ISSN:0975-9646

An Ontology Based Requirement Rule Consistency Checking Framework

Dr. Farheen Siddiqui
Dept. of computer science, Jamia Hamdard, Handard Nagar, New Delhi

Abstract: In this paper, we investigate how ontologies developed for use in Semantic Web technology could be used in automated consistency checking of application requirements. Ontology Driven Architecture allows developers would discover shareable domain models and knowledge bases from a variety of interrelated repositories and then wire them together with the remaining object-oriented components for user interface and control. Domain knowledge base(domain Ontology) captures domain concepts, relationships and rules. Requirements rules should note violate these rules or contradict the usual business behaviour. This paper suggests a rule editing and validation framework RECC (REquirement-rule Consistency Checking)that guides an analyst to enter (application specific) requirement rules. It rests on Semantic Web technologies together with reasoning engines, which operate with semantic representations.A practical validation of the approach by instantiating this framework with OWL and reasoning engines is presented here. When requirement rules are authored with RECC the acquired requirements would comply with both business needs and domain knowledge.

Key Words: Knowledge authoring, Semantic Web, Ontology rule editor, reasoning engines.

I. INTRODUCTION

Knowledge authoring has become a fundamental process in the current knowledge society, since it allows organizations and entities to obtain and manage valuable information when taking decisions. This process usually consists of three stages [32], involving domain experts and knowledge system administrators. This paper is particularly focused on the phase wherein an expert adds application specific knowledge (requirement rule and validates it .. The processes studied in this paper are based on the use of Semantic Web technologies [3,11]. The adoption of the Semantic Web overcomes the search and integration limitations of knowledge management systems [15]. More specifically, ontologies are adopted based on Description Logic (DL) [1, 2] as the representation (Fig 1) of the domain model in such processes. Knowledge models based on DL ontologies are usually divided into TBox (terminological) and ABox (assertion) components

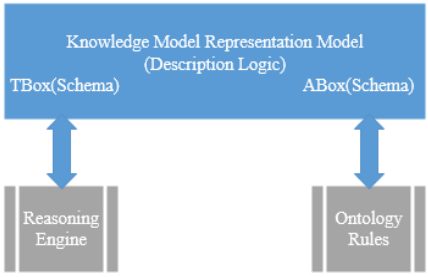

Fig 1: Knowledge models represented by means of DL ontologies with ontology rules.

The TBox contains the vocabulary and schema that define domain concepts, their properties,and the relationships (called roles in DL) among them. Hence, a concept represents a set of elements with similar characteristics (e.g., Students , StudyProgram, Instructor etc.), whereas roles symbolize binary relationships among elements (e.g., Student EnrollIn StudyProgram). Apart from these two atomic components, the DL language offers some axioms and restrictions over them to represent more complex domains (e.g., a RegularStudent is a specialization of the Student concept; a Student must have at least Enrollment). On the other hand, the ABox is populated with instances of these concepts and roles, representing a specific situation in the domain according to that schema. Let us see a simple example of an ontology rule. Suppose the domain model of a system which manages the task of a university system is represented by an ontology where Student and StudyProgram are concepts, and EnrollIn is a role that relates Student to their enrollment in a specific course. Moreover, Student has a specialization MCAStudent.The ABox of the ontology contains the assertions Student(Mark), StduyProg(MCA) and EnrollIn(MarkMCA). Now, suppose that a system administrator wants to express that all students Enrolled in MCA are MCA students. This conditional statement can be defined through the knowledge rule by using the previous ontology elements as follows:

Student(?x) StduyProg(MCA) ^ EnrollIn(?x, MCA) ->MCAStudent(?x)

The main contribution of this paper is the development of a generic rule authoring system in a Semantic Web framework which includes the characteristics listed in the above paragraph. To this end, we have developed a framework RECC which consist of a GUI that guided the user through simple steps when editing rules. RECC enable a comfortable rule editing, testing, debugging and validation. RECC consists in a graphical front end which guides the user in the management application specific business rule. RECC is a stand-alone application aimed to graphically edit, test, debug and validate ontology rules in any domain. Therefore, the RECC is intended to be used in developing application that needs to work with Ontology, ontology rules and reasoning processes.

In this work we have focused on university scenario where there is a necessity of modeling and monitoring knowledge rules to illustrate the RECC functionalities. Particularly, this scenario is based on the management of information directed to manage task of university. The rest of the paper is structured as follows. The next section introduces the underlying elements on which RECC based, and then gives an abstract architecture of the rule authoring system.

