Tool Support for the Integration of Light-Weight Ontologies

Thomas Heer[1], Daniel Retkowitz[1], and Bodo Kraft[2]

[1] Department of Computer Science 3, RWTH Aachen University,
Ahornstr. 55, 52074 Aachen, Germany
{heer|retkowitz}@i3.informatik.rwth-aachen.de
[2] AMB Generali Informatik Services GmbH,
Anton-Kurze-Allee 16, 52064 Aachen, Germany
bodo.kraft@amb-informatik.de

Abstract. In many areas of computer science ontologies become more and more important. The use of ontologies for domain modeling often brings up the issue of ontology integration. The task of merging several ontologies, covering specific subdomains, into one unified ontology has to be solved. Many approaches for ontology integration aim at automating the process of ontology alignment. However, a complete automation is not feasible, and user interaction is always required. Nevertheless, most ontology integration tools offer only very limited support for the interactive part of the integration process. In this paper, we present a novel approach for the interactive integration of ontologies. The result of the ontology integration is incrementally updated after each definition of a correspondence between ontology elements. The user is guided through the ontologies to be integrated. By restricting the possible user actions, the integrity of all defined correspondences is ensured by the tool we developed. We evaluated our tool by integrating different regulations concerning building design.

Key words: Knowledge Management, Ontology Engineering, Information Integration Tools, Human Factors

1 Introduction

Our approach to ontology integration has been developed in the context of the ConDes research project [1]. In this project we have developed new concepts for software tools to support the conceptual design phase in building design. Thereby a knowledge-based approach has been followed. The relevant terminology is defined in several domain-specific ontologies. Based on theses ontologies restrictions for the conceptual design of a building can be specified. Therefore, the example ontologies in this paper come from the domain of building engineering, but our approach for interactive ontology integration is applicable to many other domains.

There is a broad field of different types of structures that are all subsumed by the term *ontology* [2]. Ontologies can be simple *vocabularies*, i. e. lists of terms, which denote the entities of a certain domain. If a generalization relation is

[HRK09] T. Heer, D. Retkowitz, B. Kraft
Tool Support for the Integration of Light-Weight Ontologies
In: Joaquim Filipe, José Cordeiro (eds.): Enterprise Information Systems, 10th International Conference, ICEIS 2008, Barcelona, Spain, June 12-16, 2008, Revised Selected Papers, volume 19 of LNBIP, pages 175-187. Springer-Verlag, 2009
www.se-rwth.de/publications

defined for these terms, one speaks of a *taxonomy*. Both of these types are called *light-weight ontologies*. Light-weight ontologies define *concepts*, classifications of these concepts, properties and relations. In contrast to that, *heavy-weight ontologies* comprise further semantical information about a domain. This additional information is specified by axioms or constraints. Ontologies can describe concepts on different levels of abstraction. Ontologies, which define general concepts, are called upper ontologies, foundation ontologies, or *top-level ontologies* [3]. Ontologies, which contain knowledge about a specific domain, are called *domain ontologies*.

With regard to ontologies the term *integration* is used with several different semantics. Three types of ontology integration can be distinguished [4]. The first type is integration in terms of *reuse*. This means constructing a new ontology based on already existing ontologies, which are incorporated in the new ontology. A second type is integration in terms of *merging*. In this case, two or more ontologies are unified into a single ontology by merging corresponding concepts in the original ontologies. The third type is integration in terms of *use*. This type of integration is applied, when applications are built, which are based on one or more ontologies. In our approach, we use the term *integration* in the second sense, i. e. in terms of *merging*.

Before merging different ontologies into one unified ontology, a prior alignment of these is required [4]. *Alignment* is the process, in which the relations between the concepts contained in the different ontologies are determined. This is usually done by the definition of a mapping between the ontology elements. This mapping defines how the source ontologies have to be merged into one integrated ontology, so that the resulting ontology contains all the semantic information of the source ontologies, not more and not less. One difficulty in the alignment of different ontologies comes from the fact, that the structure of an ontology is not only determined by the comprised knowledge, but also by the design decisions made during its development. Therefore, even ontologies, which model the same part of a certain domain, may be structured significantly different. This makes the integration of the ontologies a difficult task.

The paper is structured as follows. First, we give an overview of related work in section 2. Then, in section 3, we will describe how to define semantic correspondences and how to generate an integrated ontology from these correspondences. The following section 4 contains the description of our developed integration algorithm. Next, in section 5, we describe, how the integrity of the defined semantic correspondences is ensured. In section 6, we give a short overview of the tool we developed to implement the integration approach. Finally, we give a conclusion and an outlook on further possible developments at the end of the paper.

