Requirements Eng (2011) 16:133–160
DOI 10.1007/s00766-011-0116-1

ORIGINAL ARTICLE

Integrating ontologies, model driven, and CNL in a multi-viewed approach for requirements engineering

Paulo F. Pires · Flávia C. Delicato · Raphael Cóbe · Thais Batista · Joseph G. Davis · Joo Hee Song

Received: 21 June 2010 / Accepted: 16 January 2011 / Published online: 17 February 2011

Abstract Research in requirements engineering (RE) has been growing in the last few years. RE researchers are generally concerned with a set of open issues such as: (i) the need for a well-defined process to identify and specify the requirements scope, (ii) suitable mechanisms to support communication among different stakeholders and development teams involved in the RE process, (iii) mechanisms to deal with the inherent volatility of requirements, and (iv) the need for a traceability scheme to help managing requirements in the downstream phases of the development process. In this work, we address some of these open issues by proposing the use of an iterative and incremental model-driven RE process combined with the employment of different notations such as controlled natural language and ontology in each activity of RE process. Based on the argument that there is no single notation suitable to represent requirements from the different perspectives of all the stakeholders and development teams, we propose a RE process encompassing different views, representing each perspective. This paper describes the proposed process, its tool support, and presents a controlled experiment that illustrates the proposal and evaluates its benefits.

Keywords Requirement engineering process · Model-driven development · MDA · Ontology · Controlled natural language

P. F. Pires (✉) · F. C. Delicato
Department of Computer Science, Institute of Mathematics, Federal University of Rio de Janeiro (UFRJ), Bloco E, CCMN/NCE, Cidade Universitária, PO BOX 68.530, Rio de Janeiro, RJ 21941-590, Brazil
e-mail: paulo.f.pires@gmail.com

F. C. Delicato
e-mail: fdelicato@gmail.com

R. Cóbe · T. Batista
Department of Informatics and Applied Mathematics (DIMAp), Federal University of Rio Grande do Norte (UFRN), Natal, RN 59072-970, Brazil
e-mail: natalpunk@gmail.com

T. Batista
e-mail: thaisbatista@gmail.com

J. G. Davis · J. H. Song
School of Information Technologies J12, University of Sydney, Sydney, NSW 2006, Australia
e-mail: joseph.davis@sydney.edu.au

J. H. Song
e-mail: json8472@uni.sydney.edu.au

1 Introduction

The success of a software system can be measured by the degree to which it meets its envisaged purpose. Software system requirements engineering (RE) can be defined as the process of discovering such purpose by identifying stakeholders and their needs and documenting them in a way that is amenable to analysis, communication, and subsequent implementation [1]. RE activities aim at managing all the requirements-related knowledge. It is common that such knowledge is concretized in a set of artifacts such as use cases, story boards, natural language documents, and business process specifications. These artifacts comprise the so-called *Requirements Document*. The production of such document is often regarded as one of the most difficult activities in the software development process [2]. The resources applied in building a solid RE process have been shown to pay off. However, studies conducted by renowned IT consulting groups as the Standish, the Gartner, and the Forrester groups have pointed out that a large number of projects still fail to achieve their goals and some of them are even canceled due to requirements-related issues [2–4].

According to Pressman [5] and Sommerville [6], a typical RE process is divided into three phases, which are executed in an iterative way: (i) requirements elicitation and analysis, (ii) requirements specification, and (iii) requirements validation. During these phases, the following activities are commonly performed:

1. Knowledge acquisition [7–9], which is the process of gathering requirements about the system to be built. This process is usually performed by the requirements engineer through interviews, group brainstorms with stakeholders, or analyses of reports, forms, spreadsheets, and/or any other type of information relevant to the system development;
2. Knowledge representation [10–12] whose goal is to facilitate the communication and sharing of the gathered requirements between system designers and stakeholders (end-users, procurement agents, etc.) as well as to document such information for further use. Such documentation can be both textual and graphical.
3. Knowledge conflict management and requirements validation [13–15] aim at assuring that the gathered requirements are not conflicting with each other and that they meet all client demands. These activities encompass the understanding of and reasoning on large system specifications, the propagation of design decisions throughout the system description, and the merge of possibly conflicting requirements that have been acquired from groups of analysts working simultaneously.

Besides these common activities, since requirements are inherently volatile, knowledge evolution management [16–20] is another import activity that has not always been considered in traditional RE processes. Evolution of requirements refers to the changes that take place in a set of requirements after the initial phases of requirement engineering [21]. According to such definition, changes in requirements that may happen in initial elicitation, analysis, specification, and validation phases are not evolutionary. Such changes in requirements are additions, omissions, or modifications of requirements [22]. Requirements commonly evolve due to the knowledge brought up during software development or they change because of unforeseen organization needs, environmental pressures, or the advent of new technologies [16]. Requirements also change over time because they are collected from several different sources (typically interviews with stakeholders), which may have different or even contradictory points of view about the system to be built [23].

1.1 The addressed problem: open issues in the RE activities

Since a good requirement engineering process brings valuable benefits in the development of software systems such as preventing errors, improving the quality of the final product, and reducing risks, both academia and industry have been investing in research focusing on improving the aforementioned requirement engineering activities. Researchers are especially interested in identifying the most significant drawbacks of the existing requirement models and processes. Some of these research endeavors [16, 24–27] have already given some clues on what is lacking in extant RE processes. According to these authors, the most common open issues in the RE are

1. The need for a well-defined process to identify and specify the *requirements scope*;
2. Better mechanisms to support *communication* among different stakeholders and development teams involved in the RE process;
3. Better mechanisms to deal with the inherent *volatility* of requirements; and
4. The need for a *traceability* scheme to help manage requirements in the downstream phases of the development process.

Each open issue is directly related to one or more of the RE activities. In the following sections, we briefly discuss these issues and their inter-relationships and review current research efforts that address each open issue.

1.1.1 Scope issues

The system scope is defined in the RE activity of knowledge acquisition. Such RE activity aims at establishing the boundary conditions and the goal for the target system. Currently, knowledge acquisition is mainly carried out by the requirements engineer who has to interpret the outcome of the interviews with clients. During the interpretation process, the requirements engineer should be able to distinguish the useful information from the useless information provided by the stakeholders [5, 17]. Misinterpretations in the activity of knowledge acquisition can lead to requirements that are incomplete, not verifiable, unnecessary, and unusable. To cope with this ***requirement scope*** issue, several works have proposed methodologies to acquire knowledge directly from stakeholder descriptions. The authors in [28–30] suggest the employment of a restricted natural language (i.e., *controlled natural languages*) to represent the requirements. Elicitation methodologies that produce requirements in the form of specific modeling RE notations such as use cases [5] are prone to generate ambiguous requirements to the stakeholders. These requirements may not be verifiable by the stakeholders since they cannot adequately understand the used notation. In contrast, the use of controlled natural languages allows requirements to be represented in a way closer to the natural human representation while keeping it

free of ambiguities and imprecision, thus minimizing misinterpretations since users can easily validate such descriptions.

1.1.2 Communication issue

After gathering requirements from the stakeholders, the knowledge representation activity takes place. In this activity, the appropriate notation(s) to represent requirements is crucial to facilitate communication among the different development teams and stakeholders [31]. A misjudgment during the knowledge representation may lead to an RE process that suffers from communication issues. Many works [31–33] argue that the employment and integration of different notations are necessary to represent requirements according to the point of view of different stakeholders and development teams. Moreover, there is a consensus in the RE community that such notations should be standardized in order to minimize communication issues. For instance, the different UML views have been widely used as standard notations to represent requirements [6].

1.1.3 Volatility issue

The next activities in a RE process are knowledge conflict management and requirements validation. A failure during the validation of the new requirements may lead to the occurrence of problems related to *volatility* issues in the RE process. Research in the area of requirements validation commonly use formal logic and inference mechanisms to check the consistency of the requirements and to assure that they are not in conflict with each other. Following this idea, Wang et al., Lenzerini, Kaiya and Saeki [13, 15, 34] proposed the use of first-order logic and ontologies to specify and validate requirement models.

The knowledge evolution management activity is also related to the occurrence of volatility issues in the RE process. The main approach to deal with the inherent volatility of requirements is to consider that the requirements engineering process of eliciting, specifying, and validating should not be executed only once during system development, but rather should be an iterative and incremental process [16, 20].

1.1.4 Traceability issue

Both conflict management and knowledge evolution activities are deeply related to the traceability scheme. The RE traceability is concerned with relating requirements with other system artifacts and it allows the life cycle of a requirement artifact to be followed both forwards and backwards throughout a software development process [35]. Such links between artifacts should allow the recording of meta-data about the RE process [36] such as the stakeholder name, who the interviewer was, when the interview occurred, etc. [37]. A traceability scheme can help to identify changes in requirements, predicting their impacts on the later development process phases, thus reducing the costs associated to requirements volatility. Works in the traceability [37, 38] area are mostly concerned with proposing (semi-) automatic processes to maintain trace dependencies (element that links to artifacts) that are updated throughout all views used to represent requirements. Researchers are also concerned with defining the means to trace the artifacts contributors (the agents who have contributed to artifact production) as, for instance, the *Contribution Structures* proposed by Gotel et al. [36].

1.2 The proposed approach

The main goal of this work is to address the aforementioned RE open issues through the use of an iterative and incremental model-driven RE process combined with the employment of different notations in each activity of such RE process. As already mentioned, there is no single notation suitable to represent requirements from the different perspectives of all the stakeholders and development teams. Therefore, we propose a RE process encompassing different views, representing each perspective.

In order to represent the requirements from the client's point of view, we propose the adoption of *Controlled Natural Language*—CNL [39]. This adoption aids the activities of knowledge acquisition and knowledge representation. It provides a notation close to the client native language, which is fully understandable by both requirements engineer and client, thus facilitating the comprehension of, reasoning on, and validation of the requirements specification. Therefore, the use of CNL partially addresses both scope and communication issues.

To represent the requirements from the development team's point of view, we propose the use of ontologies. This representation is initially built by using the system descriptions specified in CNL, further refined, and incremented by the requirements engineer. The development team needs a more structured and manageable representation of requirements in comparison with the clients'. Moreover, they need mechanisms to access, search, and reason over the requirements. The use of ontologies fulfills these needs; it allows a high-level and precise representation of a given domain in terms of the key concepts and their relations. Moreover, it is possible to reason about a given ontology using available reasoning mechanisms. In the proposed approach, ontologies are used in the knowledge representation, validation and evolution RE activities, addressing communication and volatility issues.

The client and development views must be integrated in order to avoid inconsistencies and conflicts, thus addressing communication issues, and to allow the traceability of requirements between such views. We propose a model-driven [40] solution to integrate such views and keeping them consistent. More specifically, we adopted model-driven architecture (MDA) [41] approach. The use of automatic model transformations allows the seamless synchronization of these views and also helps in the provision of a traceability scheme. In our work, the MDA paradigm provides the underpinning of the RE process that ties together all the activities. Besides, its use minimizes the negative effects of requirements evolution, since MDA transformation can assure that requirements are synchronized and up to date in all views. Regarding the MDA abstraction levels [40], we focused on the *Computer Independent Model*, CIM, to represent the requirements document, since requirements should not contain any information on the technology to be used in the implementation of the system.

In this paper, we present the proposed model-driven RE process and show how each different notation (ontologies and CNL) is used to represent requirements in each considered view. Moreover, we present the tool implemented to support the proposed process. We also report on a controlled experiment [42] conducted to illustrate the proposal and assess its usability and benefits. The remainder of this work is organized as follows. Section 2 details the proposed approach. Section 3 describes the implemented tool. Sections 4 and 5 present the controlled experiment and its analysis. Section 6 describes related work, and finally, Sect. 7 discusses the results obtained and point out future research directions.

2 Multi-viewed approach for requirements engineering

To encompass all the aforementioned activities involved in the requirement engineering as well as to address the open issues related to each activity, we propose an iterative and incremental *model-driven RE Process* [40]. The phases encompassed in our process are aligned with traditional RE processes, as defined in the literature [5, 6], although we have proposed new activities in some of these phases. In our approach, the *Requirements Document* is represented at the MDA Computational Independent Model (CIM) abstraction level. To address the different needs, we propose the division of the CIM in two views: the client view and the development team view. It is important to note that although a software system encompasses both static (structural) and dynamic (behavioral) features, our work is focused only on the static representation of the system (dynamic representation is out of scope of our proposal).

Even though we focus on the CIM model, the outcome of the proposed process is a skeleton of an MDA Platform Independent Model (PIM) representing the gathered requirements. Therefore, our model-driven RE process also encompasses a phase to generate this PIM. We suggest the adoption of UML (Unified Modeling Language [43]) class diagrams as the main notation for this PIM. In fully compliant MDA development process, such artifact will be useful in the downstream phases of the system development lifecycle. The proposed PIM, represented as a UML class diagram, also aims to facilitate the communication through the entire software development lifecycle since the design phase often uses UML notations to represent the system [44].

The following sections describe the models and views used in our proposal and show how these models and views are used to address the RE open issues discussed in foregoing. Details of the proposed process follow.

2.1 Proposed models and views

We argue that due to its complexity, the knowledge encompassed in RE cannot be fully expressed with a singled view model. To deal with this expressiveness problem, we propose organizing the requirement specification in two main abstraction levels, the **client view** and the **development team view**. In the proposed MDA RE process, the client view lays at the CIM level while the development team crosses both the CIM and PIM levels. We also propose a mapping strategy between these two views which facilitates the communication between the different actors involved in the RE process, thus addressing *communication* issues. The proposed mapping strategy is also able to trace dependencies between views, making it possible to follow the requirements throughout the development cycle, keeping metadata about the requirements elicitation process. Such capability addresses the *traceability* issue. For instance, it enables to know, at a design phase, which stakeholder has provided information about a given entity of the system.

2.1.1 CIM: Client view

This view is used in the RE activities of knowledge acquisition and representation to build a representation of the gathered knowledge at an abstraction level that facilitates the client to understand such knowledge, thus promoting his/her active participation in the requirements elicitation process. To achieve this goal, requirements are represented in this view through a controlled natural language (CNL). This approach borrows ideas from existing research on CNL such as [29, 45]. However, it differs from those in that it is based on the MDA approach as well as

being aligned to existing RE processes. Representing requirements with natural language unburdens the stakeholders of learning specific RE modeling notations, which are not typically part of their expertise. Therefore, the adoption of natural languages facilitates the communication between stakeholders and requirements engineers, minimizing potential misunderstandings and misinterpretations, thus partially addressing *communication* issues.

The knowledge represented in the client view is obtained through interviews or documents related to the business being modeled. These documents have to be aligned with the restrictions imposed by the use of CNL, so they need to be re-structured according to the CNL syntax and grammar. In our process, we used the *Attempto Controlled English*—ACE [39] CNL. Our choice for the ACE controlled language was based on its expressiveness, simplicity to learn and use and on the fact that it is also powerful enough to represent the fundamental concepts used to specify requirements, such as *entities*, *relationships* and *restrictions*.