Section III provides a description of the RECC architecture implementation, focusing on technical aspects of the authoring process. A scenario in which RECC is used to generate knowledge rules to control an intelligent building is exposed in section IV. Section V discusses several related tools for managing ontologies and rules. Finally, section VI summarizes our contribution and points out the future work.

II. RULE AUTHORING SYSTEM IN A SEMANTIC WEB FRAMEWORK

A. *Motivation: A University scenario*

In order to illustrate how RECC manages the authoring cycle of knowledge rules, it has been integrated into university system as a case study. The scenario corresponds to the management of a university. This university offers several study Programs in regular and online modes The set of rules applicable to students are different depending on mode and Study Program. Also the rules governing student may change in every academic year. Thus university systems need to manage the edition and monitoring of rules, presenting an excellent field to test the authoring framework developed here. In particular, the scenario focuses on the management of Students enrolled in different study Programs .There are two types of rules: domain rules, which are applicable to all application of a specific domain and requirement rule which are applicable for the general operations a specific application under consideration. Suppose that Bob is a Student who has enrolled in a study Program with regular mode.

The domain rule states that every student enrolled in Study Program must appear in examination. This domain rule will be represented as a knowledge rule, denoted by ExamRule henceforth (see section 4.1). The use of rules to detect inconsistencies in property values over elements in the domain is a desirable feature of these systems. If the relation between Student and Exam is missing, then RECC will be able to detect the missing relationship during requirement rule authoring. Thus, the generation of new knowledge through rules should be permitted in a simple manner.

Now consider a specific application requirement that a student with more a backlog paper is not allowed to appear in examination. This requirement rule will be represented as a knowledge rule, denoted by BacklogRule henceforth

Student(?x) ^ StudyProg(?p) ^ course(?c)
^ hasCourse(?p,?c) ^ EnrolledIn(?x,?p)
^exam(?e) ^ examHeld(?e,?c) ^ course(?d)
^ hasBacklog(?x, ?d) ^ notEqual(?c, ?d) ->Detained(?s)

Now suppose that Bob is enrolled in MCA course and is having a backlog held with him. Therefore, it is possible to reach a situation in which domain rule states that Bob should appear in exam while the requirement rule states that Bob is not allowed to appear for Exam . Now, the system analyst should be able to detect and manage the conflict generated due to these two rules.

The following relationships between the domain concepts are captured: an StudyProg has assigned one or more Courses through the HasCourse relation; a Student has enrollment in a course by means of EnrollIn relation; Notice that these relationships are not modeled here as simple properties , but as concepts. The fundamental for this decision resides in that the types of such relationships could be used to classify them in different categories. Thus, EnrollIn is a special type of association that an student has with StudyProg Eventually, observe that the domain and the range of these associations are described through roles.

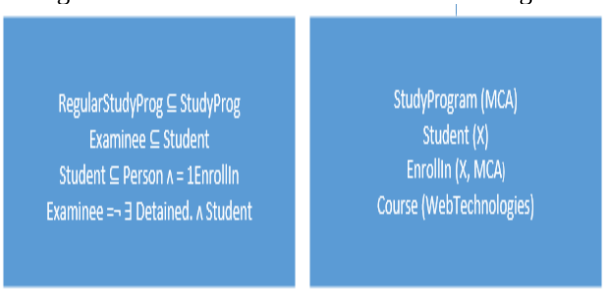

Fig 2: The University scenario expressed in an OWL-DL ontology.

Fig 2 reflects the current state of affairs in the scenario by means of concept (C(x)) and role (R(x, y)) assertions..

B. *Generic rule authoring system architecture*

Once the knowledge and rule models have been introduced, the next step developing the generic rule authoring system resides in establishing the architecture to deal with both of them. Figure 3 depicts the abstract architecture of the rule authoring system RECC. The core system (1) is formed by the RECC which encloses the methods for managing ontology rules; through GUI, a graphical interface of these tasks; the OWL API[4] , which manages the ontologies representing the knowledge model it provides a set of methods for loading and managing OWL ontologies, together with a group of reasoning engines with different capabilities.

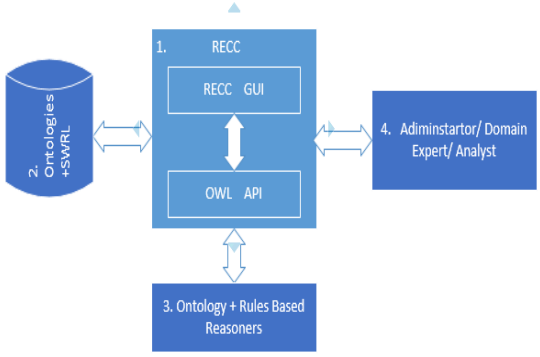

Fig 3: An abstract architecture view of RECC connected with its external resources.

