فلت – فلي – فلن |
| ب | لت – لي – لن |
| كال | |
| ل | |

**لواحق الكلمات:**

أيضاً اللواحق تقسم إلى قسمين: لواحق للأفعال، ولواحق للأسماء.

لكن هنا لدينا بعض اللواحق المشتركة التي يمكن أن تضاف إلى الأسماء، وأيضاً يمكن أن تضاف إلى الأفعال.

اللواحق المضافة إلى الأسماء بشكل أساسي تفيد في تحديد ملكية الاسم تعداده، وجنس المتكلم، مثل كتابه – كتابهم – كتابها.

أما اللواحق المضافة إلى الأفعال توضح الفاعل والمفعول به بشكل أساسي. مثل كتبه (الهاء تدل على المفعول به) ، كتبته (التاء تدل على الفاعل، الهاء على المفعول به).

الجدول التالي يوضح جميع اللواحق.

**(جدول 2–2 اللواحق المضافة للكلمات العربية)**

| لواحق الأفعال | لواحق الأسماء | اللواحق المشتركة بين الأفعال والأسماء |
|---|---|---|
| تما | ة | ا |
| تم | ات | نا |
| تن | | ان |
| تا | | ين – ون |
| وا | | ه – ها |
| و | | هن – هم – هما |
| ني | | كن – كم – كما |

### 2-4-2- تمثيل الخوارزمية:

اخترنا تمثيل الخوارزمية كأوتومات، يبين هذا الأتومات البنية الأساسية للخوارزمية على الشكل التالي:

1. مجموعة من الحالات تمثل الكلمة الناتجة بعد عملية معينة عليها، حيث أن الدائرة تمثل حالة الكلمة، والأقواس تمثل العمليات عليها.

2. قد ينتج لدينا أكثر من حل في نهاية الأمر بعد المرور على جميع الحالات الممكنة – أي الحصول على أكثر من كلمة بسيطة، أو الحصول على نفس الكلمة باجتياز أكثر من طريق-.

### 2-4-3- تفصيل خوارزمية تحليل النص:

إن برنامج محلل النص يمر على الأتومات باستخدام أسلوب العمق أولاً (depth first)، ثم يقوم بالتراجع ليحصل على جميع الحلول الممكنة، هذه هي أفضل الطرق التي وجدناها تناسب النص العربي.

إن ترتيب الأقواس والشروط الموضوعة عليها في الأتومات وضعت بترتيب يؤدي إلى أن الأقواس التي يمر عليها أولاً هي التي تعطي أفضل الحلول.

كما هو موضح في (الشكل2-1) ، فإن الأقواس الخارجة من الحالة ($S_0$) مرتبة كالآتي: أولاً حذف البادئة (أول بادئة ملاحظة في الكلمة) ثم تكرر هذه العملية على باقي الكلمة حتى تتوقف.

استخدمنا هذا الترتيب اعتماداً على حقيقة إحصائية تقول بأن الأسماء تتكرر في النص أكثر بكثير من الأفعال.

للتوضيح: كلمة "فَجْر" عند تمريرها على الأوتومات، إذا كانت ستمر على القوس المرتبط به عملية "حذف البادئة_2" أولاً، سيكون الخرج كالتالي:

1. الكلمة البسيطة: "جر".

2. بادئة_2 : "فـ".

أما في حال كانت الكلمة ستمر على القوس الذي ترتبط به العملية "مطابقة" –التي تقوم بمطابقة الكلمة مع قاعدة البيانات (corpus)– سنحصل على الخرج التالي:

1. الكلمة البسيطة : "فجر" (أي حل واحد لهذه الكلمة هو أن تكون كلمة بسيطة).

هذا مثال بسيط يوضح أهمية ترتيب العمليات في الأوتومات.

- لاحظ أننا سنستخدم كلمة (جذر) و(كلمة بسيطة) بشكل متبادل، أي أننا لا نقصد بكلمة الجذر هي الفعل المجرد الثلاثي أو الرباعي أو الخماسي، وإنما أبسط كلمة يمكن الوصول لها كما شرحنا سابقاً.

- أيضاً سنستخدم كلمة "قاموس" لندل بها على الـ(Corpus) الذي تحدثنا عنه سابقاً .

**2-4-3-1-المخطط الذي يوضح خطوات الوصول للكلمة البسيطة (الأوتومات):**

المخطط كما ذكرنا سابقاً عبارة عن شبكة من الحالات التي تصل بينها أقواس والتي تمثل العمليات التي قادت للحالة التالية.

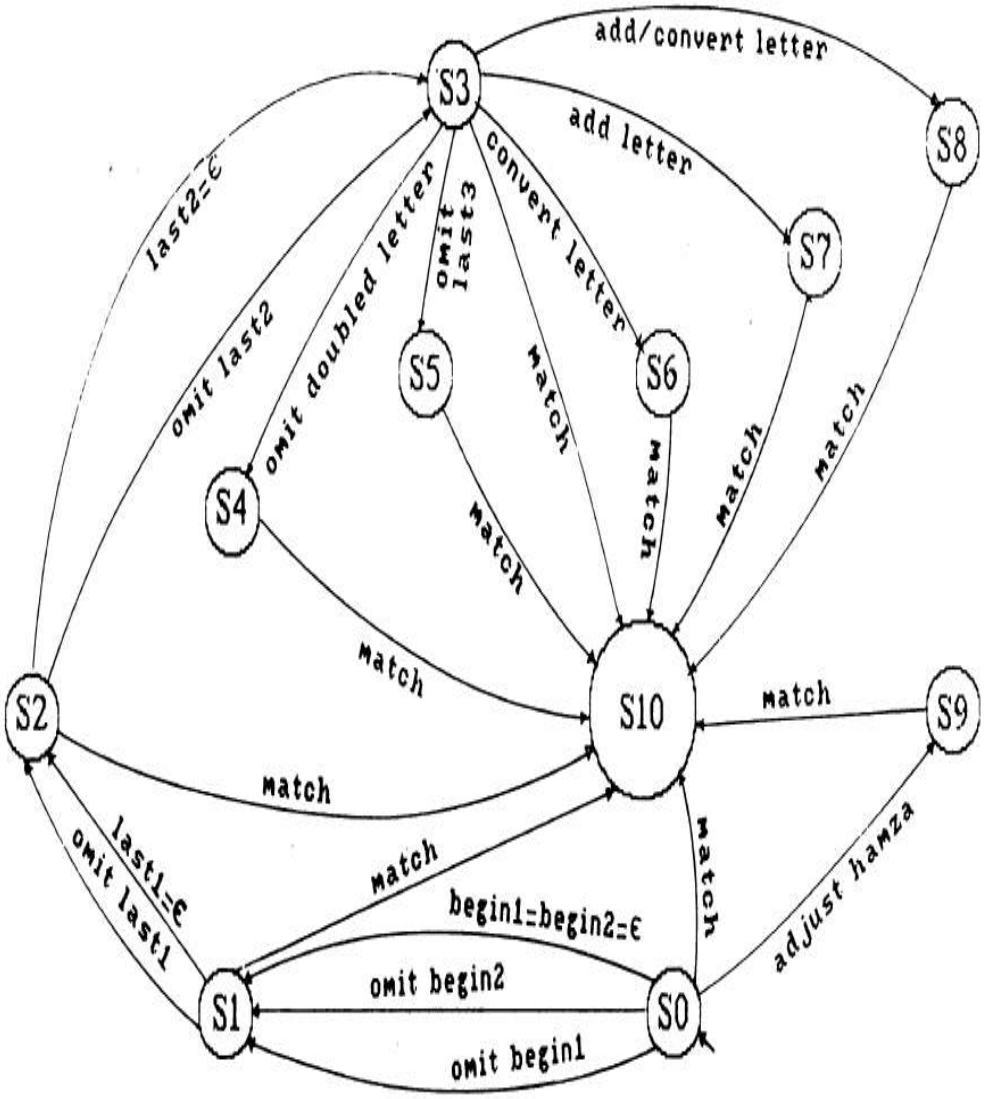



*Figure 1. ATN representing the relation between the additions and stem of an inflected Arabic word*

**(الشكل 2–1 أتومات تحليل الكلمة العربية)**

### 2-4-4- كتابة الأحرف في اللغة العربية:

إن الأحرف في اللغة العربية ليست كمثيلاتها في اللغة الإنكليزية، ففي الإنكليزية للحرف رسم واحد فهو منفصل دوماً عما قبله وما بعده، أما في اللغة العربية فيتغير الحرف تبعاً لوروده ضمن الكلمة، فحرف الواو مثلاً له شكلان في الكتابة "و" إذا كان لم يمكن اتصاله بما قبله أو كان في بداية الكلمة وأيضاً "ـو" إذا أمكن اتصاله بما قبله وكان في وسط الكلمة، ولا يمكن أن يتصل بما بعده. مثال ثانٍ: حرف العين: يكتب بأربعة أشكال: "ع" ، "ـــع" ، "ـعـ" ، "عـ".

لماذا أوردنا هذا الكلام؟

لنوضح أنه في جدول البوادئ هناك بعض الأحرف التي لم نوردها كحرف الواو "و" عندما يرد كحرف عطف، وحرف "لا" الناهية أو النافية، هذه الأحرف يمكن أن تكتب منفصلة عن الكلمة التي بعدها بفاصل، دون أن يؤثر ذلك على قواعد اللغة، أما بعض الأحرف مثل الكاف أو الفاء لا يمكن أن تكتب منفصلة. مثلاً جملة: "قرأ و كتب"، يمكن أن تكتب: "قرأ وكتب" ، أما جملة "قرأ فكتب" لا يمكن أن تكتب: قرأ ف كتب" ، فهذا خطأ في الكتابة.

على الرغم من أن المحلل النصي، يستطيع معالجة مثل هذه الأحرف وذلك بإضافتها إلى جدول البوادئ؛ إلا أنه كان من الأفضل ألا يتم ضم هذه الأحرف إلى أحرف البوادئ، وإنما تكتب في النص بشكل منفصل لأن ذلك يزيد من فعالية أداء محلل النص، ويقلل من الأخطاء.

### 2-4-5- القواعد التي يعتمد عليها محلل النص:

إن القواعد المرتبطة بالأقواس في الشكل السابق يمكن تصنيفها في خمسة تصنيفات (بعض المراجع تصنفها في ستة تصنيفات[5] والتصنيف الأخير يعالج حالات الهمزة، وهي حالات خاصة لا تفيدنا في بحثنا وإنما تزيد من زمن المعالجة ويمكن التغاضي عنها وقد أوردناها في الشكل للفائدة العامة)، التصنيف الأول -التصنيف هو عملية- هو عبارة عن قاعدة واحدة، أما بقية التصنيفات فيتألف كل منها من عدة قواعد*.

[5] Lexical Analysis of Inflected Arabic Words using Exhaustive Search of an Augmented Transition Network, Ahmed Rafea and Khaled Shaalan.

* إن جميع هذه القواعد تم كتابتها بشكل (pseudo code).

**1**- العملية (match): هي العملية المرتبطة بالقوس المؤدي إلى آخر حالة، هذه العملية مهمتها مطابقة الجذر الناتج مع الجذور المخزنة في القاموس.

مثال على هذه القاعدة:

```
IF The stem is matched with a corpus entry
THEN Retain this situation as a possible solution
```

**2**- العملية (omission): المهمة الأساسية لهذه العملية هي حذف الأحرف الزائدة من الكلمة، وذلك بمقارنة الأحرف الأولى أو الأخيرة من الكلمة مع أحد حقول جداول البوادئ واللواحق التي أوردناها سابقاً.

مثال على هذه القاعدة:

```
IF Initial characters of the inflected word are in
Begin1_set
THEN Omit these characters and keep them in
     Begin1_register;
IF Last characters of the inflected word are in
     Last2_set
THEN Omit these characters and keep them in
     last2_register.
```

مثلاً بتطبيق هذه القاعدة على كلمة "الناخبون" يعطي:

آ- begin1_register: "ال".
ب- last2_register: "ون".
ج- stem_register: "ناخب".

**3**- القاعدة الثالثة تهتم بتحويل أحد أحرف الجذر الناتج إلى وضعه الصحيح بعد الحذف، ووضع القيمة المناسبة في مسجل الحالة (status flag). يتم ذلك اعتماداً على وجود أحرف محددة في مواقع محددة من الجذر، بالإضافة إلى القيمة الموجودة في مسجلات الحذف (omission registers). هذه القاعدة رمزنا لها في الشكل السابق بـ"تحويل (conversion)".

مثال على هذه القاعدة:

```
IF Last letter stored in stem_register is 'و' AND
((last2_register='ان' OR (last2_register= 'ون') OR
((last2_register = 'ات'))
THEN Convert 'و' to 'ء';
```

بتطبيق هذه القاعدة على كلمة "صحراوان" كمثال تعطي:

آ- last2_register: "ان".

ب- stem_register: "صحراء". (الكلمة بدون تحويل هي "صحراو").

ج- status_flag: تحويل الحرف الأخير "و" من الجذر إلى "ء".

