# Model-driven Development for Adapting Question Answering Systems to Restricted Domains

**Katia Vila, Jose-Norberto Mazón and Antonio Ferrández**
Department of Software and Computing Systems
University of Alicante, Spain
{kvila,jnmazon,antonio}@dlsi.ua.es

*A Question Answering (QA) system must provide concise answers from large collections of documents to questions stated by the user in natural language. However, although many QA systems for open domain exist, its adaptation for restricted domains (such as those of healthcare, agriculture, transportation, or science) is a far from trivial task. The principal problem is that domain experts ask more precise questions (and expect more precise answers), including specific terminology, and this is costly to integrate into a QA system. To overcome this drawback, this paper presents an innovative approach based on model-driven software development. It uses restricted-domain resources to automatically and effortlessly adapt open-domain QA systems in order to make them useful in restricted-domain scenarios. Finally, a set of experiments has been carried out to show the approach's applicability.*

*Keywords: Question Answering systems, Model-Driven Software Development*

*ACM Computing Classification System: H.3.4 Systems and Software, I.2.7 Natural Language Processing*

## 1. INTRODUCTION

Question Answering (QA) is defined as the task of searching for and extracting the text that contains the answer for a specific question, stated in natural language, from a collection of text documents or a corpus (Mollá and Vicedo, 2007). QA systems can be classified in two types, depending on the application domain: Open-Domain Question Answering (ODQA) or Restricted-Domain Question Answering (RDQA) systems. While the former is concerned with a wide spectrum of questions (e.g. *who is the president of Spain?*), the latter is properly adapted to a particular area (e.g. *what antipyretic is recommended to patients with a risk of gastrointestinal bleeding?* in a medical domain), thus obtaining more precise results regarding a specific topic.

The development of ODQA systems has been stimulated by evaluation forums such as TREC (Text REtrieval Conference, http://trec.nist.gov/) and CLEF (Cross-Language Evaluation Forum, http://clef-campaign.org/). From these forums, it will be observed that most QA systems present a common architecture (as shown in the *Searching Module* of Figure 1). This architecture consists of three different sequential phases: (i) question analysis for understanding the question by detecting the expected answer type and extracting the significant keywords; (ii) these keywords are used by an Information Retrieval (IR) system in order to select and retrieve the relevant passages or documents; and (iii) finding and extracting the expected answer by using natural language

*Manuscript received: 1 May 2011*
Communicating Editor: Rafael Valencia García

processing tools (such as PoS tagger, syntactical parser, entity annotator, semantic role parser, etc.) to analyze this set of passages or documents. If this architecture is analyzed, then it is obvious that the phases of question analysis and answer extraction are dependent on the knowledge from the domain which is usually included in patterns. Note that for the purpose of this paper, we shall refer to all the possible strategies for detecting relationships between elements in both question and answer (e.g., logic forms, regular expressions, syntactical relations and so forth) as patterns. Moreover, it is important to note that within the question analysis phase (see Figure 1) it is crucial to determine the semantic type of the answer or Expected Answer Type (EAT) by means of a predefined taxonomy (also known as question hierarchy (Li and Roth, 2006) or question ontology (Metzler and Croft, 2005)). A correct specification of the EAT taxonomy implies an accurate EAT detection, thus reducing the search space of possible answers and providing a more precise answer in a more efficient manner (Li and Roth, 2006; Hovy, Hermjakob and Ravichandran, 2002). Indeed, more than 36.4% of QA errors are related to an incorrect EAT detection of the question (Moldovan, Pasca, Harabagiu and Surdeanu, 2003).

Bearing these considerations in mind, for ODQA systems to perform well in restricted domains both patterns and EAT taxonomy must be adapted. However, two main problems arise in the current approaches dealing with this adaptation: (i) manually tuning QA patterns (Peñas, Forner, Sutcliffe, Rodrigo, Forascu, Alegria, Giampiccolo, Moreau and Osenova, 2009) and EAT taxonomies (Sekine, Sudo and Nobata, 2002; Hovy *et al*, 2002; Li and Roth, 2006) for restricted domains requires a huge effort in time and cost owing to the inherent complexity of the concepts provided by these domains (Mollá and Vicedo, 2007); and (ii) defining restricted-domain QA patterns and EAT taxonomies by analyzing potential questions to be answered (Kosseim and Yousefi, 2008) is not realistic, since restricted-domain questions are highly complex and difficult to acquire. For example, IBM's computer system Watson (http://www-03.ibm.com/innovation/us/watson/) has been quite popular in the media thanks to its participation and success in the quiz show Jeopardy! (http://www.jeopardy.com/). The sources of information for Watson include encyclopedias, dictionaries, thesauri, newswire articles, databases, taxonomies, and ontologies; and it is based on an ODQA system named DeepQA (Ferrucci, Brown, Chu-Carroll, Fan, Gondek, Kalyanpur, Lally, Murdock, Nyberg, Prager, Schlaefer and Welty, 2010). Recently, there is an increasing necessity for Watson's capabilities to be adapted to other domains, e.g., the medical domain, as a clinical decision support system to aid the diagnosis and treatment of patients. This adaptation is not trivial because it implies that DeepQA would refer to new sources of information and the development of new patterns and EAT taxonomies is required.

To overcome these drawbacks we propose an approach with which to design EAT taxonomies for restricted domains, and to obtain existing patterns from an ODQA system and adapt them by using knowledge resources from a specific domain in a systematic and comprehensive manner. Our approach is based on model-driven software development (Selic, 2003) which has proved useful for defining and managing several kinds of software models in an easy and well-structured manner with a high degree of automatization (Lasheras, Valencia-García, Fernández-Breis and Toval, 2009). Our initial hypothesis is that the adaptation of ODQA systems to a new domain can be seen as a model-driven software development scenario, in such a way that existing QA patterns and knowledge from the domain are captured in a model which will guide the derivation of the new patterns and the EAT taxonomy for the restricted domain, thus significantly reducing the amount of manual labor required. Moreover, a set of experiments has been conducted with the objective of validating our approach and showing its applicability in a real-world case study in the agricultural domain.

The remainder of this paper is structured as follows. Section 2 presents related work. Section 3 describes our model-driven approach for adapting QA systems to restricted domains. Section 4 validates our approach by means of a set of experiments. Finally, Section 5 shows our conclusions and future work.

## 2. RELATED WORK

Some of the current approaches for adapting both patterns and EAT taxonomy are described as follows.

### 2.1 Adapting QA Patterns for Restricted Domains

The process of adapting existing QA systems to a specific domain is currently done manually by using linguistic resources (Peñas *et al*, 2009; Roger, Vila, Ferrández, Pardiño, Gómez, Puchol-Blasco and Peral, 2008), which is thus costly and prone-to-fail. Potential questions to be answered are traditionally analyzed (Kosseim and Yousefi, 2008). This is feasible for open domains, in which repositories of questions are easily acquired from the Web, but difficult to apply in restricted domains since sufficiently comprehensive training corpora are hard to find (Mollá and Vicedo, 2007). Moreover, there exist some QA systems based on ontologies (Valencia-García, Sánchez, Nieves and Fernández-Breis, 2011; Ferrández, Izquierdo, Ferrández and Vicedo, 2009; Lopez, Uren, Motta and Pasin, 2007) that could be easily adapted to new domains, although they do not consider textual sources, but structured information.

From our point of view, patterns should be tuned by using different kinds of knowledge resources from a specific domain. These resources are also known as KOS (Knowledge Organization Systems) which include a variety of schemes that organize, manage, and retrieve information. This term is intended to encompass all types of schemes for promoting knowledge management (Hodge, 2000), e.g., dictionaries, thesaurus, or ontologies. According to the level of detail or granularity of the knowledge they refer to, two kind of KOS exist: generic KOS (such as WordNet, http://wordnet.princeton.edu/) or the more precise domain KOS (such as Agrovoc thesaurus, http://www.fao.org/agrovoc/) for the agricultural domain). However, these KOS have their own formats and interfaces which must be unified by the QA system, thus being a costly task (Mollá and Vicedo, 2007).

Our model-driven approach overcomes this scenario since it allows question answering patterns in the ODQA system to be automatically adapted to a restricted domain from the collection of documents by integrating available KOS.