2 Related Work

Ontologies and ontology integration are still emerging topics in the field of computer science. Many approaches for the use and integration of ontologies have been proposed in research.

In [5] different techniques for the alignment of ontologies are described. These are *manual definition of correspondences*, use of *linguistic heuristics*, *top-level grounding* and the use of *semantic correspondences*. These techniques are not exclusive, but rather complement each other. The first technique requires a knowledge engineer, developing an ontology, to manually define certain correspondences between the concepts of the ontologies to integrate. These correspondences mainly have the semantics of equivalence, but are not restricted to 1:1 relations. In the second method, heuristics are applied to find correspondences automatically based on linguistic features of the terms, representing the concepts. The method of top-level grounding requires a common top-level ontology for all ontologies to be integrated. This top-level ontology is then used to identify related concepts, and to use this information as a basis for the integration. Finally, semantic correspondences can be defined. In this method, different types of semantic relations are used to relate the concepts of the ontologies to integrate. This way, not only equivalence relations, but also relations with other semantics can be defined. In our approach, we use the techniques of top-level grounding and semantic correspondences.

In [6] and [7] surveys over existing approaches to ontology alignment are presented. Both works give an overview over theoretical frameworks and several current research projects. The surveyed works range from formal over heuristic approaches to approaches, which use machine learning to automate the process of ontology alignment. However, most of the presented works more or less neglect the issues involved with the interactive part of the integration process. In the following we present two examples of related works, which use heuristics for the alignment of ontologies.

One alternative to align ontologies is, to consider lexical similarities between the terms, which represent the defined concepts. Such a lexical integration approach is implemented by the tool *Chimaera* [8]. Chimaera is an environment, which can be used for merging and testing ontologies. When integrating ontologies, Chimaera generates lists of suggestions for equivalent terms from the ontologies. These suggestions are based on lexical similarity measures. Besides that, Chimaera can identify parts of the class hierarchy, which probably need to be reorganized. These parts are identified by means of heuristic strategies. Since Chimaera uses heuristics based on lexical analysis, the identified similarities may contain mismatches. Thus, it is necessary that the user verifies all suggestions made by the tool. However, Chimaera does not propose any solutions in case of conflicts, which may arise during the integration process.

In [9], an algorithm for semi-automatic merging and alignment of ontologies called PROMPT is presented. This algorithm realizes a semi-automatic integration of ontologies. The Anchor-PROMPT algorithm [10] is an extension to PROMPT. It is used to generate suggestions, which are not only based on linguistic similarity, but also on structural properties of the ontologies. In the first step, PROMPT generates suggestions for correspondences between the classes of the ontologies, to be integrated. These initial suggestions are based on linguistic similarities of the class names and the structure of the ontologies. The latter is analyzed by the

Anchor-PROMPT algorithm. In the next step, the user selects for each suggestion an operation to perform or defines a different operation manually. PROMPT then automatically performs the selected operation and applies additional modifications to the merged ontology, if required. Subsequently, the list of suggestions is updated and a list of conflicts, which resulted from the previous operation, is generated. After this, the procedure is executed again, until no more operations have to be performed, and all suggestions are processed.

In [11], ontologies are used to integrate database schemata. To perform the integration, the schemata are augmented by corresponding ontologies that define the schema semantics. These ontologies are then integrated to a global ontology from which a unified schema can be derived. To integrate the ontologies, similarity relations between schema concepts are defined. In [11] the same four types of similarity relations are used as in our ontology integration approach. It is described, how the resulting integrated ontology can be derived from the source ontologies and the defined correspondences. There are different suggestions, how to define the similarity relations between ontology elements, neither of which is discussed in detail. One suggestion is, to provide common references by using a higher level ontology. Other possibilities are, to use thesauruses, experts familiar with both ontologies, or a hybrid semiautomatic method. However, no concepts are proposed, how an expert could be supported in defining the similarity relations, which includes finding corresponding elements and choosing the right relation types. Nothing is said about, how to ensure the integrity of the defined correspondences, and how to take the effects of defined correspondences on the integration result into account during the alignment of the ontologies.

3 Semantic Correspondences

In our approach of merging lightweight ontologies, semantic correspondences are used to relate the concepts of different ontologies to each other. Following an interactive incremental process, a knowledge engineer defines correspondences between elements of the ontologies to be integrated. Based on these correspondences an integrated ontology is automatically generated.