**4-** هذه العملية هي (addition) (وهي مجموعة من القواعد) تقوم بإضافة أحرف إلى الجذر - في حال عدم مطابقته لأي جذر ضمن القاموس- في أماكن محددة وذلك بدلاً عن الأحرف المحذوفة في العملية (omission).

يتم ذلك بتفحص طول الجذر الناتج، أو فحص الأحرف المحذوفة إن كانت مساوية لقيمة معينة، في هذه العملية لدينا ثلاثة أنواع من القواعد:

1.4- إضافة أحد الأحرف الأربعة التالية اعتماداً على طول الجذر الناتج: "ا"،"و"،"ي"،"أ" . مثال:

```
IF The length of the unrecognized stem stored in Stem
     register is two letters
THEN Add 'و' at the beginning of the stem;
```

كشرح على هذه القاعدة إذا حصلنا بعد عملية الحذف على الكلمة "عد"، بتطبيق هذه القاعدة عليها ينتج:

آ- stem_register: "وعد".

ب- status_flag: إضافة حرف "و" إلى بداية الكلمة.

2.4- هذه المجموعة من القواعد تفحص مسجلات الأحرف الزائدة المحذوفة قبل إضافة أي حرف إلى الكلمة:

```
IF ((last1 register = 'وا')
     OR (last1 register = 'ي')
     OR (last2 register ='ون')
```

```
     OR (last2 register = 'ين')
     OR (last3 register = ','))
THEN Add 'ى' last;
```

بتطبيق هذه القاعدة على كلمة "مصطَفَون" ينتج لدينا:

آ- last2_register: "ون".

ب- stem_register:"مصطفى" (بعد الحذف أصبحت "مصطف").

ج- status_flag: إضافة "ى" إلى آخر الكلمة.

3.4- هذه القاعدة يمكن اعتبارها قاعدة تعتمد على الحدس لتحديد ما إذا تم حذف حرف الياء من آخر الجذر -الذي سيكون في هذه الحالة اسماً منقوصاً- كي تتم إضافته، هذه القاعدة تطبق في حال المرور على كل القواعد السابقة، ومازال الجذر غير مميز (أي غير مطابق لأي جر ضمن القاموس).

بكل بساطة هذه القاعدة هي عملية إضافة لحرف الياء إلى آخر الكلمة بدون أي شرط:

```
Add 'ي' at the last of the stem;
```

مثلاً إذا حصلنا على كلمة "محام"، بتطبيق هذه القاعدة ينتج:

آ- stem_register: "محامي".

ب- status_flag: إضافة حرف "ي" إلى نهاية الكلمة.

**5**-هذه العملية تضم قواعد تقوم بتحويل أحد حروف الجذر الناتج، وإضافة حرف في مكان محدد للجذر الغير مميَّز ضمن القاموس.

تنفيذ الشرط في هذه القاعدة يعتمد على وجود حرف معين في مكان معين من الكلمة وعلى الأحرف الزائدة التي تم حذفها، هذه المجموعة من القواعد نراها في الشكل السابق على السهم المسمى (addition/conversion).

مثال على هذه القاعدة:

```
IF   Middle letter stored in stem_register is 'يـ'
     AND ((begin2_register='أ')
     OR (begin2_register='نـ')
```

```
     OR (begin2_register='يـ')
     OR (begin2_register='تـ ')
     OR (begin2_register='سنـ')
     OR (begin2_register = 'سأ')
     OR (begin2_register = 'سـ')
     OR (begin2_register = 'ستـ ')
     OR (begin2_register = 'فسأ')
     OR (begin2_register = 'فسنـ')
     OR (begin2_register = 'فستـ')
     OR (begin2_register = 'فسيـ')
     OR (begin2_register = 'فلتـ')
     OR (begin2_register = 'فليـ'))
THEN Convert 'يـ' to 'أ', middle AND add 'ا', first.
```

بتطبيق هذه القاعدة على كلمة "يستطيع" ينتج لدينا:

آ- begin2_register: "يـ".

ب- stem_register:"استطاع" (بعد الحذف أصبحت "ستطيع").

ج- status_flag: تحويل الياء "ـيـ" في وسط الكلمة إلى "ا" وإضافة "ا" في بداية الكلمة.

**2-4-6- الوصول إلى الجذر الأكثر احتمالاً:**

يتم الوصول إلى جذر الكلمة بالتنقل في الشبكة السابقة، للبحث عن جميع الطرق الموصلة من الحالة الابتدائية إلى الحالة النهائية.

ترتيب الحلول الناتجة يعتمد على ترتيب الطرق التي سلكها البرنامج للوصول إلى الحالة النهائية. وقد تم ترتيبها في الشكل السابق بحيث يكون الحل الأول هو الحل الأفضل، وهو الحل الذي نعتمد عليه في تطبيقنا.

بعد الوصول إلى الحل الأول يتم حفظ هذا الحل، ويقوم البرنامج بالمتابعة لإيجاد بقية الحلول.

**الخلاصة :**

قدمنا في هذا الفصل محاولة لتحليل نصوص اللغة العربية، حيث يتم ذلك بإيجاد أبسط حالة لكل كلمة من كلمات النص.

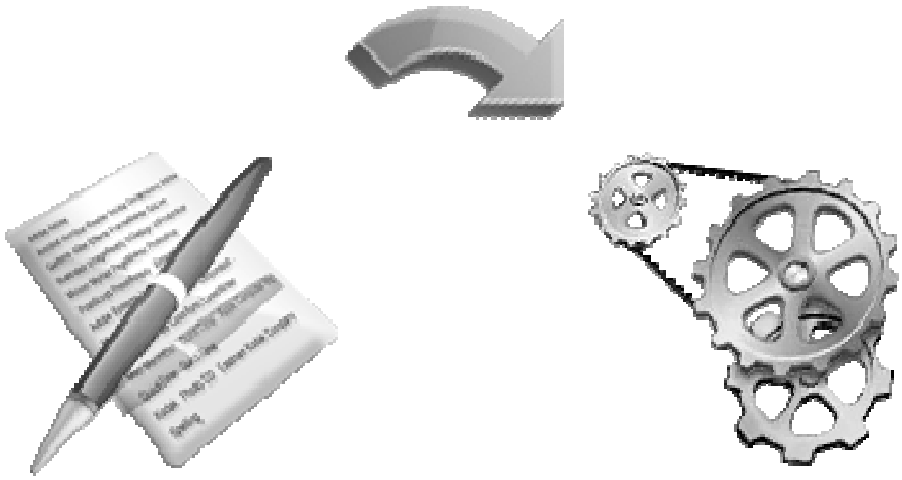
هذه المرحلة هي المرحلة الأولى والأساسية في عملية تحليل وثيقة المتطلبات، وتعتمد دقة المراحل التالية على دقة مخرجات هذه المرحلة.

سننتقل في الفصل القادم بإذن الله لتفصيل كيفية الاستفادة من تحليل اللغات الطبيعيـــة (NLP) في هندسة البرمجيات وبالتحديد في عملية تطوير الأنظمة البرمجية.

الفصل الثالث

# استخدام معالجة اللغات الطبيعية في هندسة البرمجيات

# USING NATURAL LANGUAGE PROCESSING IN SOFTWARE DEVELOPMENT



محتويات الفصل:

يتمحور هذا الفصل حول:

- الاستفادة من تحليل اللغات الطبيعية في هندسة البرمجيات.
- المنهجيات المتبعة في استخراج الصفوف.
- شجرة الإعراب و مجال المعرفة.
(Domain knowledge and Parsing Tree).
- خوارزمية (TCM) لاستخراج الصفوف.

إن جهاز الكمبيوتر ليس ذلك الجهاز الذكي كما يظن أغلب الناس العاديين، إنه ليس سوى كومة من الدارات الكهربائيـــة التـــي لا تفهم إلا لغة الصفر والواحد!

– مبرمج–

**مقدمة:**[6]

كما نعلم أنه لبناء أي نظام برمجي نحن بحاجة إلى تحليل دقيق لوثيقة المتطلبـــات و تحويلها إلى شكل التصميم وهذه المرحلة تتضمن اكتشاف مجموعة الصفوف التي هي الأساس في بناء النظام.

يقوم محلل النظام بقراءة الوثائق المتوفرة حول النظام المطلوب وترجمة أفكاره إلـــى شكل تصميمي، أي يقوم بقراءة المشكلة التي تكون مكتوبة باللغة الطبيعيـــة (informal)، و يقوم بتحويلها إلى شكل تصميمي (formal) باستخدام إحدى لغات النمذجة.

نلاحظ وجود فرق كبير بين اللغة الطبيعية و الشكل التصميمي، فمن جانـــب تكـــون المشكلة مكتوبة باللغة الطبيعية (لغة الزبون) ، ومن جانب آخر الـــشكل التـــصميمي الممثـــل بواسطة (UML). لكن السؤال المطروح هنا هل يمكن بناء أداة تقوم باستخراج الـــصفوف بشكل منهجي مؤتمت؟

إن جوهر المشكلة يكمن في أن اللغة الطبيعية تعتبر غامضة، وأن على الأداة أن تقوم بالتحويل من شكل عامّي (informal) إلى شكل منهجي (formal).

كما ذكرنا سابقاً إن تطوير البرمجية يبدأ عادة بمجموعة من الوثائق المكتوبة و التـــي تشرح المشكلة و طبيعة النظام المطلوب.

إن اللغة الطبيعية التي نستخدمها في حياتنا و التي تستخدم لوصف النظام كثيراً مـــا تكون معقدة و غامضة و تؤدي إلى حصول التباس. فالجملة تكون معقـــدة عنـــدما نـــستخدم التعابير التي تصف أشياء مرتبطة مثل ( رقم الزبون، اسمه، رقم هاتفه) هنا (الهاء) تعبر عن الزبون، أيضاً استخدام بقية الضمائر والضمائر المستترة و غيرها ..

إن استخدام اللغة الطبيعية خلال مراحل تطوير هندسة البرمجيات تعاني من المشاكل التالية:

- تعتبر اللغة الطبيعية مبهمة، بالتالي التحليل الدقيق يعتبر من الأمور الصعبة.

[6] Using Natural Language in Software Development (Nik Boyd)

- الجملة التي لها نفس الدلالة يمكن أن تمثل بأكثر من طريقة.
- هناك بعض الصفوف المخفية (hidden classes) التي لا تذكر صراحة في النص، هناك حاجة إلى نظام معرفة خبير (Expert Domain Knowledg) لتحديد هذه الصفوف.

من ناحية أخرى تتطلب البرامج الدقة و البساطة و الانتظام في شرح المشكلة، كمثال: إن لغات البرمجة تقتصر على بضع عبارات لها صيغة محددة، هذه العبارات مقيدة بقواعد لغة البرمجة (syntax of programming language).

نظراً لهذه العوامل، من غير المستغرب أن الترجمة من اللغة الطبيعية إلى شكل منتظم يشكل تحدياً كبيراً. نحن بحاجة إلى أداة تقوم بالتحويل من الشكل العامي إلى شكل مناسب للنموذج التصميمي كما ذكرنا.

### 3-1- المنهجيات المتبعة في أدوات التحليل والتصميم[7] :

شهد العقدان الماضيان تنامياً في الاهتمام بالأدوات المساعدة لمهندسي البرمجيات (CASE tools). وقد لوحظ وجود نهجين أساسيين لتوفير الأدوات المؤتمِتة التي تدعم المراحل المبكرة ( التحليل و التصميم) في تطوير البرمجية:

النهج الأول يقوم على تطبيق أساليب الذكاء الصنعي (AI Artificial Intelligence)، حيث يقوم بتطبيق خطط استنتاجية و استراتيجيات في البحث لتكوين نظام ذكي، لكن هذا النهج عانى من التعقيد الكبير في الخوارزميات المتبعة.

أما النهج الثاني فهو يقوم على تحليل اللغات الطبيعية (لغات البشر)، هذا المنحى يعتمد على أن معظم البيانات الهامة في تطوير النظام تكون موجودة في الوثائق، النظام المبني في بحثنا هذا يتبع المنهج الأخير.

في البدايات تم اتباع منهجية تقوم على التحليل اللغوي فقط، و استخراج الأسماء و الأفعال والأحرف في النص. حيث تم اقتراح الأسماء على أنها تمثل الصفوف، و الأفعال على أنها تمثل العلاقات[8].

[7] CM-Builder: An Automated NL-based CASE Tool (H.M. Harmain, R. Gaizauskas)

هذا النظام يعاني بشكل واضح من الغموض، وذلك لأن هذا الأسلوب لن يتمكن من تمييز الكلمات الهامة عن غيرها، فبعض الأسماء تمثل صفات (attribute) و بعضها لشرح المشكلة فقط و هي لا تمثل صف (Class).

كمثال على ذلك لنأخذ الجملة التالية: ( مطلوب نظام لتأجير أشرطة الفيديو)، كما نلاحظ أن كلمة (نظام) لا تمثل صف (Class) في بنية النظام المطلوب.