Current tool support for ACE [46] does not have a defined traceability scheme. Therefore, in order to deal with *traceability issues* we have added the capability of tagging ACE sentences in our client view. Tags are metadata used to mark sentences. Using tags allows keeping track of *when* a statement was given, *who* made that statement and *what* is it about (its subject). The tags are kept through all the RE process. So, it is possible to know which concepts were produced and by whom, in each interview. For example, it is possible to know that the domain class *client* is related to the concept *client* that first appeared during an interview with the business specialist on *December 10*, *1966*. The tagging mechanism is highly generic and any kind of metadata information can be assigned to sentences.

2.1.2 CIM: Development team view

The needs of the development team regarding the handling of requirements are different from the client needs. Based on the study by Kaiya et al. and Breitman et al. [34, 45], we enumerated the most common of such needs.

1. A more structured notation to represent requirements that allows identifying concepts, their properties and relationships;
2. Strategies to manage the complexity and evolution of requirements;
3. Tools that facilitate the access to, authoring of, reasoning on, and searching for requirements.

In order to fulfill all these needs, the proposed Development Team View at the CIM level is represented through a domain ontology. More precisely, we propose the use of OWL [47] to model ontologies in this view. The first need is addressed since the use of ontologies has the potential to allow a high-level and precise representation of a given domain in terms of its concepts and their relations.

The second need is addressed by two mechanisms that are available when using ontologies: reasoning and merging, as well as by a semi-automatic MDA mapping provided as part of our proposal. The use of ontologies allows reasoning about the knowledge they represent. Commonly, the reasoning encompasses two activities. The first one is the consistency checking, where the reasoning mechanism verifies if there is any individual breaking any class constraint (ABox reasoning). The reasoning mechanism also verifies if any class structural constraint is violated (TBox reasoning), for instance, if any disjoint class have descendants in common. The second activity executed by a reasoner is the taxonomic classification, where the mechanism uses the logical description of the concepts and their relationships and infers new relations between concepts. All the activities performed by a reasoner are a valuable aid for managing complex requirements since they provide the possibility of validating relationships, inferring new relationships among concepts, and identifying conflicts among requirements at an early developmental stage. The evolution and change of requirements represented as ontologies can be managed by using well-established ontology merging strategies, such as in references [48–50]. The merging enables reusing part of or the entire previous version of a domain ontology (requirements specification). In our proposed process, the development team view is built by a (semi-)automatic MDA mapping that takes the client view (in CNL) as input and through a set of transformations generates a preliminary domain ontology as output. Therefore, the MDA mapping aids managing the complexity of requirements since the outcomes of interviews are automatically parsed and transformed into structured knowledge, i.e., concepts, properties and relationships.

The third need is achieved by using ontologies notations and languages that are supported by standards bodies such as OMG and W3C.[1] Standardization efforts have been producing specifications that are broadly implemented as tools provided by several companies, many of them as open-source solutions. Such tools provide a set of functionalities to easily access and handle requirements.

Regarding the RE open issues addressed in this paper, the adoption of ontologies to represent the proposed Development Team View deals with both *communication* and *volatility issues*. Communication is addressed by the ontology capability of providing a sharable domain knowledge that stakeholders and developers should agree

[1] http://www.w3c.org.

on. Volatility is addressed by ontology reasoning and merging mechanisms which contribute to keep the requirement knowledge base free of inconsistencies possibly generated by inclusion of new requirements and/or changes in old requirements. The MDA mapping used to convert the CNL to ontology addresses both communication and traceability issues. From the development team point of view, since the transformation between different notations (CNL to ontology) is automatically done by the mapping, it avoids misinterpretation or human errors. The traceability is addressed by the automatic propagation of CNL tags to ontology annotations [47].

2.1.3 PIM: Development team view

In order to help the development team in the design phase, our approach generates a PIM skeleton in UML notation to avoid the need of starting the modeling from scratch. The provided PIM contains an UML class diagram, with the mapped entities and their relationships derived from the CIM domain ontology. The provision of this PIM skeleton through an automatic MDA transformation addresses both the *communication* and the *traceability* issues. Similarly to the approach to generate the ontology from CNL statements, since the transformation between ontology and UML classes is automatically done, it also avoids misinterpretation or human errors. The transformation process responsible for the PIM generation is also able to propagate the metadata about interviews from ontology and annotations to class stereotypes, thus ensuring that the information provided during the client interviews is present at the PIM classes. Such information is used for traceability purposes, helping to group classes, for instance, by the interview subject.

2.2 Proposed process

The proposed model-driven RE process that we developed follows the RE process definition stated by Pressman [5] and Sommerville [6]. It is divided into the following phases, which are executed in an iterative way: (i) requirements elicitation and analysis, (ii) requirements specification, and (iii) requirements validation. However, we added new activities to manage requirements evolution and also to produce the UML class diagram that is used as a start-up for the design phase.

Each activity in our process is carried out by a specific actor. We have included one new actor to the set described in works as [5, 6], namely the ontology engineer. Therefore, the set of actors in our process is composed of:

- Requirements engineer: responsible for collecting information from the client and registering all information that is useful for the system modeling.
- Ontology engineer: responsible for managing requirements represented as a domain ontology, aided by tools for ontology analyze, merge and validation.
- System Analyst: responsible for the system design.

The process with its artifacts and actors is illustrated in the UML Activity Diagram in Fig. 1.

The first phase of our process is the requirements elicitation and analysis. The activities performed in this phase are conducted by the requirements engineer with the main goal of gathering system information from the stakeholders. Such activities are (i) interview stakeholders (activity 1a, Fig. 1), (ii) review documents (activity 1b, Fig. 1), and (iii) register relevant information (activity 1c, Fig. 1). Often the interviews and documents provided by the client include unnecessary information, so the relevant information has to be initially filtered in order to build a concise view of the system requirements. Activity 1c is responsible for accomplishing such goal. In this activity, the requirements engineer builds a system description in CNL represented in the proposed RE process as the Client view of the system. In our proposal, CNL descriptions are written using *Attempto Controlled English*—ACE [39]. In order to provide traceability support, during this activity the requirements engineer must include metadata information about the process of requirement acquisition, such as date of the interview, name of the interviewed stakeholder and subject of the interview. Such metadata information is stored as tags at each sentence of the interview.

The second phase is the RE specification, in which the CIM of the development team view is built, i.e., the system domain ontology. The activities performed in this phase are: build partial domain ontology (2a.), search for reusable ontologies (2b), and merge ontologies (2c). The partial domain ontology model is automatically derived from the CNL descriptions through a MDA transformation provided as part of the Attempto project [39]. Such partial ontology encompasses all the knowledge described in the CNL sentences produced in the previous phase.

After building the partial ontology, the ontology engineer can augment it by reusing some previously defined knowledge base (activity 2b). Since we are proposing an iterative and incremental process, such predefined ontology may have been built on previous iterations of the process, through automatic MDA transformations from CNL to ontology. On the other hand, this predefined ontology can also be a well-known domain ontology used to represent a specific aspect of the system, for example, a monetary ontology to represent monetary concepts. Once a set of ontologies is selected for reuse, an ontology merging operation is carried on (activity 2c). Such reuse of ontologies is not mandatory (however it is encouraged), and the process can go on without any ontology merging. In this

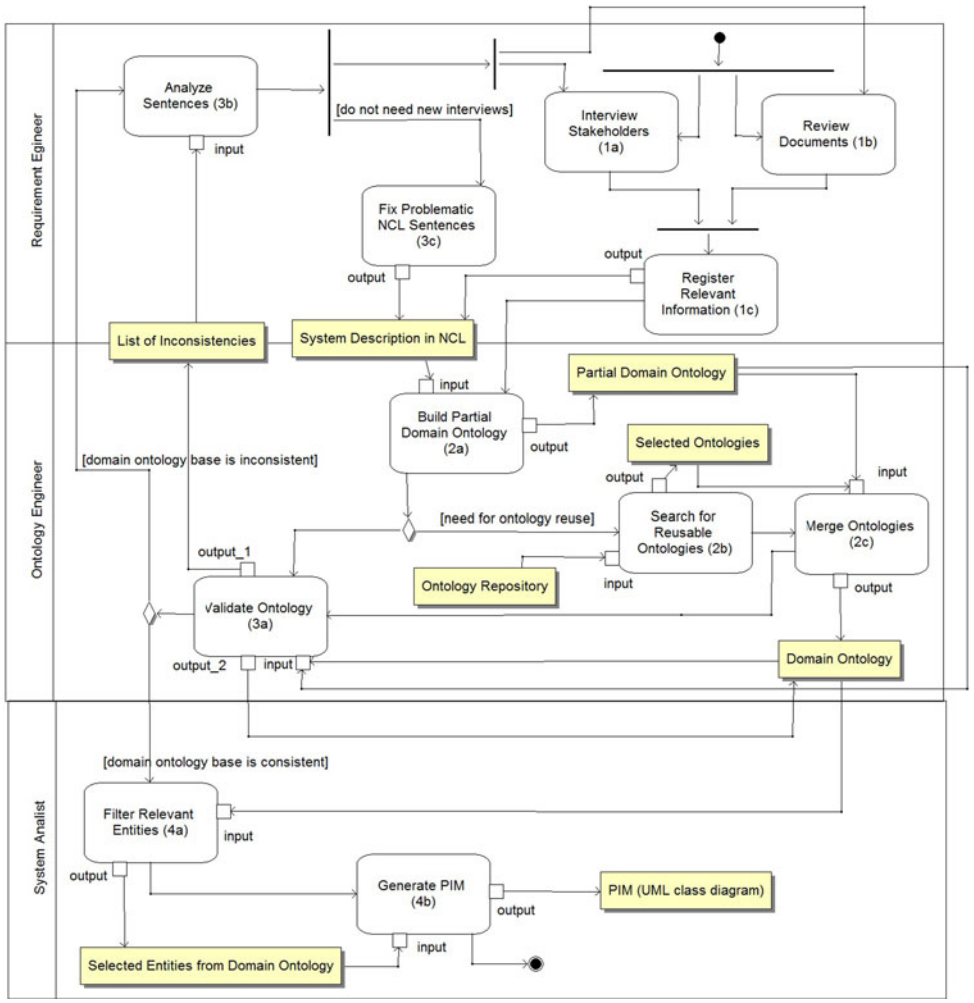

Fig. 1 UML 2.0 activity diagram of the proposed process

case, the partial ontology is promoted to domain ontology. By the end of the second phase, the CIM view is complete and ready to be validated.

The third phase of the proposed process is the requirements validation that aims to find inconsistencies or conflicts at the requirements knowledge base. The activities performed in this phase are: validate ontology (activity 3a), analyze sentences (3b), and fix problematic sentences (3c). To accomplish the goal of such phase, we propose a (semi-)automatic approach that uses inference mechanisms implemented by ontology reasoners. During the validation phase, two types of inconsistency are checked: *ABox* and *TBox*. TBox sentences describe a system in terms of controlled vocabularies, for example, a set of classes and properties, while ABox are TBox-compliant sentences about instances of those vocabularies. Therefore, *ABox inconsistencies* refer to inconsistencies between facts (individuals) and structural rules (concepts and relations), while *TBox inconsistencies* are inconsistencies between concepts and relations constraints (constraints violations). If ABox inconsistence occurs, the inconsistent knowledge base becomes locked until the conflict is solved. This happens because most of the reasoners are unable to reason over a knowledge base with this kind of inconsistency. Whenever any of these two types of inconsistence occurs, the process cannot go forward, i.e., it enters in a loop and only leaves the loop if the problem is fixed.

The outcome of the ontology validation (activity 3a) is a list of detected inconsistencies which includes the concepts related to such inconsistencies as well as the CNL sentences that originated each of these concepts. The relationships among concepts and CNL sentences are tracked down by the traceability scheme provided by the proposed approach (see details in Sect. 2.2).

The requirements engineer analyzes the generated inconsistency list (activity 3b) verifying if it is possible to solve the detected problems by himself/herself. If so, the requirements engineer performs the necessary modifications in the existent CNL system descriptions (changing and/or removing the problematic sentences) to resolve the

conflicts (activity 3c). Following, the process is resumed to activity 2a. Otherwise, new interviews or document reviews should be carried out. In this case, the process is resumed to its initial activities (1a, 1b).

Besides detecting inconsistencies in the knowledge base, the use of a reasoner during the validation phase has also the power to uncover new relationships of type *is-a* between the concepts, during the taxonomic classification occurred as part of the ontology validation (activity 3a). These relationships are built based on inference rules executed against the domain ontology concepts (i.e., ontology classes) and their constraints that were defined by the CNL declarations and transformed into ontology sufficient constraints. These constraints are typically built from sentences that start with "*Every*" or "*Everything*" and define new concepts from old ones or from object properties. For example: suppose that at a medical domain scenario, a stakeholder declared that "*Every person that has a disease is a patient*", where the concept *patient* is defined in function of the *has* property. This sentence also constrained the *has* property range to only accept values of the *disease* concept, i.e.

$$C = \{\forall x \in Person \exists y \in Disease \mid\mid has(x,y) \rightarrow x \in Patient\} \quad (1)$$

Afterward, another stakeholder states that "*Every victim is a person and every victim has a disease*" i.e.

$$V_1 = \{\forall x \in Victim \mid\mid x \in Person\} \quad (2)$$

$$V_2 = \forall x \in Victim \exists y \in Disease \mid\mid has(x,y)\} \quad (3)$$

Thus, during the taxonomic process, the reasoner can infer, based on axioms 1, 2 and 3 that "*Every Victim is a Patient*" i.e.

$$P = \{\forall x \in Victim \mid\mid x \in Patient\} \quad (4)$$

The next phase of our RE process is the generation of the PIM skeleton, which is not commonly included within traditional RE processes. We consider this phase as a transitional phase to the next software development phase. The goal of this phase is to automatically build a useful artifact for the design phase of software development life cycle. The PIM generation is carried on by a System Analyst that executes a model-driven transformation taking as input the RE knowledge base generated in the developer team's view and producing the PIM as the output. In this phase two activities are performed: filter relevant entities (activity 4a) and generate PIM (activity 4b). In the activity 4a the System Analyst selects the relevant information from the domain ontology. Finally, in activity 4b, the PIM skeleton is generated from the selected elements of the domain ontology, through a set of MDA transformations provided by the tool developed to support our approach.

3 Tool support

We developed a tool to support the proposed RE process, named *RETool*, comprising a Protégé plug-in, named *REView*, as well as a set of components, which implement functionalities related to model-driven transformations. Figure 2 depicts the tool architecture, with its components drawn in gray. The *REView* plug-in is responsible for registering the descriptions written in CNL, merging and validating ontologies. The model-driven transformation components are responsible for transforming CNL descriptions into OWL ontologies and building UML class models from such OWL ontology models. The following subsections describe the details of the implemented *RETool*.

