CoDR: A Contextual Framework for Diagnosis and Repair*

Klaas Dellschaft
ISWeb Working Group
Universität Koblenz-Landau
D-56070 Koblenz, Germany
klaasd@uni-koblenz.de

Qiu Ji
Institute AIFB
Universität Karlsruhe
D-76128, Karlsruhe, Germany
qiji@aifb.uni-karlsruhe.de

Guilin Qi
Institute AIFB
Universität Karlsruhe
D-76128, Karlsruhe, Germany
gqi@aifb.uni-karlsruhe.de

ABSTRACT

Ontologies play a central role for the formal representation of knowledge on the Semantic Web. A major challenge in collaborative ontology construction is to handle inconsistencies caused by changes to the ontology. In this paper, we present our CoDR system which helps to diagnose and repair collaboratively constructed ontologies. CoDR integrates RaDON, an ontology diagnosis and repair tool, and Cicero, which provides discussion functionality for the ontology developers. CoDR is realized as a plugin for the NeOn Toolkit. It helps to use discussions held in Cicero as context information during repairing an ontology with RaDON. But it is also possible to use the diagnoses from RaDON during the discussions in Cicero.

1. MOTIVATION

A typical problem in collaborative ontology engineering is that conflicting or inconsistent statements are introduced in an ontology. The main reason is that ontologies are often too complex for foreseeing all side-effects of modifications to them. Many side-effects may be detected by discussing the issues and possible solutions with other ontology developers.

In Section 2, we shortly summarize our work on Cicero [1]. It is used for facilitating discussions between ontology engineers. But even the most thorough discussion can not detect all possible side-effects of a modification. A further problem of online collaboration is that people are often reluctant to discuss every problem especially if its solution seems to be obvious and straightforward to realize.

Thus, on a regular basis conflicting or inconsistent statements will be introduced when it comes to changing an ontology. In Section 3, we shortly summarize our work on RaDON [2] which helps in detecting conflicting or inconsistent statements. Furthermore, it may provide possible solutions for resolving the inconsistencies.

Finally, in Section 4 we describe the CoDR framework which establishes a feedback circle between discussions in Cicero and the debugging process in RaDON. For example, the discussions can be used as context information during selecting one of the possible solutions proposed by RaDON. Furthermore, one can reuse the diagnoses of RaDON as arguments in the discussion process or for starting a completely new discussion how to best resolve an inconsistency.

*This work has been partially supported by the European project Lifecycle Support for Networked Ontologies (NeOn, IST-2006-027595). http://www.neon-project.org/.

2. DISCUSSIONS

Cicero [1] is a wiki based solution that facilitates an asynchronous discussion and decision making process between participants of an ontology engineering project. Its two main objectives are a more efficient discussion process and an enhanced documentation of the design rationale of an ontology.

The argumentation model, that is underlying the Cicero tool, is based on the DILIGENT argumentation framework [4] and the Potts and Bruns model [5]. In Cicero, a discussion always starts with an issue raised by a user of the ontology which may be annotated to one or more axioms in the ontology. After creating the issue, possible solutions can be discussed by the different participants in the ontology engineering project. Cicero also supports the users in deciding which of the solutions should be implemented. Finally, during implementing the solution, the discussion is annotated to all deleted, modified or newly introduced axioms.

3. DIAGNOSIS AND REPAIR

RaDON is used for diagnosing the reasons for the unsatisfiability or incoherence of an ontology. The diagnosis is based on minimal subsets of axioms from the ontology which preserve the unsatisfiability or incoherence (see [2, 6]). The ontology can be repaired by removing or modifying axioms contained in these minimal subsets.

The number of available diagnoses depends on the number of axioms in the minimal subsets. For example, if there exist n minimal subsets with m_i axioms ($i = 1, ..., n$) then there are $m_1 \times \cdots \times m_n$ diagnoses and ways to repair the ontology. For example, if each of 3 minimal subsets contains 4 axioms then there exist 64 different diagnoses.

In a first step, the user may identify axioms from the minimal subsets which are correct and thus shouldn't be removed or modified. This decision may be based on the discussions that are annotated to the axioms. For example, if from each subset 2 axioms can be identified to be correct, only 8 of the originally 64 diagnoses remain.

After selecting one of the remaining diagnoses, the user has to decide whether he needs to completely remove the axioms or whether it is sufficient to weaken and/or modify some of them. Again, the decision may be based on the annotated discussions. All in all, in both steps a tighter integration of Cicero and RaDON eases the decision taking process during repairing an ontology.