تمت معالجة الغموض في البداية عن طريق واجهات تسأل المحلل عن قبوله أو رفضه لكل صف من الصفوف، بالتالي تطلب ذلك تدخل كبير من المحلل في هذه المرحلة، و هذا الحل يعتبر حل نصف مؤتمت (semi automated).

فيما بعد تم إدخال العديد من التقنيات لمعالجة حالة الغموض كتحليل بناء تركيب الجملة (شجرة الإعراب أو Syntax tree) و استخدام مجال المعرفة للمشكلة المدروسة (Domain Knowledge) و الاعتماد على تكرار الكلمة في النص و أساليب حدسية أخرى (heuristic rules).

الفقرات التالية سوف تعرض الأساليب الإضافية لمعالجة الغموض.

- **تحليل بناء تركيب الجملة وشجرة الإعراب (Syntax Tree Structure) :**

استناداً إلى قواعد النحو، يمكن تجميع قواعد كتابة الجملة و حصرها في مجموعة من القواعد المنطقية التي تصف تركيب الجملة.

هذه القواعد تصف جميع تشكيلات الجملة ويمكن تمثيلها كأوتومات منتهي حيث أن الجملة يمكن تحليلها إلى مجموعة جمل فرعية و هكذا.

كمثال لدينا الجملة التالية (الزبون يمتلك حساب في المصرف)، يمكن حصر قواعد تركيب هذه الجملة كما يلي :

**جدول (3-1) قواعد تركيب الجمل**

| Phrase |
|---|
| $S \rightarrow NP+VP$ |
| $NP \rightarrow Noun$ |
| $NP \rightarrow NP + PP$ |
| $VP \rightarrow Verb + NP$ |
| $VP \rightarrow Verb + NP + PP$ |
| $PP \rightarrow prepostion + NP$ |

ملاحظة:

S : تمثل الجملة كاملة وهي الجذر root.

NP : تمثل الجملة الاسمية (Noun Phrase).

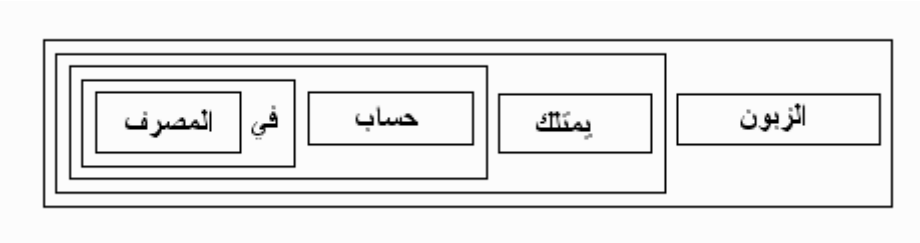
VP : تمثل الجملة الفعلية (Verb Phrase).

PP : تمثل شبه الجملة (جار ومجرور) (Prepositional Phrase).

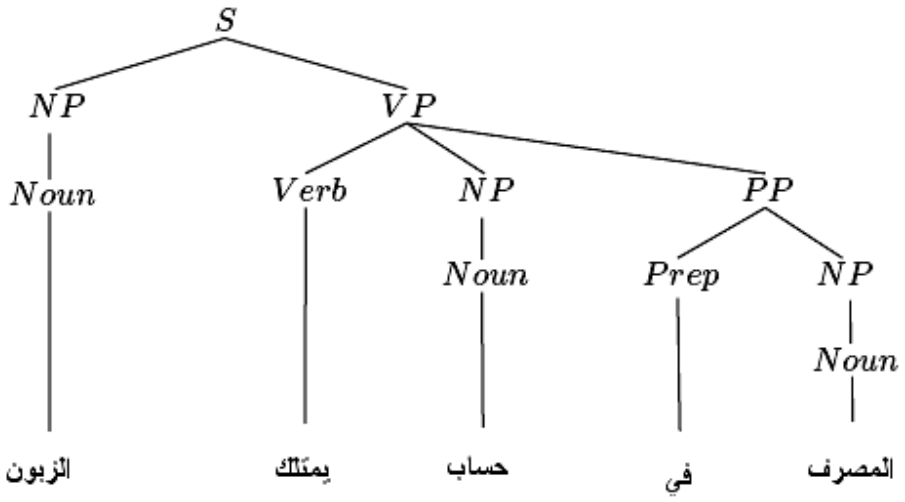
القاعدة الأولى تعني أن الجملة تتألف من عبارة اسمية (NP) و عبارة فعلية (VP).

القاعدة الثانية و الثالثة تعني أن العبارة الاسمية يمكن أن تتألف من اسم فقط (Noun) أو من اسم ملحوق بشبه جملة (جار ومجرور) (Prepositional phrase)، و هكذا .

و بناء على ذلك يتم تمثيل تركيب الجملة كما يلي :




**(شكل 3-1 تمثيل تركيب الجملة)**




**(شكل 3-2 شجرة الإعراب لجملة)**

و يتم بناء بقية القواعد لمختلف تراكيب الجمل الممكنة بشكل مشابه.

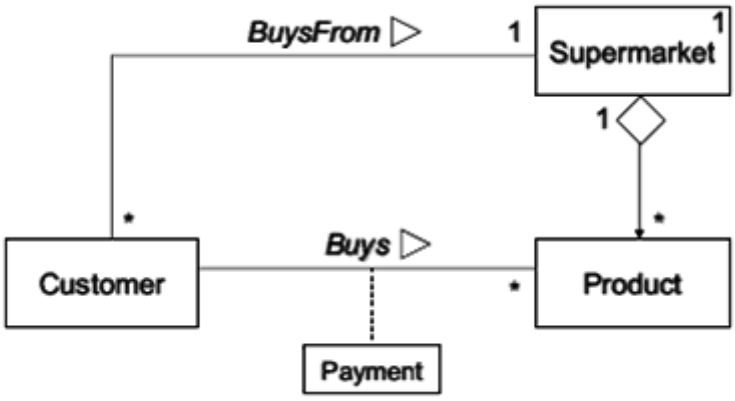
يفيد التمثيل السابق في تحديد حدود الجمل (استخراج الجمل) و في فحص نماذج معينة في النص (pattern) و تحديد الصفوف بناء على ورود هذا الاسم في النموذج، وأيضا في استخراج العلاقات.

كمثال بسيط على ذلك ليكن لدينا النموذج التالي (اسم فعل اسم) حيث أنه إذا طابقت إحدى الجمل هذا النموذج (موظف يمتلك حساب) فيمكننا الحكم على أن هذين الاسمين هما صفان و الفعل يمثل علاقة بينهما.

- **مجال المعرفة Domain Knowledge :**

دراسة الأغراض في العالم الحقيقي التي تخص مجموعة من الأنظمة المتشابهة و إيجاد الأغراض المشتركة و تخزينها ضمن قاعدة معرفة لاستخدامها مستقبلاً في مشاريع لاحقة حيث يتم تخزين المخطط المفاهيمي (conceptual Model).

كمثال لنأخذ المعرفة عن مجال (مؤسسة تجارية)، من الطبيعي أنه في أي مؤسسة تجارية يوجد زبون و منتج و علاقة شراء (payment) بين الزبون و المنتج كما هو موضح بالشكل:




**(شكل 3-3 علاقة الزبون بالمنتج)**

نقوم بدراسة مجالات مختلفة مثلاً (المستشفيات، المدارس، الجامعات، .. إلخ) و تخزين الأغراض الموجودة في العالم الحقيقي والموجودة في كل مجال، و يتم تصنيف هذه الأغراض ضمن فئات (class category)، هذه التصنيفات موضحة بالجدول التالي:

**(جدول 3-2 تصنيفات الفئات class categories)**

| | |
|---|---|
| Role of People | تمثل الأشخاص الذين يؤدون بعض الوظائف الهامة، من أمثلة ذلك الطالب (Student)، الموظف (Employee)، الزبون (Customer). |
| Places | الأماكن: تمثل أماكن الأنشطة التجارية الهامة، مثل: مكتب (Office)، المخزن (Store). |
| Physical Things | الأشياء المادية: و هي تمثل الأشياء الملموسة التي تعتبر هامة في الأنشطة التجارية، من أمثلة ذلك الآلات (Machine)، المنتجات (Product)، الأجهزة (Devices). |
| Organization | المنظمات: تمثل الوحدات التجارية الهامة مثل: شركة (Company)، فريق (Team)، دائرة (Department). |
| Event | الأحداث الهامة التي تحتاج إلى تسجيل وقت حدوثها مثل : طلب (Order)، الدفعات (Payment). |
| Concept | تمثل الأفكار غير المادية التي تستخدم لإبقاء على مسار الأنشطة التجارية مثال : حساب (Account)، مشروع (Project). |
| Interaction | تمثل علاقة بين صفين مثل صف يمثل علاقة بين المسافرين (Passenger) و الرحلات (Flights). |
| Container | تمثل الصفوف التي سوف تحتوي صفوف أخرى، أمثلة ذلك: مخزن (Store) ،رف (Shelf)، قائمة (Catalogue). |
| Thing in Container | تمثل الصفوف التي سترد في صفوف أخرى، مثال ذلك: شريط فيديو (Tape) سيرد ضمن القائمة (Catalogue). |
| Financial Instruments and Service | تمثل الأحداث المالية مثل الأسهم (Stock)، السندات (Bond). |

نلاحظ أن مجال المعرفة يستخدم كاقتراح لتحديد الصفوف، و هو يعتمد على المجال المدروس، و يساعد في معالجة الغموض.

- **قواعد حدسية أخرى (Other Heuristics Rules):**

يمكن الاعتماد على قواعد حدسية كعدد مرات تكرار الاسم في النص ، فكلما كان تكراره أكبر يكون هذا الاسم مرشحاً أكثر لأن يكون صفاً (Class). و يمكن اقتراح قواعد أخرى قد تفيد في زيادة احتمالية ترشيح الصف.

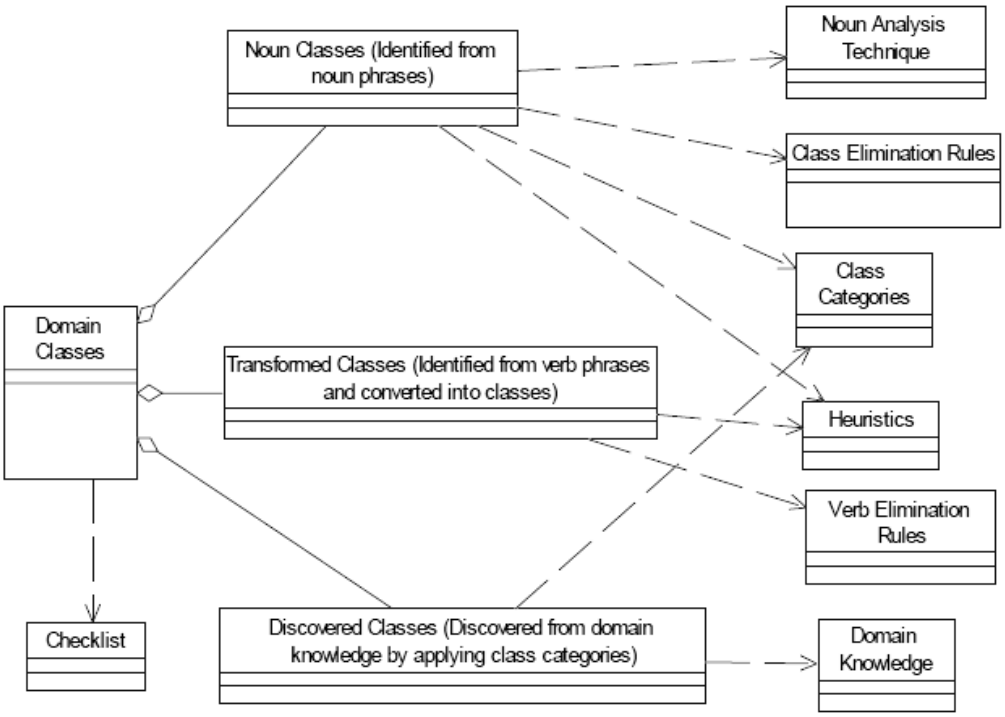
ظهرت العديد من المنهجيات و الأبحاث من أشهرها (TCM, cm-builder, …)، سوف نقوم فيما يلي بشرح تفصيلي لآلية عمل (TCM).

## 3-2- خوارزمية TCM :

**THE TAXONOMIC CLASS MODELING (TCM) METHODOLOG**

هذه الطريقة حاولت أن تدمج بين عدة أساليب لاكتشاف الصفوف، حيث قسمت صفوف المجال المدروس كما هو مبين في (الشكل 3-4 ) إلى ثلاثة مجموعات: الصفوف المستخرجة من التعابير الاسمية (Noun Classes) ، الصفوف المستخرجة من الأفعال (Transformed Classes)، الصفوف المخفية (Hidden Classes).