3.1 The *REView* plug-in

The developed *REView* plug-in (Fig. 2) builds on the existing *ACEView* plug-in [46] and incorporates several standard capabilities of Protégé. We chose the Protégé editor [51] for handling ontologies due to its popularity as an OWL editor, simplicity to use and availability of its source code. Moreover, Protégé is the native platform for *ACEView*, which is one of the few currently available ACE language editors.

In the implemented plug-in, *ACEView* is used as a CNL editor, thus every interaction with the CNL descriptions occurs through the *ACEView*. We extended the *ACEView* to add the tagging capability in order to implement our interview traceability scheme. The implemented extension is represented as the component *ACEViewWithTagging* in Fig. 2. The requirements engineer uses the editor to annotate each CNL statement with a set of tags that identifies the interview to which the sentence belongs to. To encapsulate an interview, we proposed a structure called *Subject-Date-Stakeholder*—SDS, which is composed of information (tags) about the requirements elicitation process (date of interview, name or role of Stakeholder and subject of the interview). For example, suppose that at 1 April 2009 the requirements engineer interviewed a security analyst from a given company in order to collect information about the user control subsystem. The SDS tags for this situation would be: "*User Control*", "*1/4/2009*" and "*Security Analyst*". The SDS structure must be filled for each sentence. The set of sentences marked with the same SDS composes a complete interview.

In the current version of the implemented *REView* plug-in, the activities of ontology merging and validating are accomplished by the Pellet reasoner [52]. However, we have tested other reasoners, such as Fact++ [53], and they provided the same results of Pellet, indicating that different reasoners can be possibly chosen for use by the ontology

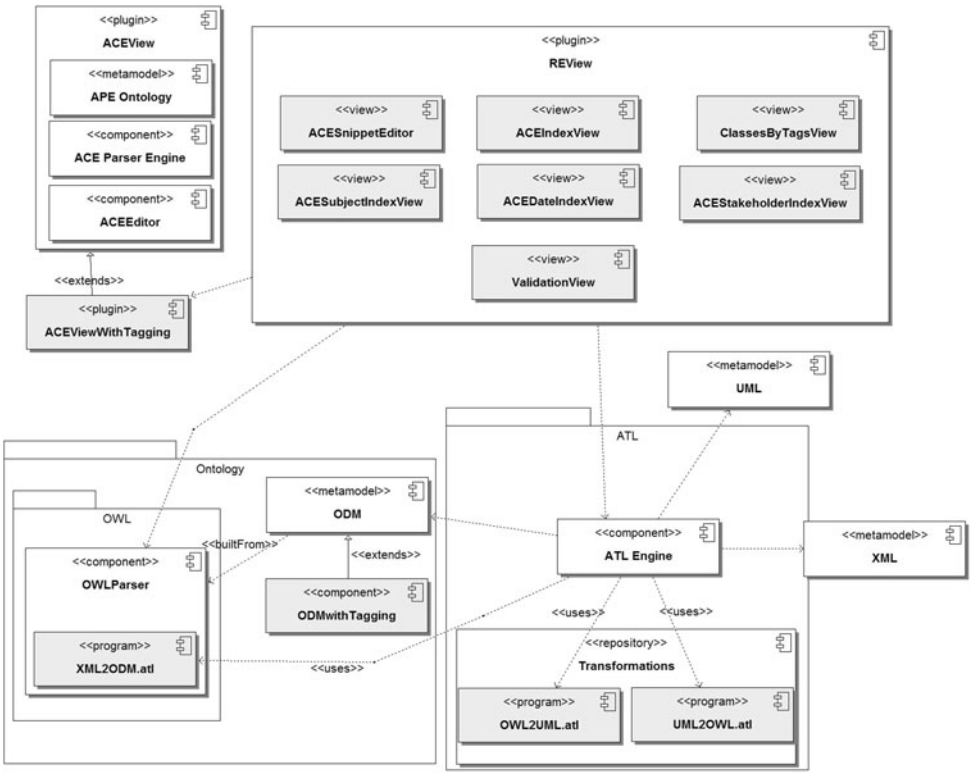

Fig. 2 *RETool* architecture

engineering without affecting the outcome of the proposed RE process.

Besides extending the *ACEView* plug-in with a traceability scheme for interviews, the *REView* plug-in defines a set of views that aid both the requirements engineers and the ontology engineers to interact with the implemented *RETool*. Such views are represented as components tagged with the <<view>> stereotype in Fig. 2. For each created new view, we implemented a new component, responsible for listening to Protégé events and showing in the respective view the correct information according to such events. Following we discuss each one of these views.

1. ACESnippetEditor (Fig. 3a): this view is used to create, remove and edit ACE sentences (snippets in the ACEView plug-in jargon) along with user defined tags and the SDS tags associated with each sentence. This view is composed of: (a) a list of all sentences registered by the requirements engineer; (b) fields to edit and inspect ACE sentences; and (c) fields for inspecting metadata about the registered ACE sentences (for instance the list of user tags used to annotate a sentence);
2. ACEIndexView (Fig. 3b): this view lists ontology concepts (OWL classes) and the ACE sentences that make reference to them;
3. ClassSortedByTagsView (Fig. 3c): this view groups ontology classes according to the general tags that annotate them;
4. ACESubjectIndexView (Fig. 3d): this view groups ACE sentences by the subject tag which annotates them;
5. ACEDateIndexView (Fig. 3e): This view groups ACE sentences by the date tag which annotates them;
6. ACEInterviewIndexView (Fig. 3f): This view groups the ACE sentences by the stakeholder tag which annotates them;
7. ValidationView (Fig. 4): this view provides the functionality to check the consistency of the underlying ontology and, whenever there are conflicting concepts, it shows the snippets which contain them, classified by subject and date tags.

3.2 Components for MDA transformations

The implemented *RETool*, encompasses three different model-driven transformations. The first transformation has the goal of translating ACE textual descriptions into OWL files representing the domain ontology. Such translation process synchronizes the client view with the first model of the development team view. The first model-driven transformation is performed by the *ACE Parser Engine*—APE [54] component, developed and provided by the Attempto project. This component provides features to edit and parse textual descriptions as well as to build ontologies from such descriptions. We extended the ontology metamodel of APE to add the tagging mechanism used for implementing

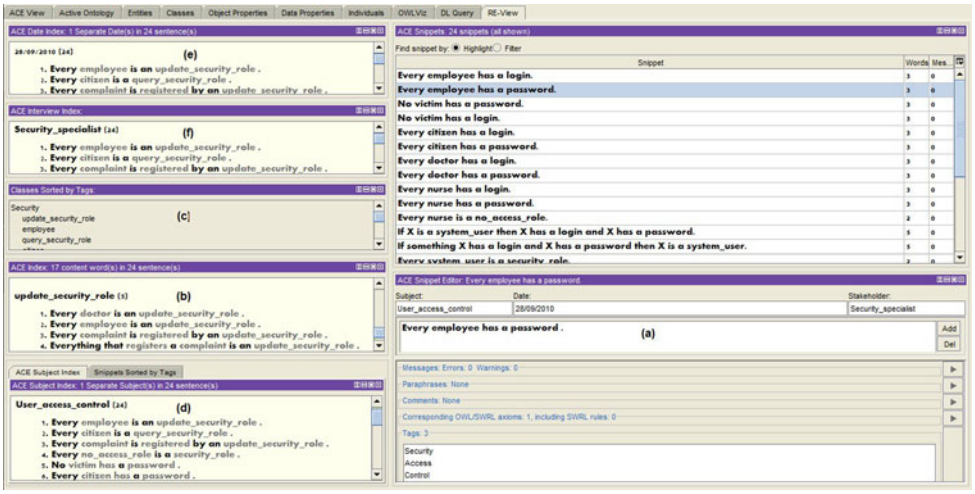

Fig. 3 The views of the implemented *REView* plug-in

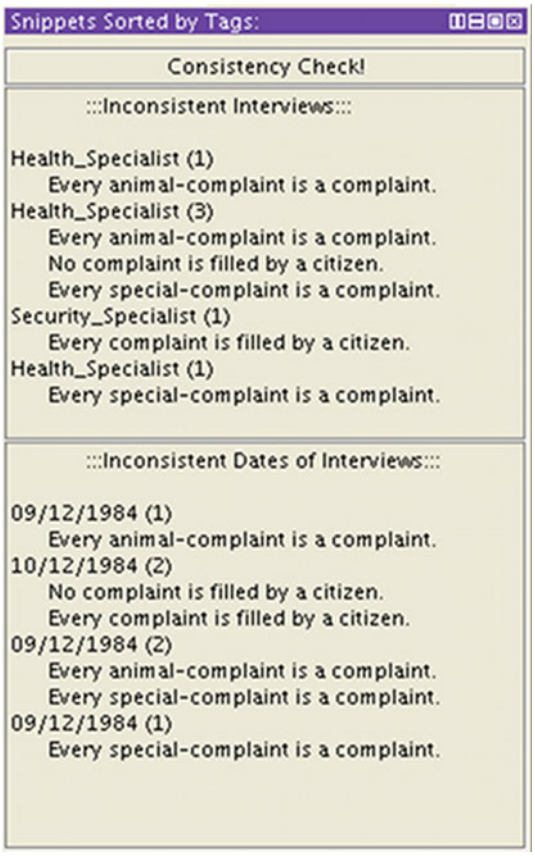

Fig. 4 *ACEView*'s validation view

the traceability scheme. The extended metamodel is included in the *ACEViewWithTagging* component in Fig. 2. The tagging was implemented by a generic mechanism so that any kind of metadata information can be associated to the sentences, besides the proposed SDS itself. The output of APE is an OWL ontology represented in XML. However, to be used as input in the further chain of MDA transformations performed by our tool, we chose to adopt the OMG standard metamodel for ontologies, ODM (Ontology Definition Metamodel [55]). We made a minor modification in ODM, by introducing the *Tag* concept, which is responsible for keeping the SDS metadata (for purposes of traceability, as previously explained). Therefore, the second transformation in *RETool* is performed by the *OWLParser* component (represented in the package Ontology of Fig. 2) using a set of ATL rules that are executed against the XML metamodel. We chose the ATL model transformation language [56] to implement these transformations due to its maturity level and active development community.

The third transformation has the goal of translating the domain ontology into an UML class diagram. This translation process synchronizes the high-level model of the development team view (CIM level represented as an OWL ontology) with a lower level model of the development team view (PIM level represented as an UML class diagram) that is more suitable to be manipulated in further software development phases. This third model-driven transformation, represented as the package ATL in Fig. 2, is performed by a set of ATL rules that map OWL concepts formally specified according to ODM into UML classes.

Besides ODM, the implemented *RETool* makes use of two other metamodels: UML [57] and XML [58]. The XML metamodel is used to build ODM models from XML files and the UML metamodel is used to build the output model produced by the implemented ATL rules. The current version of the *RETool* uses the OMG UML metamodel provided by the eclipse UML2 project.[2] To accommodate the traceability scheme, we opted for using the UML extensibility mechanism, named stereotypes [43]. A noticeable advantage of this design decision is that the UML metamodel can be used without modifications.

[2] http://www.eclipse.org/uml2.

Therefore, the output of the ATL rules mapping is an UML model usable by any modeling tool compatible with the standard UML2 metamodel.

4 The controlled experiment

We have conducted an experiment in order to determine whether the proposed process and its tool support meet the goals stated in Sect. 1 and also to assess the complexity and benefits of using the tool when compared to traditional methods for requirement elicitation and representation. The planning and execution of the experiment reported in this section are based on the guidelines proposed in [59].

4.1 Experiment goals

We define the objectives of the experiment using the template of the Goal/Question/Metrics (GQM) method [60] as suggested by [59]. GQM is a top-down approach to establish a goal-driven measurement system for software development. The approach is divided into three levels: (i) the organizational **goals** which define measurement goals (conceptual level), (ii) the **questions** to address the previously stated goals (operational level), and (iii) the **metrics** that provide answers to the defined questions (quantitative level). GQM aims at defining the measurement goals according to the established business/research goals. Our primary research goal is to assess the effectiveness of our process and *RETool* in dealing with RE open issues. Additionally, we defined a secondary goal of assessing the usability and actual applicability of the proposed tool. We discuss below our *measurement goals* written according to the GQM template defined by Basili et al. [60].

First goal ***Analyze*** *the proposed process and RETool* ***for the purpose of*** *evaluating their effectiveness* ***with respect to*** *dealing with RE open issues presented in the paper* ***from the point of view of*** *the Requirement sengineer and Stakeholders* ***in the context of*** *Information System development process.*

Second goal ***Analyze*** *the proposed RETool* ***for the purpose of*** *evaluating it* ***with respect to*** *its complexity of use and applicability* ***from the point of view of*** *the Requirements engineer and Stakeholders* ***in the context of*** *Information System development process.*

4.2 Experiment design

Following the GQM method, we first refined the stated goals in a set of questions and then defined metrics that provide the grounding to answer the questions.

4.2.1 Questions

The following research questions are considered in the experiment. Questions Q1–Q4 are related to the first research goal while questions Q5–Q7 refine the second research goal.

- Q1: How effective are the process and *RETool* to deal with RE scope issues?
- Q2: How effective are the process and *RETool* to deal with RE communication issues?
- Q3: How effective are the process and *RETool* to deal with RE validation issues?
- Q4: How effective are the process and *RETool* to deal with RE traceability issues?
- Q5: Do real users have difficulties to use the *RETool*?
- Q6: Do real users feel that the *RETool* aids them to generate a RE document?
- Q7: Do real users feel that the *RETool* aids them to read a RE document?

4.2.2 Planning

In order to collect data to answer the proposed questions according to the established metrics (Sect. 4.2.3), an experiment in a controlled setting [42, 61, 62] was carried out. The experiment was organized in two different projects, each one addressing one of the two stated research goals.

The first project, named **PRJ1**, encompassed the development, using the proposed process and *RETool*, of a real software system that was previously built using standard analysis process and tools, such as use cases and class diagrams of the UML [43]. The existing software artifacts, denoted as the *base system*, were used as baseline for comparison with the software artifacts generated when developing the system with the *RETool*, which was denoted as the *generated system*. The subjects in this project acted as requirement/ontology engineers, system analyst and stakeholders. The subject playing the role of engineer/analyst was extensively trained in the proposed RE process and *RETool*. The subject playing the role of stakeholder received a textual document containing the detailed description of the selected system. Then, the requirements engineer subject conducted interviews with the stakeholders to gather the requirements of the system. After the interviews, the subjects performed tasks that correspond to all the further activities defined in the proposed RE process. For this project, the subjects were observed by the responsible researcher.