There are four types of correspondences with different semantics: *equivalence*, *overlap*, *generalization* and *disjointness*. The chosen terms for the semantic correspondence types *overlap* and *disjointness* are rooted in the field of set theory. In our ontology integration scenario, the elements of ontologies are terms, structured in a generalization hierarchy. The terms represent concepts. Hence, correspondences between ontology elements relate concepts to each other. A concept defines a mental collection of objects or circumstances, which have common attributes. This collection is called the *extension* of the concept [11]. Extensions are basically sets. Thus, between two extensions one of the four possible relationships between sets must hold. The extensions of two concepts can be equal or disjoint, one can be a subset of the other, or the two extensions can overlap, i. e. they have a nonempty intersection, but are neither equal nor in a set-subset relation. From these four possible relations between sets the

four correspondence types equivalence, disjointness, generalization and overlap are derived. In figure 1, examples of corresponding ontology elements and their extensions are shown. For example the generalization correspondence from ladies restroom to restroom implies that the extension of the former concept is a subset of the extension of the latter concept. Informally speaking, all ladies restrooms are restrooms. The disjoint correspondence is a special case insofar as it is not explicitly represented by an edge between ontology elements. Whenever no correspondence of one of the other three types is defined between two concepts, they are implicitly defined as disjoint.

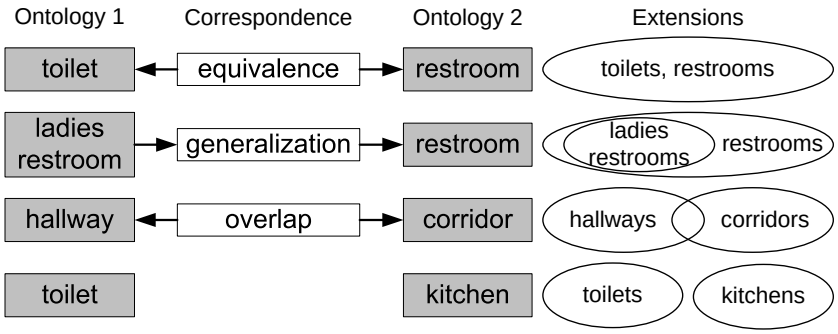

Fig. 1. Corresponding concepts and their extensions

The defined correspondences between ontology elements allow for the automatic generation of a merged ontology. In our approach an arbitrary number of source ontologies can be merged into one ontology. To illustrate this, we consider here the result of the integration example from section 4. In figure 3 c) cutouts of three ontologies from the domain of building construction are shown along with the resulting merged ontology. Several correspondences are defined between the elements of the source ontologies, e. g. the terms toilet and restroom are defined as equivalent. The terms dining room and living room are defined as overlapping, since there are rooms that have the functionality of both, like e. g. a family room. Therefore family room is defined as a specialization of dining room.

If several ontology elements are defined as equivalent, then the merged ontology only contains one representative for the equivalence class. Therefore the merged ontology in figure 3 c) only contains one element room and only the element restroom for the two equivalent ontology elements toilet and restroom. For each defined generalization correspondence a generalization edge is generated in the merged ontology. Afterwards, redundant generalization edges are removed. An overlap correspondence indicates that there are objects in the modeled domain, which belong to both corresponding concepts. When specifying an overlap correspondence, the knowledge engineer, who carries out the ontology integration, can choose between two options. Either an ontology element, which represents the intersection of the overlapping concepts, is generated for the overlap correspon-

dence in the merged ontology, or the overlap correspondence is simply defined to indicate the overlapping of the two concepts without any direct influence on the resulting ontology. In the example in figure 3 c) no ontology element is generated.

During the execution of the integration procedure, the knowledge engineer selects elements of the merged ontology to define correspondences, but the correspondences are established between the elements of the source ontologies or correspondences, which generate the selected elements of the merged ontology.

The merged ontology is not generated at once, after all correspondences have been defined. It is incrementally updated throughout the execution of the integration algorithm, which is described in the following section.

4 Integration Algorithm

In our integration approach an arbitrary number of ontologies can be merged into one ontology, which contains all the semantic information of the source ontologies. In the first step two ontologies are integrated. After that, all remaining ontologies are integrated one by one into the merged ontology, where each ontology is integrated with the current intermediate result. Except for the choice of terms, which represent the concepts in the merged ontology, the integration result is independent of the order, in which the source ontologies are integrated. This is guaranteed, because the correspondences defined throughout the integration process are established between elements of the original source ontologies. The source ontologies remain unchanged in the knowledge base, and the merged ontology can be generated from the source ontologies and the defined correspondences at any time.

In our ontology integration approach there is no strict distinction between the phases of ontology alignment and merging. In the related work about ontology integration [6, 7] different approaches are presented for the alignment of ontologies, which is often regarded as the first step of the integration procedure. The actual merging of ontologies is then carried out in a second step, based on the previously defined correspondences. This way, many dependencies between the defined correspondences are not taken into account. Especially, if the alignment is performed by a human being, e. g. a knowledge engineer, it is difficult for this person to oversee all effects of defined correspondences on the integration result. Therefore we follow a different approach. The knowledge engineer always works with the current intermediate result of the ontology integration. He defines correspondences between elements of this merged ontology. Whenever the knowledge engineer defines a correspondence, the merged ontology is immediately updated, so that he can directly see the effects of his action. So ontology alignment and merging steps alternate throughout the integration process.