### 2.2 Adapting EAT Taxonomies for Restricted Domains

Many EAT taxonomy proposals for ODQA systems currently exist which are concerned with a wide spectrum of questions (Metzler and Croft, 2005; Li and Roth, 2006; Sekine *et al*, 2002; Hovy *et al*, 2002). Within these approaches, EAT taxonomies are manually developed from large collections of questions (from the Web, e.g. Ask.com: http://www.ask.com; Yahoo Answers: http://answer.yahoo.com or from TREC or CLEF conferences) by obtaining knowledge from WordNet. These approaches take into account the question stem or interrogative clause (e.g., What, Which, Who questions, etc.), later adding semantic knowledge to obtain more accurate answers. Of all the answer types, those having an ambiguous question stem (What, Which) are the most difficult to analyze, since they can be related to any answer type (What object, What substance, What enzyme, etc.), unlike the question stems Who, When and Where, which may correspond to person, date, and location concepts. In fact, the more ambiguous the question is the more semantic knowledge is required to specify an adequate EAT taxonomy for the application domain of the QA system. Furthermore, in spite of being hierarchical, these approaches are not sufficiently refined to be useful in restricted domains, and specific approaches for developing restricted-domain EAT taxonomies are required. For example, the EAT taxonomies defined in Ely, Osheroff, Gorman, Ebell, Chambliss, Pifer and Stavri (2000) and Sang, Bouma and de Rijke (2005) were obtained from a collection of 1001 questions asked by physicians and from 435 questions in the RSI (Repetitive Strain Injury) corpus, respectively. Both works manually develop the EAT taxonomy by using the UMLS metathesaurus as the domain KOS. In (Ferrés and Rodríguez, 2006) an EAT taxonomy of a baseline QA system is manually tuned by using domain ontology concepts and relationships. A common drawback of these approaches is that the EAT taxonomy is based on analyzing potential questions from users, which may not be feasible in real applications, since acquiring a large number of restricted-domain questions from domain experts is difficult.

Our model-driven approach again contributes towards overcoming these problems since it uses several KOS in order to automatically adapt an EAT taxonomy to a restricted domain from the collection of documents.

## 3. MODEL-DRIVEN QA SYSTEMS ADAPTATION TO RESTRICTED DOMAINS

Our model-driven approach for adapting QA systems to restricted domain is depicted on the right-hand side of Figure 1, while a common architecture for a QA system is represented on the left-hand side. Our approach is based on defining (i) a set of models containing the most relevant terms in the collection of documents, and useful concepts from different kind of KOS and QA patterns; and (ii) a set of transformations to be applied to these models in order to obtain the code of the new patterns to be used by the adapted QA system.

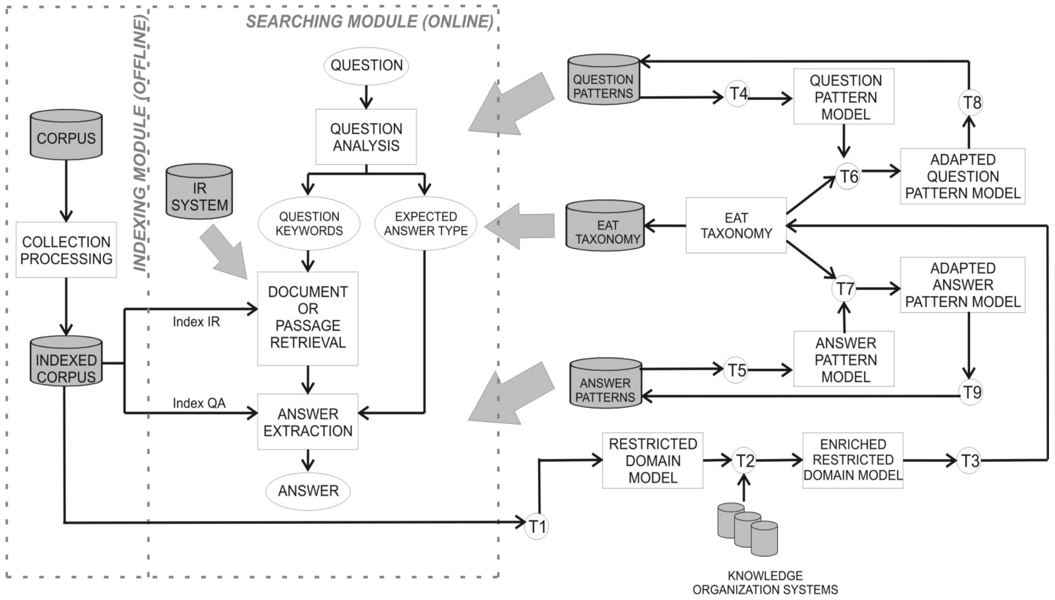


**Figure 1: Model-driven adaptation of QA systems to restricted domains**

## 3.1 Metamodels for Adapting QA Systems

Under the model-driven umbrella, and according to Kleppe, Warmer and Bast (2003), "a model is a description of (part of) a system written in a well-defined language", while "a well-defined language is a language with well-defined form (syntax), and meaning (semantics), which is suitable for automated interpretation by a computer". Therefore, on the one hand, a model must focus on those important parts of a system, thus abstracting superfluous details. On the other hand, well-defined languages can be designed by means of metamodeling (Bézivin, 2005), which provides the foundation for creating models in a meaningful, precise and consistent manner. Our metamodels for restricted domains, question patterns, and answer patterns are described as follows.

### *3.1.1 Restricted Domain Metamodel*

A *restricted domain* metamodel (see Figure 2) has been defined to create models for representing terms from a restricted-domain corpus and joining them to their corresponding concepts from the available KOS.

The core element in this metamodel is the *RestrictedDomainModel* metaclass which is useful for creating a model for a particular restricted domain. The *CorpusTerm* metaclass is useful for representing any of the terms appearing in a corpus. A metaattribute value is used to store the lemmatized value of each term. There are several lexical types of corpus terms, such as adjectives, nouns or verbs, which are represented as several subclasses of the *CorpusTerm*, i.e. *AdjectiveTerm*, *NounTerm* or *VerbTerm* metaclasses. It is worth noting that syntactical relations between these terms (which can be easily provided by a PoS tagger and a syntactical parser when the corpus is processed) are valuable for use in further steps of our approach. Specifically, the *VerbTerm* metaclass has relations to indicate which *NounTerm* can be seen as a subject or as an object. The *NounTerm* can also be related to an adjective or to other nouns. These relations are important to detect the multi-words which often appear in restricted domains (e.g. "calcium hydroxide" or "adrenal cortex hormones" in the chemical domain). Each kind of *CorpusTerm* also has its own type (originating from several *Enumerations* as shown in Figure 2). Finally, each *CorpusTerm* may also contain some semantic information (*SemanticLabel* metaclass). This semantic information can be provided by open-domain tools when the corpus is being processed in the QA task, such as semantic

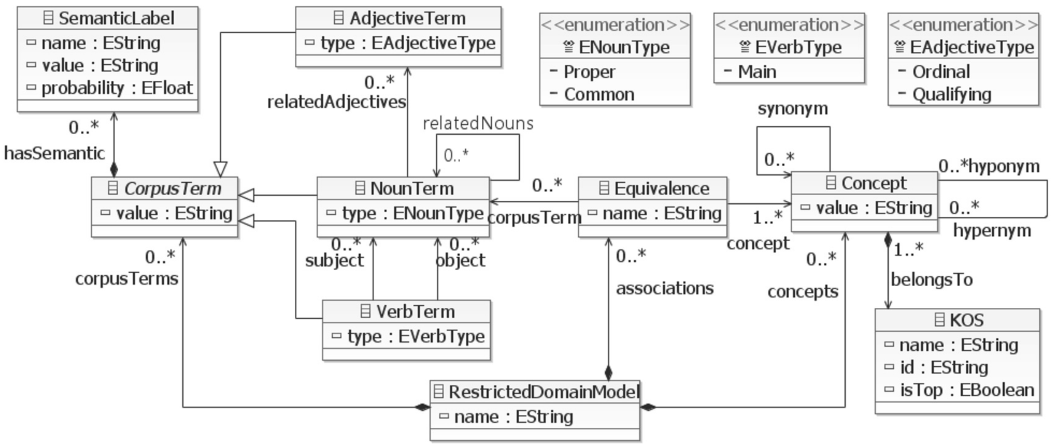


**Figure 2: Restricted domain metamodel**

role parser, name entity recognizer, temporal or numerical expressions recognizer, etc. The *SemanticLabel* metaclass indicates the *name* of the technique used to acquire the semantic information, the *value* obtained by applying these techniques and, also the *probability* of the certainty of this *value*.