4. ONTOLOGY EDITING WORKFLOW

In the following, we will describe the CoDR ontology editing workflow (see Fig. 1). It allows for moving from one

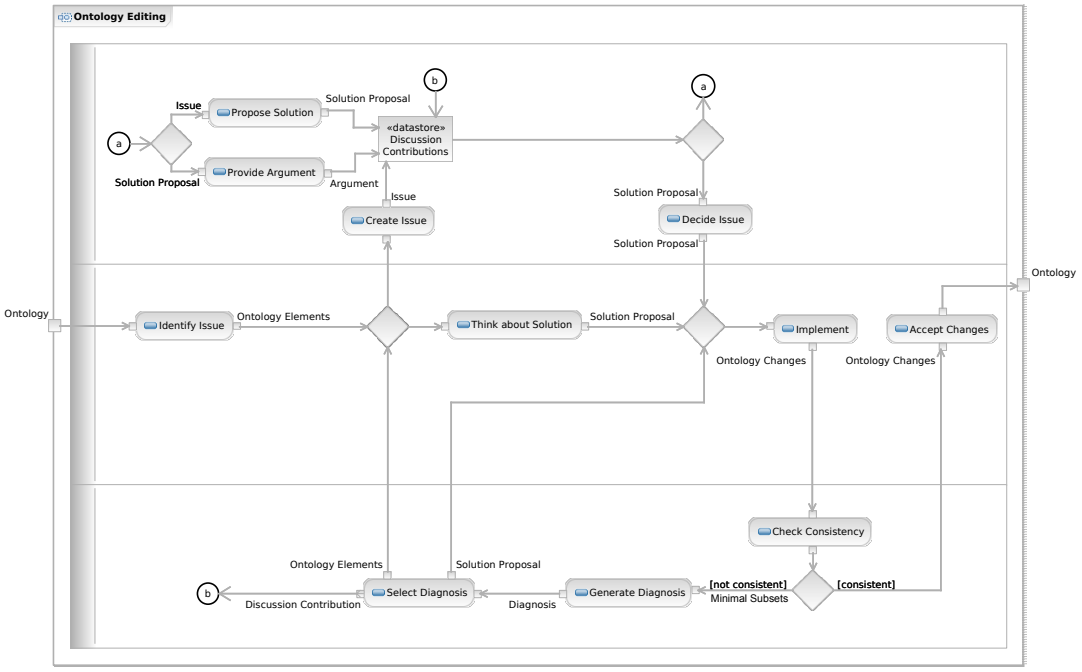

Figure 1: The CoDR ontology editing workflow.

consistent state of an ontology to the next consistent state and for resolving inconsistencies in collaboration with other developers. The CoDR workflow is realized by different plugins for the NeOn Toolkit[1]. Besides the RaDON and Cicero plugin, it also uses the Oyster plugin and server. Oyster is a central ontology registry which allows for exchanging ontologies, ontology changes and ontology related metadata [3]. In our case, it is used to capture ontology changes and to propagate them to the other collaborating developers.

The CoDR workflow always starts with a consistent or empty ontology. For this ontology, first an issue is identified, e. g. the ontology should be extended or existing elements should be modified. If the issue is explicitly formulated, one may use its description to create an issue in Cicero and discuss possible solutions with the other collaborating developers (see [1]). After the discussion, the developers decide on which of the solution proposals should be implemented.

But an ontology developer can also skip the discussion process in Cicero. Then the developer thinks about solutions for the issue and decides which solution he implements. Skipping the discussion process may be an option for quite simple issues. Disadvantages are that side effects are easier overlooked and that the design rationale is not documented.

In any case, after deciding on the solution, the developer starts implementing the changes in the ontology. After finishing the implementation, the user checks with the help of RaDON whether the ontology is still consistent. If the ontology is consistent, the ontology changes are accepted and propagated to the other developers of the ontology. Otherwise, RaDON will be used for diagnosing the problem.

For each diagnosis, RaDON is able to show relevant discussions in Cicero which are annotated to the ontology elements in the minimal subsets (see Section 3). After inspecting the diagnoses and the related discussions in Cicero, the user has several choices how to further proceed: (1) He may select one of the diagnoses as a solution proposal and implement it in the ontology, (2) he may attach the diagnoses as possible solution proposals to an already existing discussion in Cicero, or (3) he may start a new discussion about how to deal with the issues that were spotted by RaDON.

[1] `http://www.neon-toolkit.org/`

5. CONCLUSIONS

We have presented the CoDR ontology editing workflow. It helps to quickly spot and resolve logical inconsistencies which were introduced during previous changes to an ontology. The workflow is realized by plugins for the NeOn Toolkit. The plugins and installation instructions are available on the RaDON website at `http://radon.ontoware.org/demo-codr.htm`.

6. REFERENCES

[1] K. Dellschaft, H. Engelbrecht, J. M. Barreto, S. Rutenbeck, and S. Staab. Cicero: Tracking design rationale in collaborative ontology engineering. In *Proc. of ESWC*, pages 782–786, 2008.

[2] Q. Ji, P. Haase, G. Qi, P. Hitzler, and S. Stadtmüller. Radon - repair and diagnosis in ontology networks. In *Proc. of ESWC*, pages 863–867, 2009.

[3] R. Palma, P. Haase, and A. Gómez-Pérez. Oyster: sharing and re-using ontologies in a peer-to-peer community. In *Proc. of WWW*, pages 1009–1010, 2006.

[4] H. S. Pinto, S. Staab, and C. Tempich. DILIGENT: Towards a fine-grained methodology for Distributed, Loosely-controlled and evolving Engineering of oNTologies. In *Proc. of ECAI*, 2004.

[5] C. Potts and G. Bruns. Recording the reasons for design decisions. In *ICSE*, pages 418–427, 1988.

[6] S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proc. of IJCAI*, pages 355–362, 2003.