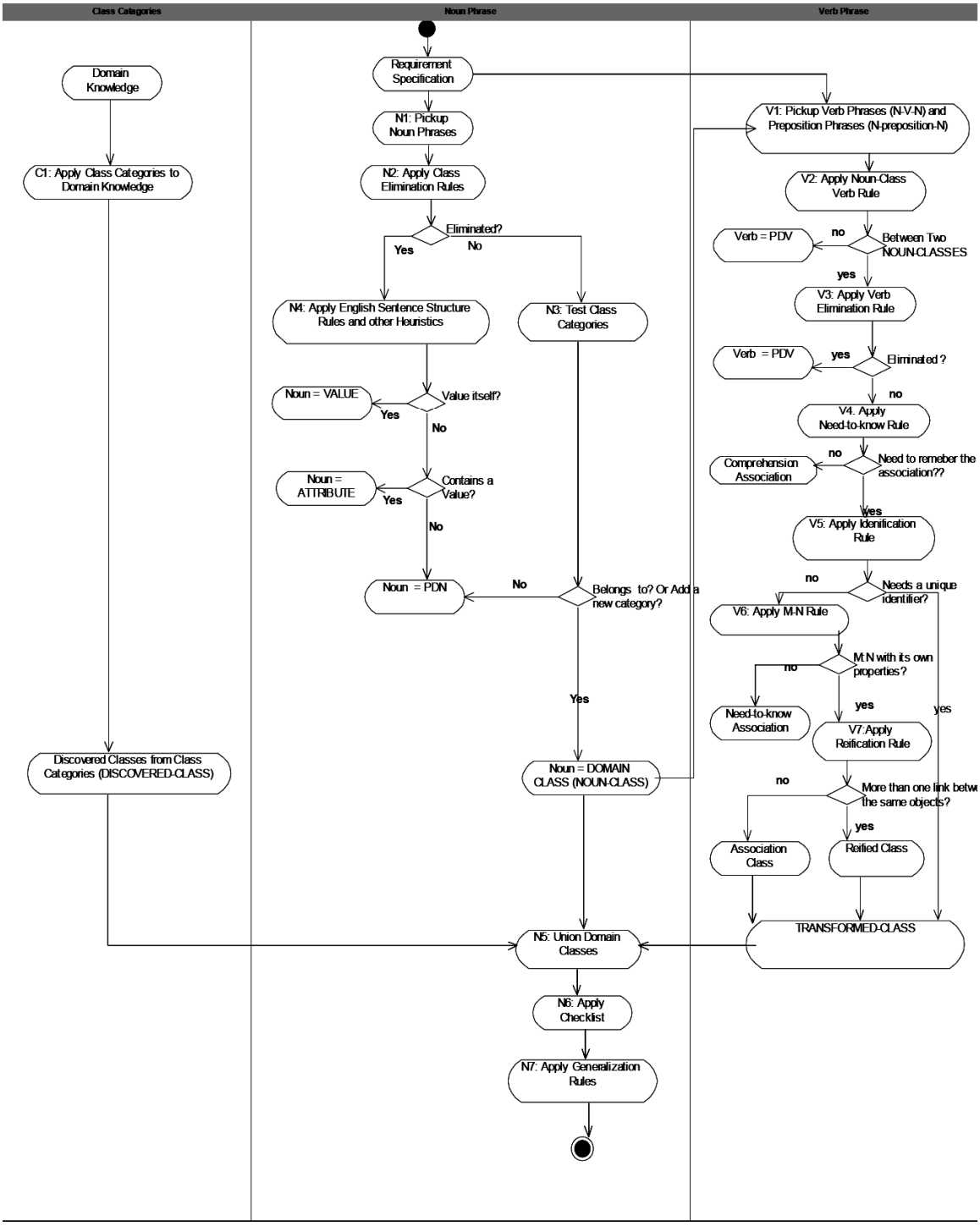
1. الصفوف المستخرجة من الأسماء (Noun Class): و هي الصفوف المستخرجة من التعابير الاسمية في وثيقة المتطلبات.
2. الصفوف المستخرجة من الأفعال (Transformed Class): وهي الصفوف المستنتجة من الجمل الفعلية في وثيقة المتطلبات.
3. الصفوف المخفية (Hidden Class): الصفوف التي يتم اكتشافها بواسطة تطبيق مجال المعرفة و القواعد الحدسية الأخرى.




**(الشكل 3-4 صفوف المجال في خوارزمية TCM و تقنيات استخراجها)**

### 3-2-1- المنهجية المتبعة في طريقة TCM :

في الشكل (3-5) يوضح المنهجية المتبعة خطوة بخطوة على شكل مخطط نشاطات (activity diagram)، كما نلاحظ لدينا في الشكل ثلاثة فروع (swimlane) مقسمة إلى فرع يساري وفرع أوسط و فرع يميني.




**(الشكل3-5 مخطط النشاطات (Activity Diagram) لخوارزمية TCM)**

- الفرع الأوسط: الهدف من نشاطات هذا المجرى هو استنتاج الصفوف عن طريق الأسماء التي ذكرت صراحة في وثيقة المتطلبات. ندعو هذه الصفوف بـ (noun classes).
- الفرع اليميني: الهدف منه تحديد الصفوف التي يمكن استخراجها من الأفعال، ندعو هذه الصفوف بـ (transformed classes).

- الفرع اليساري: الهدف من هذه النشاطات هو اكتشاف الصفوف المخفية و التي لم تذكر صراحة في وثيقة المتطلبات، و التي هي ضرورية بالنسبة لمجال المشكلة. يتم اكتشاف هذه الصفوف بتطبيق مجال المعرفة الخاص بالمشكلة المدروسة ندعو هذه الصفوف بـ (hidden classes).

سيتم توضيح القواعد المستخدمة في كل فرع في الفقرة التالية.

### 3-2-2- القواعد المستخدمة في منهجية TCM:

**النشاطات في الفرع الأوسط (middle swinlane):**

نبدأ من وثيقة المتطلبات:

- **خطوة N1:** نقوم باستخراج التعابير الاسمية (Noun Phrase) من الجملة المعالجة:
  - سوف يتم تصنيف هذه الأسماء إلى أسماء خاصة بشرح المشكلة (Problem Description Noun (PDN)) و أسماء خاصة بحل المشكلة (Problem Solving Noun (PSN)). حيث أن الـ PSNs هي تلك التي تصبح صفوف (Classes) أو صفات (Attributes) أو قيم (Values).

- **خطوة N2:** قاعدة استبعاد الصفوف (Class Elimination Rules):
  - إذا كانت الجملة الاسمية تطابق أحد قواعد الاستبعاد، عندها هي ليست صف، و يتم استبعادها من قائمة الصفوف المرشحة.

- **خطوة N3:** تطبيق قاعدة الفئات (Class Category Rule):
  - إذا كان الاسم يطابق أحد الصفوف الموجودة في فئات الصفوف (Class categories)، عندها هذا الاسم يمثل صف ندعوه بـ (Noun Class)، أما إذا كان الاسم لا يطابق أحد الصفوف في فئات الصفوف، عندها نستخدم طرق حدسية لإبقائه كصف أو استبعاده.

- **خطوة N4:** نطبق قواعد تركيب الجمل العربية (Arabic Sentence Structure) و أدوات مساعدة أخرى، نصنف هذه الأسماء إلى صفوف (classes) أو صفات (attributes) أو قيم (values).
  - إذا كان الاسم يمثل صف أو صفة أو قيمة، إذن هو (PSN)، وإلا نعتبره (PDN).

**النشاطات في الفرع اليميني (Rightmost swimlane) :**

- **خطوة V1:** نقوم بتحليل الجملة التي لها الصيغة (اسم-فعل-اسم) أو (اسم-حرف جر-اسم) كآلية من أجل ترشيح العلاقات association.
  - نقوم بتصنيف الأفعال إلى أفعال شرح المشكلة (Problem Description Verbs (PDV)) وأفعال حل المشكلة (Problem Solving Verbs (PSV)). الـ (PSV) هي الأفعال التي يمكن أن تمثل علاقة (association) في مخطط الصفوف.
- **خطوة V2** (Noun Class-Verb Rule):
  - إذا كان واحد من الأسماء التي تحيط بالفعل ليست صف، عندها الفعل هو (PDV)، فنقوم باستبعاد هذا الفعل.

```
The Noun-Class Verb Rule
IF one of two nouns surrounding the verb or the
preposition is not a NOUNCLASS
THEN the verb is a problem-description verb.
Eliminate it.
```

- **خطوة V3** (Verb Elimination Rule):
  - إذا كان الفعل في قائمة الأفعال المستبعدة، عندها الفعل هو (PDV)، نقوم باستبعاده (سوف نقوم بشرح هذه القواعد لاحقا).
- **خطوة V4** (Identification Rule):
  - إذا كان المفهوم الذي يمثله الفعل يدل على وجود صف، عندها نقوم بتحويل الفعل إلى صف. ندعو هذا الصف بـ (Transformed Class)، هذه الخطوة تعتمد بشكل كبير على القواعد الحدسية.

    مثال : الفعل (يحاسب) يدل على وجود صف (محاسب) .
- **خطوة V5** (M:N Rule):
  - إذا كان الفعل يمثل علاقة بين صفين، و هذه العلاقة تملك مجموعة من الخصائص، نقوم بتحويله إلى صف رابط (association class)، ندعو هذا الصف أيضاً بـ (Transformed class).

**النشاطات في الفرع اليساري (Leftmost Swimlane):**

- **خطوة C1** (Apply domain knowledge to class category):

- نقوم بتطبيق معرفة المجال و استنتاج صفوف من قاعدة المعرفة، ندعو الصفوف المستخرجة في هذه الخطوة بـ (hidden classes).

و بالتالي تكون مجموعة الصفوف التي تمثل المجال هي اجتماع الصفوف المستخرجة من الفروع الثلاثة :

$$\{\text{Domain Classes}\} = \{\text{Discovered Classes}\} \cup \{\text{Noun Classes}\} \cup \{\text{Transformed Classes}\}$$

**3-2-3- تفصيل التقنيات المستخدمة :**

**قواعد الإستبعاد (Elimination Rules):**

قواعد الاستبعاد المقترحة في الخطوة N2 هي:

- **Redundant classes:** اسمان يمثلان نفس مفهوم التجريد، نختار الأكثر تخصصاً ، مثال نستخدم (Customer) بدلاً من الـ (User).
- **Irrelevant classes:** الأسماء ليس لها علاقة بالمشكلة المدروسة (خارج نطاق المجال).
- **Vague Classes:** الاسم له معنى واسع جداً ، مثل (Business).
- **Operations:** الاسم يمثل عملية (operation) مثل (حساب الفاتورة)، الجملة الإسمية (حساب الفاتورة) هو اسم لعملية.
- **Attributes:** الاسم يمثل صفة أو عدد، على سبيل المثال الاسم و العمر و رقم الهاتف تمثل الصفات التي تحمل قيمة، هناك بعض الحالات التي يحدث غموض في كون الجملة الاسمية تمثل صف (class) أو خاصية (attribute)، لذلك نقوم باستخدام القاعدة التالية:

```
IF a noun has only one property to remember
THEN it is an attribute of another class
ELSE it is a class
```

**مثال :**

في وثيقة المتطلبات (اسم المدينة) ذكرت لمرة واحدة، هـذا يعنـي أن اسـم المدينة هو خاصية لصف ما. أما إذا ذكر (اسم المدينة، المحافظة التابعـة لهـا، موقع المدينة) عند ذلك (المدينة) تمثل صف لأنها تمتلك أكثر من صفة.

العبارة الاسمية التي تجتاز القواعد السابقة نرشحها كصف.

```
The Class Elimination Rule
IF the noun candidate belongs to one of the CER rules
THEN it is not a class.
```

**قاعدة فئات الصفوف (Class Category Rule):**

إذا كانت الأسماء المرشحة التي اجتازت قواعد الاستبعاد تنتمـي إلـى قائمـة تصنيفات الصفوف، إذن هي تمثل صف يخص المجال المدروس:

```
The Class Category Rule
IF the candidate noun which passed the Class Elimination Rules belongs to one of the class categories
THEN it is a domain class and we call it a NOUN-CLASS
ELSE use domain knowledge to decide whether to keep the class.
```

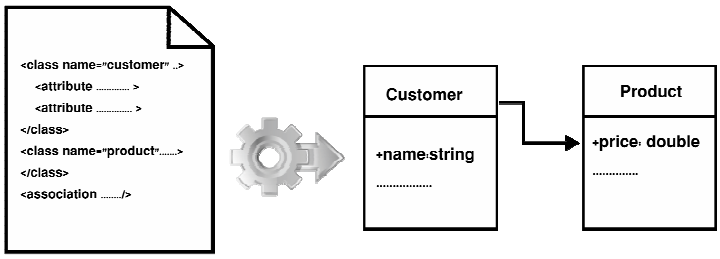
ندعو الصفوف التي اجتازت قواعد الاستبعاد، و التي تنتمي إلى أحد الفئـات بـ (Noun Classes).

**خرج هذه المرحلة:**

بعد الانتهاء من عملية التحليل للنص العربي يتم توثيق الخرج الناتج من هذه المرحلة على شكل مستند XML يوصف داخله الصفوف و خصائصها و العمليات التي تتم عليها مع سرد للعلاقات بين هذه الصفوف .