Furthermore, the *Concept* and *Equivalence* metaclasses allow the elements of this *restricted-domain* metamodel to be semantically enriched with concepts and relationships from several KOS. The *Concept* metaclass refers to an element from a particular KOS. Each of these elements is represented with a value. Each concept can additionally be related to one or more concepts through relations of synonymy, hypernymy and hyponymy. Each concept may be related to more than one *KOS* for which the *name* is indicated and also an *ID* for the concept within this KOS. This metaclass has an *isTop* metaattribute that states whether it is a top concept in that KOS. Equivalences between a term and a concept can be defined: the metaclass *Equivalence* represents an association between *Concept* and *NounTerm*.

### *3.1.2 Question and Answer Patterns Metamodels*

Existing question patterns from ODQA systems must be represented in a model to enable them to be adapted to the domain represented in a restricted domain model. To this aim we have defined the *question pattern* metamodel which contains the elements needed to create a variety of these question pattern models (see Figure 3). These models will define the system question typology, i.e. question types that the system will be able to answer, thus detecting the kind of expected answer and the keywords of the question. A pattern is represented as a *Pattern* metaclass in order for it to contain several associated expressions (i.e. *Expression* and *Association* metaclasses) which represent a pattern. Moreover, a pattern is associated with an answer type (i.e. *AnswerType* metaclass), in such a way that the kind of expected answer is known when the question is classified by choosing the pattern that best fits the question. A metaclass *Expression* is used to consider every kind of expression. For example, syntactical labels such as PP-preposition, PtDt-interrogative pronoun or determinant, VBC-verbal head, SNP-simple noun phrase, SPP-simple preposition phrase and their values (e.g., an expression PtDt could have “which” as a value). Expressions may

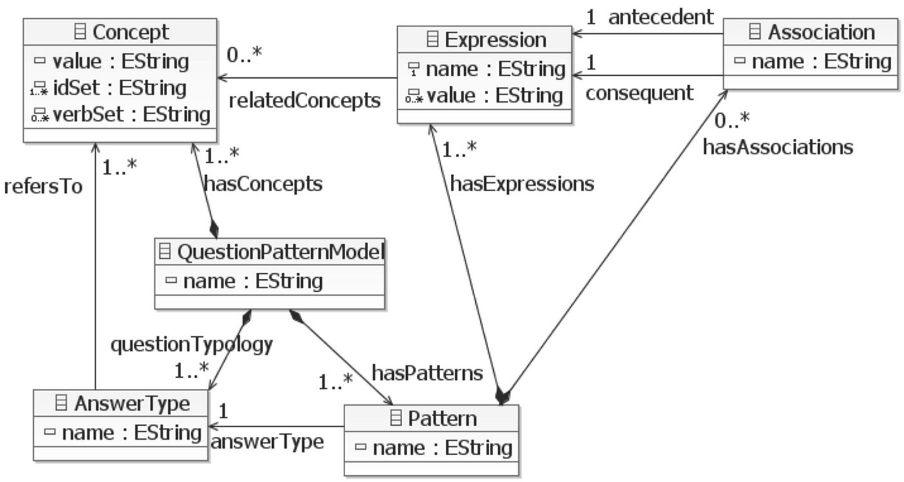


**Figure 3: Question pattern metamodel**

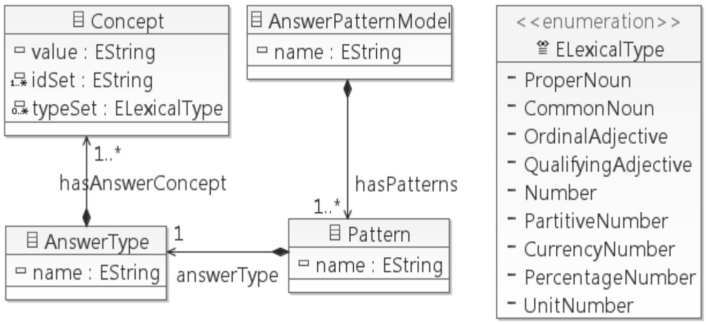


**Figure 4: Answer pattern metamodel**

have some related concepts (e.g., an SNP may have hyponyms of certain concepts within their expression). A metaclass *Association* relates expressions in order to ascertain a sequential order. It has an antecedent and a consequent. An example could be an association with the name "PtDt-VBC" whose antecedent is an expression "PtDt" and whose consequent is "VBC". An *AnswerType* metaclass refers to one or more concepts in order to determine the type of the answer. A metaclass *Concept* contains one or more IDs stored in the attribute that identifies the different KOS in which this concept is supposed to appear, along with several verbs that are frequently associated with the concept.

In the same way, the *Answer Pattern* metamodel has been defined (see Figure 4). In this case, each *Concept* metaclass has a set of valid lexical types (*typeSet*), which are represented by means of the *ELexicalType* enumeration.

## 3.2 Transformations for Adapting QA Systems

Having defined our metamodels we shall now describe the transformations required to adapt the existing QA systems to a restricted domain.

### *3.2.1 Obtaining Restricted-domain EAT Taxonomies*

A set of transformations has been designed to automatically define an EAT taxonomy for restricted-domain QA by using KOS within them (see transformations T1, T2, and T3 in Figure 1): the most relevant restricted domain terms from the corpus are defined in a model (by using transformation T1). This model is then enriched with concepts from restricted and open domain KOS (transformation T2). Finally, once this knowledge is represented, a restricted-domain EAT taxonomy is derived (transformation T3).

**Obtaining a restricted-domain model from the corpus.** Transformation T1 (see Figure 1) obtains a *restricted domain model* by selecting the most relevant terms appearing in the corpus based on two constraints: lexical (each term must originate from a meaningful category, such as a noun, an adjective or a verb), and statistical (terms must have certain frequencies, e.g. relative frequency (*fr*) or *tf-idf* frequency (Baeza-Yates and Ribeiro-Neto, 1999)). It is worth noting that the threshold values for these frequencies can be modified depending on the specific domain. A class *CorpusTerm* is created (*AdjectiveTerm*, *VerbTerm* or *NounTerm*) from each selected term, with its corresponding lexical, syntactic and semantic information obtained from the corpus processing in the QA task (see *Indexing Module* in Figure 1), including the different kinds of relationships between them.

To illustrate the benefits of our approach throughout this paper, consider the following motivating example based on the agricultural domain from the Cuban Journal of Agricultural Science or RCCA (*Revista Cubana de Ciencia Agrícola*, http://www.ica.inf.cu/productos/rcca/). This journal comprises topics related to agricultural science, such as Animal Science, Pastures and Forages, etc. In this paper, we use the English part of this journal as the corpus. One sample question about the agricultural domain is: "*Which is the enzyme that increases the animals' digestibility of organic phosphorus?*"

This question cannot be answered by our open-domain QA system, denominated as AliQAn (Roger *et al*, 2008). This system has an EAT taxonomy with two levels, based on WordNet Based-Types, that consists of the categories shown in the part of Figure 5 labeled as *OD-EATT*. By using the EAT taxonomy of AliQAn, the classification of the previous sample question is *object*. It is worth pointing out that AliQAn's classification of the question could be considered to be correct since *enzyme* has a top-concept hypernym *object* (the whole hypernym path is shown by means of rectangles in Figure 5). However, *object* is too wide a concept which may accept some incorrect candidate answers such as: artifact, ground, yeast, acids, salts, etc. as being semantically correct.

It is therefore necessary to apply our set of transformations to obtain an EAT taxonomy for the agricultural domain to be used in AliQAn. Figure 6 shows how transformation T1 works, by taking the sample question and a passage of a document related to that question.