The second project, named **PRJ2**, was organized in two laboratory sessions. In each session, the subjects performed three tasks as follows: (i) requirements elicitation, (ii) modeling of static system requirements using UML class diagrams and use cases, (iii) modeling of static system requirements using *RETool*. The subjects were separated

into two groups. One group played the role of the project stakeholder and the other played the role of the requirements engineer. The stakeholders group received a textual document containing the detailed description of a system to be modeled. Then, the requirement engineer group conducted interviews with the stakeholders to gather the requirements of the system. After the interview, the requirement engineer group indentified the system requirements using both UML diagrams and the *RETool*. The produced requirement document was then validated by the stakeholder group. After the first session, both groups interchanged their roles; stakeholders group became requirement engineer group and vice versa. To avoid a priori knowledge on the system, a different system description was given to the stakeholders group in each session. All the subjects received two questionnaires, one to assess their perception of the provided training and description of experiment tasks, as well as to collect information on their education and experience in the experiment tasks (QT1), and other to evaluate the RETool (QT2).

4.2.3 Metrics

Since there is no standardized set of metrics available to evaluate the defined research questions, we specified our own metrics tailored to the purpose of the experiment. Following we describe each metric, denoted M_{ij}, where i corresponds to the question identifier, and j is a counter in the case that more than one metric is defined per question.

M_{11}. Difference between the number of entities at the specification of the *base system* and the *generated system* This metric intends to assess the process and tool capability of correctly discovering all entities relevant to the system specification. It is defined as:

$$Ne_{gen-base} = Ne_{gen} - Ne_{base},$$

where:

Ne_{base} Number of Entities (UML classes) at the base system specification.
Ne_{gen} Number of Entities (UML classes) at the generated system specification.

M_{12}. Highest number of entities per subject tag This metrics intends to characterize the process capability of isolating the entities related to each subject. This number can be used to evaluate the complexity (based on the number of entities) of each part of the system concerned with a certain subject. These numbers are compared among them and the highest one indicates the subject with the highest number of entities (and the highest complexity). This metric is defined as:

$$\max\left(N_{ej}\right),$$

where:

N_{ej} Number of entities at the domain ontology marked with the jth subject tag.

M_{13}. Number of inferred relationships Represented as N_{infe}, this metrics denotes the number of relationships among entities discovered by the activity of taxonomic classification performed by the *RETool*. This metric is useful for answer two different research questions. In this context, it is used to show that the whenever the proposed tool is able to reveal new relationships, it aids in the definition of the system scope.

M_{21}. Percentage of sentences removed after the validation activity This number shows the capability of the proposed process to detect sentences that generate inconsistencies in the knowledge base, possibly originated by communication issues. This metric is defined as:

$$N_{aer} = \frac{A_{rs}}{N_s} \times 100$$

where:

N_s Number of sentences from the client interviews.
A_{rs} Total amount of sentences removed after the validation activity.

$M_{22} = M_{13}$. Number of inferred relationships For the purpose of this research question, the finding of new relationships that were not made explicit during the elicitation process can indicate some kind of communication problem such as vagueness or ambiguity in the collected sentences.

M_{31}. Number of conflicts per each interview subject This metric intends to characterize the process ability of discovering the degree of complexity of each subject, assuming that more complex subjects tend to raise more inconsistencies. It is denoted as:

S_{ci} Total amount of conflicting sentences at the ith subject.

M_{32}. Percentage of inconsistencies between concepts This metric intends to characterize the process ability of automatically discovering inconsistencies in the sentences gathered during the interviews. It is defined as:

$$N_{ibc} = \left(\frac{A_{cs}}{N_s}\right) \times 100,$$

where:

N_s Total number of sentences.
A_{cs} The total amount of conflicting sentences.

M_{41}. Difference between the number of interviews conducted with stakeholders and the number of SDS tags at the domain ontology This metric intends to characterize the process ability of keeping metadata about the RE process after the construction of the domain ontology. This number is greater or equal to zero, where zero indicates that all interviews were kept and a non-zero value indicates that some interview information are missing. This metric is defined as:

$$Ni_{Cli-Ont} = Ni_{cli} - Ni_{Ont},$$

where:

Ni_{Cli} The number of interviews given by the stakeholders.

Ni_{Ont} The number of interview tags at the domain ontology.

M_{42}. Difference between the number of SDS tags in the generated domain ontology and the number different stereotypes representing SDS tags in the generated PIM This metric intends to characterize the process ability of keeping metadata about the RE process after the PIM generation. This number is greater or equal to zero, where a zero result indicates that all interviews were kept and a non-zero value indicates that some interview information are missing. This metric is defined as:

$$Ni_{Ont-PIM} = Ni_{Ont} - Ni_{PIM},$$

where:

Ni_{Ont} The number of interviews represented as SDS tags at the domain ontology

Ni_{PIM} The number of interviews represented as stereotypes, after the PIM generation.

M_{51}. Difficulty to use the *RETool* This metric measures the perception of the users regarding their difficulty in using the proposed tool. It is denoted as:

D_u %Subjects that had difficulties in using the *RETool*.

M_{61}. Effectiveness of *RETool* in extracting requirements This metric assess the perception of the users regarding the *RETool* capacity of significantly aids the requirements engineer in the extraction of the requirements in comparison with the use of traditional UML techniques alone. It is denoted as:

E_{er} % Subjects that felt the *RETool* significantly aids generating of the requirement document in comparison with traditional UML techniques alone.

M_{62}. Effectiveness of *RETool* in documenting requirements This metric assess the perception of the users regarding the *RETool* capacity of significantly aids the requirements engineer in the documentation of the requirements in comparison with the use of traditional UML techniques alone. It is denoted as:

E_{dr} % Subjects that felt the *RETool* significantly aids generating of the requirement document in comparison with traditional UML techniques alone.

M_{71}. Effectiveness of *RETool* in facilitating the reading of requirement document This metric assess the perception of the users regarding the easiness of reading the requirement document generated by the *RETool* in comparison with UML use cases. It is denoted as:

E_{rd} % Subjects that found easier to read the requirement document generated by the *RETool* than the UML use cases.

4.3 Preparation and execution of project PRJ1

The first project was conducted in a software laboratory at Federal University of Rio Grande do Norte (UFRN), Brazil, with two (2) subjects. The subjects were Computer Science M.Sc. Students in their second year, with basic knowledge in both requirements and software engineering. The subjects were extensively trained to use the *RETool* in a classroom at the University. The project was developed during 1 week, in 3 sessions of 3:00 h each. Previously to the sessions, the subjects attended an introductory lesson where detailed instructions on the tasks to be performed were presented. This lesson had the duration of 30 min and aimed at highlighting the purpose of the experiment, while details of the research questions were not provided. After the introductory lesson, the subjects performed the execution of PRJ1, which consisted in the task of generating the RE document of a real system using the *RETool*.

4.3.1 System description

The selected system was the Health Watcher system (HW), a well-known testbed used in the AOSD Europe project,[3] whose goal is to register public health system complaints. Health Watcher system describes health units and their specialties and can be used as a reference for software systems to support the public health care system.

The system registers three kinds of complaints: *animal-complaint* (*report either on sick/maltreated animals or diseases caused by animals*), *food-complaint* (cases where there is a suspicion infected food ingestion), and *special-complaint* (*other types of* complaint such as restaurants

[3] http://aosd.di.fct.unl.pt/.

Table 1 Interview data

Stakeholder	Date	Subject
Medical specialist	29/09/2010	Health system organization
Health system specialist	30/09/2010	Complaint system
Security specialist	28/09/2010	User access control

with hygiene problems, leaking sewerage, etc.). Each complaint has a set of data to identify the complaint occurrence such as the complainer name, address, the registry of the victim (if any), etc. The system also keeps track of each health unit and the respective diseases treated by it.

The HW system was chosen due to the availability of documents describing its requirements and use cases. After a preliminary analysis, we observed that the original HW specification was too simple to suitably depict all the potential benefits of our proposed process, so we have added information about the drugs prescribed by the doctors, medical procedures, and health unit staff (doctors, nurses, etc.) to the systems specification. Also, we have extended its specification to describe requirements related to *security* issues.

The complete description of the original HW system can be found in the study by Soares et al. [63]. The system has several specification documents and implementation versions. In each version, new requirements (mostly non-functional) were added to the system. We used the first version of the requirements specification and the system implementation as the base for our experiment. The HW UML class diagram was used for purpose of comparison with the PIM skeleton generated by our tool. From now on, we will refer to the HW UML specification as the "base system". We employed a manual process to lexicographically compare the names of the entities of the generated HW specification with those of the base specification.

The following subsection explains the execution of the PRJ1 and its outcomes in the context of the experiment.

4.3.2 RE process execution in the context of PRJ1

One of the subjects was selected to be the stakeholder. This subject played different stakeholder roles during the development of the project: (i) the stakeholder responsible for non-functional requirements of system access control, named *Security Specialist*;(ii) the stakeholder responsible for the whole complaining system, named *Health System Specialist*; and (iii) the stakeholder responsible for the health service, identified as *Medical Specialist*. The other subject was selected to play the roles of: (i) requirements engineer; (ii) ontology engineer; and (iii) system analyst. The selection of both subjects was random.

During the first process activity, the requirements engineer gathers information from stakeholders. Table 1 lists an excerpt of the interviews carried out in the experiment. The full set of transcriptions can be found at http://labdist.dimap.ufrn.br/twiki/OntologyMDAEC. In our experiment, we consider interviews as the only source of information. We did not use documents describing the domain because such approach would require the employment of some revision technique in order to meet the CNL restrictions. These techniques are out of the scope of this work.

After the interviews, the requirements engineer should register the gathered relevant information about the system by feeding the CNL editor with it. The requirements engineer should also register the metadata (tags) related to each information with the fragment of the interview that raised it. Figure 5 illustrates the *RETool* after the execution of activity 1(c) of the proposed process. View (a) shows the (ACE) sentences sorted by the Subject tags in which they have appeared. View (b) shows the sentences sorted by the

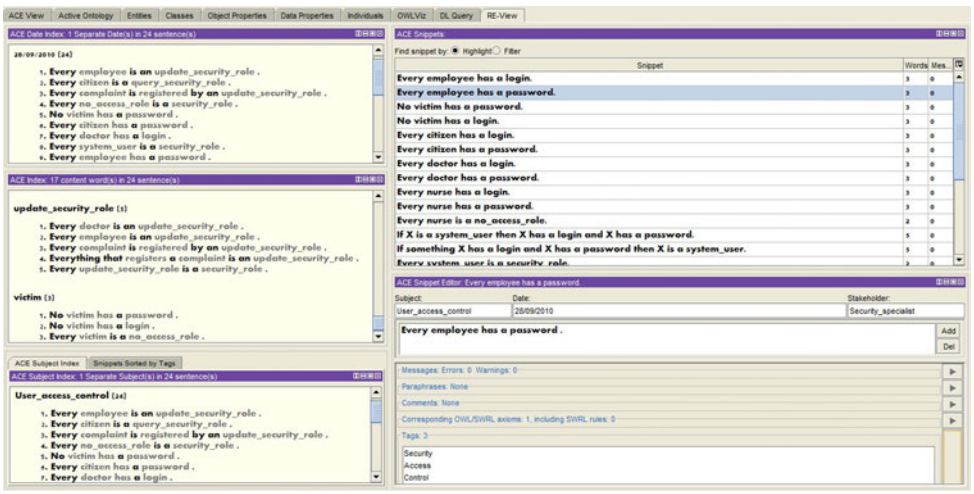

Fig. 5 The *REView* plug-in fed with the interview information

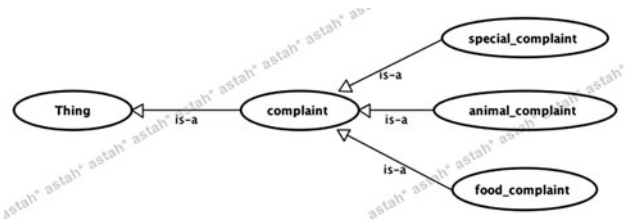

Fig. 6 Fragment of the generated domain ontology

date when they occurred. View (c) shows the other tags used to mark the selected sentences. Finally, view (d) shows all the sentences that contain each concept.

Figure 5 shows that the sentence "*Every employee has a password.*" is marked with three ordinary tags: "*Security*", "*Access*", and "*Control*". Moreover, such sentence is also marked with the subject tag "*User_Access_Control*", the date tag "*28/09/2010*" and the stakeholder tag "*Security_Specialist*". So, from the subject, date and stakeholder tags, one can conclude that the sentence was given by the security specialist, at September 28, 2010 and it is related to the subject "user access control".

According to our proposed RE process, after feeding the tool with the collected requirements information, the next step is to build the domain ontology. Since the *RETool* is integrated in the Protégé tool, the ontology, which represents part of the development team view, is built on-the-fly while the interview sentences are fed into the tool. Figure 6 shows a fragment of the generated ontology using a graph notation. The graph should be read from right to left and the edges represent *is-a* relationship between the concepts, expressed as nodes. For example, the ontology fragment of Fig. 6 defines the concepts *animal_complaint*, *food_complaint*, and *special_complaint* as sub-concept of *complaint* concept, which is a sub-concept of the special concept owl:*Thing*.

The next phase of the proposed process is the requirements validation where the ontology is checked for consistency by a reasoner. In our tool, this is done by the *Pellet reasoner* bundled with the Protégé tool. Besides the checking of domain ontology consistency, new relationships may be discovered through the automatic taxonomic classification process, also provided by the reasoner. The following subsection summarizes the inconsistencies discovered during the validation phase carried on during the experiment.

4.3.2.1 Inconsistencies and taxonomic classification In the performed experiment, 24 TBox and 12 ABox inconsistencies were found. Table 2 illustrates five of these TBox Inconsistencies. In such table, the stakeholder that provided the sentence appears between brackets. These inconsistencies will be explained throughout this section along with the adopted strategy to solve them. Such inconsistencies along with the stakeholders and interview dates related to them are presented in the ValidationView of the *REView* plug-in.

Our process also identified 12 ABox knowledge inconsistencies, i.e., inconsistencies between the individuals at the ontology and the definitions of the concepts. They were all solved using the same strategy used to address TBox inconsistencies, i.e., conducting new interviews followed by removing and/or changing sentences that originate the inconsistency. Table 3 lists an excerpt of the discovered ABox sentences along with the TBox sentence with which each of them is in contradiction (the stakeholders that provided the statements appear between brackets).

Besides using the ACEIndexView and the ValidationView provided by *REView*, the inferred hierarchy view provided by Protégé can also be used by the requirements engineer to analyze the detected inconsistencies and track their source. For example, Fig. 7 shows the Protege inferred hierarchy view with conflicting classes *animal_complaint*, *food_complaint* and *victim*. Assuming that the RE engineer wants to investigate the *victim* class, he should select such concept and access the ACEIndexView to get the information about each sentence where the concept *victim* has been quoted, as shown in Fig. 8.