**الانتقال من مرحلة التحليل إلى المخطط التصوري:**

تأخذ هذه المرحلة كدخل ملف الـ XML الذي هو خرج المرحلـة السـابقة و يـتم إظهار المخطط الناتج، و توفر هذه الأداة إمكانية تعديل الـصفوف و إضـافة الـصفوف و العلاقات.




**الشكل (5 -5) الإنتقال إلى الشكل التصويري**

## خرج هذه المرحلة:

بعد الانتهاء من عمليات التعديل فإننا نحصل كخرج على ملف XML له نفس توصيف ملف الدخل الذي تم إدخاله إلى هذه المرحلة و زيادة عليه التعديلات التي تم إضافتها من خلال هذه الأداة.

## الخلاصة:

قمنا في هذا الفصل بعرض التقنيات التي تساعد على استخراج الصفوف من وثائق المتطلبات.

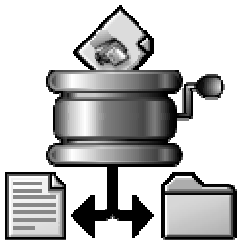
في البداية رأينا أهمية استخدام معالجة اللغات الطبيعية (NLP) من أجل ترشيح الصفوف، ثم وضحنا بعد ذلك أن استخدام هذه التقنية وحدها لا تكفي، لذلك استخدمنا أساليب أخرى تساعد في معالجة الغموض، مثل استخراج شجرة الإعراب، واستخدام قاعدة معرفة للمشكلة المدروسة وقواعد حدسية أخرى.

ثم قمنا بعرض خوارزمية (TCM) كطريقة من أجل استخراج الصفوف و العلاقات. و تجدر الإشارة في النهاية إلى أن خوارزمية (TCM) تعتبر من أفضل الطرق المعروفة حتى يومنا هذا.

الفصل الرابع

# مـولـِّـــد الشِّـــيفرة

# CODE GENERATOR



محتويات الفصل:

ينـــاقش هذا الفصل:

- ما هو مولد الشيفرة (Code Generator).
- ما الذي يمكن توليده بواسطة مولد الشيفرة.
- الفوائد المجنية من استخدام مولد الشيفرة.
- أنواع مولدات الشيفرة.
- أشكال ملفات الدخل التي يمكن عن طريقها توليد الشيفرة.
- آلية عمل مولد الشيفرة.

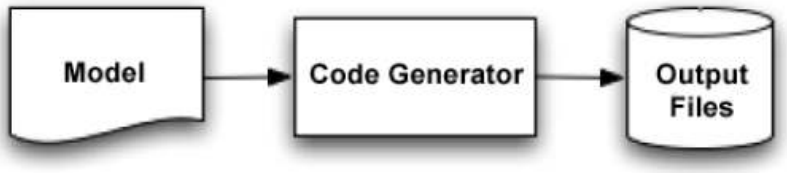
إن البرمجة اليوم هي سباق بين مهندسي البرمجيات لبناء برامج نظيفة أكثر، ومقروءة أكثر.

-ريتش كوك-

**مقدمة[9]:**

يمكن تعريف مولّد الشيفرة (Code generator) بأنه تقنية تستند على كتابة تطبيق يستخدم لتوليد شيفرة برنامج آخر، استخدام مولّد الشيفرة يُمكّننا من كسب الوقت و المال الذي يهدر بكتابة شيفرة تقليدية باليد.

يقوم البرنامج السابق بقراءة دخل معيّن (Schema File) وهو توصيف للنظام بواسطة لغة معيارية، التوصيف يجب أن يحتوي على معلومات (metadata) عن النظام، مثلاً مخططات (UML)، توصيف بواسطة الـ (XML)، (metadata) لقاعدة المعطيات للنظام، ثم نطبق عليها مجموعة من القوالب (templates) التي تحقق التوصيف المدخل، الخرج يمكن أن يكون شيفرة برمجية (Java, C++) أو (DDL code أي SQL code) أو (HTML code) ... إلخ حيث يوجد لكل خرج قالب محدد (templat) .




**(الشكل 4-1 مراحل توليد الشيفرة)**

## 4-1- ماذا يمكن أن يولّد مولد الشيفرة (code generator)[10]؟

توليد هيكل قاعدة البيانات (SQL DDL Code) :

1. توليد طبقة الإتصال مع قاعدة البيانات (Generating the database access layer)
2. توليد طبقة منطق العمل (generate business logic layer).
3. توليد طبقة واجهة المستخدم (generating GUI).
4. توليد طبقة الاستدعاء عن بعد (Remote Procedure Call Layer RPC Layer).
5. توليد توثيق النظام (generate documentation).

[9] http://today.java.net/pub/a/today/2004/05/12/generation1.html
[10] Code Generator in Action (JACK HERRINGTON) Chapter 1

يوفر مولد الشيفرة (Code Generator) من وجهة نظر هندسة البرمجيات المزايا التالية[11]:

1. **تحسين الإنتاجية (Productivity) :**

يقوم مولد الشيفرة بتوليد الشيفرة المصدرية بشكل أسرع من الكتابة العادية حيث يقوم بتوليد مئات الصفوف في ثواني. كما أن إجراء أية تعديلات ستتم بسرعة بتعديل ملف الدخل، وبالتالي تتحسن الإنتاجية.

2. **التناسق (Consistency):**

برنامج توليد الشيفرة دائماً يتبع مقاييس كتابة الشيفرة كتسمية المتحولات (Variable Naming) التي يمكن أن يتجاهلها المبرمج، ككتابة التعليقات و المقاييس البرمجية الأخرى، وهذا يؤدي إلى شيفرة مفهومة وواضحة وسهلة الاستخدام.

3. **توفير قاعدة معرفة مركزية (A Single Point of Knowledge):**

أي تغيير في (schema file) سينعكس على كل أجزاء النظام. كمثال في طريقة الكتابة اليدوية و في أفضل حالاته عند تغيير اسم حقل في الجدول فيجب أن تشمل التغييرات في الجدول نفسه، و في طبقة الأغراض (object Layer) وفي توثيق النظام (documentation).

باستخدام مولد الشيفرة يمكن تغيير اسم الحقل و إعادة توليد الشيفرة، التغييرات ستنعكس مباشرة على كل طبقات النظام.

4. **التجريد (Abstraction):**

يمكننا من إنشاء توصيف للنظام بشكل مجرد بعيداً عن التحقيق و اللغة المستخدمة، لذلك النظام لا يوجه لمنصة عمل معينة.

5. **إعطاء وقت أكبر للتصميم (More Design Time):**

بما أن مولد الشيفرة يعتمد بشكل كلي على ملف الدخل، و الذي هو توصيف للنظام، فإن اهتمام الفريق سوف ينصب بشكل كبير على تصميم النظام حيث أن كتابة الشيفرة سوف تتولد بشكل آلي و سريع.

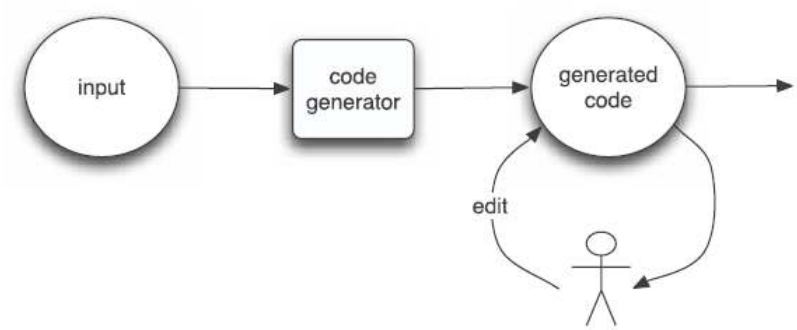
النقاط السابقة يجب أن تأخذ بعين الإعتبار عند تصميم أي (Code Generator).

[11] Code Generator in Action (JACK HERRINGTON) Chapter 1

### 4-2- تصنيف مولدات الشيفرة [12]:

يمكن تصنيف مولدات الشيفرة (code generator) حسب أسلوب التوليد إلى صنفين: مولدات فعّالة (active) ومولدات سلبية (passive).

#### 1. مولّد الشيفرة السلبي (Passive style: one-time code generation) :



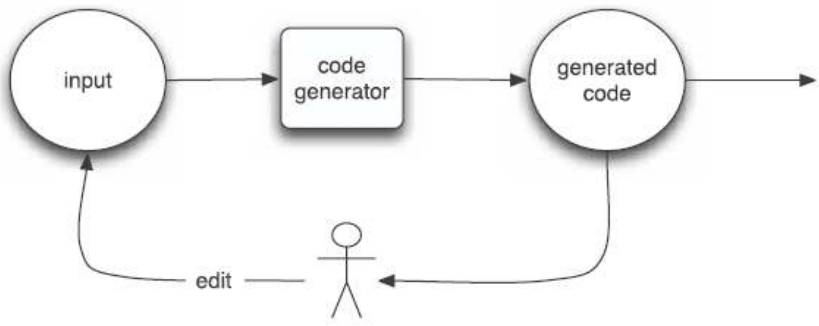

In passive code generation, code is generated once and then modified by the developer.

**(الشكل 4-2 مولد الشيفرة السلبي)**

يقوم بتوليد الشيفرة لمرة واحدة و يكون للمطوّر الحرية في التعديل على ملف الخرج، حيث تعتبر مرحلة توليد الشيفرة كنقطة بداية فقط بعد ذلك يتم دمج ملف الخرج مع المشروع، إذا قمنا بتعديل ملف الدخل و قمنا بإعادة توليد الشيفرة، أية تعديلات قمنا بها سوف تضيع.

هذا النوع من المولدات يقوم بتوليد الهيكل (structure) فقط، لذلك مرحلة توليد الشيفرة تبدأ بعد الإنتهاء من بناء النظام.

#### 2. مولد الشيفرة الفعّال (Active style: integrated code generation):




With active code generation, the developer edits only the input to the generator, not the generated code.

**(الشكل 4-3 مولد الشيفرة الفعال)**

[12] *XDoclet in Action (CRAIG WALLS NORMAN RICHARDS) CHAPTER 1*

في هذا النوع بإمكان المطور التعديل على ملف الخرج فقط عن طريق التعديل على ملف الدخل و إعادة توليد الشيفرة مرة أخرى (Regeneration) .

لذلك بإمكاننا دمج مولد الشيفرة مع مراحل تطوير النظام (development cycle)، حيث يمكن توليد الشيفرة في أي مرحلة من مراحل التطوير، و كلما حسّنا في تصميم النظام يمكن إعادة التوليد (Regeneration).

## 4-3- أشكال ملفات الدخل لمولد الشيفرة (Code generation input sources) :[13]

إن شكل ملف الدخل يؤثر بشكل كبير على طبيعة مولد الشيفرة، هل هو فعّال أو سلبي. سنعرض بشكل موجز أشكال ملفات الدخل من أجل مولد الشيفرة الفعّال.

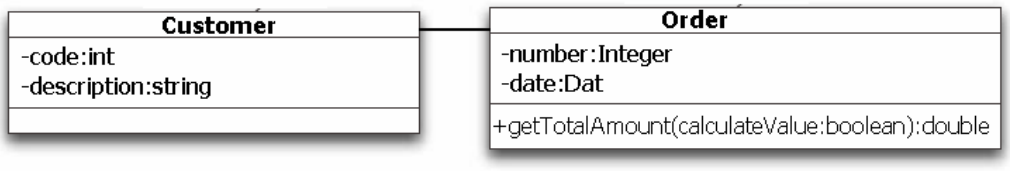
### 1. التوصيف كدخل (Model as input source) :

كما نعلم في الوقت الحالي أصبحت للغة التوصيف المعيارية (UML) دور بارز في مرحلة تصميم النظام، حيث توفر معايير لتمثيل الصفوف و هرميتها، الاتصال بين الأغراض، هيكلية النظام .

مخططات الـ (UML) تعتبر خيار جيد من اجل توليد هيكلية النظام، لكنها تحتاج إلى أدوات نمذجة (Visualization tools)، و التي تعتبر مكلفة و بالإضافة إلى صعوبة التنسيق بين الفريق، حيث يتطلب وجود نفس أداة النمذجة عند كل أعضاء الفريق.

المثال التالي يمثل صف زبون (customer) وصف يمثل طلب الشراء للزبون (order).

باستخدام أدوات النمذجة (Modeling tools) يمكننا الحصول على شكل مشابه للشكل:




**(الشكل 4-4)**

[13] *XDoclet in Action (CRAIG WALLS NORMAN RICHARDS) CHAPTER 1*

تسمى مولدات الشيفرة التي تستخدم مخططات الـ (UML) كدخل بمولدات الـشيفرة المقادة بالتوصيف (Model-driven).

**2. ملفات البيانات كدخل (Data files as Input):**

حيث يمكن أن تأخذ هذه الملفات عدة أشكال، يمكن أن تكون بيانات مفصولة بفاصلة (CVS Comma Separated Value) أو (XML file). يعتبر هذا النوع أفضل بسبب إمكانية تبادل ملفات الدخل بدون الحاجة إلى أية أدوات أخرى .