OWL API is employed in the core of the system to obtain a working model from the domain ontology, not as a reasoning engine. RECC receives OWL ontologies and SWRL rules as input (2). RECC is in charge of SWRL rules, which can also be graphically edited, tested, debugged and validated in the GUI. Inference processes in RECC are accomplished by the combination of different reasoning engines (3). These engines can be distinguish according to the two types of inference that have been

explained in the introduction (see figure 1, the reasoning engines box), namely ontology and rule-based reasoning. Regarding ontology reasoning engines, there exist several proposals such as Pellet [31], Jena[6], Euler [30], Fact++ [25]. Likewise, there are several implementations available in the field of rule-based engines, such as SweetRules, JEOPS, JLisa, Prova, OpenRules, Jess, RDFExpert[9], Pellet and Jena.

III. ARCHITECTURE IMPLEMENTATION

The architecture previously presented in figure 3 has been implemented as framework for authoring of ontology rules. The following subsections present a description of all RECC framework components.

A. *Details on implementation*

Testing and debugging tasks in the rule authoring process demand that the developers get the full control in the execution of the rules, step by step, during the whole authoring session. Moreover, developers need to track the reasons for which rules have been fired in order to discover errors in the design of the rules. At the same time, both ontology and rule-based reasoning processes have to be taken into account in RECC to test and validate the edited rules.

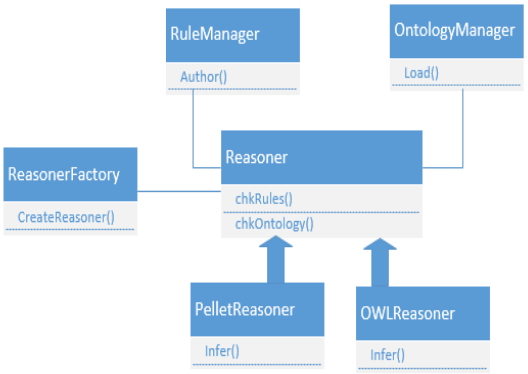

Fig 4: class diagram for RECC

The selected engine combination of the current RECC version for testing and debugging purposes consists of Pellet both as the ontology reasoner and rule reasoner. If the user decides to insert a new engine combination, the RECC framework offers an easy method to achieve it. Figure 4 depicts the class diagram of the RECC reasoning architecture. A RuleReasoner is in charge of providing a debugger component in RECC. A Debugger is a composition of one or more reasoner implementations. Each Reasoner implementation inherits from an abstract class Reasoner which provides basic features related to rule and ontology managements.

Essentially, in case a user decides to extend the rule reasoner support, he has to create a new class implementing the interface Reasoner. This interface only has one abstract method for carrying out the inference process. Then, the user can retrieve all the needed information from OntologyManager and Rule- Manager utility classes using the methods provided by the Reasoner classes. This information is then combined with the API of the new reasoner and the results obtained from the reasoning process are published using the analogous methods.

B. *RECC: Rule Authoring framework for software applications*

RECC software Analyst which need knowledge rule authoring services. These services are directly implemented in the RECC as a set of methods to enter business rules, perform ontology inference processes, and test business rules, retrieve the inferred knowledge generated by ontology and rule reasoning processes, and accept/Reject rules during the authoring process.

The integration of the reasoning processes in RECC has been depicted in figure 6. It starts dividing the initial knowledge model into rules on one hand (SWRL), and ontology model (OWL) on the other hand. Next, the ontology model is loaded into the Pellet ontology reasoner producing a semantic enrichment of this ontology model as output. The reason for doing this separation between SWRL and OWL is for isolating the execution of the rules in order to get the full control over their execution for debugging and validating purposes. Then, ontology rules have to be transformed from the SWRL syntax managed in the RECC architecture to the specific rule format imposed by the rule reasoner in case it is necessary. Next, these rules and the semantically enriched ontology model are inserted in the rule reasoner. Finally, this reasoner infers new facts that could be grouped as rule-based knowledge as shown in figure 6. For example, the property Enrollment can only take one value for the same Student; therefore a semantic inconsistency occurs when this property takes more than one different value, for a specific application business needs. Thus, RECC is able to discover inconsistencies, and to notify it to the Analyst. Then, Analyst can decide to include the requirement rule again in the initial knowledge base in order to start a new step in the authoring process or just simply discard the requirement rule because a rule is violating the general domain restriction.