Within this example, the text is lexically and syntactically labeled by using Freeling (http://nlp.lsi.upc.edu/freeling/), specifically by means of its PoS-tagger and its shallow syntactic parser, respectively. Freeling's labels starting with: (i) "NN:common noun" or "NNP:proper noun" are added to the *restricted domain model* as *NounTerm* with "NC" or "NP" *Type* respectively, (ii)

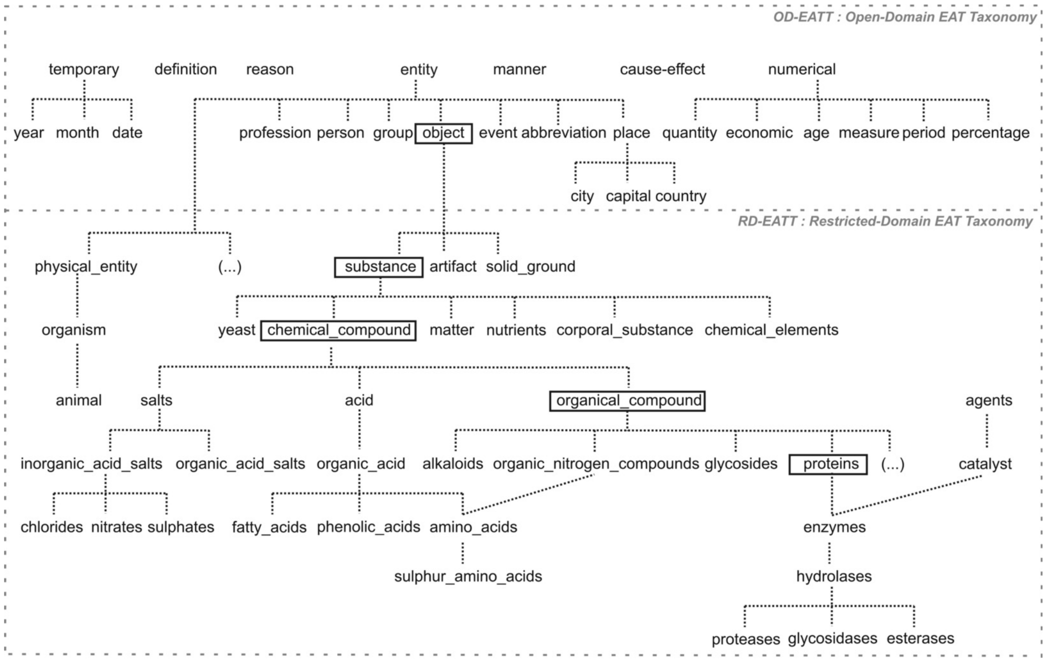
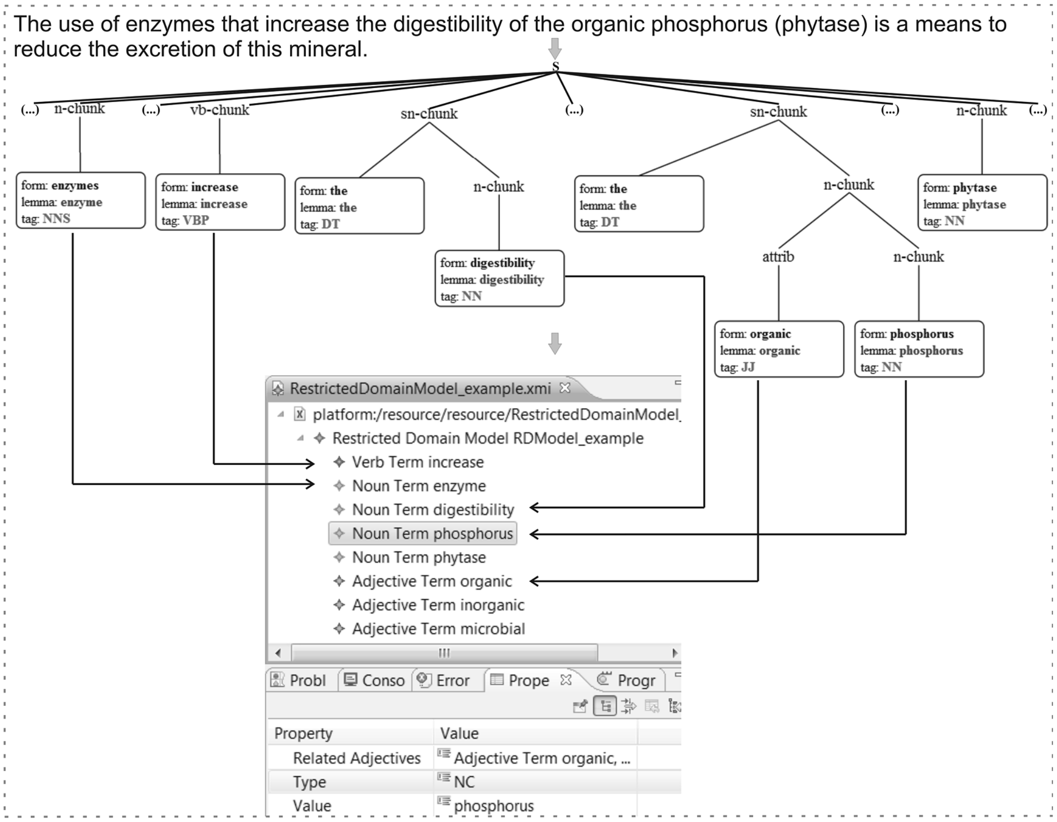


**Figure 5: Excerpt of an EAT taxonomy for open (OD-EATT) and restricted (RD-EATT) domains**


The use of enzymes that increase the digestibility of the organic phosphorus (phytase) is a means to reduce the excretion of this mineral.
S
(...) n-chunk (...) vb-chunk sn-chunk (...) sn-chunk (...) n-chunk (...)
form: enzymes lemma: enzyme tag: NNS
form: increase lemma: increase tag: VBP
form: the lemma: the tag: DT
n-chunk
form: the lemma: the tag: DT
n-chunk
form: phytase lemma: phytase tag: NN
form: digestibility lemma: digestibility tag: NN
attrib
n-chunk
form: organic lemma: organic tag: JJ
form: phosphorus lemma: phosphorus tag: NN
RestrictedDomainModel_example.xmi
platform:/resource/resource/RestrictedDomainModel_
Restricted Domain Model RDModel_example
Verb Term increase
Noun Term enzyme
Noun Term digestibility
Noun Term phosphorus
Noun Term phytase
Adjective Term organic
Adjective Term inorganic
Adjective Term microbial
Probl
Conso
Error
Prope
Progr

| Property | Value |
|---|---|
| Related Adjectives | Adjective Term organic, ... |
| Type | NC |
| Value | phosphorus |

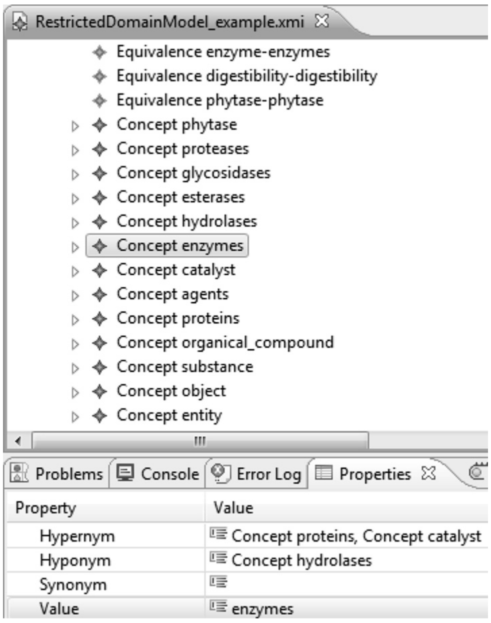
**Figure 6: Overview of how to obtain a restricted-domain model**

"VB:verb" as *VerbTerm* with "VM" *Type* and (iii) "JJ:adjective" as *AdjectiveTerm* with "AO" *Type*. The following syntactical information was also extracted: the "phosphorus" noun is related to the "organic" adjective, while the "digestibility" noun is the object of the verb "increase" and the "enzyme" noun is the subject. These syntactical relationships are obtained by using the following chunking tags for English in the Freeling shallow parser: "S:sentence", "n-chunk:noun", "sn-chunk:noun with definite article" and "vb-chunk:verb" in the following way: (i) the noun and adjective within the same "n-chunk" or "sn-chunk" are related by means of a *Related Adjectives* attribute; (ii) nouns within the same "n-chunk" or "sn-chunk" are related by means of a *Related Nouns* attribute; and (iii) noun and verbs within the same "S" are related by means of a *Subject* or *Object* attribute, according to the function of the name with regard to the verb.

**Enriching the restricted-domain model.** The second step of our approach consists of adding semantic knowledge to the already defined elements of the restricted-domain model by means of concepts and relationships from different kinds of KOS in order to create an *enriched restricted domain model*. This enrichment step takes place in the T2 transformation (see Figure 1) which allows heterogeneous KOS to be managed (from a simple taxonomy to a complex ontology) by ensuring integration and interoperability among them. The reason for this is that our metamodel is sufficiently sound to specify in a model those parts of KOS that will be useful in the following steps of our approach, thus abstracting unnecessary details. This transformation also makes the system adaptable since if a new KOS has to be considered, we are not obliged to change the whole QA system, but can adapt T2 to the new KOS.