يعتبر توصيف الـ (XML) الأكثر رواجاً لأنه يتبع معيارية معينة في التوصيف (standardization)، و بسبب دعمه في كل أنظمة و بيئات العمل (و هو التوصيف المستخدم في النظام المدروس).

تسمى مولدات الشيفرة التي تستخدم ملفات الـ (Data File) كدخل بمولدات الشيفرة المقادة بالبيانات (Data Driven).

كمثال : الملف المبين في (الشكل 4-5) يمثل الصفين Customer و Order و علاقة association بين الصفين.

**3. ملف شيفرة كدخل (Source file as input):**

حيث يكون ملف الدخل يحتوي على الشيفرة (source code) و إشارات مميزة (special tags). ندعو هذه المولدات بالمولدات المقادة بالشيفرة (code-driven).

كمثال لدينا الـ (JavaDoc) : تأخذ كدخل ملف الجافا الذي يحتوي على الشيفرة مع التعليقات (comments) و تقوم بتوليد (HTML Code) يمثل التوثيق (documentation) لهذا الملف.

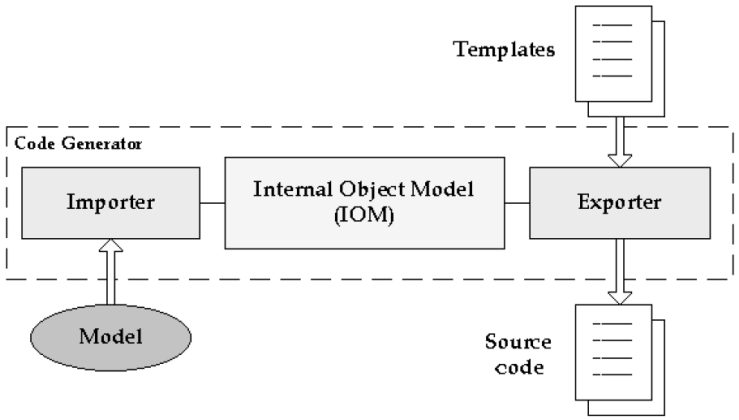
```
<?xml version="1.0" encoding="UTF-8"?>
<Content>
     <Class name="Customer" stereotype="">
          <Attribute name="code" type="integer"/>
          <Attribute name="description" type="string"/>
     </Class>
     <Class name="Order" stereotype="">
          <Attribute name="number" type="integer"/>
          <Attribute name="date" type="date"/>
          <Operation name="getTotalAmount"
     returnType="double">
               <Parameter name="calculateVAT"
          type="boolean"/>
          </Operation>
     </Class>
     <Association name="">
          <Role class="Order" multiplicity="*" name=""
          type="start"/>
          <Role class="Customer" multiplicity="1" name=""
          type="end"/>
     </Association>
</Content>
```

**(الشكل 4-5)**

## 4-4- بنية مولد الشيفرة :[14]

يتألف مولد الشيفرة من ثلاثة مركبات : (الشكل 4-5)

- المستورد (Importer): يقرأ نموذج الدخل (Model as Input) و يقوم بترجمتـــه إلى بنية وسيطة مستقلة عن منصة العمل ندعوها بنموذج الأغراض




**(الشكل 4-6 بنية مولد الشيفرة)**

[14] http://www.ftponline.com/javapro/2004_03/online/code_gnaccarato_03_03_04/page2.aspx

- توصيف الغرض (Object Model): يمكن تطويره ليقبل أي نوع من نماذج الدخل.
- نموذج الأغراض الداخلي (Internal Object Model IOM): و هو تمثيل لنموذج الدخل في المرحلة الأولى، يوصف صفوف النظام و العلاقات .
- المصدّر (Exporter): يقوم باستخدام المعلومات الموجودة في نموذج الأغراض لتوليد ملف الخرج المناسب إعتماداً على القالب، حيث يوجد قالب لكل خرج معين (source code)، مثلا قالب لتوليد (Java) ، و قالب لـ (C#) ... إلخ.

**4-5- شرح آلية العمل:**

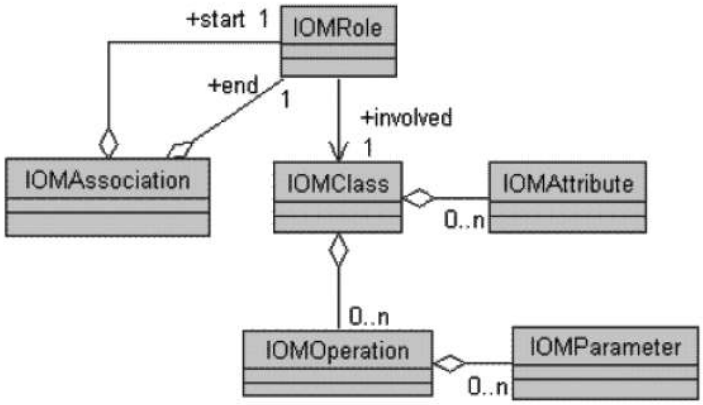
يتم استيراد ملف دخل له أحد الأنواع السابقة، ولنفرض أن هذا الملف عبارة عن ملف (XML) له الصيغة كما في (الشكل 4-7) .

يقوم الـ(IOM module) بعملية تحويل محتويات الملف إلى نموذج أغراض داخلي، الشكل التالي يوضح مخطط الـ (UML) الذي يستخدم من أجل تمثيل نموذج الأغراض الداخلي (IOM) :

```
<?xml version="1.0" encoding="UTF-8"?>
<Content>
     <Class name="Customer" stereotype="">
          <Attribute name="code" type="integer"/>
          <Attribute name="description" type="string"/>
     </Class>
     <Class name="Order" stereotype="">
          <Attribute name="number" type="integer"/>
          <Attribute name="date" type="date"/>
          <Operation name="getTotalAmount"
     returnType="double">
               <Parameter name="calculateVAT"
          type="boolean"/>
          </Operation>
     </Class>
     <Association name="">
          <Role class="Order" multiplicity="*" name=""
          type="start"/>
          <Role class="Customer" multiplicity="1" name=""
          type="end"/>
     </Association>
</Content>
```

**(الشكل 4-7 بنية ملف الدخل)**

بعد ذلك يقوم المصدّر (Exporter) بتوليد ملفات خرج مختلفة (ملفات C#، java، ...) بالاعتماد على مخطط التمثيل الداخلي.




**(الشكل 4-8 نموذج الأغراض الداخلي)**

## الخلاصة:

قمنا في هذا الفصل بعرض تقنية مولد الشيفرة (code generator) كأداة مساعدة في تطوير البرمجيات، حيث أنه كما ذكرنا يوفر تجريد للنظام و سرعة في توليد الشيفرة ويحسن الإنتاجية ويعطي الوقت الكافي لعملية التصميم، كما أنه يمكن استخدامه في توليد ملفات خرج تحوي شيفرات بلغات مختلفة موافقة لنفس ملف الدخل.

سنقوم في الفصل القادم بدمج جميع التقنيات التي ذكرناها في الفصول السابقة كنظام متكامل يساعد مهندسي البرمجيات خلال العمل.

الفصل الخامس

# التطبيق العملي

# PRACTICAL APPLICATION

محتويات الفصل:

ستتعرف في هذا الفصل على النظام المقترح:

- بنية العامة النظام المقترح.
- الواجهات الرئيسة للتطبيق.
- كيفية استخدام التطبيق من البداية إلى النهاية.

ليس هناك أمر عجيب، كل ما عليك أن تقوم به هو أن تضغط على المفاتيح المناسبة في الوقت المناسب وستعمل الأداة تلقائياً.

-يوهان سيباستيان باخ-

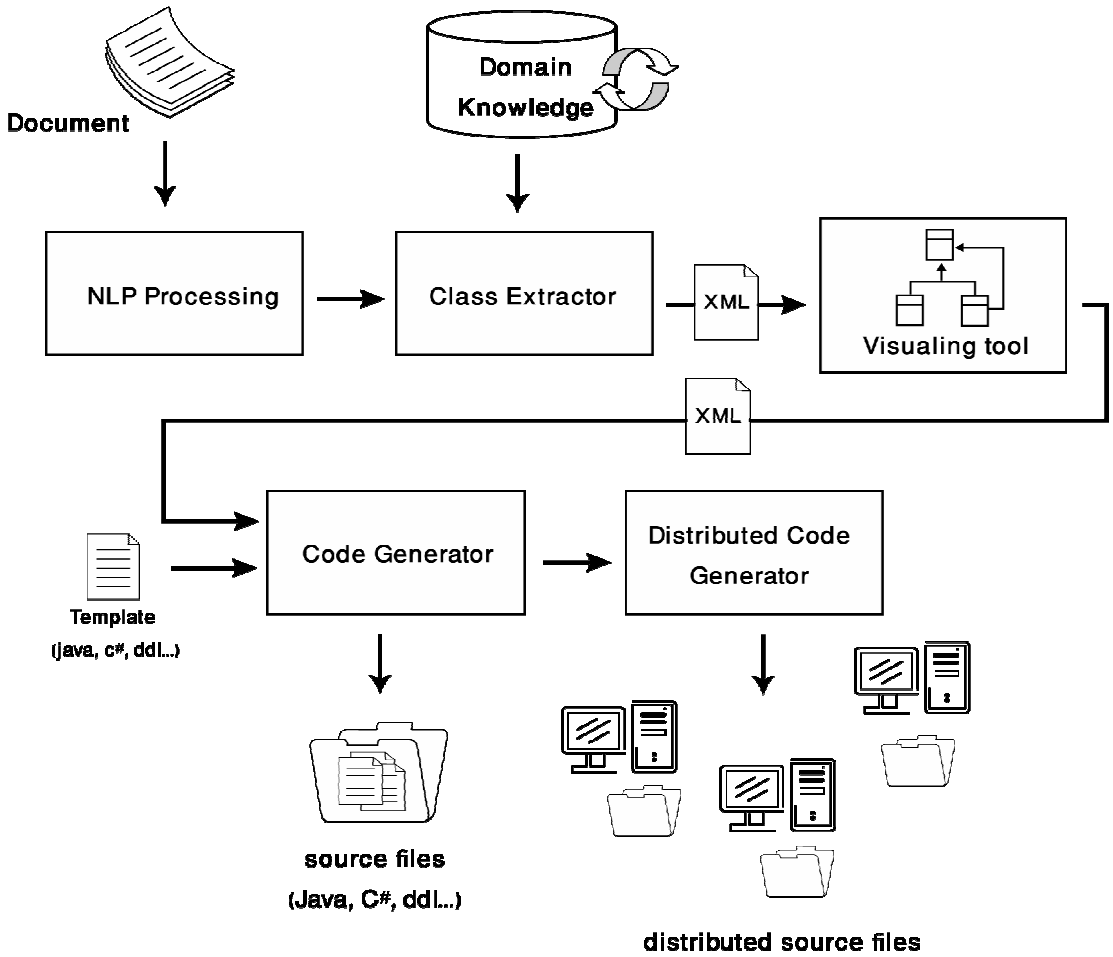
**مقدمة:**

في هذا الفصل سنقوم بعرض التطبيق المقترح لأتمتة تطوير البرمجيات، حيث يشمل هذا التطبيق التقنيات السابقة التي تحدثنا عنها من تحليل للغة العربية، ثم توصيف للنظام المدروس بلغة نمذجة وهي (UML) ثم توليد الشيفرة المحلية (Local) أو الموزعة (Distributed).

**5-1- البنية العامة للنظام المقترح:**

يتألف النظام المقترح من الوحدات (Component) التالية:

محلل النص (NLP Processing)، مكتشف الصفوف (Class Extractor)، أداة تمثيل الصفوف (Class Diagram Visualization tool)، مولد الشيفرة ( Code Generator)، مولد شيفرة التطبيقات الموزعة (Distributed Code Generator).




**(الشكل 5-1 بنية النظام المقترح)**

**1. محلل النص (NLP Processing):**

مهمة هذه المكونة هي تحليل معجمي النص، دخل هذه المرحلة هي وثيقة المتطلبات التي تشرح المشكلة، ولا يشترط وجود بنية محددة أو أية شروط في طريقة الكتابة، مراحل المعالجة هي كما يلي:

1. (Tokenization) : تقوم بتقسيم النص إلى (tokens).
2. (Sentence splitting) : يقوم بتقسيم النص إلى جمل.
3. (Part of speech): يقوم بتخصيص وسم (tag) لكل كلمة (اسم، فعل، صفة ...).
4. (Morphological Analysis): تقوم بإرجاع جذر كل كلمة.

خرج هذه المرحلة هو ملف يحتوي على الكلمات مع توصيفها، أي له الشكل التالي: (الكلمة، وسمها (tag)، الجذر)

**2. مكتشف الصفوف (Class Extractor):**

و مهمة هذه المكونة هي ترشيح الصفوف و العلاقات المقترحة لبناء التطبيق، دخل هذه المرحلة هو الملف الناتج عن محلل النص.

