Translation of Heterogeneous Requirements Meta-Models Through a Pivot Meta-Model

Imed Eddine Saidi[1], Mahmoud El Hamlaoui[2], Taoufiq Dkaki[1],
Nacer Eddine Zarour[3] and Pierre-Jean Charrel[1]
[1]*IRIT Laboratory, University of Toulouse-Jean Jaurès, Toulouse, France*
[2]*IMS-ADMIR Laboratory, ENSIAS, Rabat IT Center, Mohammed V University in Rabat, Rabat, Morocco*
[3]*LIRE Laboratory, University Constantine 2, Constantine, Algeria*

Keywords: Requirements Engineering, Pivot Model, Translation, Interoperability.

Abstract: Companies use these different approaches to elicit, specify, analyse and validate their requirements in different contexts. The globalization and the rapid development of information technologies sometimes require companies to work together in order to achieve common objectives as quickly as possible. We propose a Unified Requirements Engineering meta-model (UREM) that allows cooperation in the requirements engineering process between heterogeneous RE (Requirement Engineering) models. In this paper, we explore UREM as a pivot meta-model to ensure interoperability between heterogeneous RE models.

1 INTRODUCTION

The globalization and the rapid development of information technologies require that nowadays systems and organizations cooperate with each other. Many research works have emerged to support this cooperation which gave birth to the Cooperative Information Systems (CIS) in order to deal with the problem of heterogeneity at all levels of software development lifecycle. Most of the work focuses on the architectural (conceptual) level of software projects. Very little work focuses on the most abstract level (early stage) of the software development lifecycle namely the requirements engineering (RE); which aims to describe and manage upstream phases of software projects. This does not mean that there are no problems in this stage of RE when it comes to inter-company cooperation. It should be noted that one of the reasons affecting the failure of software projects is the bad definition of requirements. Hence, focusing on requirements engineering becomes gradually one of the most important concerns in the software development lifecycle. This has resulted in the emergence of different kinds of requirements engineering approaches such as goal, viewpoint, and scenario oriented approaches. This variety of RE approaches makes cooperation among companies stakeholders in this stage a difficult activity due to the heterogeneity of these approaches. The aim of this paper is to propose a solution which allows companies that use different kinds of RE approaches to cooperate without forcing companies to migrate to a unique approach which is very time and cost consuming.

Bendjenna, (Bendjenna and al., 2010) has proposed an integrated approach MAMIE which combines different kinds of concepts: goal, scenario and viewpoint in order to allow cooperation between companies. In i* approach, there exists different variations for particulars usages. Carlos (Carlos and al., 2011) has defined super meta-model hosting identified variations of i* and implementing a translation algorithm between these different variations oriented to semantic preservation. Our work intends to be a combination between the two works. We propose an abstract meta-model which allows cooperation and translation of information between different kinds of RE approaches.

This paper is organized in six sections. In section two, we present requirements engineering meta-models. In section three, we present the idea behind the unified meta-model and in section four, we present our unified meta-model UREM. In section five, we deduct translation rules between concepts.

Saidi, I., El Hamlaoui, M., Dkaki, T., Zarour, N. and Charrel, P.
Translation of Heterogeneous Requirements Meta-Models Through a Pivot Meta-Model.
DOI: 10.5220/0006789903760382
In *Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2018)*, pages 376-382
ISBN: 978-989-758-300-1

In section five, we illustrate a simple example of translation between RE models. Finally, we conclude and draw perspectives of this paper.

2 REQUIREMENTS ENGINEERING META-MODELS

In requirements engineering, different models exist and each model is composed of a set of concepts, these approaches are commonly grouped in three groups. In this section, we draw the meta-model of what we believe is the most widely approach of each group or type. We explore respectively the meta-model of i* as a goal oriented approach, PREview as a viewpoint oriented approach and CREWS as a scenario oriented approach. Most of the concepts in the following meta-models share two common attributes: ID and Name. Intuitively, each concept has a name. The attribute ID is added in order to keep a trace (index) to each instance of each concept. Figure 1 illustrates the meta-model of i* according to the description given in (Castro and al., 2011).