Defining semantic correspondences between ontology elements is a difficult task [12]. It is difficult to identify those ontology elements, which are related. Especially in case of large ontologies, without any guidance a knowledge engineer often does not know where to find corresponding elements. If corresponding elements have been identified, it is often hard to decide, which type of correspondence

should be established. Sometimes correspondence types are chosen, which conflict with other correspondences. This happens, because effects of previously defined correspondences are hard to oversee.

Our ontology integration approach provides solutions to the aforementioned problems. It aims at providing tool support for the interactive integration of ontologies. The effects of defined correspondences are on the one hand taken into account by the fact, that the knowledge engineer always works with the current intermediate result. On the other hand, defined correspondences restrict the possibilities for defining new correspondences.

The problems of finding corresponding ontology elements and defining the correspondences in the right order are addressed by two aspects of our integration algorithm. First, a restrictive traversing order of the merged ontology is enforced. And second, those ontology elements, for which correspondences can be defined at a certain point in the integration processes, are restricted to a manageable number. This way the knowledge engineer is guided through the merged ontology, and his attention is focused to a relatively small part of the possibly large ontology.

Our integration approach relies on the assumption, that all ontologies use a common top-level ontology. When integrating two ontologies, one can assume that the roots are equivalent concepts. Thus, in a first step a top-level grounding of the two ontologies is performed. After the definition of equivalence correspondences between the roots of the ontologies, a first version of the merged ontology is generated. In this merged ontology the corresponding ontologies are combined by unifying their roots. After the top-level grounding an adapted breadth-first traversing is performed. The traversing is steered by the defined correspondences. At each point during the integration of two ontologies some elements are highlighted. The knowledge engineer is only allowed to define correspondences between these highlighted elements.

In figure 2 the highlighting of ontology elements depending on the type of a previously defined correspondence is shown. For example in figure 2 a) an equivalence correspondence has been established between the ontology elements A and 1. At a later point in the integration procedure the knowledge engineer is asked to define correspondences between the highlighted elements B, C, 2 and 3. In figure 2 b) a generalization correspondence between A and 1 has been established. The ontology element A is defined to be a specialization of 1. Hence, in a later step the relationships between A and the specializations of 1, namely 2 and 3, have to be clarified. The case of an overlap correspondence constitutes a special case. An overlap correspondence provides the least information about the relationships of the specializations of the linked objects. Thus, the highlighting of ontology elements is conducted in three steps. In a first step, one of the corresponding elements and the specializations of the other are highlighted, like it is shown in figure 2 c). In this step it is not allowed, to define equivalence correspondences between A and either 2 or 3, because this would imply, that A is a generalization of 1. For the same reason it is furthermore not allowed to define a generalization correspondence with A as source and one of the specializations of 1 as target. In a second step the highlighting of the ontology elements is the other

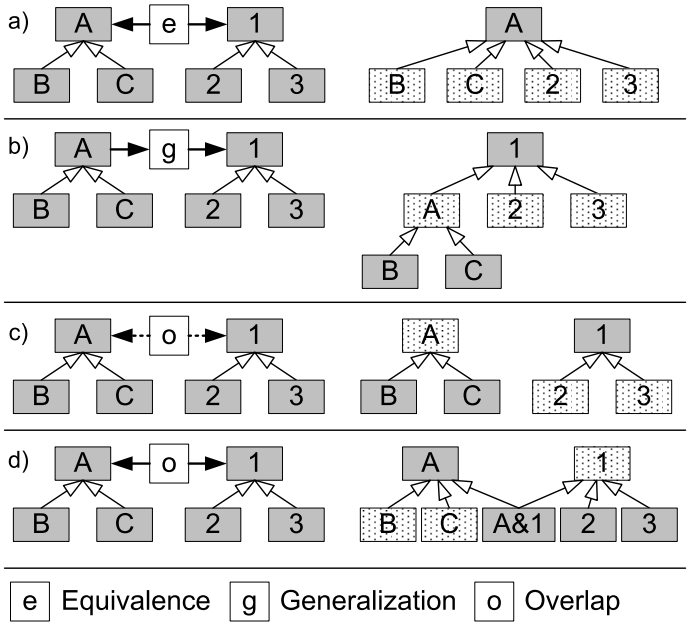

Fig. 2. Highlighting of ontology elements.

way round as in the first step, while the same restrictions apply. This situation is depicted in figure 2 d). While in figure 2 c) the case of an overlap correspondence without generation (indicated by the dashed arrows) of an ontology element is shown, in figure 2 d) the element for the intersection is generated. This element is not highlighted in steps one and two, because its relationships to the overlapping concepts are already defined. Finally, in the third step all specializations of the overlapping concepts are highlighted, including a potentially generated element, to give the knowledge engineer the opportunity to clarify their relationships. This situation is not depicted in figure 2.