مراحل المعالجة هي كما يلي:

1. تمثيل الجمل على شكل شجرة إعراب أي استنتاج شجرة الإعراب (parsing tree).
2. تطبيق خوارزمية (TCM) و الاعتماد على قاعدة المعرفة لاستخراج الصفوف و العلاقات.

خرج هذه المرحلة هو ملف (XML) الذي يوصف الصفوف و العلاقات المقترحة بينها.

**3. أداة رسم الصفوف (Class Diagram Visualization Tool):**

و مهمة هذه المكونة هي تمثيل الصفوف (Class Diagram) المستنتجة من مكتشف الصفوف، و الموصفة بواسطة ملف الـ (XML)، هذه الأداة تحتوي على أدوات تعديل على مخطط الصفوف، مثل إضافة و حذف الصفوف و العلاقات.

خرج هذه المرحلة هو ملف (XML) مشابه لملف الدخل لكنه معدل حسب وجهة نظرة المحلل.

**4. مولد الشيفرة (Code Generator):**

يقوم بتوليد ملفات الشيفرة، اعتماداً على القالب الذي نختاره، مثلاً نختار قالب (java) لتوليد (java source file)، وقالب (DDL) لتوليد هيكل قاعدة البيانات (DDL script)... و هكذا.

**5. مولد شيفرة التطبيقات الموزعة (distributed code generator):**

تستخدم هذه المكونة كخيار إضافي لتوليد الشيفرة اللازمة من أجل توزيع التطبيق على عدة أجهزة، (RPC Layer)، مثل (RMI).

## 5-2- تمثيل الأغراض داخل النظام:

كما ذكرنا سابقاً أننا اعتمدنا لغة التوصيف المعيارية (XML) لترميز أغراض النظام كونها لغة معيارية مدعومة من معظم لغات البرمجة، وكونها توفر إمكانية توصيف الـ (Meta Data) و هذا يسهل تبادل هذه الملفات بين الأدوات المساعدة المختلفة.

الملف التالي يظهر كيفية ترميز صف :

```
<class name="Class1">
        <Attribute name='att1" type="string" />
        <Attribute …… />
</class>
```

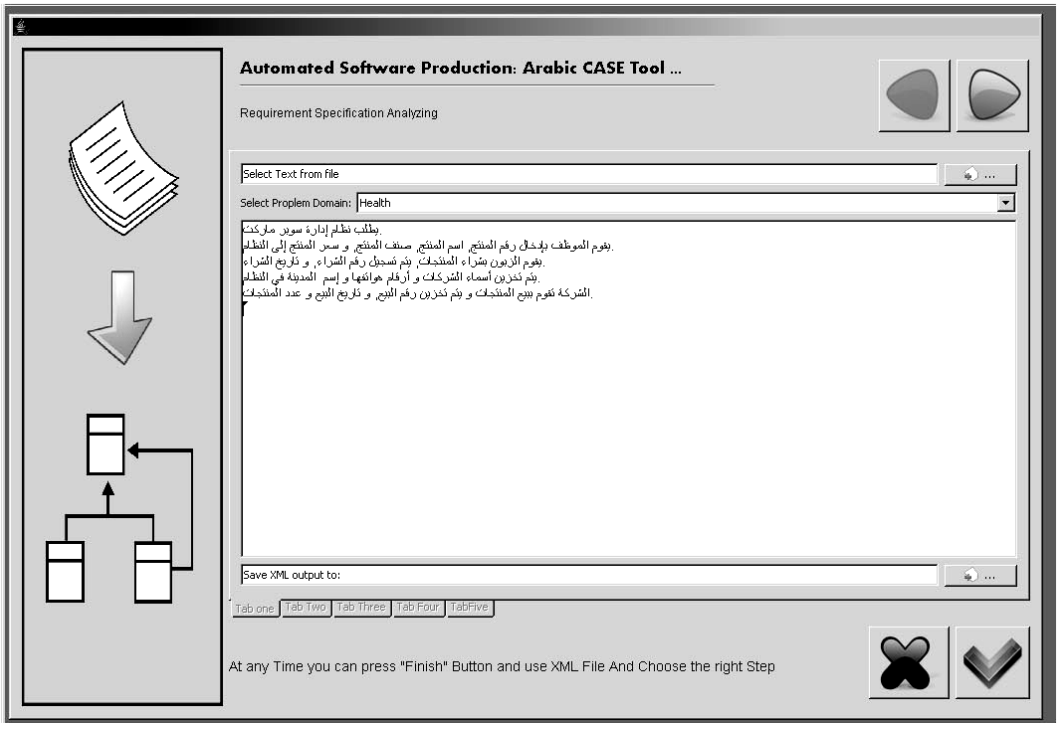
و الملف التالي يظهر كيفية تمثيل علاقة:

```
<Association name="assoc">
        <Role type="start" class="Class1" />
        <Role type="end" class="Class2" />
</Association>
```

## 5-3- الواجهات الرئيسية للتطبيق :

### واجهة تحليل النظام:

عند تشغيل البرنامج تظهر للمستخدم نافذة تسمح له بتحميل ملف مكتوب (وثيقة متطلبات) باللغة العربية، و عند اختياره للملف تظهر محتوياته داخل المساحة المخصصة للكتابة كما في الشكل:



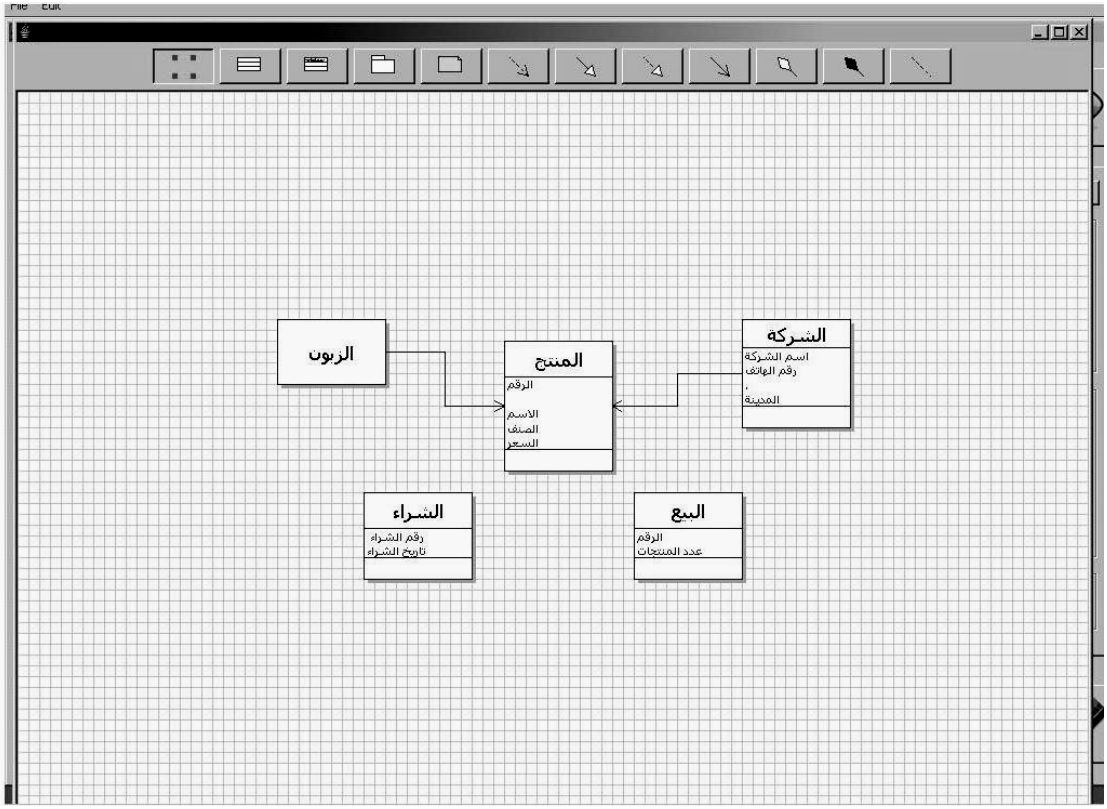

**(الشكل 5-2 واجهة تحليل النظام)**

يترتب على المستخدم اختيار مجال المشكلة من القائمة، ففي مثالنا نختار مجال مشكلة الـ (market).

يقوم النظام بتحليل الملف الذي قام المستخدم بإختياره، و يكون قد تم تنظيم محتويات الملف بشكل منتظم و تم استنتاج الصفوف و العلاقات (المخطط المفاهيمي) تمهيداً للمرحلة الثانية.

### واجهة تمثيل النظام (Visualization step):

المرحلة التالية لمرحلة التحليل هي مرحلة تمثيل للصفوف المقترحة، حيث تظهر للمستخدم نافذة رسم الصفوف،و يتم إظهار الصفوف المقترحة لهذا النظام ، تحتوي هذه النافذة على أدوات رسم و تعديل وحذف الصفوف كما في الشكل:

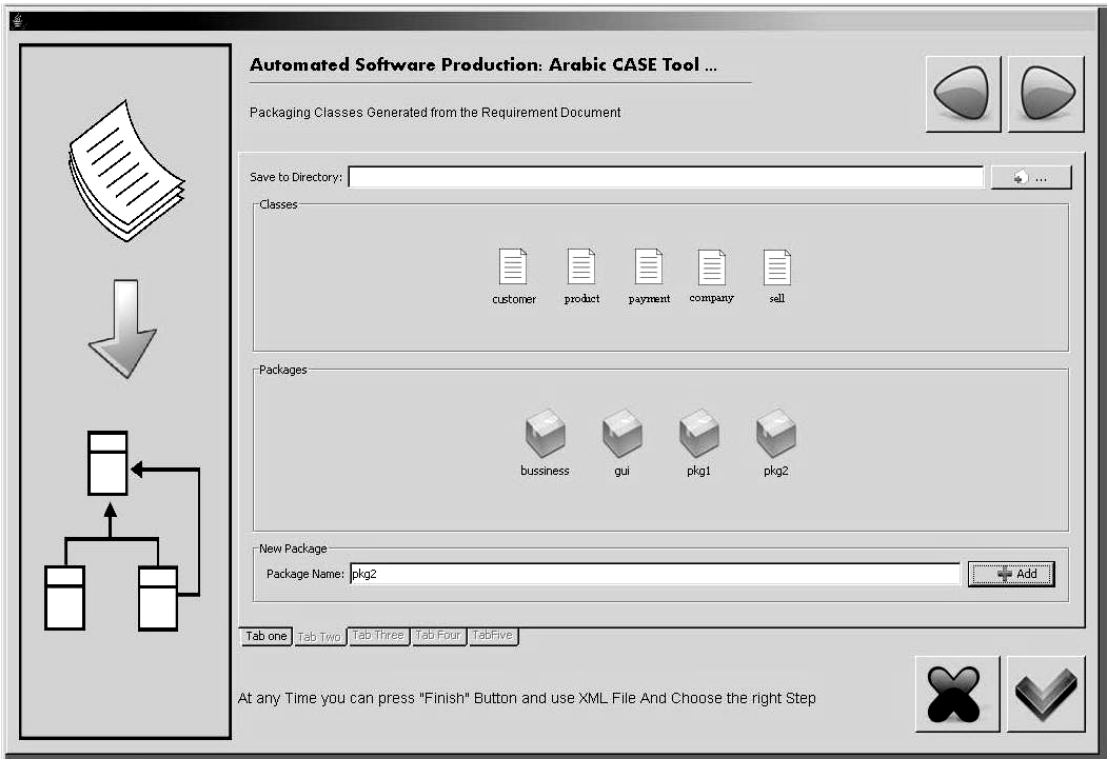



**(الشكل 5-3 واجهة تمثيل النظام)**

بعد قيام المستخدم بإجراء التعديلات و قبوله بها، يقوم بإغلاق هذه النافذة لينتقل إلـــى الخطوة الثالثة و هي التحزيم ، أي وضع كل مجموعة من الصفوف في حزمة (pacakge) .