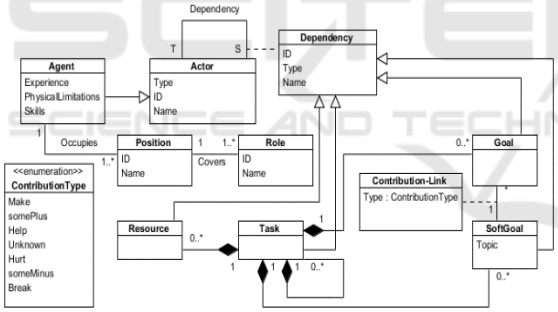

Figure 1: i* meta-model.

In i*, Actors depend on each other through four intentional elements: Task, Goal, SoftGoal and Resource. An Actor can be an Agent which occupies a certain position. Each position covers a set of Roles. The Task concept can be decomposed into a set of intentional elements: Sub-Tasks, Sub-Goals, Sub-SoftGoals and Resources. A soft goal can contribute positively or negatively to the achievement of one or more goals.

Figure 2 illustrates the meta-model of PREview (Sommerville and al., 1997).

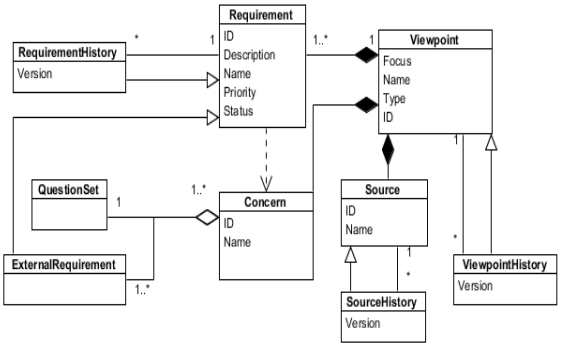

Figure 2: PREview meta-model.

PREview is a viewpoint-oriented approach where each viewpoint has a Focus, a Name, and a Type and is composed of a set of concerns, sources, requirements and history which aims to ensure the traceability of the focus, requirements and sources of information. We can observe the attribute Version in the three classes that represent the history concept.
Figure 3 illustrates the meta-model of CREWS as described in (Sutcliffe and al, 1998).

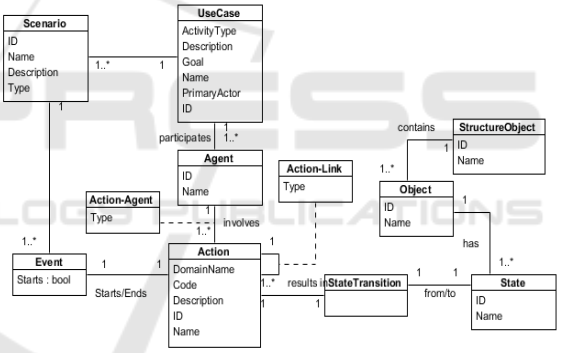

Figure 3: CREWS meta-model.

CREWS is a scenario oriented approach starting by defining use cases of the problem to be solved. Each use case can generate one or more scenarios knowing that each scenario is a sequence of events. Many agents can participate to a use case and involve one or more actions. Actions are inter-connected through eight types of links. An action can use an object and can change its state through a state transition.

3 INTEROPERABILITY OF HETEROGENEOUS MODELS

Conceptually, the interoperability of heterogeneous models can be performed either directly without

intermediate transformation or after a transformation tin order to express models within the same "pivot" language. A pivot model is a model used as an intermediate representation to align the input models to the same formalism.

The concept of a pivot model has been introduced in several research areas related to the model-driven engineering especially in taking into account the interoperability. Commonly, the term “pivot” means the point of rotation in a lever system. It is also the term used to describe an interpreter who translates a low level language “Maltese” (national language of Malta) to a language (e.g : English). The translated text is then used as a source of translation for other languages (Beleg and al., 2009).

One of the first uses of the term “pivot” in Computer Science referred to the quicksort algorithm numbers (quicksort). The algorithm consists in choosing a number (called pivot) from a list of disordered numbers and switch all the elements, so that all those who have a lower value to the pivot are placed to the left and all those who a higher value on his right.