Figure 3 shows some steps of the integration algorithm by example. Figure 3 a) shows the situation directly after the top-level grounding of ontology 1 and ontology 2, as explained earlier. The root concepts of the ontologies are linked by an equivalence correspondence. Hence, the generated merged ontology depicted in figure 3 a) on the right contains only one element room and all specializations of room from the two ontologies are children of this root element. The elements sanitary room, toilet and dining room are highlighted. In the following, the knowledge engineer defines a generalization correspondence from toilet to sanitary room. This is the only correspondence, he defines for the highlighted elements, and thus the algorithm proceeds. Because of the defined generalization correspondence, in figure 3 b) the ontology elements restroom and toilet are highlighted. The knowledge engineer defines an equivalence correspondence between these elements. In this case, the generalization correspondence is modified, so that it references the equivalence correspondence instead of the ontology element toilet. This is

depicted in figure 3 c). Figure 3 c) shows the final situation, in which a third ontology has been integrated with the other two ontologies. The resulting merged ontology contains all the semantic information of the source ontologies.

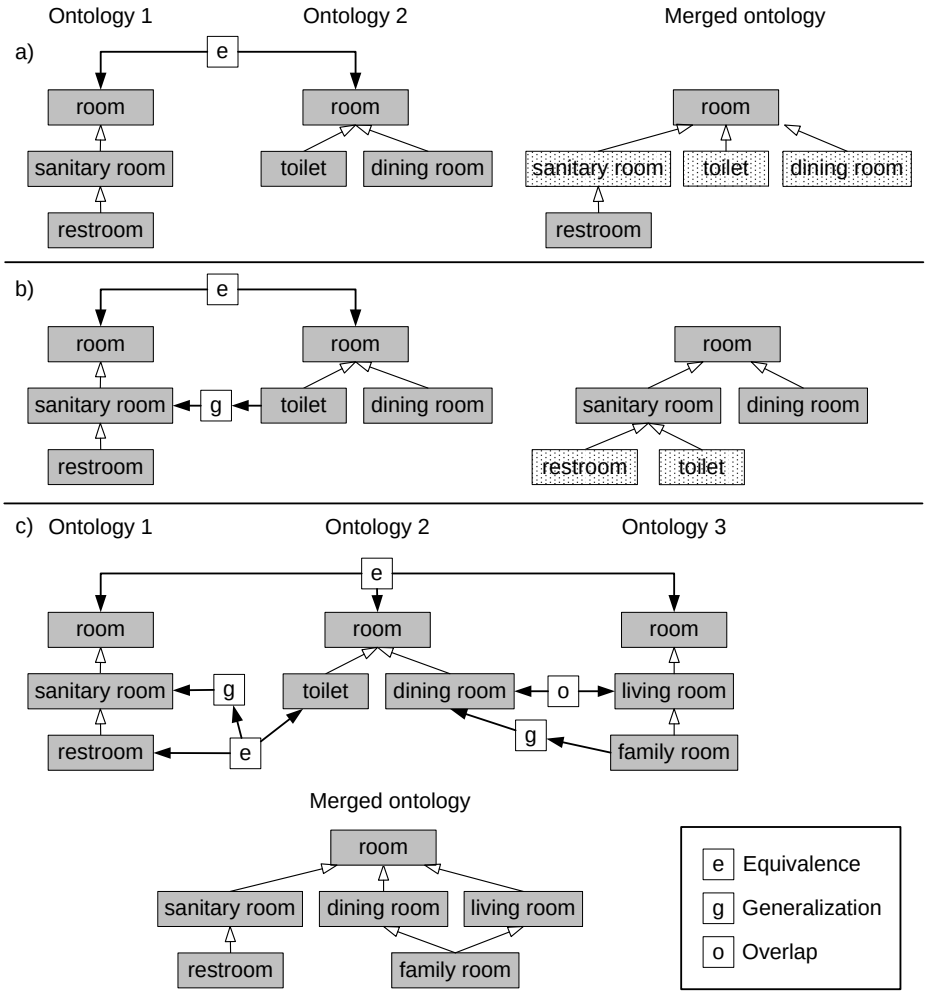

Fig. 3. Example steps of the integration procedure.