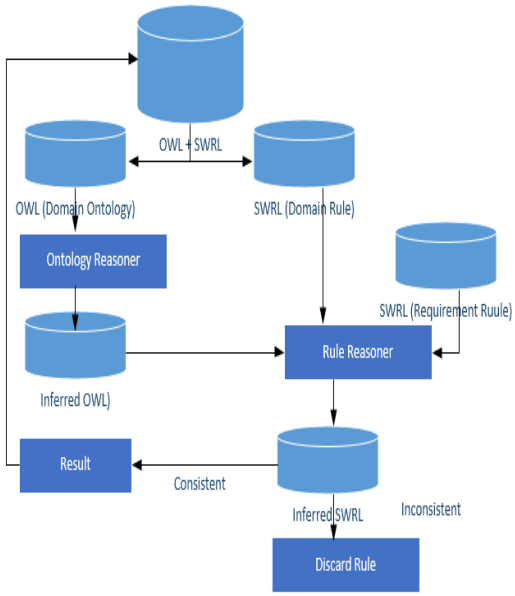

Fig 6: Ontology processing for knowledge authoring

C. *RECC GUI*

RECC provides a GUI intended for being used by system administrators. All the functionalities involved in the knowledge rule authoring are offered here in a graphical mode. RECC provides a full control in the navigation

across the knowledge models managed in RECC the rule authoring process is performed hiding the details of the underlying SWRL syntax to the user. Analyst adds a requirement rule through a wizard that guides the creation or modification tasks. These tasks are executed in an easy and intuitive selection manner, where ontology elements (concepts, roles, individuals, etc.) are selected to the correspondent part of the rule structure displayed in the GUI. This structure is based on SWRL syntax, as previously exposed in section 2.2. To this end, the rule editor in RECC offers a complete vision on the domain ontology, where the user can navigate across all the information represented in it. Both antecedents and consequents of a rule are defined in the same manner: For example, according to the University scenario introduced in section 2, We can add a SWRL rule saying that an individual X from the Person class, which has Enrollment in study Program Y, belongs to a new class Student. Such rule is best described in the SWRL

Person(?x)^ StudyProg(?y)^EnrollIn(?x, ?y) -> Student(?x)

To edit this rule in RECC-GUI, the Analyst should first select the class Student and Study Program from the domain model listed in the class list box.The properties that are displayed in the property list box are strictly according to the object properties in domain ontology having selected concepts either as range or domain. Then analyst can then select the Enrolln Property (0bject property of Student)to the rule. The edition task is validated by the wizard, avoiding the appearance of syntactical errors during the rule definition.

Regarding consistency checking of requirement rule provided by RECC-GUI, it offers some options such as two objects can be related in head of a rule only if such relation exists in domain ontology. Also a requirement rule can be added only if it is not violating domain rule and already added requirement rule and domain rules .RECC informs about relationships in the antecedents of rule that are missing in the domain ontology or inconsistent with domain rule. After a rule has been entered, the analyst decides whether he wants to insert the inferred facts into the initial knowledge base in order to perform the next requirement rule insertion step or discard these facts totally or partially. As a result, RECC can be seen as a framework for an ontology rule editing, testing, debugging and validation.

The inference process depicted in figure 6 and performed by the selected reasoning combination is done just by a click action. Not only will this action provide new inferred knowledge and a consistence validation, but it also offers the different knowledge bases showing all the information involved in each step of the knowledge rule authoring process. This last feature confers an extra debugging power to RECC over Prot´eg´e [26] or Ontotrack [19], which do not implement it.