Milanovic introduced in (Milanovic and al., 2009): R2ML (Rewerse Rule Markup Language), a pivot meta-model for bidirectional alignment taking into account in one hand the ontologies that are the backbone of the semantic web and in the other hand MDA concepts. In the field of ontologies, central area of the Semantic Web, the models are described by OWL (Ontology Web Language) and SWRL language (Semantic Web Rule Language) for expressing validation rules for the semantic web. Whereas in the field MDA, models are described in UML with OCL as constraints expression language. Thus models expressed in UML / OCL can be exploited in the field of semantic web by translating them into OWL/SWRL models and vice versa through neutral model: R2ML.

Similarly Sun and al. in (Yu Sun and al., 2009) have defined a pivot model. Many tools according to them are developed to automatically detect redundant codes in a program and represent them into appropriate statistics. The problem is that each of these tools has a different representation of the obtained result which gives the integrator a hard task to know each of these representations in order to act on the portion of the appropriate program.

The idea presented is to provide a common graphical representation in SVG. This is achieved by defining a meta-model pivot GCC (Generic Code Clone) that contains common concepts and characteristics of redundant code blocks detection tools. This meta-model will serves as (intermediate) between redundant code detection tools and SVG (Scalable Vector Graphics) model.

The use of pivot as an intermediate model makes it easy to centralize and optimize the data format in order to represent the input models in the same formalism. The interoperability process is thus simplified and can continue with less complexity.

In the next section we explore our Pivot model (UREM) which is proposed to perform translation between different RE models.

4 UNIFIED REQUIREMENTS ENGINEERING META-MODEL

UREM is an intermediary of communication and information translation between different types of RE models. RE models are instances of different types of RE meta-Models where each meta-Model is composed of a set of concepts.

The idea behind the pivot UREM is to create a new meta-Model which is composed of a set of classes where each class is an abstraction of a set of concepts (similar concepts) that exist in different RE meta-models. To find abstractions between RE concepts, we have adopted a rigorous process that is concerned with the meaning of concepts (Semantic Process). Our process is based on WordNet (George, 1995) to find semantic relationships and similarities between words which represent RE concepts (words are the only thing that we get to apprehend RE concepts).

Our aim is to perform cooperation between different types of approaches. In the unification process, we have chosen one approach from each type of RE approaches in order to achieve our goal, regardless of the RE approach chosen, our unification process is applicable to various other approaches. In this paper, we deal with approaches that are widely used: i* (Castro and al., 2011) as goal oriented approach, CREWS (Sutcliffe and al, 1998) as scenario oriented approach and PREview (Sommerville and al., 1997) as viewpoint oriented approach.

The following sub-sections give us an overview of the unification process.

4.1 Concepts Categorization

The first step of the unification process is to categorize all concepts of the three RE approaches mentioned above under two categories:

- Concepts of category one: the most of these concepts are represented as one word and we can get directly the definition and the different semantic relationships between them from WordNet;
- Concepts of category two: the most of these concepts are composed of more than one word and we cannot get directly the definition and the different semantic relationships between them from WordNet.

We adopt an incremental process in order to create the unified meta-model UREM. We start with concepts of category one. Next, we use results of category one to complete the unification process with the concepts of the second category and conclude UREM.

4.2 Dealing with Concepts of Category One

The algorithm of unification of this category of concepts is composed of two steps:

4.2.1 Semantic Relatedness and Word Sense Disambiguation (WSD)

In English language, a word can have more than one sense that can lead to ambiguity. Disambiguation is the process of finding out the most appropriate sense of a word (concept) that is used in a given context.

The Lesk algorithm (Lesk, M., 1986) uses dictionary definitions (gloss) to disambiguate a polysemous word in a sentence context. The idea of the algorithm is to count the number of words that are shared between the two glosses. The more overlapping (overlap scoring) the words, the more related the senses are. We have used an adapted version of Lesk (Satanjeev B., and al., 2002) which uses WordNet to access a dictionary with senses arranged in a hierarchical order. This extended version uses not only the gloss/definition of the synset, but also considers the meaning of related words.