Transformation T2 associates each corpus term previously detected with a particular concept from the domain KOS. Simple words from the *NounTerm* are searched for first, followed by multi-words by using the *Related Adjectives* and *Related Nouns* attributes. An *Equivalence* class is created to associate each new *Concept* class (including its corresponding *KOS* classes) with some existing *NounTerm* classes. The following step is to search for synonyms, hyponyms and hypernyms of the new *Concept* class in a domain KOS until a top concept is reached. Every top concept from the domain KOS is then checked for its subsequent association with a particular concept from a generic KOS (a disambiguation algorithm can be used at this stage), and if this association does not exist then hyponyms (and their synonyms) of this top concept are checked. For each concept belonging to a generic KOS, its hypernyms (and its synonyms) are added to the restricted-domain model until a top concept is found. The rationale behind the T2 transformation is to associate corpus terms with concepts from the domain KOS and not with the concept from the generic KOS, since (i) it is more likely that restricted-domain terms will appear in the domain KOS, and (ii) polysemy is avoided because restricted-domain terms that appear in the generic KOS are previously disambiguated with the domain KOS.

Following our running example, we have chosen to use the Agrovoc thesaurus as the agricultural domain KOS, and WordNet as the generic KOS. The following *NounTerms* in the previously obtained restricted-domain model (see Figure 6) are found in Agrovoc: “phytase”, “digestibility” and “enzyme”. These are specified as *Concepts* in the enriched restricted-domain model (see Figure 7). From these concepts, transformation T2 uses Agrovoc and WordNet to obtain 249 new *Concepts* (185 from Agrovoc, 15 from WordNet, and 49 from both) and 9 levels in accordance with their hypernym-


RestrictedDomainModel_example.xmi

Equivalence enzyme-enzymes
Equivalence digestibility-digestibility
Equivalence phytase-phytase
Concept phytase
Concept proteases
Concept glycosidases
Concept esterases
Concept hydrolases
Concept enzymes
Concept catalyst
Concept agents
Concept proteins
Concept organical_compound
Concept substance
Concept object
Concept entity

| Property | Value |
|---|---|
| Hypernym | Concept proteins, Concept catalyst |
| Hyponym | Concept hydrolases |
| Synonym | |
| Value | enzymes |

**Figure 7: Example showing how to obtain an enriched restricted-domain model**

hyponym hierarchical structure. For example, the *Concept enzymes* is obtained from the *NounTerm enzyme* along with its hypernyms in Agrovoc until the top concept *agents* is attained (see the part of Figure 5 labeled as RD-EATT). This Agrovoc top concept *agents* is intended to be mapped with the same concept in WordNet. However, it has five senses, which causes a high degree of polysemy in our enriched restricted-domain model. A simple disambiguation strategy is therefore used in which an Agrovoc concept is mapped with its WordNet counterpart only if it has one sense. Otherwise, some of the hyponyms of the concept from Agrovoc are intended to be mapped with one WordNet concept that has only one sense. In our example, the concept from Agrovoc that is successfully mapped is *enzymes*, thus obtaining the concepts from WordNet shown in Figure 5 starting at *proteins*.

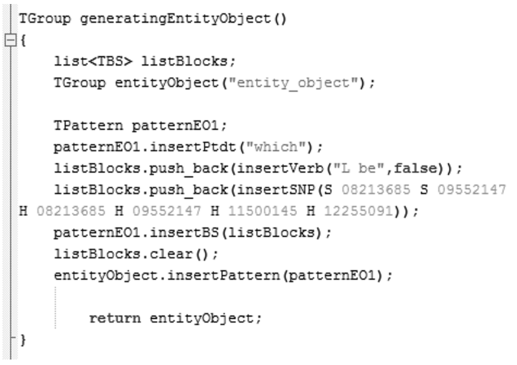
**Obtaining EAT taxonomies from the enriched restricted-domain model.** The *enriched restricted-domain model* obtained previously is used to create an EAT taxonomy for the restricted domain by applying transformation T3 (see Figure 1). Concepts in this taxonomy can obviously be refined to a greater or lesser extent, so a level of granularity should be selected in T3 by applying certain criteria over the *enriched restricted-domain model*: if a loose criterion is chosen, such as "including those concepts without a hypernym in the EAT taxonomy", then a generic taxonomy is obtained, while if a tighter criterion is defined, such as "including those concepts that have a number of hyponyms greater than N in the EAT taxonomy", then a more refined taxonomy is obtained. A tight criterion is more appropriate in restricted-domain QA because it is highly advisable for these kinds of taxonomies to be refined in order to improve their precision. We advocate the creation of an EAT taxonomy from the terms in the enriched restricted-domain model (and not directly from the domain KOS) in order to ensure that it contains those semantic classes that are more closely related to the domain. Finally, it is worth noting that the result of transformation T3 is a sub-model of the *enriched restricted-domain model* (see Figure 1).

By taking the first criterion, a generic EAT taxonomy is obtained for our running example with a single level with two concepts *entity* and *agents*. By using this taxonomy, the example question can be classified as *entity* (thus retrieving erroneous candidate answers such as any kind of profession, event, group, etc.). The question could also be unclassified since the *enzyme* class is not in the taxonomy. Conversely, if the second criterion is used, a refined EAT taxonomy is obtained with 9 levels and 13 concepts (as shown in the taxonomy excerpt in Figure 5). The question would therefore be classified as *enzymes* and the search space would be restricted to kinds of enzymes (such as "hydrolases" or its hyponyms) which would be accepted as a correct answer. Finally, by using this EAT taxonomy, the answer to the question will be "phytase" (a hyponym of the EAT taxonomy concept "hydrolase"). Furthermore, this EAT taxonomy would be useful for more specific questions, e.g. the sample question could be redefined as "*Which is the esterase that increases the animals' digestibility of organic phosphorus?*"

### *3.2.2 Adapting QA Patterns for Restricted Domains*

The EAT taxonomy previously defined is used in several transformations (see transformations T4, T5, T6, T7, T8, and T9 in Figure 1) to adapt both question and answer patterns to a restricted domain in an automatic manner.

**Obtaining QA pattern models.** In order to adapt existing patterns to a new domain, they must first be acquired from the baseline ODQA system. Transformations T4 and T5 are responsible for obtaining existing question and answer patterns in their respective models. These transformations depend on the kind of the implementation of the system, and our approach can therefore manage every kind of pattern by simply updating T4 and T5.

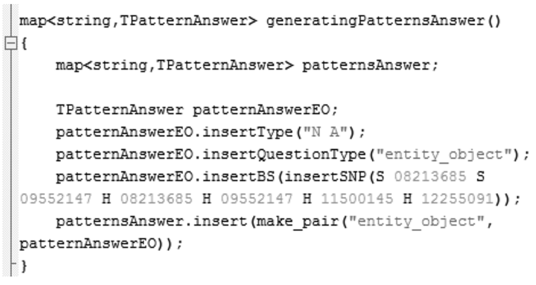


```
TGroup generatingEntityObject()
{
    list<TBS> listBlocks;
    TGroup entityObject("entity_object");

    TPattern patternEO1;
    patternEO1.insertPtdt("which");
    listBlocks.push_back(insertVerb("L be",false));
    listBlocks.push_back(insertSNP(S 08213685 S 09552147
H 08213685 H 09552147 H 11500145 H 12255091));
    patternEO1.insertBS(listBlocks);
    listBlocks.clear();
    entityObject.insertPattern(patternEO1);

        return entityObject;
}
```

**(a) Question pattern code**



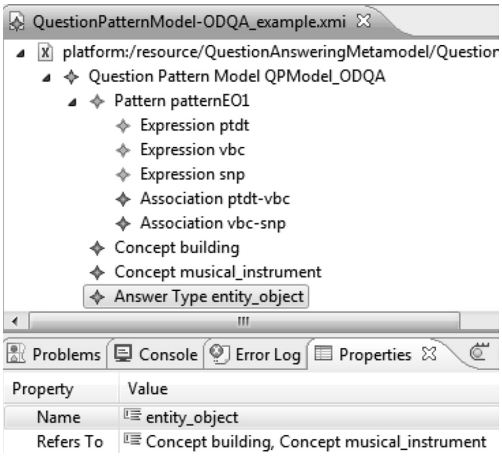
```
map<string,TPatternAnswer> generatingPatternsAnswer()
{
    map<string,TPatternAnswer> patternsAnswer;

    TPatternAnswer patternAnswerEO;
    patternAnswerEO.insertType("N A");
    patternAnswerEO.insertQuestionType("entity_object");
    patternAnswerEO.insertBS(insertSNP(S 08213685 S
09552147 H 08213685 H 09552147 H 11500145 H 12255091));
    patternsAnswer.insert(make_pair("entity_object",
patternAnswerEO));
}
```