IV. USING THE RULE AUTHORING IN UNIVERSITY SCENARIO

The domain ontology only captures domain concepts and neglects domain-restricted rules, or shortly domain rules. Domain rules are the description of some definitions and restrictions of a business, which can be used to maintain business structure and control or influence business behavior. Those rules are the restrictions or constraints imposed on business behavior. They usually exist deep in the mind of users as important background information which is not easily documented. requirement rules are the description of some business process, which is to be used for a specific business structure Hence requirement rule vary for different application of same domain(same domain rules). If analysts are not familiar with the domain, especially the domain rules, they would model a requirement rule that violates domain rules and hence making contradiction to the usual business behavior. Therefore, it is necessary to model requirement rules for application to achieve agreement not only on the domain concepts but also without violating the domain-restricted rules. We can capture all rules together in the application rules base. The later can be used to check if any requirement rule is inconsistent with domain rule. This scenario is used here to illustrate the entire knowledge authoring process described in the introduction of the paper, and it specially focused on the management of rules. Regarding the first stage, dedicated to the acquisition of knowledge, the Analyst has modeled the education domain by means of an OWL ontology based on the DMTF-CIM standard, called OWL-CIM [8]. This OWL ontology represents the concepts as partially shown in the TBox of figure 2.A complete vision of the scenario is depicted in figure 7. The creation of the OWL-CIM ontology that represents this specific domain has been performed in Prot´eg´e [26], a broadly used OWL editor. The specific scenario involving Student and the Exam management (partially represented in the ABox of figure 2) has been modeled following the same idea. The OWL files of this scenario exposed in figure 7 has been distributed within the RECC framework, as a case of study It describes a university offering Study Programs in different Departments For simplicity, just the two Department involved in the scenario, Computers and Management, has been depicted. The composition relationship has been modeled using the hasStudyProg association (see section 2.2). Each StudyProg contains in turn some Courses (through the hasCourse association).

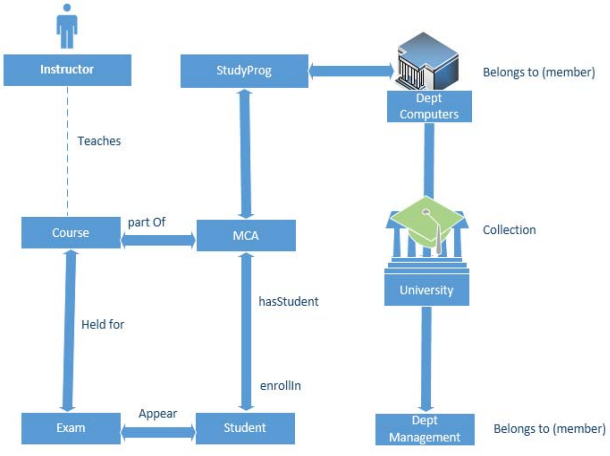

Fig 7 : A running scenario

S is a student of the university whose enrolled StudyProg is MCA (by means of the EnrollIn association). MCA is offered in Computers Department Each Study Program conducts examination for the courses it contains. The Student is linked to Exam by the AppearIn association and course is linked to Exam by the Heldfor association.From the domain ontology we can know the following knowledge: Every student who is enrolled in a shall appear in exam for that study program corresponding rule:

Exam Rule:
Student(?x) ^ StudyProg(?p) ^ course(?c)
^ hasCourse(?p,?c) ^ EnrolledIn(?x,?p) ^
exam(?e) ^ examHeld(?e,?c) ->Examinee(?s)
Particularly, university scenario deals with the management of Students and Instructor. On one hand, a domain rule states that every enrolled student appear in exam This rule has been recorded as domain rule in a domain rule base.Contrarily, the policy of a particular university states that student must not appear in an exam if he is having a pending backlog course. This rule is added to the by a system administrator through authoring process of RECC

Backlog Rule :
Student(?x) ^ StudyProg(?p) ^ course(?c)
^ hasCourse(?p,?c) ^ EnrolledIn(?x,?p) ^
exam(?e) ^ examHeld(?e,?c) ^ course(?d) ^
hasBacklog(?x, ?d) ^ notEqual(?c, ?d) ->Detained(?s)
where examinee and detained are defined to be disjoint classes in domain ontology.

This scenario is used here to illustrate the entire knowledge authoring process described in the introduction of the paper, and it specially focused on the management of rules. Regarding the first stage, dedicated to the acquisition of knowledge, the system administrator has modeled the University domain by means of OWL ontology This OWL ontology represents the concepts as partially shown in the figure 2. The creation of the OWL ontology that represents this specific domain has been performed thanks to Protégé a broadly used OWL editor.