After solving the inconsistencies, the taxonomic classification activity takes place and a new ontology is built. This activity is carried out by the ontology engineer aided by Protégé Inferred Axioms view. In our experiment, the taxonomic classification was able to discover thirty-three new relationships. Each line of the Protégé Inferred Axioms view shows a *subClassOf* axiom, used to express the subclass relationship between two concepts. Figure 9 shows the Protégé axiom view with some of the inferred relationships. In Fig. 9, the axioms marked as (c), stating that the concepts *nurse*, *doctor*, *observer* and *patient* are sub-concepts of the *person* concept, were inferred by the reasoner after processing the sentence "*Everything that has a name is a person*" and those declaring that the *citizen*, *nurse*, *doctor*, *observer* and *patient* concepts have names.

The following phase of the process execution is the generation of the PIM skeleton. The first activity of such phase is the filtering of relevant entities. In the experiment we performed a very simple filtering that consisted in removing the properties which we considered as primitive types, like: *name*, *phone_number*, *address*, etc. However, this activity can be much more complex depending on the RE knowledge base, for instance involving a full ontology cleaning [64]. After the filtering activity, a MDA transformation is run to generate the PIM skeleton. This is done outside Protégé by an Ant script specifying the defined transformation flow.

Table 2 Discovered inconsistencies and proposed solutions

Conflicting sentences	Solution
TBox Inconsistencies Originated from the Sentence "*Every victim is a system_user*" (**Health System Specialist**)	
"*No victim has a password.*", "*No victim has a login*" and "*If something X is a system_user then X has a login and a password*" (Security Specialist)	The requirements engineer conducted a new interview with both conflicting stakeholders and decided to remove the conflicting sentence "Every victim is a system_user"
TBox Inconsistencies Originated from the Sentence "*Everything that has a problem_location_data is an animal_complaint*" (**Health System Specialist**)	
"*Every special_complaint has a problem_location_data.*" and "*No animal complaint is a special_complaint*" (Health System Specialist)	The RE engineer conducted a new interview with the Health System Specialist and he decided to remove the sentence "*Everything that has a problem_location_data is an animal_complaint*". So, now *special_complaints* also have *problem_location_data*
TBox Inconsistencies Originated from the Sentence "*Everything that executes medical_procedures is a doctor*" **and** "*Every nurse executes at least 1 medical_procedure*" (**Medical Specialist**)	
"*No doctor is a nurse*" (Medical Specialist)	To solve such conflict, the RE engineer interviewed again the Medical Specialist and he recognized that both doctors and nurses execute *medical_procedure*, so the RE engineer removed the sentence "*Everything that executes medical_procedures is a doctor*"
TBox Inconsistencies Originated from the Sentence "*Everything that has a symptom is a disease*" (**Medical Specialist**)	
"*Every food_complaint has at least 1 symptom*" and "No Complaint is a disease" (Health System Specialist)	In order to solve such conflict, the RE engineer conducted new interviews with both conflicting stakeholders and they come to the conclusion that a *food_complaint* should also have *symptoms*, this was done by relaxing the restriction that states that "*Everything that has a symptom is a disease.*", so the interviewer changed such sentence to "*Every disease has at least 1 symptom*"
TBox Inconsistencies Originated from the Sentence "*Every suspicious_food_establishment is a problem_location_data*" (**Health System Specialist**)	
"*Everything that has a problem_location_data is an animal_complaint*" and "*Every suspicious_meal has a suspicious_food_stablishment*" (Health System Specialist)	These sentences conflict due to the fact that *complaint* and *suspicious_meals* are disjoint and that every suspicious meal has a suspicious food establishment. In order to solve this conflict, the RE engineer conducted a new interview with the Health System Specialist and he discover that *suspicious_meals* are actually a type of complain. So the sentence "Every *suspicious_meal is a food_complaint*" is added to the knowledge base

Figure 10 shows the fragment of the PIM (UML class diagram) for the entities related to the *complaint* concept. The complete class diagrams can be found at http://labdist.dimap.ufrn.br/twiki/MDARE. This figure shows the concepts presented in Fig. 6 after the transformation workflow. It shows the classes *complaint*, *special_complaint*, *food_complaint* and *animal_complaint*. Figure 10 also shows that our transformation was able to keep the is-a relationships between the concepts. For example: the *special_complaint* class, which is a subclass of the complaint class, corresponds to the Special Complaint concept that is a subconcept of the Complaint concept. Moreover, the metadata about the interview process are propagated from the ontology domain model to the PIM where they are represented as UML stereotypes. It is worth noting that although the use of stereotypes for interview visualization may seem to negatively affect the PIM readability, most of current UML graphic tools provide functionalities that selectively inhibit stereotypes presentation, thus avoiding this potential drawback.

4.4 Preparation and execution of project PRJ2

The PRJ2 experiment was specifically devised and prepared to evaluate the second research goal. PRJ2 was conducted in a software laboratory at University of Sydney (USYD), Australia, with nine (9) subjects. The subjects were volunteers' Computer Science PhD Students, with different levels of knowledge in both requirement and software engineering. The PRJ2 experiment started with an introductory lesson. In the lesson, detailed instructions on the tasks to be performed were presented to the subjects. The lesson aimed at highlighting the goal of the experiment, while the details on the experimental research questions were not provided. After that, the subjects were trained to use the *RETool*. The training was divided into two steps: in the first step, concepts related to the ACE language were presented. Next, the subjects were introduced to the *RETool* and they were given hands-on tutorial with examples to familiarize themselves with the main

Table 3 Contradicting ABox and TBox sentences

ABox sentence	TBox sentences
"*A victim has a password*" (*Health System Specialist*)	"*No victim has a password*" (Security Specialist)
"*A food_complaint has no victim*" (*Medical Specialist*)	"*Every food_complaint has at least 1 victim*" (Health System Specialist)
"*A nurse has no password*" (*Health System Specialist*)	"*Every nurse has a password*" (Security Specialist)
"*A flu is a disease that has no symptoms*" (*Health System Specialist*)	"*Every disease has at least 1 symptom*" (Medical Specialist)
"*A visitor_doctor is a doctor that has no login*" (*Medical Specialist*)	"*Every doctor has a login*" (Security Specialist)
"*A general_practitioner is a doctor that has no specialty*" (*Medical Specialist*)	"*Every doctor has at least 1 specialty*" (Health System Specialist)
"*An auxiliar_nurse is a nurse that executes no medical_procedure*" (*Medical Specialist*)	"*Every medical_procedure is executed by a nurse.*" and "*Every nurse executes at least 1 medical_procedure*" (Medical Specialist)
"*A natural_medicine is a drug that has no side_effects*" (*Medical Specialist*)	"*Every drug has at least 1 side_effect*" (Medical Specialist)
"*An observer registers a complaint*" (*Health System Specialist*)	"*If something X registers a complaint then X is a citizen.*" and "*No observer is a citizen*" (Health System Specialist)
"*An animal_poisoning_complaint is an animal_complaint*" **and** "*An animal_poisoning_complaint is a food_complaint*" (*Health System Specialist*)	"*No food_complaint is an animal_complaint*" (Health System Specialist)

Fig. 7 Conflicting concepts shown at the Protégé inferred hierarchy view

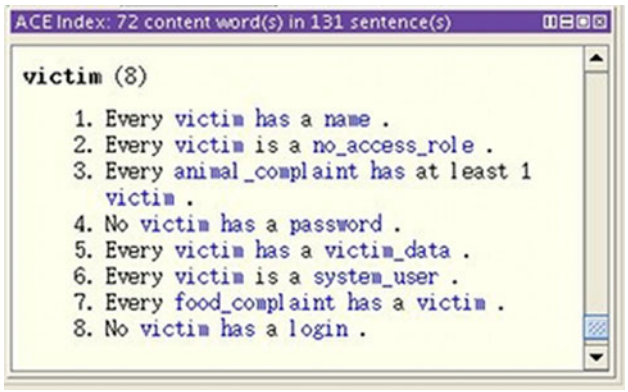

Fig. 8 Sentences at which the conflicting concepts were used shown at the ACEIndex view

functionalities of *RETool*. The preparation phase consumed 2:00 h.

After the preparation phase, the experiment was carried out in two sessions. Each session took 2:00 h. The subjects were randomly divided into 3 groups of two individuals and 1 group of 3 individuals. In each group, the subjects interchanged their roles according to the experiment planning (Sect. 4.2.2).

Similarly to the works presented by Briand et al. and Gravino et al. [61, 62] at the end of the second laboratory session, two survey questionnaires were distributed to the subjects. The first questionnaires (QT1) aimed at assessing the overall quality of the provided material, the comprehension of the tasks, and the level of the subject knowledge in UML, ontologies, and CNL. The second questionnaire (QT2) aimed at assessing the subjects' perceived usability and the usefulness of the *RETool* compared to UML class diagram and use cases.

The survey questionnaire QT1 was composed of fourteen questions (Table 4). The answers to the questions Q1–Q9 were based on a five-point Likert scale [65]: from strongly agree (1) to strongly disagree (5). The questions from Q10–Q14 expected answers according to a different five-point Likert scale ranging from very high (1) to very low (5).

The survey questionnaire QT2 was composed of five questions (Table 5). The answers to the questions in QT2 were based on a five-point Likert scale ranging from: strongly agree (1) to strongly disagree (5). A neutral judgment could be also be expressed by the subjects.

5 Experiment results and interpretation

This section describes the experiment results and their interpretation.

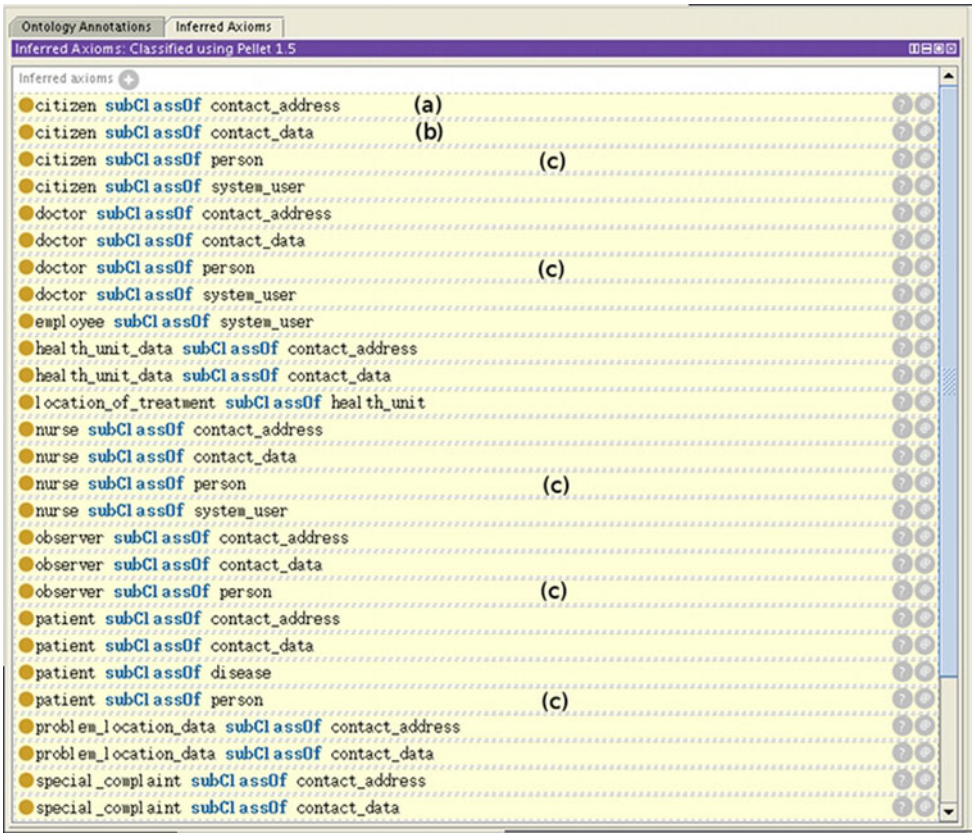

Fig. 9 Protégé inferred axioms view

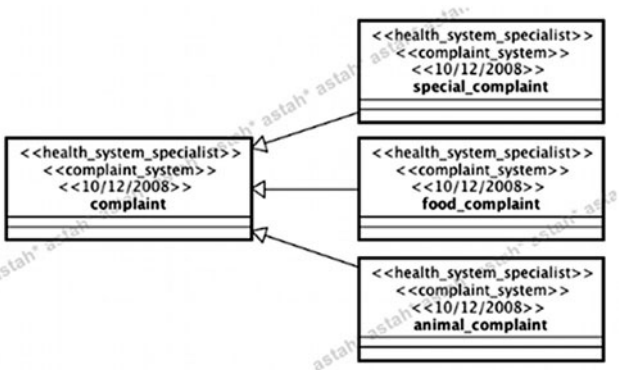

Fig. 10 Fragment of the generated PIM

5.1 Results of PRJ1

Tables 6, 7, 8, and 9 show the results of the data collected during the execution of PRJ1of the conducted experiment, for each one of the defined research questions.

Regarding the question addressing the *scope issues*, for the first metric, $Ne_{gen-base}$, we found a value of 25 for the number of entities (meaning UML classes) generated by the proposed process against 15 entities specified in the *base system*. The 25 entities generated by our process include all the entities in the *base system* specification and additional ones. Therefore, the process succeeded in identifying the same scope of the *base system*. The difference between the numbers of entities in the two specifications comes from the fact that the base specification does not modularize some concepts. For instance, *contact_data* is represented as primitive attributes repeated in several classes as all kinds of complaints (*special*, *animal* and *food*). The same applies to the concepts: *victim_data*, *complaint_data* and *contact_address*. Also, some of the concepts captured at the process were not represented as classes at the base system. We can cite the concepts *suspicious_food_stablishment*, *suspicious_meal* and *symptom*. The result of this metric shows that the proposed process is able of correctly uncovering all relevant entities in the system being modeled.

To obtain the second metric, $\max(N_{ej})$, we first computed the number of entities in the domain ontology marked with each one of the j subject tags, and then computed the highest value among them. The following subjects were considered: (i) Health_System = 1; (ii) Complaint_System = 2; (iii) User_access_control = 3.