**(b) Answer pattern code**

**Figure 8: Sample of open-domain pattern code**

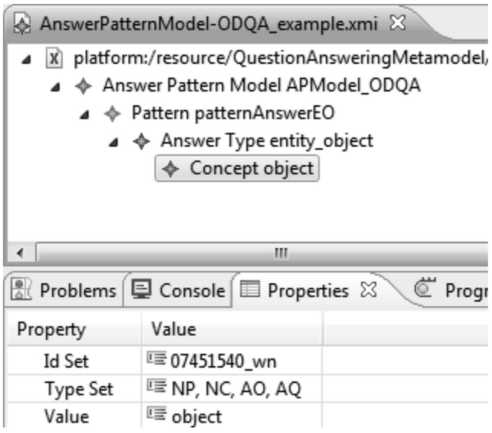
Figure 8(a) shows an example of a question pattern from AliQAn, called “patternEO1”, which is implemented in C++. It is important to note that the answer type (e.g. “entity_object”) is defined through the TGroup class, while the pattern (e.g. “patternEO1”) is defined with the TPattern class. In AliQAn system the elements of the question (e.g. “which” as PtDt, “be” as Verb and SNP) and the syntactical relations between them are defined in the TPattern class. Finally, the SNP type elements are defined by the identifier of the concepts that are related to the answer type and their corresponding kind of semantic relation (i.e. L-Literal, S-Synonym, and H-Hyponym). The corresponding question pattern model (obtained when applying T4) is shown in Figure 9(a), and it has the following expressions: PtDt (value: “which”), VBC (value: “be”) and SNP (whose related concepts are “musical_instrument”, “building”, etc.); the associations are “PtDt-VBC” and “VBC-SNP”; and the answer type is “entity_object”. Lastly, it is worth noting that this pattern signifies that ODQA system is able to identify questions of the kind: “*Which is the musical instrument that Beethoven played?*”



QuestionPatternModel-ODQA_example.xmi
platform:/resource/QuestionAnsweringMetamodel/Questior
Question Pattern Model QPModel_ODQA
Pattern patternEO1
Expression ptdt
Expression vbc
Expression snp
Association ptdt-vbc
Association vbc-snp
Concept building
Concept musical_instrument
Answer Type entity_object
Problems
Console
Error Log
Properties

| Property | Value |
|---|---|
| Name | entity_object |
| Refers To | Concept building, Concept musical_instrument |

**(a) Question pattern model**



AnswerPatternModel-ODQA_example.xmi
platform:/resource/QuestionAnsweringMetamodel/
Answer Pattern Model APModel_ODQA
Pattern patternAnswerEO
Answer Type entity_object
Concept object
Problems
Console
Properties
Progr

| Property | Value |
|---|---|
| Id Set | 07451540_wn |
| Type Set | NP, NC, AO, AQ |
| Value | object |

**(b) Answer pattern model**

**Figure 9: Sample of open-domain pattern models**

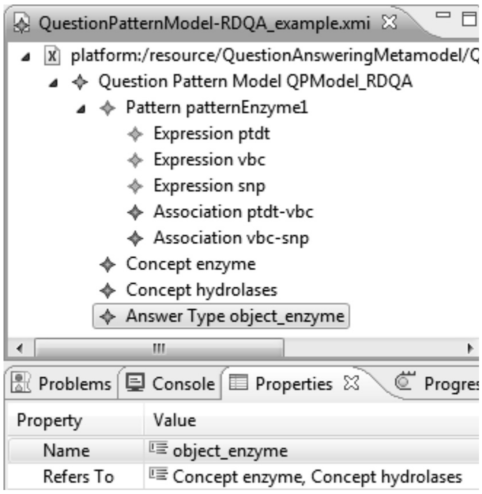
We must also obtain the patterns that make the answer extraction possible in a similar way. For example, Figure 8(b) shows an answer pattern from AliQAn system called "patternAnswerEO". The corresponding answer pattern model (obtained when applying T5) is shown in Figure 9(b). This model has the following elements: an element of the *Pattern* class called "patternAnswerEO", an element of the *AnswerType* class that represents the answer type associated with this pattern (value: "entity_object") and several elements of the *Concept* class correlated to this answer type (value: the concept of "inanimate_object" and their hyponyms). This answer pattern permits the answer to the question previously formulated to be found in the following text: "*[…] Beethoven, the last great representative of the Viennese classic style, played the piano from youth […]*".

**Adapting QA pattern models.** Transformation T6 (see Figure 1) collects both the EAT taxonomy of the *enriched restricted-domain model* and models of existing question patterns in order to generate new question pattern models that are specifically tuned to the restricted domain. The first step is to obtain the concepts from the EAT taxonomy of the *enriched restricted-domain model* (previously obtained in Sect. 3.2.1) and represent them with the *Concept* metaclass of the *adapted question pattern model*, together with the information required to fill in all the attributes of this class. The next step is to search for relationships between concepts which have been added to the *adapted question pattern model* (new concepts), along with those existing concepts in the *question pattern model* (old concepts) that allow us to decide whether patterns from an old concept can be used to define patterns for a new concept. Assuming that every top-level concept of the generic KOS used by the ODQA system has patterns, we will create a new pattern derived from an existing one if the following conditions hold when comparing the old concept and the new concept: (i) they are equal, (ii) they have a common hypernym provided by the same KOS, i.e. they are siblings in the hierarchy, or (iii) they maintain a hyponym-hypernym relation, i.e. they are parent and child in the hierarchy.

Transformation T7 aims to derive an adapted answer pattern model from existing patterns by using the same strategy as T6. This transformation collects both the EAT taxonomy from the *enriched restricted-domain model* and models of existing answer patterns in the QA system. The result of this transformation will be the answer pattern model adapted to the new restricted domain.

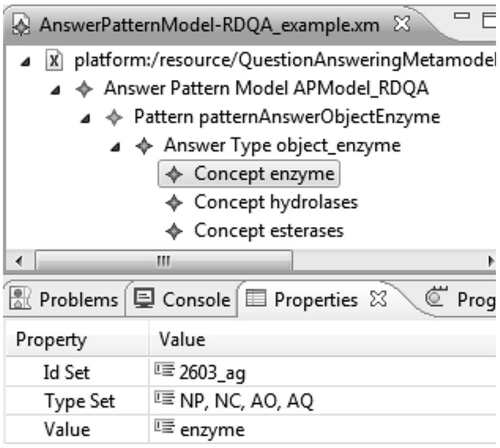
For example, if we recall our case study, the "enzyme" concept's "object" is the hypernym top concept which is an old concept with defined patterns. A new pattern for the "enzyme" concept is therefore also created from the old pattern previously explained for the "object" concept (i.e. "patternEO1" in Figure 9(a)). The name of the new pattern is "patternEnzyme1" (see Figure 10(a)), and it has the following expressions: PtDt (value: "which"), VBC (value: "be") and SNP (whose related concepts are "enzyme", "hydrolases", "esterases", "proteases", and "glycosidases"); the associations are "PtDt-VBC" and "VBC-SNP", and the answer type is "object_enzyme". Finally, it is worth pointing out that a new kind of question can be answered: "*Which is the enzyme that increases the animals' digestibility of organic phosphorus?*"

We must also obtain the adapted answer patterns "patternAnswerObjectEnzyme" (see Figure 10(b)) for the "enzyme" concept and its hyponyms in a similar way, as occurred with the open-domain answer pattern "patternAnswerEO", by associating the "object" concept (see Figure 9(b)). This model has the following elements: an element of the *Pattern* class called "patternAnswerObjectEnzyme", an element of the *AnswerType* class that represents the answer type associated with this pattern (value: "object_enzyme") and several elements of the *Concept* class correlated to this answer type (value: the concept of "enzyme" and its hyponyms). This answer pattern permits the answer to the question previously formulated to be found in the following text: "*[…] The use of enzymes that increase the digestibility of the organic phosphorus (phytase) is a means to reduce the excretion of this mineral […]*".



```
QuestionPatternModel-RDQA_example.xmi
platform:/resource/QuestionAnsweringMetamodel/Q
  Question Pattern Model QPModel_RDQA
    Pattern patternEnzyme1
      Expression ptdt
      Expression vbc
      Expression snp
      Association ptdt-vbc
      Association vbc-snp
    Concept enzyme
    Concept hydrolases
    Answer Type object_enzyme
```

| Property | Value |
|---|---|
| Name | object_enzyme |
| Refers To | Concept enzyme, Concept hydrolases |



```
AnswerPatternModel-RDQA_example.xm
platform:/resource/QuestionAnsweringMetamodel
  Answer Pattern Model APModel_RDQA
    Pattern patternAnswerObjectEnzyme
      Answer Type object_enzyme
        Concept enzyme
        Concept hydrolases
        Concept esterases
```

| Property | Value |
|---|---|
| Id Set | 2603_ag |
| Type Set | NP, NC, AO, AQ |
| Value | enzyme |