A. Rule authoring process

In order to show an example of the rule authoring process, suppose that a analyst is responsible of editing the different domain restrictions by means of rules. To this end, the analyst uses a kind of templates which gathers such users' preferences to convert this information into rules As a result, the analyst defines a rule that a student appear in exam of the course that belong to study Program in which student is enrolled with. This rule is called examRule and it is given below in SWRL abstract syntax. examrule can be read as follows: if a student is enrolled in a study Program with course ,and exam is held for that course and the student, then student appear in exam

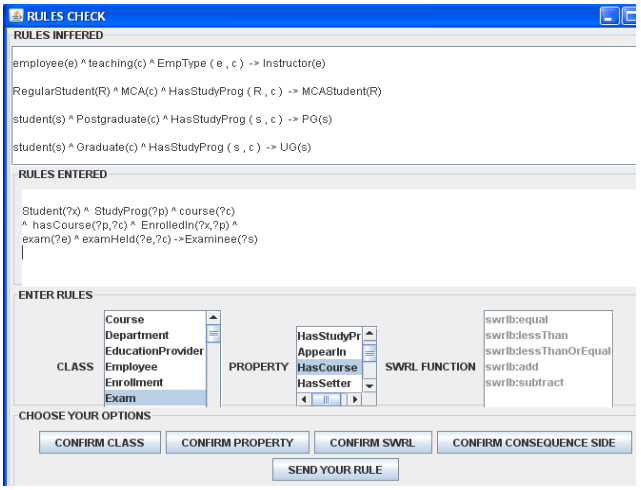

Fig 8: Domain Restriction edited in RECC-GUI as a rule by a analyst.

The edition of the rule through a rule in RECC-GUI can be seen in figure 8. First, the upper- corner shows the domain restriction modeled as rule in current domain. Third, panel shows the concepts, properties and SWRL functions representing the current domain model. Essentially, these lists are the friendly, graphical and structured representation of the OWL ontology and they allow the user to utilize the ontology concepts without any knowledge about the OWL syntax. .Then, the user can select concepts, properties and SWRL build ins from these lists to the items which compose the rule atoms. As analyst select concepts from class list box the corresponding properties are populated in property list box. In case the Analyst would like to create an atom belonging to the antecedent of the rule, he will be guided by a wizard to control and avoid any possible mistake in the atom definition. The wizard guides the user thought messages in the GUI in order to notify the next action in the authoring process. In an atom definition, the user will start selecting a concept from the list in the class area and validate the action by clicking the button "next" button. This process is repeated in the same manner for the predicate and object of the atom. Then, in order to finish the process, the user has to press in "confirm Class", "Confirm Property, "confirm SWRL" or "Confirm Consequent" according to their aim. The snapshot in figure 8 represents a situation
in which all the atoms of the ExamRule antecedent and consequent have been already inserted in by the analyst He is now using RECC to "Send Rule" which will save the rule as domain restriction and finish with the rule edition session.

B. Rule testing, debugging and validation tasks

After defining ExamRule, it is saved in rule file containing domain rules by clicking on the button "Save Rules. Analogously, the Analyst uses RECC to insert the requirement of the particular university as a rule, i.e. BacklogeRule .Furthermore, in this scenario the RECC of the system administrator is also configured to automatically execute an inference process whenever a new rule is added through RECC, as for example if a student s enrolled d in study program MCA with course Web Technology will be

Examinee for exam held for web technology. Now if s has a pending backlog other than web technology then s will be detained. Since examinee and detained are disjoint concepts in the underlying domain model, an inconsistency in the requirement is detected by RECC. The inference process provide all the inferred facts produced by the both ontology and rule reasoning .The Analyst could perform some actions. On one hand, he can decide to insert the inferred information into the system passing to the next step in the debugging session. On the other hand, he can decide to discard totally or partially the inferred information and to repeat the inference process changing any rule definition or enabling/disabling totally or partially some rules in order to test and validate the correctness of the rules available in the system. In this case, the RECC framework has detected an inconsistency in the knowledge base due to the existence of a conflict between the inferred facts according to ExamRule and BacklogRule. Both rules force the S to be Examinee and detained at same time .This situation violates the disjoint ness of Examinee and detained classes. RECC is able to detect this conflict and offers mechanisms to solve it manually. These mechanisms consist of the selective deletion of one of the conflicting facts.

V. RELATED WORK

As the popularity of the Semantic Web has rapidly increased, several ontology tools has been developed at the same time. Prot´eg´e [26] is a famous ontology tool with an OWL plug-in that allows the user to define her own ontologies, and to export them into a variety of formats including RDF(S), OWL, and XML. It also supports the edition and execution of SWRL rules [27]. SWOOP [28] is an IDE for developing ontologies, based on a Web browser interface. Hence, it allows browsing through hyper-links, which can be considered as an initial idea of showing ontologies in an intuitive way to the user. This tool has demonstrated to be useful for ontology debugging. However, it does not permit to debug or validate rules.