The following values were achieved for each subject:

$$N_{e1} = 22, \quad N_{e2} = 33, \quad N_{e3} = 9$$

The Complaint_*System* had the highest number of entities (33). Therefore, the complaint system is probably the

Table 4 Experiment survey questionnaire QT1

ID	Question
Q1	I judge that the received training was sufficient to perform the tasks I was assigned to
Q2	I judge that the provided training material was sufficient to perform the tasks I was assigned to
Q3	I had enough time to perform the task of specifying the CoffeeMaker system with the RETool
Q4	I had enough time to perform the task of specifying the CoffeeMaker system with UML
Q5	I had enough time to perform the task of specifying the Cafeteria system with the RETool
Q6	I had enough time to perform the task of specifying the Cafeteria system with UML
Q7	The task objectives were perfectly clear to me
Q8	The description of the CoffeeMaker system was perfectly clear to me
Q9	The description of the Cafeteria system was perfectly clear to me
Q10	I judge the level of difficulty of the task on specifying the CoffeeMaker system as:
Q11	I judge the level of difficulty of the task on specifying the Cafeteria system as:
Q12	Assessing your experience level on the analysis phase of object-oriented system modeling (UML use cases and class diagrams)
Q13	Assessing your experience level on the Natural Controlled Languages
Q14	Assessing your experience level on the Ontologies

Table 5 Experiment survey questionnaire QT2

Q15	I judge the task of generating the RE document with the RETool more complex than the task of generating the RE document with UML
Q16	I judge the task of reading the RE document easier with the support of the RETool than with UML use cases and class diagrams
Q17	I judge the task of generating the RE document with the RETool should became easier with practice
Q18	I judge that the RETool significantly aided the extraction of the requirements during the task of interviewing the stakeholders
Q19	I judge that the RETool significantly aided the documentation of the requirements during the task of interviewing the stakeholders

Table 6 Results for the scope issues

Metric	Result
$Ne_{gen-base}$	10
$\max(N_{ej})$	33
$N_{\text{inf}e}$	33

Table 7 Results for the communication issues

Metric	Result
N_{aer}	12.83
$N_{\text{inf}e}$	33

Table 8 Results for the validation issues

Metric	Result
S_{ci}	$S_{c1} = 11$ (5 TBox, 6 ABox)
	$S_{c2} = 21$ (15 TBox, 6 ABox)
	$S_{c3} = 4$ (4 ABox)
N_{ibc}	24

Table 9 Results for the traceability issues

Metric	Result
$Ni_{cli-Ont}$	0
$Ni_{Ont-PIM}$	0

largest subsystem in the modeled domain. Moreover, such values can provide hints on how well a requirement was elicited. Subjects with a N_{ej} value far below the average may indicate that the requirement related to such subject is not well elicited, requiring further interviews. On the other hand, such requirement may be irrelevant to the system thus, it should be considered out of the system scope.

Finally, the $N_{\text{inf}\,e}$ value shows that 33 relationships were not made explicit during the requirements specification activity in the *base system*. Since the mechanisms provided by the *RETool* were able to reveal these new relationships, the proposed process aided in the definition of the correct scope of the domain.

Regarding the question addressing the *communication issues*, for the first metric we achieved the following values: $N_s = 148$; $A_{rs} = 19$.

The resultant value of $N_{aer} = 12.83$ indicates that about 13% of the sentences generated some kind of

inconsistence. These sentences were detected by the validation carried out by the RETool. Since these inconsistencies were generated by communication problems, the proposed process succeeded in minimizing such issues.

The value of the second metric, $N_{\text{inf } e}$, indicates that 33 relationships were not made explicit during the elicitation process, possibly due to some kind of communication problem such as vagueness or ambiguity in the collected sentences. However, since such sentences arose when the inference mechanisms of the RETool were employed, the generated system does not suffer from this issue.

Regarding the question addressing the *validation issues*, for the first metric the same three subjects described above (Health_System, Complaint_System, User_access_control) were considered. The resultant numbers show that the process is able to classify the subjects according to their complexity, assuming that there is a direct relation between inconsistency and complexity. In our experiment, the *Health_System* subject had the highest number of inconsistencies (21) and the *Security* subject the lowest (4).

For computing the second metric, from a total of 148 sentences we achieved 36 conflicting sentences (24 TBox and 12 ABox). This number shows that our process was able to automatically detect 24% of inconsistent sentences. Without such support, some of these inconsistent sentences could be propagated to the further phases of the software development thus, increasing the cost to solve them.

The results of the two metrics defined to assess the *traceability issues* show that no information about the interview process was missed during the first transformation process, when the domain ontology is generated and also that our process were able to keep track of all interview metadata information throughout the whole MDA generation process.

Summarizing the relevant points of the results concerning our first stated goal, we can observe that regarding *scope issues*, besides showing that the proposed RE process succeeded in capturing all the entities present in the specification of the *base system*, the performed evaluation demonstrates that the process was able to provide hints on the complexity of a given requirement and on how well it was elicited. The performed evaluation also demonstrates that the proposed process minimizes both *communication* and *validation issues* through its mechanisms of conflict detection and taxonomical classification. Finally, *traceability issues* were successfully addressed by the MDA process that synchronizes the different models and correctly propagates the metadata on interviews among them.

5.2 Results of PRJ2

The metrics regarding the second stated goal are extracted from the survey questionnaires QT1 and QT2 obtained after the conduct PRJ2 experiment. Since these questionnaires were based on Likert-type scales, it is necessary to calculate the internal consistency reliability of the adopted scales [66]. Cronbach's alpha [67] is an index of reliability widely used to analyze Likert-type questionnaires [66]. Such index is capable of determining if a given questionnaire will always elicit consistent and reliable responses even if the questions were replaced with other similar questions. Cronbach's alpha index is associated with the variation accounted for by the true score of the *underlying construct*. Construct is the hypothetical variable that is being measured [68]. Alpha index (α) ranges in value from 0 to 1; the higher the score, the more reliable the generated scale is. Nunnaly [69] has indicated 0.7 to be an acceptable reliability coefficient, but lower thresholds are sometimes used in the literature.

The answers to the questionnaires QT1 and QT2 are shown in Tables 10 and 12, respectively. The descriptive statistics of the questionnaires responses are shown in Tables 11 and 13, respectively. In order to assess the questionnaire reliability using the Cronbach's alpha index (α), we grouped the questions according to the following constructs: (i) training (Q1 and Q2); (ii) time RETool (Q3 and Q5); (iii) time UML (Q4 and Q6); (iv) task description (Q8 and Q9); (v) task complexity (Q10 and Q11); RETool benefits (Q16–Q19). The value of α for each construct was calculated using the IBM SPSS statistical software [70].

Based on the answers to questions Q1 and Q2, the training for the subjects was not enough to carry out the tasks, although the training material was considered relatively proper for the experiment. In particular, it was revealed that the subjects were unfamiliar with ACE language and the RETool. The value of α obtained for the *training* construct was 0.695, which denotes an acceptable reliability.

From the questions Q3–Q6, it can be readily inferred that the time to perform the experiment was considered proper and sufficient. More than 55% of the subjects agreed that the time to carry on the tasks using both RETool and UML were sufficient and the values of α obtained were respectively 0.706 for the *time RETool* construct and 0.914 for the *time UML* construct.

According to the answers to Q7, more than 66% of the subjects agreed that the task objectives were clear. Since there is a single question to analyze this item, it is not possible to assess the reliability of the answers related to clearness of the task objectives by using α.

The answers to the questions Q8 and Q9 show that the descriptions of the systems in both sessions were clear enough to perform the task. The value of α obtained for the *task description* construct was 0.869, which denotes a good reliability.

Table 10 Experiment survey questionnaire QT1 answers (%)

	Strongly agree	Agree	Neutral	Disagree	Strongly disagree
Q1	0.00	22.22	11.11	55.56	11.11
Q2	0.00	44.44	33.33	11.11	11.11
Q3	0.00	66.67	22.22	11.11	0.00
Q4	11.11	44.44	33.33	11.11	0.00
Q5	0.00	55.56	11.11	33.33	0.00
Q6	11.11	55.56	33.33	0.00	0.00
Q7	11.11	55.56	11.11	22.22	0.00
Q8	22.22	44.44	33.33	0.00	0.00
Q9	22.22	44.44	22.22	11.11	0.00
	High	Medium/high	Medium	Medium/low	Low
Q10	11.11	22.22	44.44	11.11	11.11
Q11	11.11	33.33	33.33	11.11	11.11
Q12	22.22	11.11	33.33	22.22	11.11
Q13	0.00	11.11	22.22	33.33	33.33
Q14	0.00	22.22	0.00	22.22	55.56

Table 11 Descriptive statistics of survey questionnaire QT1

	N	Minimum	Maximum	Sum	Mean	SD
Q1	9	1	4	22	2.44	1.014
Q2	9	1	4	28	3.11	1.054
Q3	9	2	4	32	3.56	0.726
Q4	9	2	5	32	3.56	0.882
Q5	9	2	4	30	3.33	1.000
Q6	9	3	5	34	3.78	0.667
Q7	9	2	5	32	3.56	1.014
Q8	9	3	5	35	3.89	0.782
Q9	9	2	5	34	3.78	0.972
Q10	9	1	5	28	3.11	1.167
Q11	9	1	5	30	3.33	1.118
Q12	9	1	5	28	3.11	1.364
Q13	9	1	4	19	2.11	1.054
Q14	9	1	4	17	1.89	1.269
Valid *N* (listwise)	9					

Table 12 Experiment survey questionnaire QT2 answers (%)

	Strongly agree	Agree	Neutral	Disagree	Strongly disagree
Q15	22.22	55.56	22.22	0.00	0.00
Q16	0.00	44.44	44.44	11.11	0.00
Q17	33.33	33.33	33.33	0.00	0.00
Q18	0.00	66.67	22.22	11.11	0.00
Q19	0.00	66.67	33.33	0.00	0.00

Table 13 Descriptive statistics of survey questionnaire QT2

	N	Minimum	Maximum		Mean	SD
Q15	9	3	5	36	4.00	0.707
Q16	9	2	4	30	3.33	0.707
Q17	9	3	5	36	4.00	0.866
Q18	9	2	4	32	3.56	0.726
Q19	9	3	4	33	3.67	0.500
Valid *N* (listwise)	9					

It can be shown that the subjects think that specifying both systems is considered challenging based on the answers to the questions Q10 and Q11. The value of α obtained for the *task complexity* construct was 0.907, which denotes an excellent reliability. This result can be partially due to their limited knowledge of the requirement engineering process, especially the use of and the unfamiliarity to tools such as the ACE language and ontologies as the feedback from the questions Q12–Q14 shows.

The answers to the questionnaires QT2 were used to generate the metrics to the questions related to the second stated goal. Table 14 presents the results of such metrics. The first metric, D_u, shows that almost 78% of the subjects considered that RETool is more complex to use than UML alone when generating requirement documents (this result was derived from question Q15). This metrics indicates that despite the benefits of the RETool, its use is not trivial and proper training is required to take advantage of it. Since there is a single question to analyze this metrics, it is not possible to assess the reliability of the answers related to the RETool complexity by using α.

The value of metric E_{rd} (derived from question Q16) shows that more than 44% of the subjects thought that reading the requirement documents with the support of RETool is easier than UML diagrams. However, this result is not conclusive because the same percentage (44%) of subjects responded QT16 with a neutral judgment.

Despite the difficulties and unfamiliarity of using RETool, the result of the metric E_{er} (derived from question Q19) indicates that the majority of the subjects (66.67%) considered that RETool significantly aided the extraction of the requirements during the interviews with the stakeholders. Moreover, the result of the metric E_{dr} (derived from question Q19) shows that the majority of the subjects (66.67%) found that RETool significantly aided the requirement documentation. The answers to the questions Q18 and Q19 along with Q1 suggest that the benefit of using RETool can be increased with proper and sufficient trainings. Also, the subjects reported (Q17) that if they have had enough training they would have been able to generate the requirements documents more easily.

Table 14 Results for the metrics addressing the usability and applicability of the RETool

Metric	Result
D_u	77.78
E_{er}	66.67
E_{dr}	66.67
E_{rd}	44.44

Q16–Q19 were grouped in the *RETool benefits* construct, whose obtained α value was 0.701.

The accomplished analysis indicates the potential applicability of using RETool in the RE process provided that sufficient training is given to the users.

Besides the questionnaires, the observation of PRJ2 raised some interesting facts. First, regarding the complexity of use of the RETool, PRJ2 showed that even with few hours of training it is possible to start using the RETool to extract and document requirements even with users with no previous knowledge on ACE and OWL. This is mainly due to one of the features of RETool borrowed from ACEView. Since ACEView offers an English-based (ACE) syntax for OWL, it greatly simplifies the creation and manipulation of OWL knowledge bases [46]. Moreover, the RETool completely hides the MDA transformations from the user. Another important observation is regarding RETool ways of use and its current limitations. The subjects found easier the specify OWL object properties as well as data properties directly from the Protégé standard views than with ACE sentences. The problem with this approach is that data properties specified directly in Protégé standard views do not appear as ACE sentences. One important limitation detected during the experiment was also related to OWL object properties and data properties. The current implemented mapping from ACE sentences to both object and data properties is incomplete. Besides the ACE sentences, the user also needs to specify the correct OWL Domain and Range of each property in the Protégé standard views. All the identified limitations in RETool are currently being addressed.

5.3 Threats to validity

In this section we describe the different threats that could affect the validity of the experiment presented in this paper, following the framework proposed in [42]. The discussed threats affect the construct, internal, external, and conclusion validity of the results. Overall, we tried to address the possible threats to validity through the careful planning of the experiment (Sect. 4.2.2).

Construct validity is concerned with the relationship between theory and observation [42]. Construct validity threats that may be present in this experiment were addressed by using a fairly simple and standard design. Another threat that could be raised is due to evaluation apprehension. To mitigate this threat, the subjects were not evaluated on their performance during the experiment. Moreover, subjects were not aware of the experiment goals.

The *internal validity* threat is concerned with factors that may affect the dependent variables (the metrics in our experiment) without the researcher's knowledge [42]. In particular, the presented experiment aimed at assessing whether stakeholders perceive benefits in using the *RETool*. Regarding internal validity, we identify the following threats:

(i) *Differences between subjects*: the different level of experience with UML, ontologies and CNL could lead to biased results. We tried to minimize this threat by including a training session on such languages. However, the analysis of the questionnaires' responses indicates that the results of PRJ2 were biased through some variables. The questionnaire QT1 revealed that the training given during the PRJ2 was clearly not sufficient. Moreover, QT1 also revealed that the subjects are not homogeneous regarding their experience with UML, Ontologies and CNL. Therefore, it is very alike that the results would be different if a proper training and a more homogeneous group of subjects were used. Since the questionnaires (questions Q12–Q14) indicate that the subjects are more familiar with object-oriented modeling than with ontology and CNL, the bias of the results was toward UML. In particular, considering that only one subject had previous contact with the *RETool* and that the training was not sufficient, we can conclude that the results indicating that the *RETool* is more complex to use than UML (Q15) are strongly biased toward UML.

(ii) *Differences between requirements models and tasks*: The requirements models used in each task were different. Moreover, the tasks presented to the subjects were different. Therefore their understandability could be different. We mitigated these threats by the design of the experiment, in which each group worked over two laboratory sessions, alternating the different tasks (using *RETool* or using UML). The survey questionnaire also showed that the subjects found clear everything regarding the controlled experiment.