5 Correspondence Integrity

Many tools for the integration of ontologies generate suggestions for possible semantic correspondences between ontology elements, but provide no assistance for choosing the right correspondences in the right order. In our view the main

functionality of a tool, which provides support for the alignment of ontologies, should be, to ensure the integrity of user-defined semantic correspondences. The integrity is ensured, if no conflict between defined correspondences exists regarding their semantics.

In our ontology integration approach, the integrity of defined correspondences is ensured by several means. Some of these have already been described in section 4. The breadth-first traversing order ensures, that correspondences between general concepts of the ontologies are defined before correspondences between their specializations. The former restrict the possibilities for defining the latter. The incremental integration approach reduces the possibility of defining inconsistent correspondences, because changes like the unification of equivalent elements are immediately performed on the merged ontology. The highlighting of ontology elements and the restrictions, which hold for defining correspondences between them, also contribute to ensuring the integrity of defined correspondences, as described in section 4. These restrictions can be seen as *static restrictions*, as they do not depend on previously defined correspondences between the highlighted elements and their generalizations.

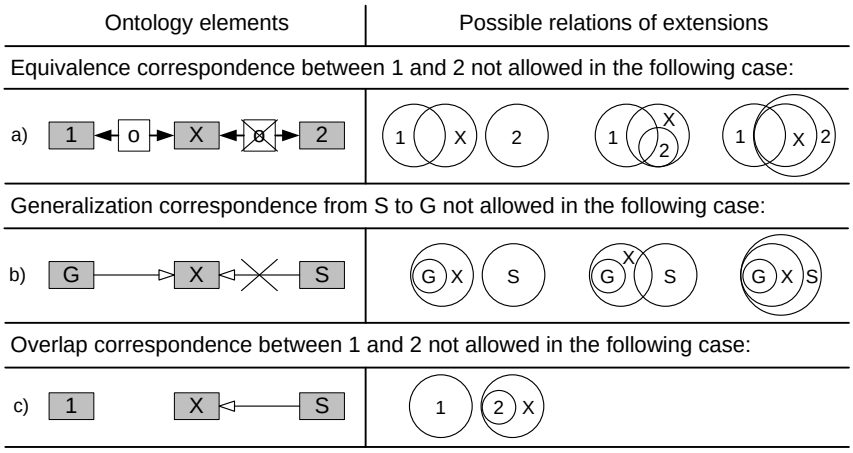

Fig. 4. Examples for dynamic restrictions for correspondences.

However, when defining correspondences between the highlighted ontology elements, all previously defined correspondences between the highlighted elements, between their generalizations, and between the former and the latter have to be taken into account. Thereby, we consider the type of the correspondence to be defined as well as the types of previously defined correspondences. Restrictions, which arise through this consideration, are called *dynamic restrictions*. To determine all dynamic restrictions for a certain correspondence type, we examined the extensions of those concepts that should be linked by the new correspondence. In figure 4 for each correspondence type one example case is shown, in which the definition of a correspondence is prohibited. The restrictions result from the

relationships of the concepts' extensions. Crossed out correspondences mean, that in the defined case there must not be a correspondence of the given type, e. g. in figure 4 b) the pattern includes all cases, where there is no generalization correspondence from S to X.

The example in figure 4 b) is a case in which no generalization correspondence can be established between the ontology elements G and S, where G should be defined as the more general concept and S as its specialization. The ontology element G is a specialization of another ontology element X, while S is not. On the right side, the possible relations between the extensions of G, S and X are shown. The extensions of X and S can be disjoint, they can overlap, or the extension of X can be a subset of the extension of S. However, in any case it is impossible, that the extension of S is a subset of the extension of G. Hence, a generalization correspondence from S to G is not allowed.

The fewest restrictions apply for the definition of an overlap correspondence, because it provides the least information about the relation of the linked concepts, and is thus compatible with most other correspondences.

Whenever a knowledge engineer is going to define a new correspondence, all dynamic and static restrictions are checked. If some restriction would be violated by establishing the correspondence, then the according user action is prohibited by our tool and the user is informed about the reason for the denial.

Our integration algorithm can be extended by using heuristics for finding correspondences between the highlighted ontology elements. This could be linguistic similarity measures like in [8] or heuristics, which take the structure of the ontology graph into account, like in [10]. In this way, it could be possible, to achieve a high degree of automation of the integration process. Correspondences between highlighted ontology elements would be automatically generated, if their similarity measure would succeed a certain threshold, and their definition would not violate any restrictions. User interaction would only be required in undecidable cases. We did not follow this approach, because our main focus lied on the support for interactive ontology integration.