AEGONT [24] is an ontology development environment on .NET framework, whose major innovation lies in the Rule and Query Views, although they both are not fully functional yet. SWeDE [29] (Semantic Web Development Environment), an extensible framework built on the Eclipse IDE including an OWL editor with features like syntax highlighting, autocompletion, and error-detection. It also integrates existing tools like the OWL Validator and DumpOnt (an ontology visualizer). The second one is RuleVISor [20], an alpha-tested rule editor. Also Pronto [17] is a reasoning engine with probabilistic reasoning support that may be included in RECC framework. Nevertheless, they lack a debugging mechanism and finding inconsistency between domains restricted rule and requirement rule.

VI. CONCLUSIONS

This paper presents a framework to manage the authoring process of rules in knowledge systems based on Semantic Web technologies. The knowledge model of these systems is normally given by means of ontologies. From these ontologies it is possible to define production rules (i.e., "if-then") in order to describe requirements. in a natural and straightforward manner.

In this paper, RECC is presented in the form of a GUI. The management of ontology elements, the retrieval of the inferred facts generated by ontology and rule-based reasoning, or the activation/deactivation of rules during the inference process is some of the RECC features. On the other hand, RECC is a standalone application for users such as Analyst which offers the same operations to model requirement in form of rules that are consistent with underlying domain ontology. Once new rules have been created, they can be tested by means of the facilities incorporated in RECC. The platform which performs the inference process over the knowledge model is based on Jena and Pellet reasoning engines, although it may effortless be extended with new reasoner capabilities (e.g. fuzzy inference). As for debugging and validation, both syntactic and semantic checking of rule definitions has also been included in the RECC framework. The former avoids ill-formed rules, by warning the user if the rule is being bad defined. The latter detects knowledge conflicts among rules, which usually are complicated to discover. These conflicts arise because of numerous causes: contradictory consequents, ontology axiom violation, etc. The conflict is then reported to the user, and besides a manual solving mechanism is provided.

The benefits of RECC have been illustrated by integrating this framework into a university system. In this system we have developed a scenario where intelligent services are implemented by combining different kind of knowledge such as the current context, user's preferences and desired behaviors of the system. Such preferences and behaviors are expressed by means of rules in this scenario. The entire cycle of managing these rules, including the inference process and conflict detection, has been demonstrated in this scenario by means of the usage of RECC.

REFERENCES

1. Franz Baader and Ulrike Sattler. "Tableau Algorithms for description Logics". In *International Conference on Automated Reasoning withAnalytic Tableaux and Related Methods, TABLEAUX 2000*, pages 1–18. Springer-Verlag, July 2000.
2. Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and eter F. Patel-Schneider, editors. "*The Description Logic Handbook: Theory, Implementation, and applications*". Cambridge University Press, New York, NY, USA, 2003.
3. Tim Berners-Lee, James Hendler, and Ora Lassila. "The Semantic Web". *Scientific American*, 284(5):3443, 2001.
4. Matthew Horridge, Sean Bechhofer. The OWL API: A Java API for OWL Ontologies. Semantic Web Journal 2(1), Special Issue on Semantic Web Tools and Systems, pp. 11-21, 2011
5. [Bozsak et al. 2002] Erol Bozsak and et al. "KAON - Towards a Large Scale Semantic Web." In *EC-WEB '02: Proceedings of the Third International Conference on E-Commerce and Web Technologies*, pages 304–313, London, UK, 2002. Springer- Verlag.
6. [Carrol et al. 2004] Jeremy J. Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. "Jena: implementing the Semantic Web recommendations." In *Proceedings of the 13th international World Wide Web conference*, pages 74–83. ACM Press, 2004.
7. Mike Dean, Dan Connoll, Frank van Harmelen, James Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. Web Ontology Language (OWL). Technical report, W3C, 2004. http://www.w3.org/TR/owl-features/
8. Felix J. Garc´ıa, Gregorio Mart´ınez, Andr´es Mu˜noz, Juan A. Bot´ıa, and Antonio F. G´omez Skarmeta. "Towards Semantic Web-