(iii) *Fatigue effects*: The experiments were organized in daily sessions of 3:00 h (PRJ1) and 2:00 h (PRJ2). During the experiments, the subjects were observed by a responsible researcher and no signs of fatigue were detected.

(iv) *Subject motivation*: All the subjects were volunteers and they were aware that their contribution was very important to the research group they belong to. Moreover, most of the subjects reported at the end of the experiments that they were happy to participate in the experiment since they had the opportunity to learn new techniques that they perceived as useful for their carriers.

External validity is the degree to which the experiment results can be generalized within different contexts. As stated by Gravino et al. [62], threats are always present when experiments are performed with students. However, many researchers believe that experiments with students are useful as a pilot for latter industrial experiments. It is important to highlight that our experiment aims at evaluating a new RE approach and a tool that is in its very early stage of development. Considering this context, it was not viable to try an industrial experiment. Another aspect to consider is that the system descriptions used in the experiments have small size. Larger requirements models present a complexity that needs to be handled. Therefore, it is necessary to perform experiment replications with system descriptions of different sizes to confirm or contradict the results. However, since the *RETool* provides validation and traceability of requirements, we argue that with more complex systems, its use can be even more effective.

Conclusion validity threats concern the issues that affect the ability of the experiment to generate correct conclusion. The conclusion validity threats were mitigated by the experiment design and the use of statistics to evaluate the reliability of the survey questionnaires. Additionally, the survey questionnaires were designed using standard procedures and scales [65]. However, as the conclusion validity can be affected by the observation number, further replications on a larger dataset are required to confirm or contradict the achieved results.

Despite the highlighted issues with the experiment, the metrics results of PRJ1 provide initial evidences that the proposed process and *RETool* provide useful mechanisms to deal with the RE open issues. The metrics results of PRJ2, albeit not conclusive, corroborate to the results of PRJ1, indicating that the use of the *RETool* aids in the extraction and documentation of requirements.

6 Related work

As we previously mentioned, the main purpose of this work is to address four major open issues in requirements engineering, namely, *scope*, *communication*, *volatility* and *traceability* issues, by proposing a multi-viewed approach based on the integration of three different technologies: ontologies, CNL and MDA. The use of ontologies partially addresses communication and volatility issues; the use of CNL addresses scope and communication issues while MDA addresses communication and traceability issues. We organized this section in two groups. The first group encompasses works that employs Ontologies to augment the RE in different ways, thus addressing communication and volatility issues. The second group presents the works that propose the use of controlled natural languages and/or model-driven development to augment the RE process, thus jointly addressing scope, communication and traceability issues. After describing each group of works, we present a discussion about the main differences between them and our proposal. To the best of our knowledge, our work is the first to propose an approach that exploits the integration of these three technologies, thus tackling the four open issues altogether.

6.1 Ontologies in requirements engineering

Research initiatives exploring the synergy between ontologies and requirements engineering can be classified according to the way ontologies are used to support the requirements engineering process. Based on this idea, the authors in [71] organize the use of ontology in the requirements engineering phase in: (i) ontology as a **product** of RE; (ii) RE processes **guided by** ontologies; (iii) ontologies supporting **collaboration** in geographically distributed RE processes; and (iv) ontologies supporting requirements **validation**. Following we describe the related works according to this organization. However, we do not consider the works on ontologies supporting RE distributed collaboration since supporting distributed process development is out of the scope of the proposal presented in this paper.

6.1.1 Ontology as a product of RE

Breitman et al. [45] propose a requirement engineering process that encompasses a specific sub-process for building ontology. This sub-process is based on the layered ontology engineering approach [72], where a Language Extended Lexicon (LEL) [73] is used to organize the ontology building. The lexicon is built by extracting the relevant terms from the interviews or source documents, and mapping such terms to the appropriate constructs of the ontology that describes the domain representing the application being elicited. In this work, ontologies play the role of facilitating the knowledge sharing regarding the domain being elicited. Vongdoiwang [74] proposes a methodology for supporting RE in which ontologies are used as a tool to convert a problem domain textual description into an object model. Such methodology is

based on the transformation of eight different models. The first model is a Text description model (T-model) and the last model (the output of the methodology) is a Class (object) model (C-model). The T-model is processed by the Corporum OntoExtract [74] ontological engine tool, which builds an ontology in RDF schema (O-Model) that reflects concepts contained in the T-Model. Such concepts are refined by the remaining models in order to build the C-model. The other proposed models represent specific analysis activities, which the developers should accomplish in order to get benefit from using Ontologies for semiformal identification of objects, where such objects are responsible for the system functionality. Works such presented in [51] and [75] show that the Semantic Web, in special ontologies, can serve as a platform on which domain models can be created, shared, and reused. The authors argue that the end-to-end use of ontologies in the analysis and design phases as well as in the implementation is highly suitable for rapid application development. The authors propose the use of a web-based knowledge representation format that enables developers to discover sharable domain models and knowledge bases from internal and external repositories.

6.1.2 RE processes guided by ontologies

Commonly, the RE phase implies the use of different methodologies such as object-oriented analyses, goal-driven, viewpoints-oriented, and scenario-based approaches, or their combinations [76]. Since such methodologies were not designed taking into account the integration among them, they are hard to use in a collaborative way. This is a well-known problem in RE and different solutions have been proposed since the eighties [77]. More recently, Lee and Gandhi [76] proposed an ontology-based framework, named Onto-ActRE, which promotes cohesiveness between the artifacts generated from different modeling techniques and creates a shared understanding from multiple dimensions. Such framework combines several RE modeling techniques with complementary semantics in a unifying ontological engineering process, allowing building a multidimensional view of the gathered RE knowledge. The central point of this solution is a Problem Domain Ontology (PDO) that integrates (i) goal-driven scenario composition, (ii) requirements domain model, (iii) viewpoints hierarchy, and (iv) other domain specific taxonomies. The authors developed the GENeric Object Model (GenOM) tool, which is based on Jena and OWL (for representing PDO), and that allows requirements engineers to use the requirements domain model along with the goals from the goal hierarchy and the associated stakeholders in a viewpoints hierarchy. The authors [78] propose an ontology to support the requirement management process specifically tailored to the engineering design. The proposed ontology provides a terminology for design that can be shared by all the engineers involved. Furthermore, the authors define the meaning of the terminology by using first-order logic which gives a precise and unambiguous semantics for each term. This approach avoids ambiguity, possible conflicts and different interpretations by different engineers besides promoting the collaboration among developers. Moreover, the authors developed a set of axioms that capture definitions and constraints on the terminology to enable automatic inferences from the design knowledge. Such axioms allow deducing new information from the existent knowledge base and enable integrity checking of the design knowledge, i.e., detecting invalid data in the database and avoiding updates introducing conflicts among the data and the object model of design.

6.1.3 Ontologies supporting requirements verification

Requirements verification is a concern that has been investigated for a while. Among such researches, there are works that, similarly to ours, take advantage of the power of the first-order logic beneath ontologies to help uncovering inconsistencies. The work Kaiya et al. by [34] proposes a method for requirements elicitation, called ORE (Ontology based Requirements Elicitation), which uses a domain ontology to represent the requirements knowledge. In such work, after the domain ontology was manually built by the requirements engineer, the requirements elicitation proceeds with two iterative activities for requirements verification: evaluation of requirements by applying quality metrics, and revision of the gathered requirements based on the structural characteristics of the generated ontology. By using inference rules along with quality metrics on the domain ontology, the proposed methodology aids the requirements engineer to discover which requirements should be added for improving completeness of the RE knowledge base and/or which requirements should be deleted from the RE knowledge base for keeping consistency among the elicited requirements. The work by Zongyong et al. [79] suggests a formal approach to precisely describe ontology by initially using description logic, and then modeling integrity rules and derivation rules which restrict the business behavior. All the rules are represented from three perspectives: syntax, semantics and visualization. Finally, the work provides a framework for requirements model checking that combines domain ontology and domain rules, thus making the requirements elicitation process both guided by the domain ontology and restricted by the domain rules. Therefore, the acquired requirements would comply with both business needs and domain knowledge.

6.1.4 Discussion

Our work shares many similarities with the aforementioned researches. First, we also advocate the use of ontologies as a product of the RE phase. Second, we proposed a process for supporting RE that strongly relies on the use of Ontologies. Finally, we also use the inference capabilities associated with ontologies as a tool to provide validation of RE knowledge. However, our approach provides unique features that differentiate it from the aforementioned researches. While Ontologies are a good approach for defining a common language to create and share a domain as well as for harmonizing different ontology engineering approaches, ontologies demand additional modeling effort, which must be balanced by savings at other activities [80, 81]. Thus, a key to promoting the advantages of ontologies is the integration of ontological knowledge across the software engineering lifecycle. Our approach achieves such integration through the use of MDA transformations that automatically links ontological models with the other models in the software engineering lifecycle. Even though ontologies are a powerful mechanism to be used in the requirements representation and validation, as can be seen in the aforementioned works, such representation alone is not suitable for all the different stakeholders (as well as activities) involved in a RE process. For instance, ontologies are not suitable to acquire knowledge from the stakeholder; therefore, it is not able to directly address *scope* issues. Our MDA multi-viewed approach deals with such issue, providing different views for each type of actor and activity involved in the RE process. Moreover, our approach is seamless integrated with traditional RE processes and fully supported by open-source tools. Furthermore, the proposed MDA approach provides a simple and useful traceability scheme that facilitates dealing with requirements evolution and managing conflicts requirements, dealing with volatility issues. The use of ontologies alone does not provide any facilities to tackle the inherent volatility of requirements.

6.2 Controlled natural languages, model driven, and requirements engineering

Since natural language-oriented models are widely used in requirements modeling, there are several works that propose (semi-)automatic transformation to map the requirements models into conceptual object models [28, 39, 74]. More recently, Controlled Natural Languages (CNLs) have been integrated within model driven–based processes. Controlled natural languages are subsets of natural language whose grammars and dictionaries have been restricted in order to reduce or eliminate both ambiguity and complexity [82], intrinsic of natural languages [29] uses CNL integrated in an MDA approach for unifying the ideas of expressing requirements in a language close to natural, building Platform Independent Models for software components, and implementing the components via Platform Specific Models. The full transformation mapping uses a formal system of rules expressed in Two-Level Grammar (TLG) [83]. TLG is an object-oriented requirements specification language, based on natural language, with enough formalism to derive the corresponding implementation. The technique used on the transformation consists on identifying objects and relations on the problem domain based on nouns and the verbs between them. This grammar was initially created as a language to specify programming languages. With the emergence of executable models, it became an executable specification language, allowing the transformation of requirements expressed in natural language to a formal specification. Leal et al. [84] propose a Controlled Natural Language named Natural MDA language. Natural MDA is an action specification language [85] with a high abstraction level that is aligned to MDA objectives. The proposed language aims to raise the systems specification abstraction level, to complement UML and, as a consequence, to reduce the gap between the business domain objects and programming language elements. The author developed two associated tools, specifically designed to augment model-driven tools with Natural MDA language. Kalnins et al. [86] present an approach whose main goal is to show how transformations could be used to support the full path from requirements to code, in a model-driven development. In such work requirements are specified in a controlled natural language named *Requirements Specification Language* (RSL) [87] which has been developed as part of the ReDSeeDS project [88]. The required behavior specification built in RSL is precise enough so that this specification can be processed by model transformations in order to generate initial versions of the analyses and design models. All model-to-model transformations in the approach are implemented in model transformation language MOLA [89].

6.2.1 Discussion

The aforementioned works shares the same idea of using CNL to capture RE knowledge from stakeholders and also the idea of using model-driven transformations to generate software artifacts from the design phase. However, differently from our proposal, since the domain model of such works is not ontology-based, they do not take advantage of all the benefits provided by the use of an ontology-centric approach as, for instance, the capability of *requirements validation*. Therefore, differently from our proposal, such works do not fully address the four open issues in RE dealt with in this paper.

7 Final remarks

The research on the area of RE has grown fast in the last few years. In spite of this fact, there are still open issues. In our work we initially identified such issues and investigated the main existent initiatives that are addressing them. Following we presented an approach that borrows several ideas and techniques from these works and organizes them in a novel way, generating an augmented RE process that minimized the identified RE open issues. The proposed process encompasses three key technologies: Model Drive Architecture, Ontologies, and Controlled Natural Languages. The use of these three technologies allows representing requirements from the different perspectives of all the stakeholders and development teams involved in our RE process, as well as keeping such perspectives synchronized.

The proposed RE process is supported by a tool built using state of art techniques and technologies. Such tool allows collecting requirements knowledge directly from the stakeholder system descriptions. It also enables to transform such high-level descriptions into lower abstraction level representations in order to meet the needs of the development team. Also, the tool provides functionalities to manipulate, search, and validate the requirement knowledge base, thus facilitating the work of the development team.

The evaluation of the proposed process and associated tool was carried out based on a controlled experiment. The results draw from our experiment indicate that a clear communication channel between the teams involved in a RE process minimizes most of the scope and communication issues, since both parts can express their view about the system scope using a suitable notation. Moreover, the proposed process and tool provide a simple, but useful, traceability scheme. The experiment also shown that the proposed knowledge validation technique was effective to deal with volatility issues, once we were able to track down requirements inconsistencies by using ontologies along with reasoning mechanisms. Finally, the experiment shown that our approach correctly integrates the different views that represent the RE knowledge, once no information was lost during the transformation.

We argue that this work is a promising step toward an effective RE process that has proven the advantages of the integrated use of MDA, ontologies, and CNL to cope with current RE open issues. As so, it opens many different research directions to be explored. First, this work did not fully explore the potential of using ontologies in the RE process. For instance, there are many techniques to build, evaluate, and evolve domain ontologies, such as ontology cleaning [64] and ontology merging [50] that can improve the quality of the requirement knowledge base generated by the proposed RE process. Second, the proposed RE process did not include a comprehensive methodology to solve conflicts. In the current version, the tool only identifies conflicting sentences and leaves the requirements engineering in charge of solving them. Third, the proposed RE process only assures the synchronization of the different views provided that the modifications only occur from the higher to lower level models. That is, if a CNL sentence is modified, such changes are automatically and correctly propagated to the related ontology concept(s) and UML class(es). However, the opposite situation is not addressed in our current tool.