6 Tool Support

We implemented our approach by developing a graph-based tool for ontology integration. The underlying data structure for the representation of ontologies is a graph. All ontologies to be integrated are subgraphs of a common host graph. The host graph is stored in a GRAS database [13], which is a specialized database for graph structures. We used the graph rewriting system PROGRES [14] to specify transformations on a the host graph, which contains besides the representations of the source ontologies also the correspondences and the resulting merged ontology. The application logic, which constitutes the core part of the integration tool, was generated from this specification. The graphical user interface of the integration tool was realized by means of the UPGRADE framework [15].

Our visual graph-based tool provides an abstraction of the internal data structure and provides a user-friendly and problem-adequate representation. We

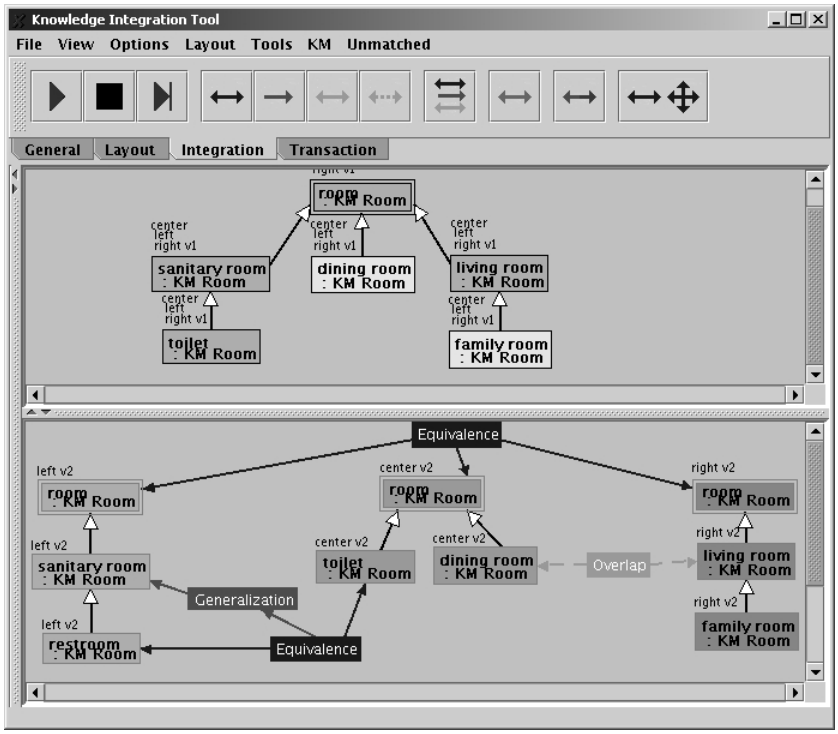

Fig. 5. Integration tool.

evaluated the applicability of our approach and the efficiency of our integration tool, by merging several large ontologies from the domain of building design. The merged ontologies contained the relevant concepts for the definition of knowledge for the conceptual design of the university hospital in Aachen.

In figure 5 a screenshot of the integration tool is depicted. The graphical user interface is divided into two views. Throughout the integration process, the upper view shows a part of the intermediate merged ontology. In this view the user can select highlighted ontology elements, and he can define correspondences between these by clicking on according toolbar buttons or context menu entries respectively. The lower view of the integration tool shows the original source ontologies together with the correspondences, which have been defined between their elements. In this view a knowledge engineer can inspect, how the correspondences are actually established between the elements of the source ontologies.

The integration proceeds as follows. The knowledge engineer selects the ontologies to be integrated via a dialog. After that, the alignment and merging of the first two ontologies takes place. After the first two ontologies have been integrated, the views are updated to the intermediate result of the integration of the first three ontologies, and so on, until all ontologies are merged into one. Whenever there are steps in the integration process, in which the user could not take any action, e. g. because no more correspondences are allowed for

the highlighted elements, then these steps are automatically skipped, and the algorithm proceeds.

7 Conclusion

In this paper we presented a novel approach for interactive ontology integration. Several different ontologies can be merged into one. The ontologies are integrated one by one, while the structure of the resulting merged ontology is independent of the order, in which they are integrated. The alignment of the ontologies relies on the definition of semantic correspondences between their elements. These correspondences are manually defined by a knowledge engineer. The knowledge engineer is supported by our graph-based tool in many ways. Alignment and merging steps alternate throughout the integration process. The intermediate result of the integration is immediately updated after each definition of a correspondence. That way, the effects of defined correspondences on the integration result become directly visible. The user is guided through the merged ontology, and his attention is focused on small parts of the ontology, where he has to define new correspondences. In this way, the common problems of finding corresponding ontology elements and defining the correspondences in the right order are substantially reduced. The integrity of all defined correspondences is ensured, because actions, which would violate it, are prohibited by the tool. Thereby, all restrictions, which arise through previously defined correspondences, are taken into account. We identified static and dynamic restrictions, which may prohibit the definition of certain correspondences, and motivated these restrictions by looking at the extensions of corresponding concepts. The integration algorithm can be extended by using heuristics for generating suggestions for correspondences. Thereby, a high degree of automation of the integration process could be achieved. The combination of the two approaches – the calculation of suggestions for correspondences using heuristics on the one hand, and the restriction of possible correspondences on the other hand – would probably enable the tool, to make reliable estimations about the correct correspondences between ontology elements. So far, our focus lied on the support for interactive ontology integration. Nevertheless, it would be a promising approach, to combine our concepts with approaches for automatic ontology integration.