4.2.2 Least Common Hypernym and Semantic Similarity between Two Senses

In this step we look up using WordNet the least common hypernym (LCH) for each pair of this category of concepts using appropriate senses that are previously assigned. Hyponymy is a 'kind of' relation, for example: tree is a kind of plant, tree is a hyponym of plant and plant is a hypernym (abstraction) of tree. We treat the taxonomy of hyponymy as a tree T_H. Once all trees are built, we establish connections between all LCH. These LCH are the set of abstraction concepts used in UREM.

4.3 Dealing with Concepts of Category Two

In this step, we use results obtained from the previous step to conclude the unified requirements engineering meta-model UREM. We are aware that where exist a common hypernym between two concepts, there exist a path between them in the tree T_H. The shorter path from the first concept to the second, the more similar they are. Regarding this category of concepts, we compute similarity scores between concepts by comparing text definitions for each pair of them with LCH elements that are archived in the previous step. The resulted meta-model UREM is shown in figure below.

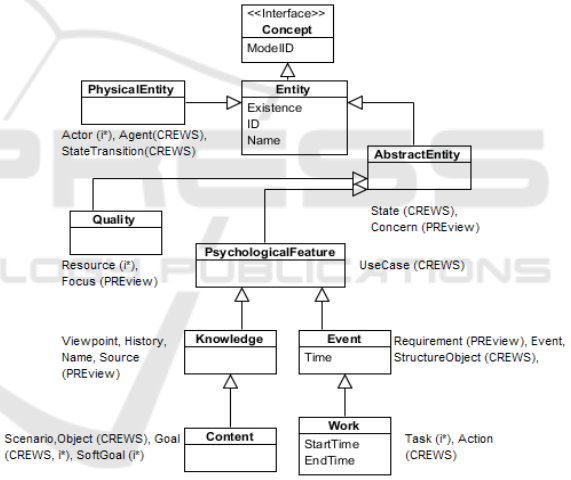

Figure 4: Unified Requirements Engineering meta-model.

Each abstract concept in UREM covers a set of RE concepts that exist in different RE models, proceeding from UREM, we are looking for all correspondences between RE concepts of the three meta-models. These correspondences will be used to translate a source concept CS of a source model MS to a target concept CT of a target model MT. In one hand, we can say that the whole set of all possible RE concepts can be in the same correspondence at a certain abstraction because all UREM concepts are related through inheritance relationship knowing that each instance of a class in object oriented design can be an instance of its parent class according to the Liskov Substitution Principle [6]. In the other hand,

we care more about details when performing translation in order to maximize the quantity of information to be translated among concepts. Each correspondence should have at least one concept for each approach (i*, CREWS or PREview) to ensure that all concepts in all approaches can be translated to other concepts in the remaining approaches. We define correspondences from the most detailed concept in UREM to the most abstract by relating each abstract concept to its parent until getting at least one concept for each approach in the correspondence. The result is given in the following correspondences:

- Corresponence 1 = {Scenario, Object (CREWS) , Goal (CREWS,i*), SoftGoal (i*), Viewpoint, History, Name, Source (PREview)}
- Correspondence 2= {Task(i*), Action, Event, StructureObject (CREWS), Requirements (PREview)}
- Correspondence 3={UseCase, State, Agent, StateTransition (CREWS), Concern, Focus (PREview),Resource, Actor (i*)}

Concepts in the same correspondence which share the same abstraction are more similar to each other than the rest of concepts in the same correspondence. This important point is proportional for the information to be translated among concepts.

5 DEDUCTION OF TRANSLATION RULES FROM UREM

In this section, we deduct translation rules from UREM illustrated in figure one. We observe in figure one a list of concepts of different RE meta-models near each class of UREM. So, each class covers a set of concepts and plays the role of a pivot between these concepts. Proceeding from UREM, we are looking to find for each source concept c_S of model M_S, a target concept or a set of target concepts c_T of model M_T. We perform two-way translation between two given models M_1 and M_2, first from M_1 to M_2 then we translate each not-translated concept of M_2 to a target concept of M_2. Two-way translation allows us to ensure that we have applied translation on all concepts of all different RE models.