based Management of Security Services". *Springer Annals of Telecommunications*, 63(3-4):183–194, 2008.
9. Mime Sweeper Research Group. RdfExpert: A Web-powered Expert System for Generic Inference tasks.http://public.research.mimesweeper.com/RDF/RDFExpert/Documentation/ HTML/Overview.html, August 2001.
10. Uwe Hansmann, Lothar Merk, Martin S. Nicklous, and Thomas Stober. "*Pervasive Computing : The Mobile World"*. Springer, 2003.
11. M. Hefke "A Framework for the Successful Introduction of KM Using CBR and Semantic Web Technologies". *Journal of Universal Computer Science*, 10(6):731–739, 2004.
12. Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. "From SHIQ and RDF to OWL: The making of a Web Ontology Language". *Journal of Web Semantics*, 1:7–26, 2003.
13. I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. Technical report, W3C, 2004. http://www.w3.org/Submission/SWRL/
14. Ian Horrocks and Peter F. Patel-Schneider. "A Proposal for an OWL Rules Language." In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 723–731, New York, NY, USA, 2004. ACM.
15. J. Joo, and S. M. Lee. "Adoption of the Semantic Web for Overcoming Technical Limitations of Knowledge Management Systems". *Expert Systems with Applications: An International Journal*, 36(3):7318–7327, 2009.
16. Michael Kifer, Georg Lausen, and James Wu. "Logical Foundations of Object-oriented and Frame-based Languages". *Journal ACM*, 42(4):741–843, 1995.
17. Pavel Klinov. Pronto: a Non-monotonic Probabilistic Description Logic Reasoner. In *Proceeding at 5th European Semantic Web Conference (ESWC08)*, 2008.
18. Graham Klyne and Jeremy J. Carroll. Resource Description Framework RDF): Concepts and Abstract Syntax. Technical report, W3C, 2004. *http://www.w3.org/TR/rdf-concepts/*
19. Thorsten Liebig, Holger Pfeifer, and Friedrich von Henke. "Reasoning Services for an OWL Authoring Tool: An Experience Report". In *Proceedings of the International Workshop on Description Logics*, 2004.
20. Christopher J. Matheus, and Kenneth Baclawski Mieczyslaw M. Kokar, and Jerzy A. Letkowski. An Application of Semantic Web Technologies to Situation Awareness. In *4th International Semantic Web Conference*, 2005.
21. Boris Motik, Peter F. Patel-Schneider, and Ian Horrocks. OWL 2 Web Ontology Language: Structural Specification and Functional-style Syntax. Technical Report, W3C, April 2008. http://www.w3.org/TR/owl2-syntax/
22. Boris Motik, Ulrike Sattler, and Rudi Studer. "Query Answering for OWL-DL with Rules". *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 3(1):41–60, JUL 2005.
23. [Horrocks and Sattler 2007] Ian Horrocks and Ulrike Sattler. A Tableaux Decision Procedure for SHOIQ In *Journal of Automated Reasoning* Springer Verlag, 39(3):245–429, 2007
24. Tugba Ozacar Murat Osman Unalir, Ovunc Ozturk. Aegean Ontology Environment (AEGONT). Project Report MSR 2003176, Ege University, Bornova-Izmir, March 2006.
25. Dmitry Tsarkov and Ian Horrocks. *Automated Reasoning*, volume 4130 of *Springer Lecture Notes in Computer Science*, chapter FaCT++ Description Logic Reasoner: System Description, pages 292–297. Springer Berlin / Heidelberg, 2006.
26. N. F. Noy, M. Sintek, S. Decker, M. Crubezy, R. W. Fergerson, and M. A. Musen. "Creating Semantic Web Contents with Protege-2000". *IEEE Intelligent Systems*, 16(2):60–71, 2001.
27. Martin O'Connor, Holger Knublauch, Samson Tu, Benjamin N. Grosof, Mike Dean, William Grosso, , and Mark Musen. "Supporting rule system interoperability on the Semantic Web with SWRL". In *Proceeding of 4th International Semantic Web Conference ISWC*, Ireland, Nov 2005.
28. Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. "Debugging OWL ontologies". In *Proceedings of the 14th international conference on World Wide Web*, pages 633–640, 2005.
29. R.G. Pereira and M.M. Freire. "SWEDE: A Semantic Web Editor Integrating Ontologies and Semantic Annotations with Resource Description Framework". In *International Conference on Internet and Web Applications and Services/Advanced AICT-ICIW '06*, 2006.
30. Jos De Roo. Euler Proof Mechanism. http://www.agfa.com/w3c/euler/, 10 2007.
31. Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. "Pellet: A Practical OWL-DL Reasoner". *Journal of Web Semantics*, 5(2):51–53, 2007.
32. Pramuditha Suraweera, Antonija Mitrovic, and Brent Martin. "The Role of Domain Ontology in Knowledge Acquisition for ITSs. In *7th International Conference on Intelligent Tutoring Systems*, pages 207–216, 2004.