References

1. Nuseibeh B, Easterbrook S (2000) Requirements engineering: a roadmap. In: ICSE—future of SE track, pp 35–46. http://blueciteseer.ist.psu.edu/article/nuseibeh00requirement.html
2. González-Baixauli B, Laguna M, Crespo Y (2005) Product lines, features, and MDD. In: EWMT 2005 workshop
3. Gómez-Pérez A, Fernández-López M, Corcho O (2004) Ontological engineering: with examples from the areas of knowledge management, e-commerce and the semantic web. Springer
4. Paulk M, Weber C, Curtis B, Chrissis M (1995) The capability maturity model: guidelines for improving the software process. Addison-Wesley/Longman Publishing, Boston
5. Pressman R (2005) Software engineering: a practitioner's approach. McGraw-Hill, New York
6. Sommerville I (2001) Software engineering, 6th edn. Addison-Wesley, Harlow
7. Reubenstein H, Waters R (1991) The requirements apprentice: automated assistance for requirements acquisition. IEEE Trans Softw Eng 17(3):226–240
8. Walz D, Elam J, Curtis B (1993) Inside a software design team: knowledge acquisition, sharing, and integration. Commun ACM 36(10):63–77
9. Wojcik R, Holmback H (1996) Getting a controlled language off the ground at Boeing. In: Proceedings of the 1st international workshop on controlled language applications, pp 22–31
10. Mylopoulos J, Borgida A, Jarke M, Koubarakis M (1990) Telos: Representing knowledge about information systems. ACM Trans Info Syst 8(4):325–362
11. Johnson W, Feather M, Harris D (1992) Representation and presentation of requirements knowledge. IEEE Trans Softw Eng 18(10):853–869
12. Gordon M (2004) Knowledge representation: logical, philosophical, and computational foundations. Distrib Syst Online IEEE 5(1):9.1–9.3
13. Wang S, Jin L, Jin C (2006) Ontology definition metamodel based consistency checking of UML models. In: 10th International conference on computer supported cooperative work in design, pp 1–5
14. Donini F, Lenzerini M, Nardi D, Schaerf A (1996) Reasoning in description logics. In: Brewka G (ed) Principles of knowledge representation and reasoning. Studies in logic, language and information. CLSI Publications, pp 193–238
15. Lenzerini M (1996) Tbox and abox reasoning in expressive description logics. In: Proceedings of KR-96. Morgan Kaufmann, pp 316–327
16. Christel M, Kang K (1992) Issues in requirements elicitation. Carnegie Mellon University, Software Engineering Institute

17. Carter RA, Anton AI, Dagnino A, Williams L (2001) Evolving beyond requirements creep: a risk-based evolutionary prototyping model. In: Proceedings of IEEE 5th international symposium on requirements engineering (RE'01), pp 94–101
18. Espindola R, Lopes L, Prikladnicki R, Audy J (2005) Uma Abordagem Baseada em Gestão do Conhecimento para Gerência de Requisitos em Desenvolvimento Distribuído de Software. In: VIII workshop on requirements engineering
19. Sage A, Palmer J (1990) Software systems engineering. Wiley, New York
20. Macaulay L, Flower C, Kirby M, Hutt A (1990) USTM: a new approach to requirements specification. Interact Comput 2(1):92–118
21. Antón A, Potts C (2001) Functional paleontology: system evolution as the user sees it. In: Proceedings of the 23rd international conference on software engineering. IEEE Computer Society, pp 421–430
22. Stark G, Oman P, Skillicorn A, Ameele A (1999) An examination of the effects of requirements changes on software maintenance releases. J Softw Maintenance 11(5):293–310
23. Dubois E, Hagelstein J, Rifaut A (1989) Formal requirements engineering with ERAE. Philips J Res 43(4):393–414
24. Al-Rawas A, Easterbrook S (1996) Communication problems in requirements engineering: a field study
25. Bhat J, Gupta M, Murthy S, Technologies I (2006) Overcoming requirements engineering challenges: lessons from offshore outsourcing. Softw IEEE 23(5):38–44
26. Goguen JA, Linde C (1993) Techniques for requirements elimination. In: Proceedings of international symposium on requirements engineering. IEEE CS Press, Los Alamitos, pp 152–164. http://blueciteseer.ist.psu.edu/goguen93techniques.html
27. Nurmuliani N, Zowghi D, Powell S (2004) Analysis of requirements volatility during software development life cycle. In: Software engineering conference, 2004. Proceedings 2004 Australian, pp 28–37
28. Breitman K, do Prado Leite J (2004) Lexicon based ontology construction. In: LNCS, vol 19–34
29. Bryant B, Lee B, Cao F, Zhao W, Burt C, Gray J, Raje R, Olson A, Auguston M (2003) From natural language requirements to executable models of software components. In: Proceedings of the monterey workshop on software engineering for embedded systems: from requirements to implementation, pp 51–58. http://blueciteseer.ist.psu.edu/bryant03from.html
30. Debnath N, Leonardi M, Mauco M, Montejano G, Riesco D (2008) Improving model driven architecture with requirements models. In: 5th International conference on information technology: new generations, 2008 (ITNG 2008), pp 21–26
31. van Lamsweerde A (2008) Requirements engineering: from craft to discipline. In: Proceedings of the 16th ACM SIGSOFT international symposium on foundations of software engineering. ACM, New York, pp 238–249
32. White S (2004) Process modeling notations and workflow patterns. In: Workflow handbook, pp 265–294
33. Kavakli E, Loucopoulos P (2008) Goal driven requirements engineering: evaluation of current methods. In: Proceedings of 8th CAiSE/IFIP8, vol 1
34. Kaiya H, Saeki M (2006) Using domain ontology as domain knowledge for requirements elicitation. In: Proceedings of the 14th IEEE international requirements engineering conference (RE'06). IEEE Computer Society, Washington, pp 186–195
35. Gotel O, Finkelstein C (1994, April) An analysis of the requirements traceability problem. In: Proceedings of the 1st international conference on requirements engineering, pp 94–101
36. Gotel O, Finkelstein A (1997, January) Extended requirements traceability: results of an industrial case study. In: Proceedings of the 3rd IEEE international symposium on requirements engineering, pp 169–178
37. Cleland-Huang J, Chang C, Christensen M (2003) Event-based traceability for managing evolutionary change. IEEE Trans Softw Eng 29(9):796–810
38. Egyed A (2003) A scenario-driven approach to trace dependency analysis. IEEE Trans Softw Eng 29(2):116–132
39. Fuchs NE, Schwertel U, Schwitter R (1999, June) Attempto controlled English—not just another logic specification language. In: Flener P (ed) Logic-based program synthesis and transformation. No. 1559 in Lecture Notes in Computer Science. 8th International workshop LOPSTR'98. Springer, Manchester
40. Mellor S (2004) MDA distilled: principles of model driven architecture. Addison-Wesley Professional, Canada
41. Kleppe A, Warmer J, Bast W (2003) MDA explained: the model driven architecture: practice and promise. Co., Inc. Addison-Wesley/Longman Publishing, Boston
42. Wohlin C, Höst M, Henningsson K (2003) Empirical research methods in software engineering. In: Conradi R, Wang AI (eds) Empirical methods and studies in software engineering, Lecture Notes in Computer Science. Springer, Heidelberg, pp 7–23
43. OMG (2009) OMG unified modeling language TM (OMG UML), Superstructure. http://www.omg.org/cgi-bin/doc?formal/09-02-02.pdf. Accessed 10 May 2010
44. Rumbaugh J, Jacobson I, Booch G (2004) Unified modeling language reference manual, The Pearson Higher Education
45. Breitman K, do Prado Leite J (2003, September) Ontology as a requirements engineering product. In: 11th IEEE international conference on requirements engineering proceedings, pp 309–319
46. Kaljurand K (2008) ACE view—an ontology and rule editor based on Attempto Controlled English. In: 5th OWL experiences and directions workshop (OWLED 2008). Karlsruhe, 26–27 October, 12 pp
47. Smith M, Welty C, McGuinness D (2004) Owl web ontology language guide. W3C recommendation 10
48. Stumme G, Maedche A (2001) FCA-merge: bottom-up merging of ontologies. In: 7th international conference on artificial intelligence (IJCAI01), pp 225–230
49. Steve G, Gangemi A, Pisanelli D (1998) Integrating medical terminologies with ONIONS methodology. In: Information modelling and knowledge bases, vol IX
50. Noy N, Musen M (2000) PROMPT: algorithm and tool for automated ontology merging and alignment. In: Proceedings of the national conference on artificial intelligence (AAAI), pp 450–455
51. Knublauch H (2004) Ontology-driven software development in the context of the semantic web: an example scenario with protege/OWL. In: Proceedings of MDSW2004, Monterey
52. Sirin E, Parsia B (2004) Pellet: an owl dl reasoner. In: 2004 International workshop on description logics. Citeseer, p 212
53. Tsarkov D, Horrocks I (2006) FaCT++ description logic reasoner: system description. In: Automated reasoning, pp 292–297
54. Fuchs NE, Kaljurand K, Kuhn T (2008) Discourse representation structures for ACE 6.0. Tech. Rep. ifi-2008.02, Department of Informatics, University of Zurich, Zurich
55. Colomb R, Raymond K, Hart L, Emery P, Welty C, Xie G, Kendall E (2006) Version 3.3: The object management group ontology definition metamodel. In: Ontologies for software engineering and software technology, pp 1–25
56. Jouault F, Allilaire F, Bézivin J, Kurtev I, Valduriez P (2006) ATL: a QVT-like transformation language. In: Companion to the 21st ACM SIGPLAN symposium on object-oriented programming systems, languages, and applications. ACM, p 720
57. Catalog Of OMG Modeling and metadata specifications. Available at: http://www.omg.org/technology/documents/modeling_spec_catalog.htm#UML

58. MOF 2.0/XMI (XML Metadata Interchange) Mapping specification, v2.1.1. OMG Available Specification, formal/07-12-01. Available at http://www.omg.org/docs/formal/07-12-01.pdf
59. Jedlitschka A, Pfahl D (2005) Reporting guidelines for controlled experiments in software engineering. In: International symposium on empirical software engineering, p 10
60. Basili V, Caldiera G, Rombach H (1994) The goal question metric approach. Encycl Softw Eng 1:528–532
61. Briand L, Penta MD, Labiche Y (2004) Assessing and improving state-based class testing: a series of experiments. Trans. Softw Eng 30(11):770–793
62. Gravino C, Scanniello G, Tortora G (2010) An empirical investigation on dynamic modeling in requirements engineering. Lect Notes Comput Sci 5301/2010:615–629. doi:10.1007/978-3-540-87875-9_43
63. Soares S, Laureano E, Borba P (2002) Implementing distribution and persistence aspects with aspect. J ACM SIGPLAN Notices 37(11):174–190
64. Guarino N, Welty C (2002) Evaluating ontological decisions with OntoClean. Commun ACM 45(2):65
65. Oppenheim AN (1992) Questionnaire design, interviewing and attitude measurement. Pinter, London
66. Gliem JA, Gliem RR (2003, October) Calculating, interpreting, and reporting Cronbach's Alpha reliability coefficient for Likert-type scales. In: Midwest research-to-practice conference in adult, continuing, and community education, pp 82–88
67. Cronbach LJ (1951) Coefficient alpha and the internal structure of tests. Psychometrika 16:297–334
68. Hatcher L (1994) A step-by-step approach to using the SAS(R) system for factor analysis and structural equation modeling. SAS Institute, Cary
69. Nunnaly J (1978) Psychometric theory. McGraw-Hill, New York
70. IBM SPSS Statistics 19. Available at: http://www.spss.com. Accessed Dec 2010
71. Gašević D, Kaviani N, Milanović M (2009) Ontologies and software engineering. In: Staab S, Studer R (eds) International handbooks on information systems, part 5, handbook on ontologies, Springer, Berlin, ISBN 978-3-540-70999-2 (print) 978-3-540-92673-3 (online)
72. Maedche A (2002) Ontology learning for the semantic web. Kluwer, Boston
73. Leite JCSP, Franco APM (1993) A strategy for conceptual model acquisition. In: 1st IEEE international symposium on requirements engineering. IEEE Computer Society Press, Los Alamitos, pp 243–246
74. Vongdoiwang W, Batanov DN (2006) An ontology-based procedure for generating object model from text description. Published in journal: Knowl Inf Syst (KAIS): February 2006. Knowl Inf Syst (2006):93–108. doi:10.1007/s10115-005-0218-5
75. Völkel M (2006, May) RDFReactor—from ontologies to programatic data access. In: Proceedings of the Jena user conference, 2006. HP, Bristol
76. Won Lee S, Gandhi R (2005) Ontology-based active requirements engineering framework. In: Proceedings of the 12th Asia-Pacific software engineering conference, pp 481–490
77. Dobson G, Sawyer P (2006) Revisiting ontology-based requirements engineering in the age of the semantic web. In: Dependable requirements engineering of computerised systems at NPPs
78. Lin J, Fox MS, Bilgic T (1996) A requirement ontology for engineering design. In: Proceedings of 3rd international conference on concurrent engineering, pp 343–351. A revised version appears in Concurr Eng Res Appl 4(4):279–291
79. Zong-yong L, Zhi-xu W, Ai-hui Z, Yong X (2007, July) The domain ontology and domain rules based requirements model checking. Int J Softw Eng Appl 1(1):89–100
80. Oberle D (2006) Semantic management of middleware, vol I of The semantic web and beyond. Springer, New York
81. Happel H-J, Seedor S (2006) Applications of ontologies in software engineering. In: International workshop on semantic web enabled software engineering (SWESE'06)
82. Kittredge RI (2003) Sublanguages and controlled languages. In: Mitkov R (ed) The Oxford handbook of computational linguistics. Oxford University Press, Oxford, pp 430–447
83. Bryant B, Lee B (2002) Two-level grammar as an object-oriented requirements specification language. In: Proceedings of the 35th annual Hawaii international conference on system sciences (Hicss'02)-vol 9 (7–10 Jan 2002), HICSS. IEEE Computer Society, Washington, p 280
84. Leal LN, Pires PF, Campos MLM, Delicato FC (2006) Natural MDA: controlled natural language for action specifications on model driven development, on the move to meaningful internet systems 2006: CoopIS, DOA, GADA, and ODBASE, doi: 10.1007/11914853_33, pp 551–568
85. Raistrick C, Francis P, Wright J (2004) Model driven architecture with executable UML. Cambridge University Press, ISBN 0-521-53771-1
86. Kalnins A, Kalnina E, Celms E, Sostaks E (2010) From requirements to code in a model driven way, Lecture Notes in Computer Science v. 5968/2010 (Advances in Databases and Information Systems), pp 161–168. Springer, Berlin, ISBN: 978-3-642-12081-7, ISSN: 0302-9743 (print), pp 1611–3349 (online), doi:10.1007/978-3-642-12082-4
87. Smialek M, Bojarski J, Nowakowski W et al (2007) Complementary use case scenario representations based on domain vocabularies. In: Engels G, Opdyke B, Schmidt DC, Weil F (eds) MODELS 2007, LNCS, vol 4735. Springer, Heidelberg, pp 544–558
88. ReDSeeDS, Requirements Driven Software Development System Project. EU 6th framework IST project (IST-33596), http://www.redseeds.eu
89. Kalnins A, Barzdins J, Celms E (2005) Model transformation language MOLA. In: Aßmann U, Aksit M, Rensink A (eds) MDAFA 2003, LNCS, vol 3599. Springer, Heidelberg, pp 62–76