We conclude two types of translation between concepts: Direct Translation and Inheritance Translation. The following sub-sections describe each type of translation.

5.1 Direct Translation

This type of translation is used if the source concept c_S of a model M_1 and the target concept c_2 of a model M_2 share the same abstraction c_G in UREM. To perform translation from the source concept to the target concept, we check the abstraction of the source concept then create the target concept by implementing the abstraction c_G in two steps: Copy shared attributes to the target concept and then translate the rest of attributes one by one.

For example, the concept *Resource* in an instance of i* meta-model share the same abstraction *Quality* in UREM with the concept *Concern* of a PREview model then *Resource* concept must be translated to a *Concern* in PREview model and vice versa.

5.2 Inheritance Translation

This type of translation is used if a source concept c_S of a model M_S can't be translated to any target concept c_T of model M_T using Direct Translation. In this type of translation, we check classes that are linked to the abstraction c_G of the source concept in order to find the abstraction of a target concept c_2. Since we care more about details of concepts, we check first child classes of c_G. If no abstraction of a target concept is founded, we check parent classes (abstractions of c_G). If no abstraction of a target concept is founded then the source concept c_S can't be translated to a target concept c_G.

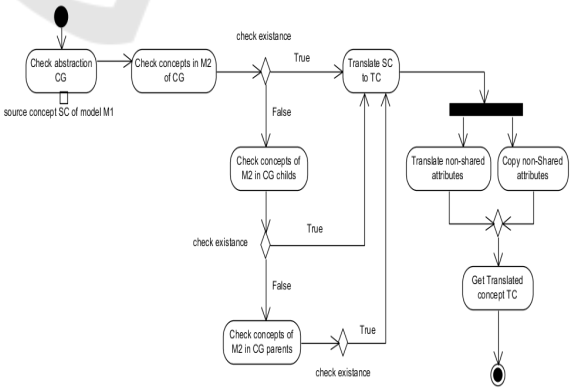

Figure 5: Activity diagram of translation between requirements engineering concepts.

For example, to perform translation from a *Requirement* of a PREview model to other concept in i* model. We observe that Requirement doesn't

share an abstraction which is *Event* with any concepts of i*. Thus, we look up child classes of *Event* class level by level. We find that the child class *Work* of *Event* covers the concept *Task* in i* then, when we perform translation from a PREview model to an i* model, the concept *Requirement* of PREview must be translated to the concept *Task* of i* and vice versa.

The activity diagram which describes the overall process of finding translation between concepts is shown in the figure 5.

6 EXAMPLE OF CONCEPTS TRANSLATION

The following example is just a simple illustration of concepts translation between different RE models in the development of a simple batch payroll system. A complete case study and translation between concepts in details will be in a future work.

CREWS, i* and PREview approaches are used in order to create a requirements specification of the desired system. One of the concepts specified in i* model is the goal 'employee payment'. To translate i* model to PREview model, the goal 'employee payment' will be translated to Viewpoint, History, Name and Source concepts as shown in figure 3.

To translate the goal 'employee payment' of an i* model, we perform the following steps:

- Create an abstraction of type content of the goal;
- Create the abstraction Knowledge of Content;
- Implement Knowledge by creating three classes of concepts: Viewpoint, Source and History. The concept Name is an attribute of the Viewpoint class and represents the identifier of the given viewpoint.

Figure below gives us a simple view of translation from Goal to Viewpoint, History, Name and Source concepts and does not give a lot of details of attributes translation.