References

1. Kraft, B.: Semantische Unterstützung des konzeptuellen Gebäudeentwurfs. Dissertation, RWTH Aachen University, Aachen (2007)
2. Gómez-Pérez, A., Fernández-López, M., Corcho, O.: Ontological Engineering: With Examples from the Areas of Knowledge Management, e-Commerce and the Semantic Web. Springer (2004)
3. Guarino, N.: Formal Ontology and Information Systems. In Guarino, N., ed.: Proc. of the 1st Intl. Conf. on Formal Ontology in Information Systems (FOIS'98), Amsterdam, IOS Press (June 1998) 3–15

4. Pinto, H.S., Gómez-Pérez, A., Martins, J.P.: Some Issues on Ontology Integration. In Benjamins, V.R., Chandrasekaran, B., Gómez-Pérez, A., Guarino, N., Uschold, M., eds.: Proc. of the IJCAI–99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends (KRR5). Volume 18 of CEUR Workshop Proceedings., Aachen, Departement of Computer Science 5, RWTH Aachen (1999) 7/1–7/12
5. Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., Hübner, S.: Ontology-Based Integration of Information – A Survey of Existing Approaches. [16] 108–117
6. Kalfoglou, Y., Schorlemmer, M.: Ontology Mapping: The State of the Art. The Knowledge Engineering Review **18**(1) (2003) 1–31
7. Euzenat, J.: State of the Art on Ontology Alignment. Technical Report D2.2.3, Knowledge Web Consortium (2004)
8. McGuiness, D.L., Fikes, R., Rice, J., Wilder, S.: An Environment for Merging and Testing Large Ontologies. In Giunchiglia, F., Selman, B., eds.: Proc. of the 17th Intl. Conf. on Principles of Knowledge Representation and Reasoning (KR-2000), San Francisco, USA, Morgan Kaufmann Publishers (April 2000) 483–493
9. Noy, N.F., Musen, M.A.: PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In Engelmore, R., Hirsh, H., eds.: Proc. of the 12th Conf. on Innovative Applications of Artificial Intelligence (IAAI-2000), Orlando, Florida, AAAI Press (2000) 450–455
10. Noy, N.F., Musen, M.A.: Anchor-PROMPT: Using Non-Local Context for Semantic Matching. [16] 63–70
11. Hakimpour, F., Geppert, A.: Resolving Semantic Heterogeneity in Schema Integration: an Ontology Based Approach. In: Proc. of the 2nd Intl. Conf. on Formal Ontology in Information Systems. Volume 2001., New York, USA, ACM Press (2001) 297–308
12. Klein, M.: Combining and Relating Ontologies: An Analysis of Problems and Solutions. [16] 53–62
13. Kiesel, N., Schürr, A., Westfechtel, B.: GRAS, A Graph-Oriented Database System for (Software) Engineering Applications. Information Systems **20**(1) (1995) 21–51
14. Schürr, A., Winter, A., Zündorf, A.: The PROGRES approach: Language and environment. In Ehrig, H., Engels, G., Kreowski, H.J., Rozenberg, G., eds.: Handbook on Graph Grammars and Computing by Graph Transformation: Applications, Languages, and Tools. Volume 2., World Scientific (1997) 487–550
15. Böhlen, B., Jäger, D., Schleicher, A., Westfechtel, B.: UPGRADE: Building Interactive Tools for Visual Languages. In Callaos, N., Zheng, B., Kaderali, F., eds.: Proc. of the 6th World Multiconference on Systemics, Cybernetics, and Informatics (SCI 2002), New York, TPA Publishing (2002) 17–22
16. Gómez-Pérez, A., Gruninger, M., Stuckenschmidt, H., Uschold, M., eds.: Proc. of the IJCAI–01 Workshop on Ontologies and Information Sharing. In Gómez-Pérez, A., Gruninger, M., Stuckenschmidt, H., Uschold, M., eds.: Proc. of the IJCAI–01 Workshop on Ontologies and Information Sharing, Orlando, Florida, AAAI Press (2001)