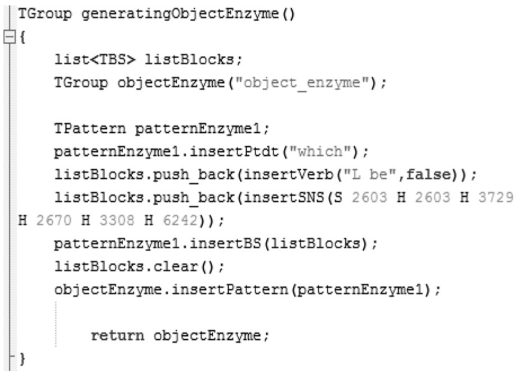
**(a) Adapted question pattern model** **(b) Adapted answer pattern model**

**Figure 10: Sample of adapted pattern models**

**Generating new QA pattern code.** Finally, transformations T8 and T9 automatically deploy the corresponding code for a specific QA system pattern. This is done by basing these transformations on the notion of customizable templates in order to capture rules for translating question and answer pattern models into corresponding code for different QA systems. As examples, an adapted question pattern "patternEnzyme1" and an adapted answer pattern "patternAnswerObjectEnzyme" resulting from applying our approach are shown in Figure 11(a) and Figure 11(b), respectively.

## 3.3 Implementation

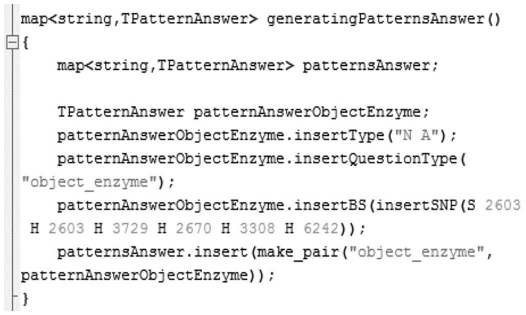
The metamodels and transformations have been implemented by using the Eclipse Framework (http://www.eclipse.org). Eclipse is an open source project conceived as a modular platform which can be extended by plugins in order to add features to the development environment. In order to



```
TGroup generatingObjectEnzyme()
{
    list<TBS> listBlocks;
    TGroup objectEnzyme("object_enzyme");

    TPattern patternEnzyme1;
    patternEnzyme1.insertPtdt("which");
    listBlocks.push_back(insertVerb("L be",false));
    listBlocks.push_back(insertSNS(S 2603 H 2603 H 3729
H 2670 H 3308 H 6242));
    patternEnzyme1.insertBS(listBlocks);
    listBlocks.clear();
    objectEnzyme.insertPattern(patternEnzyme1);

        return objectEnzyme;
}
```



```
map<string,TPatternAnswer> generatingPatternsAnswer()
{
    map<string,TPatternAnswer> patternsAnswer;

    TPatternAnswer patternAnswerObjectEnzyme;
    patternAnswerObjectEnzyme.insertType("N A");
    patternAnswerObjectEnzyme.insertQuestionType(
"object_enzyme");
    patternAnswerObjectEnzyme.insertBS(insertSNP(S 2603
 H 2603 H 3729 H 2670 H 3308 H 6242));
    patternsAnswer.insert(make_pair("object_enzyme",
patternAnswerObjectEnzyme));
}
```

**(a) Adapted question pattern code** **(b) Adapted answer pattern code**

**Figure 11: Sample of adapted pattern code**

support modeling tasks, we have used facilities provided by the Eclipse Modeling Framework (EMF, http://www.eclipse.org/emf) and the Graphical Modeling Framework (GMF, http://www.eclipse.org/gmf). To implement transformations between models, ATL (Atlas Transformation Language, http://www.eclipse.org/atl) has been used. ATL is integrated with EMF, thus allowing the definition and execution of model transformations by means of rules that match elements in source and target models. Finally, the transformations that generate code have been implemented by using Acceleo (http://www.eclipse.org/acceleo), which is also integrated into EMF in order to obtain code from models by labeling metamodels.

## 4. EXPERIMENTS

Two experiments have been conducted in order to validate our approach. The first one aims to show how unfeasible a baseline ODQA is for a restricted domain. To do so, a previous study of AliQAn, was carried out with 180 training questions over the RCCA corpus (i.e. 2024 articles from the RCCA journal), in which it was detected that 73.3% of errors in the adaptation of the system were caused by (i) a poor and incorrect EAT taxonomy, (ii) an incorrect classification of questions owing to the absence or inefficiency of the question patterns, and (iii) 17.2% of errors were owing to failures in the answer patterns. The precision of the AliQAn system obtained 28.8% with regard to only the first answer retrieved and 13.6% when considering the first three answers. These results are very low in comparison to the average (i.e. around 43% precision) attained by AliQAn in the CLEF evaluation forum (i.e. open-domain scenario). It also attained an overall recall of 33%. It is consequently crucial to adapt AliQAn to a restricted domain in order to increase its precision and obtain actionable information from the Web.

Our second experiment aims to show how our approach can be used to adapt the baseline ODQA to increase precision and recall. The first step in this experiment consisted of processing the RCCA corpus with a PoS tagger and a syntactical parser (by means of the Freeling PoS-tagger and the Freeling shallow syntactic parser), indexing it and computing frequencies for each term. Since we consider the most relevant terms to be those which have *fr>25* and *tf-idf>0.01*, 8696 relevant terms were obtained and specified in a restricted-domain model by means of transformation T1. Noun terms were then used in transformation T2 to enrich the restricted-domain model by using Agrovoc and WordNet. Table 1 shows a summary of our results: the restricted-domain model has 9022 concepts of which 3029 are multi-words. Most of the concepts (8530) originate from the domain KOS (Agrovoc) and 3473 concepts originate from the generic KOS (WordNet), which has 2981 common concepts that represent the enrichment of the restricted domain model. The EAT taxonomy was then obtained from the restricted-domain model by applying transformation T3 with the criterion of choosing those concepts with more than two hyponyms. Table 1 shows that the EAT taxonomy contains roughly 10% of concepts from the restricted-domain model. Finally, the other transformations in our approach were executed (i.e. from T4 to T9), thus generating 325 new EAT concepts and their corresponding code for about 2600 question patterns and 325 answer patterns (one for each concept). These data show how much effort would be required to accomplish restricted-domain adaptation manually and the benefit of applying our approach to support this process. Furthermore, it is worth noting that, by using our approach, the precision and recall are 58.3% and 75%, respectively. If we compare these values with the results of our first experiment, we can appreciate an increment of 29.5% in the precision and 42% in the recall. Therefore, there were 57.8% less errors for question patterns and 7.8% for answer patterns. These results show the effectiveness of our approach in the adaptation of AliQAn to the agricultural domain of the RCCA journal.

| | Restricted Domain Model | | | | EAT Taxonomy | | | |
|---|---|---|---|---|---|---|---|---|
| Levels | Agrovoc | WordNet | Multi-words | Total | Agrovoc | WordNet | Multi-words | Total |
| 0 | 438 | 174 | 212 | 462 | 149 | 77 | 69 | 161 |
| 1 | 1382 | 479 | 812 | 1429 | 133 | 83 | 67 | 149 |
| 2 | 1565 | 551 | 568 | 1627 | 98 | 70 | 40 | 121 |
| 3 | 1144 | 486 | 334 | 1233 | 79 | 77 | 24 | 111 |
| 4 | 839 | 362 | 250 | 935 | 70 | 68 | 19 | 105 |
| 5 | 896 | 375 | 261 | 979 | 76 | 53 | 24 | 94 |
| 6 | 1002 | 433 | 255 | 1053 | 66 | 52 | 17 | 82 |
| 7 | 562 | 291 | 140 | 587 | 37 | 18 | 12 | 43 |
| 8 | 284 | 156 | 61 | 289 | 32 | 19 | 4 | 32 |
| 9 | 291 | 95 | 84 | 295 | 11 | 9 | 3 | 13 |
| 10 | 72 | 41 | 28 | 77 | 5 | 2 | 1 | 5 |
| 11 | 30 | 17 | 12 | 30 | 4 | 2 | 1 | 4 |
| 12 | 20 | 11 | 8 | 20 | 1 | 1 | 0 | 1 |
| 13 | 3 | 1 | 2 | 3 | 0 | 0 | 0 | 0 |
| Total | 8530 | 3473 | 3029 | 9022 | 761 | 531 | 281 | 921 |

**Table 1: Statistics of Restricted Domain Model and EAT taxonomy created (# semantic classes)**

## 5. CONCLUSIONS AND FUTURE WORK

For a question to be efficiently and effectively answered in real-world scenarios, QA systems must be adapted to restricted domains. However, current approaches suffer from the following principal problems: (i) Expected Answer Types (EAT) taxonomies and QA patterns are manually tuned for restricted-domains, which requires a huge amount of effort in both time and cost, and (ii) the tuning of EAT taxonomies and QA patterns is based on analyzing potential questions to be answered, which is not a realistic situation since, in restricted domains, questions are highly complex and difficult to acquire.