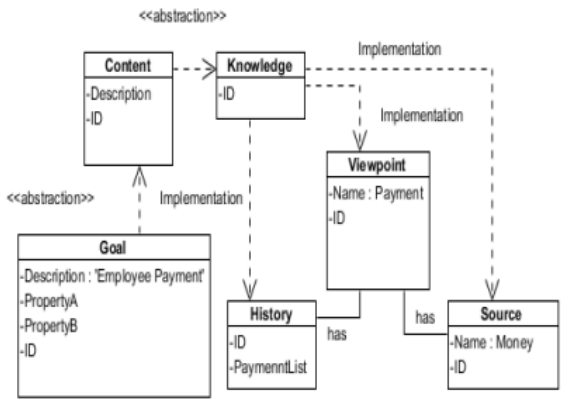

Figure 6: Translaton of goal concept in i* model to PREview concepts.

For each employee, there exists a payment viewpoint associated to this employee. This viewpoint encapsulates information about the payment such as: payment type (Hourly, Salaried, etc.), amount and so on. History class contains a list of payment records that have been carried out. Source class represents the money used to pay employees.

7 CONCLUSION

This paper has presented a unified requirements engineering meta-model that is resulted from a semantic unification process of different requirements engineering meta-models. The unification process is based on finding semantic similarities between different concepts that already exist in different types of requirements engineering meta-models. The aim of the unified requirements engineering meta-model UREM as mentioned in section two and three is to perform translation between different types of requirements engineering models in order to allow cooperation between companies that have different cultures and use different kinds of Requirements Engineering approaches. The translation rules are deducted in section four directly from the unified meta-model UREM. One of the gaps of these translation rules is the lack of concrete semantic translation at attributes level of concepts. We seek to fix this issue of semantic translation between concepts in a future work. We seek also to demonstrate other features of UREM such as evolution and composition. Evolution is how UREM is easy to update and maintain. Composition is how to compose a full requirements specification document of a project

from different pieces of requirements specifications arisen from different models. Afterward, we are looking to implement a visualization tool in order to present and illustrate the translation operation between models as graphs.

REFERENCES

Sommerville, I., Sawyer, P., 1997. Requirements Engineering: A Good Practice Guide. *John Wiley & Sons, Inc. New York, NY, USA*. ISBN: 0471974447.

Bendjenna, H., Zarour, N.E., Charrel, P.J., 2010. Eliciting Requirements for an inter-company cooperative information System. *Journal of Systems and Information Technology (JSIT).*

Cares, C., Franch, X., 2011. A Metamodelling Approach for i* Model Translations. *23rd International Conference, CAiSE 2011.*

Beleg, G. Design and prototypical implementation of a pivot model as exchange format for models and metamodels in a qvt/ocl development environment. Ph.thesis, *Université de technology Dresden, 2007*.

Milanovic, M., Gaševic, D., Giurca, A., Wagner, G., and Devedžic, V. On interchanging between owl/swrl and uml/ocl. In *Proceedings of 6th Workshop on OCL for (Meta-) Models in Multiple Application Domains (OCLApps) at the 9th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MoDELS), Genoa, Italy, pages 81–95, 2006.*

Zekai, Y. S., Jouault, F., Tairas, R. and Gray, J. Software language engineering. A Model Engineering Approach to Tool Interoperability, pages 178–187. *Springer, 2009.*

Saidi, I. E., Dkaki, T., Zarour, N. E., Charrel, P. J., 2012. Towards Unifying Existing Requirements Engineering Approaches into a Unified Model. *KMIS 2012: 311-315.*

George, A. M., 1995. WordNet: A Lexical Database for English, *Communications of the ACM. Vol. 38, pp. 39-41*

Castro, J., 2011. Goal Oriented Requirements Engineering i*, *Fifth International Conference on Research Challenges in Information Science.*

Sutcliffe, A. G, Maiden N., Shailey, M., Darrel, M., 1998. Supporting Scenario-Based Requirements Engineering, *IEEE Transactions on software engineering, vol. 24, No. 12.*

Sommerville, I., Sawyer, P., 1997. Viewpoints: principles, problems and a practical approach to requirements engineering. *Computing Department, Lancaster University, UK, Annals of Software Engineering.*

Lesk, M., 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone.

Satanjeev B., Ted P., 2002. An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet, Computational Linguistics and Intelligent Text *Processing Lecture Notes in Computer Science Volume 2276, 2002, pp 136-14.*