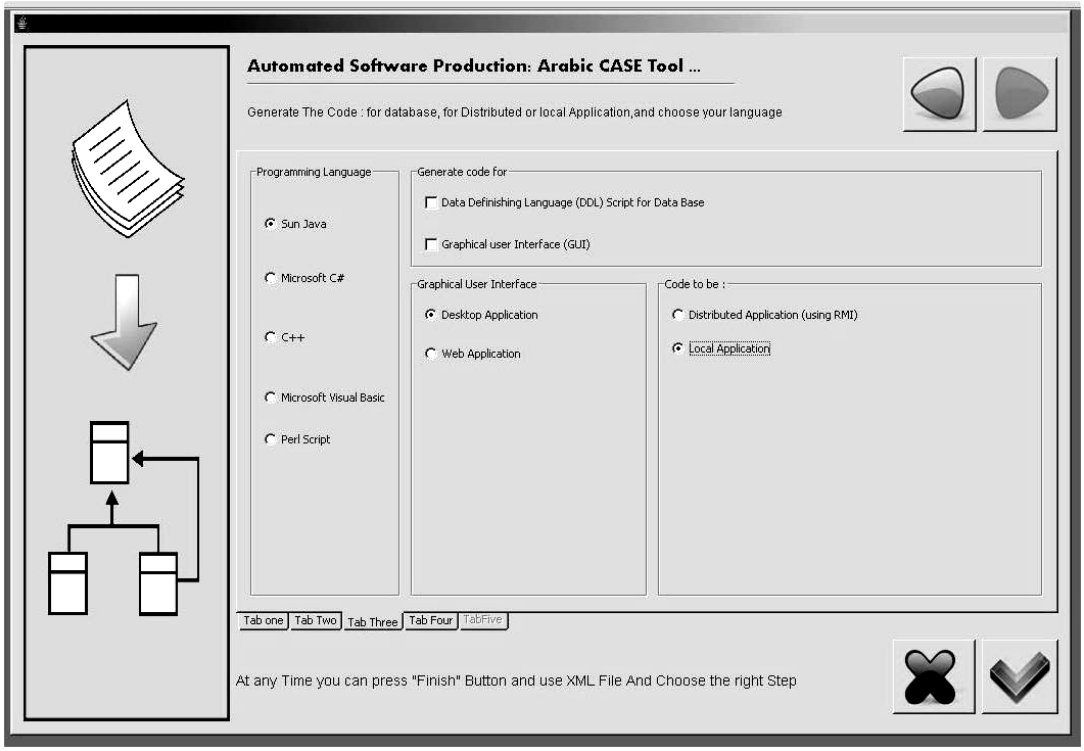
**نافذة التغليف ضمن حزم:**

حيث يتم عرض الصفوف الناتجة في الخطوة السابقة، و يمكن للمستخدم إضافة حزمة عن طريق الضغط على زر (Add) ، بعد إضافة الحزم يقوم بسحب الصفوف إلـــى داخـــل الحزم عن طريق السحب و الإفلات، ثم يختار المكان الذي سوف يتم تخزين ملفات الخـــرج بالضغط على (save to directory)، و بعد ذلك نضغط السهم للإنتقال إلى الخطوة الرابعة و هي توليد ملفات الشيفرة.




**(الشكل 5-4 واجهة وضع الصفوف ضمن الحزم)**

## واجهة توليد ملفات الشيفرة:



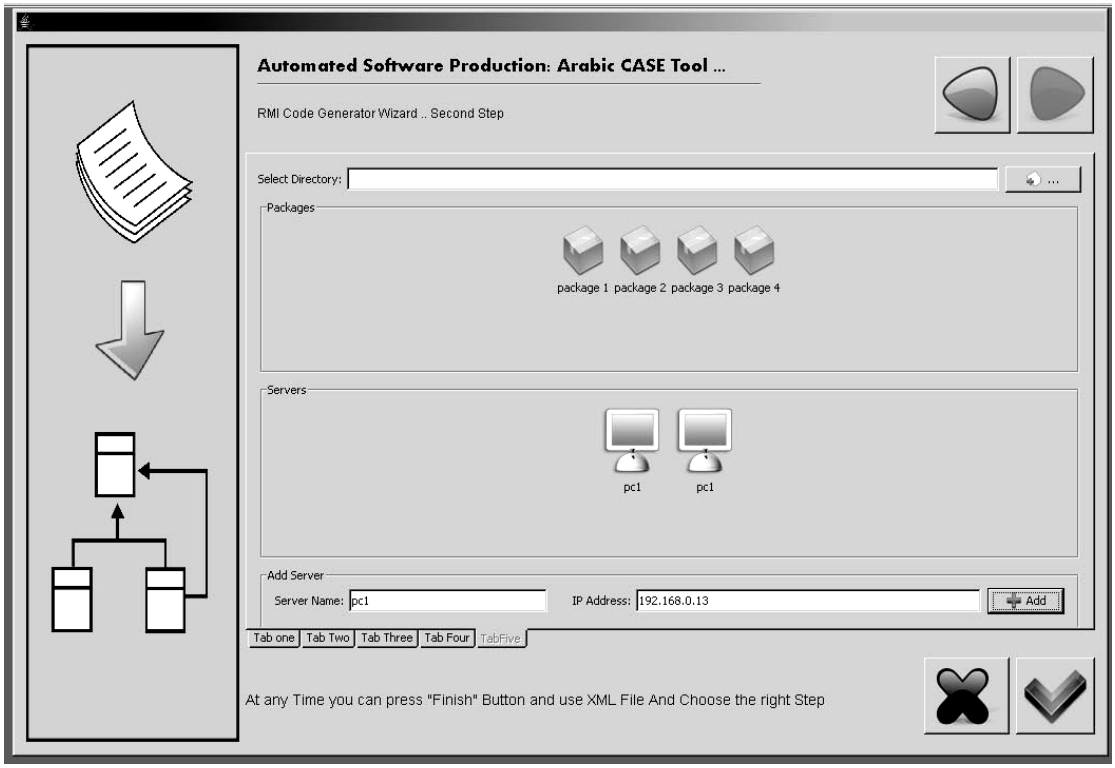

**(الشكل 5-5 واجهة توليد ملفات الشيفرة)**

و في هذه النافذة يقوم المستخدم بإختيار لغة البرمجة السمتهدفة (java, c#, ...)، و هناك امكانية لتوليد ملفات بنية قاعدة البيانات (ddl script)، أو توليد واجهات مرئية (GUI interface)، و اختيار نوع التطبيق ، هل هو تطبيق عادي، أو تطبيق ويب، وبإختيار الخاصية (Local Application) يتم بناء تطبيق محلي (Local) وتكون العملية قد انتهت، بالضغط على زر الإنهاء يكون المعالج قد قام بتوليد ملفات الخرج.

أما إذا اختار مولد التطبيقات الموزعة (Distributed Application)، فينتقل للمرحلة الرابعة و هي تحديد الأجهزة التي يريد توزيع التطبيق عليها.

**واجهة توزيع النظام :**

و في هذه النافذة تظهر للمستخدم جميع الحزم الموجودة، و يمكنه من إضافة جهاز لنشر التطبيق عليه و ذلك بإدخال اسم الجهاز في مربع النص (Server Name) و إدخال رقم الجهاز في مربع النص (IP Address)، ثم يقوم بسحب كل حزمة إلى الجهاز المطلوب.




**(الشكل 5-6 واجهة توزيع النظام)**

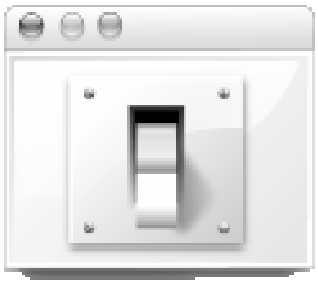
و عند الإنتهاء نضغط على زر الإنتهاء (finish) فيقوم البرنامج بتوليد الملفات الضرورية لتوزيع التطبيق.

**الخاتمة:**

نلاحظ أن الأداة المقترحة (Arabic CASE tool) قد وفرت الكثير من الوقت و العناء بالنسبة للمطور، ففي البداية قدمت له إقتراح لبنية النظام، طبعا هذا الإقتراح ليس نهائي و هو ليس إلا عبارة عن مخطط مفاهيمي (Conceptual Model)، و بهذا يكون الحاسب قد شارك في عملية الـ (Brain Storming) باقتراحه لصفوف النظام، ثم انتقلنا نحو تمثيل الصفوف و رسم مخطط النظام و تعديله إنتهاء بتوليد شيفرة التطبيق و توزيع النظام.

نلاحظ أن معظم اهتمام المطور قد انصب في عملية التصميم، و بالتالي تتحسن جودة المنتج (Quality)، و تؤدي إلى تسريع بناء النظام.

# الخاتمة
# والتطلعات المستقبلية



أول العلم الصمت، والثاني الاستماع، والثالث الحفظ،
والرابع العمل، والخامس نشره.
- الأصمعي -

قدمنا في هذه الدراسة محاولة لبناء أداة مساعدة لمهندسي البرمجيات تدعم اللغة العربية (Arabic CASE Tool).

هذه الأداة تستخدم تحليل اللغات الطبيعية (NLP) من أجل تحليل وثيقة المتطلبات المكتوبة باللغة العربية و تحويلها إلى نموذج منتظم. يستخدم هذا النموذج كدخل لأداة اكتشاف الصفوف (Class Extractor) من أجل اكتشاف الصفوف (Classes)، الصفات (Attribute)، و العلاقات بين الصفوف (Relations) أي استنتاج التصور المبدئي للنظام (Conceptual Model).

النموذج النهائي هو ملف (XML) يوصف الصفوف والعلاقات، هذا النموذج يستخدم كدخل مع أداة رسومية (Graphical CASE tool) التي تستخدم لرسم مخططات الصفوف (UML Class Diagram)، وهذه الأداة توفر لمطور النظام أدوات إدراج و حذف و تعديل الصفوف و العلاقات للوصول إلى أفضل تصميم يريده.

ثم ننتقل بعد ذلك من مرحلة النمذجة إلى مرحلة توليد الشيفرة (source code files)، حيث يمكن للمطور تحديد اللغة البرمجية التي يريد تحقيق نظامه بها، وتوليد ملفات تعريف قواعد البيانات (DDL Script)، حيث يتم توليد الهيكلية البدائية لتحقيق النظام.

و توفر الأداة أيضاً إمكانية توزيع التطبيق، أي توليد طبقة الـ (RPC layer) من أجل توزيع التطبيق على عدد من الأجهزة الموجودة على الشبكة.

هنالك الكثير من العمل مازلنا بحاجة لإنجازه ليتم الوصول إلى أداة متكاملة تساعد المبرمج في جميع مراحل تطوير البرمجية.

## الآفاق المستقبلية لتطوير الأداة:

يمكن تحديد تصور لتطوير الأداة للحصول على أداة مساعدة متكاملة بالتالي :

- تطوير خوارزمية تحليل النص لإعطاء جودة أعلى في التحليل حيث أنه كلما ازدادت دقة النتائج في هذه المرحلة كلما أعطت دقة أكبر في مجال استخراج الصفوف في المرحلة التالية.

- تطوير الخوارزمية التي تم اعتمادها في استخراج الصفوف و العلاقات و تجدر الإشارة أن هذه المرحلة هي الفيصل فإنه كلما زادت القواعد الحدسية التي تنظم هذه المرحلة و كلما ازدادت دقة و تخصص مجالات المشكلة (Domain Problem) المخزنة ضمن قاعدة المعرفة (Domain Knowledge) التي يعتمد عليها النظام في استخراج صفوف المجال كلما حصلنا على مخطط مفاهيمي (Conceptual Model) يقترب من النص المراد تحقيقه.
- إيجاد أداة لتوليد صفوف الربط مع مصادر البيانات حسب مصدر البيانات المراد اعتماده في عملية التخزين (مثلاً للربط مع قاعدة بيانات Oracle، أو mysql).
- إدراج مولدات شيفرة (Code Generators) للغات أخرى مثل (C++، C#،..) و ذلك لتمكين المطور من اختيار اللغة التي يريد بناء التطبيق بها.
- يمكن اعتماد أدوات لفحص أداء البرمجية حتى يمكن التأكد من مستوى أداء كل مكون من مكونات البرمجية الناتجة ويمكن أن تكون هذه الأداة بشكل رسومي (Graphically).

لقد حاولنا في هذه الدراسة بذل ما نستطيع من جهد لدعم اللغة العربية في صناعة البرمجيات، راجين المولى عز و جل أن نكون قد وفقنا في هذا العرض.

والله من وراء القصد

والحمد لله رب العالمين

حمص في

23/جمادى الثانية/1428

8/تموز/2007

# المـــــــراجـــــع


✓ **مراجع الفصل الأول:**

- النمذجة المرئية في التحليل و التصميم غرضي التوجه (الأستاذ بلال اسماعيل)
- Modern Systems Analysis and Design (Third Edition)
- http://en.wikipedia.org/wiki/Computer/aided_software_engineering

✓ **مراجع الفصل الثاني:**

- http://www.alarabiyah.ws/showpost.php?postid=28
  (موقع صوت العربية)
- Light Stemming and Co-occurrence Analysis: Larkey, Ballesteros, Connel , 2003
- Lexical Analysis of Inflected Arabic Words using Exhaustive Search of an Augmented Transition Network, Ahmed Rafea and Khaled Shaalan

✓ **مراجع الفصل الثالث:**

- Using Natural Language in Software Development (Nik Boyd)
- CM-Builder: An Automated NL-based CASE Tool (H.M. Harmain, R. Gaizauskas)
- A Taxonomic Class Modeling Methodology for Object-Oriented Analysis Il-Yeol Song, Kurt Yano, Drexel University, USA

✓ **مراجع الفصل الرابع:**

- http://today.java.net/pub/a/today/2004/05/12/generation1.htm
- Code Generator in Action (JACK HERRINGTON)
- XDoclet in Action (CRAIG WALLS NORMAN RICHARDS)
- http://www.ftponline.com/javapro/2004_03/online/code_generator_03_03_04/page2.asp