This paper presents an approach with which to overcome these problems in a systematic, well-structured, and comprehensive manner by using an innovative point of view borrowed from model-driven software development. Specifically, problem (i) is overcome by means of an automatic process that uses as input the different kinds of knowledge resources (KOS) from the specific domain and the most relevant terms from the corpora. With regard to problem (ii), since comprehensive training corpora are hard to find, our approach automatically obtains the EAT taxonomies and the code of the QA patterns to be used for both the classification of the query and the extraction of the answer. They are just obtained by an automatic process of the corpora and the KOS.

Finally, the implementation of our approach has been used for conducting two experiments within the agricultural domain: the first one has shown how unfeasible a baseline ODQA is for a restricted domain (precision of 28.8% with the first answer, and 13.6% with the first three answers); whilst the second one has shown how our approach can be used to adapt the baseline ODQA (an increment of 29.5% in the precision and 42% in the recall).

Our future work will consist of evaluating our approach with a more complete set of experiments in other domains and with other baseline ODQA systems.

## ACKNOWLEDGEMENTS

This research has been partially funded by the Valencia Government under the PROMETEO/2009/119 project, and by the Spanish Government under the TIN2009-13391-C04-01 project.

## REFERENCES

BAEZA-YATES, R. and RIBEIRO-NETO, B. (1999): Modern information retrieval, *ACMPress*, New York.

BÉZIVIN, J. (2005): On the unification power of models, *Software and System Modeling* 4(2): 171–188.

ELY, J.W., OSHEROFF, J.A., GORMAN, P.N., EBELL, M.H., CHAMBLISS, M.L., PIFER, E.A. and STAVRI, P.Z. (2000): A taxonomy of generic clinical questions: classification study, *BMJ* 321(7258): 429–432.

FERRÁNDEZ, Ó., IZQUIERDO, R., FERRÁNDEZ, S. and VICEDO, J.L. (2009): Addressing ontology-based question answering with collections of user queries, *Inf. Process. Manage.* 45(2): 175–188.

FERRÉS, D. and RODRÍGUEZ, H. (2006): Experiments adapting an open-domain question answering system to the geographical domain using scope-based resources, *Proceedings of the Multilingual Question Answering Workshop of the EACL 2006*, 69–76.

FERRUCCI, D.A., BROWN, E.W., CHU-CARROLL, J., FAN, J., GONDEK, D., KALYANPUR, A., LALLY, A., MURDOCK, J.W., NYBERG, E., PRAGER, J.M., SCHLAEFER, N. and WELTY, C.A. (2010): Building Watson: An overview of the DeepQA Project, *AI Magazine* 31(3): 59–79.

HODGE, G. (2000): *Systems of Knowledge Organization for Digital Libraries: Beyond Traditional Authority Files*, The Digital Library Federation Council on Library and Information Resources.

HOVY, E., HERMJAKOB, U. and RAVICHANDRAN, D. (2002): A question/answer typology with surface text patterns, *Proceedings of the second international conference on Human Language Technology Research*, 247–251.

KLEPPE, A., WARMER, J. and BAST, W. (2003): *MDA Explained. The Practice and Promise of The Model Driven Architecture*, Addison Wesley.

KOSSEIM, L. and YOUSEFI, J. (2008): Improving the performance of question answering with semantically equivalent answer patterns, *Data Knowl. Eng.* 66(1): 53–67.

LASHERAS, J., VALENCIA-GARCÍA, R., FERNÁNDEZ-BREIS, J.T. and TOVAL, A. (2009): Modelling reusable security requirements based on an ontology framework, *Journal of Research and Practice in Information Technology* 41(2): 119–133.

LI, X. and ROTH, D. (2006): Learning question classifiers: the role of semantic information, *Nat. Lang. Eng.* 12(3): 229–249.

LOPEZ, V., UREN, V.S., MOTTA, E. and PASIN, M. (2007): AquaLog: An ontology-driven question answering system for organizational semantic intranets, *J. Web Sem.* 5(2): 72–105.

METZLER, D. and CROFT, W.B. (2005): Analysis of statistical question classification for fact-based questions, *Inf. Retr.* 8(3): 481–504.

MOLDOVAN, D.I., PASCA, M., HARABAGIU, S.M. and SURDEANU, M. (2003): Performance issues and error analysis in an open-domain question answering system, *ACM Trans. Inf. Syst.* 21(2): 133–154.

MOLLÁ, D. and VICEDO, J.L. (2007): Question answering in restricted domains: An overview, *Computational Linguistics* 33(1): 41–61.

PEÑAS, A., FORNER, P., SUTCLIFFE, R., RODRIGO, Á., FORASCU, C., ALEGRIA, I., GIAMPICCOLO, D., MOREAU, N. and OSENOVA, P. (2009): Overview of ResPubliQA 2009: Question answering evaluation over European legislation, *Working Notes of Cross Language Evaluation Forum (CLEF)*.

ROGER, S., VILA, K., FERRÁNDEZ, A., PARDIÑO, M., GÓMEZ, J.M., PUCHOL-BLASCO, M. and PERAL, J. (2008): Using AliQAn in monolingual QA@CLEF 2008, *CLEF*, 333–336.

SANG, E.T.K., BOUMA, G. and DE RIJKE, M. (2005): Developing offline strategies for answering medical questions, *Proceedings of the AAAI-05 workshop on Question Answering in restricted domains*, 41–45.

SEKINE, S., SUDO, K. and NOBATA, C. (2002): Extended named entity hierarchy, *Proceedings of 3rd International Conference on Language Resources and Evaluation (LREC'02)*, Canary Islands, Spain, 1818–1824.

SELIC, B. (2003): The pragmatics of model-driven development, *IEEE Software* 20(5): 19–25.

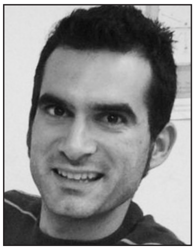
VALENCIA-GARCÍA, R., SÁNCHEZ, F.G., NIEVES, D.C. and FERNÁNDEZ-BREIS, J.T. (2011): OWLPath: An OWL ontology-guided query editor, *IEEE Transactions on Systems, Man, and Cybernetics, Part A* 41(1): 121–136.

## BIOGRAPHICAL NOTES

*Katia Vila obtained her PhD in computer science from the University of Alicante (Spain). Previously, she received her BS and MSc degrees in computer science from the University of Matanzas (Cuba). She is a member of the Research Group on Natural Language Processing and Information Systems of the Department of Software and Computing Systems (www.dlsi.ua.es). She has published several papers about restricted-domain question answering systems, noise-tolerant information retrieval systems and model-driven development in international workshops and conferences (such as ICCS, SEKE, MTSR, CLEF and so on). Her current research interests include natural language processing, question answering systems, information retrieval and model-driven development. Contact her at kvila@dlsi.ua.es*

Katia Vila

*Jose-Norberto Mazón is assistant professor at the Department of Software and Computing Systems in the University of Alicante (Spain). He obtained his PhD in computer science from the University of Alicante. He has published several papers in international and national conferences (such as ER or JISBD), and in several journals such as DSS, SIGMOD Record or DKE. He has also been co-organizer of the International Workshop on Business intelligencE and the WEB and the International Workshop on The Web and Requirements Engineering. His research interests are: model driven development, business intelligence, and requirement engineering. Contact him at jnmazon@dlsi.ua.es*


Jose-Norberto Mazón

*Antonio Ferrández is a full-time lecturer at the Department of Software and Computing Systems in the University of Alicante (Spain). He obtained his PhD in computer science from the University of Alicante (Spain). His research interests are: natural language processing, anaphora resolution, information extraction, information retrieval and question answering. He has participated in numerous projects, agreements with private companies and public organizations related to his research topics. Finally, he has supervised PhD thesis and participated in many papers in Journals and Conferences related to his research interests. Contact him at antonio@dlsi.ua.es*

Antonio Ferrández