A Tool to Support Ontology Creation Based on Incremental Mini-ontology Merging

A Thesis Proposal
Presented to the
Department of Computer Science
Brigham Young University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

by
Zonghui Lian
July 2006

Abstract. This thesis will address the problem of tool support for semi-automatic ontology mapping and merging. Solving this problem can contribute to ontology creation and evolution by relieving users from tedious and time-consuming work. As proposed in the NSF-supported TANGO project [16], this work will show that a tool can be built that will take a light weight ontology and a growing ontology as input and make it possible to produce manually, semi-automatically, or automatically an extended growing ontology as output. Characteristics of this tool include: (1) a graphical, interactive user interface with features that will allow users to map and merge ontologies, and (2) a framework supporting pluggable, semi-automatic and automatic mapping and merging algorithms.

1 Introduction

Ontologies provide a powerful explanatory mechanism for concepts and their relationships. The creation of ontologies however, is expensive and complex. In most cases, information collection and analysis, concept and relationship design, and iterative construction are tedious and time-consuming work.

Much effort has been expended to alleviate these difficulties. Currently over 90 ontology editor tools are available [9]. Although focused on different purposes and based on different ontology languages, generally, we can classify these ontology development tools into three categories according to their dominant creation processes: manual, semi-automatic, and automatic.

Manual ontology generation tools, such as [1], [3], [4], [6], [10], and [13], provide a way for users to create ontologies without using ontology languages directly. These tools are relatively simple. They provide no information extraction and analysis functions, and most of them provide no support for ontology merging. Although these tools provide some conveniences they still leave most of the information collection and analysis work to be done manually.

Some tools are semi-automatic and provide automatic support for some parts of ontology creation [14]. For example, based on machine learning and linguistic processing, Text2Onto [8] can extract concepts and relations from textual data. A limitation of this tool is the creation of a good corpus for machine learning and natured language processing techniques. If users want to use this tool to create an ontology in a certain domain, they have to have enough data in the corpus first, which in most cases, is hard to obtain. Also, the resulting ontologies are often not reliable, and in some case, the accuracies of the relationships among extracted data items are quite low. Another example of a semi-automatic ontology creation method is presented in [11]. It allows semi-automatic knowledge extraction from underlying classification schemas such as folder structures or web directories. The performance of this method depends highly on the previously generated ontologies. In addition the accuracy of distinguishing concept and data instances is an issue for this method.

There are also some available tools that provide for automatic ontology generation. However, they are all highly constrained. TOPKAT [5] just supplies a simple natural language parser that only provides partial natural language analysis to identify possible concepts and property values. The tool presented in [15] is able to extract ontologies only if the scope of the domain is very narrow. Although the authors claim that this tool is automatic, an ontology evolution session is highly user-interactive.

Researchers at BYU have proposed TANGO (Table ANalysis for Generating Ontologies) that can generate ontologies fully automatically, but can also be run semi-automatically, or manually. TANGO [16] is an ontology creation tool that assembles information provided in ordinary tables into ontologies. The aim of TANGO is to create ontologies while involving the user as little as possible. TANGO's working process includes four steps: (1) recognize and normalize table information; (2) construct mini-ontologies[1] from normalized tables; (3) discover inter-ontology mapping; and (4) merge mini-ontologies into a growing application ontology[2].

My research is part of the TANGO project. It mainly focuses on Step 3 and Step 4. Specifically, my project is to construct a tool to do the ontology mapping and merging. The input for a session using this tool is a mini-ontology and a growing ontology, the output is the growing ontology augmented by the mini-ontology. The

[1]A *mini-ontology* is a light weight ontology generated from a table by our table interpretation techniques.

[2]A *growing ontology* is a domain ontology that is being constructed as a result of merging mini-ontologies into the growing ontolog

tool should allow the ontology mapping and merging processes to be automated. It should also guide users by notifying them when interventions may be necessary and by suggesting possible resolutions to issues that may arise.

2 Thesis Statement

A tool can be built that will take a mini-onotology and a growing ontology as input and make it possible to produce manually, semi-automatically, and automatically an extended growing ontology as output. Characteristics of this tool include: (1) a graphical, interactive user interface with features that will allow users to map and merge ontologies, and (2) a framework supporting pluggable, semi-automatic and automatic mapping and merging algorithms.

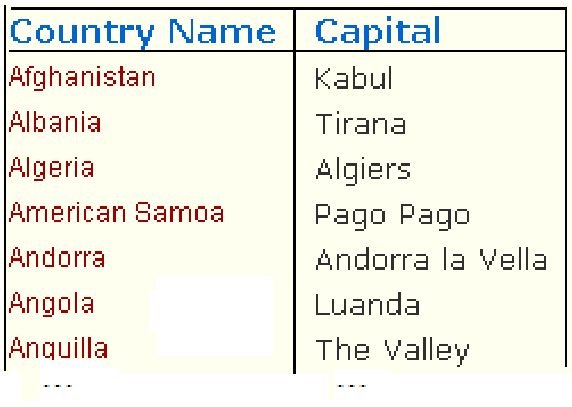
3 Project Description

To achieve the goals mentioned in the thesis statement, we plan to conduct the research as follows: (1) provide users with a tool to do ontology mapping and merging; (2) construct a framework that provides APIs that will allow automated mapping and merging algorithms to access information necessary for their use; and (3) test the success of the tool for both manual mode and semi-automatic mode of operations. The proposal elaborates on these three aspects of the research in the three subsections that follow.

3.1 A Tool for Ontology Mapping and Merging

Currently the data extraction research group at BYU already has a ontology tool, called the OntologyEditor [17], which provides an environment for ontology design and ontology maintenance. The OntologyEditor, however does not provide ontology mapping and merging functions. Our mapping and merging tool for TANGO will be built on top of the OntologyEditor.

The tool's basic functions must allow users to (1) specify mappings between related object sets, related relationship sets, and related taxonomies, and (2) merge the matched components, include the components with no match, and clean up the resulting growing ontology if necessary. Along the way issues may arise in the ontology integration process. We use Issue/Default/Suggestion (IDS) triples [7] to resolve these issues.



Country Name	Capital
Afghanistan	Kabul
Albania	Tirana
Algeria	Algiers
American Samoa	Pago Pago
Andorra	Andorra la Vella
Angola	Luanda
Anguilla	The Valley
...	...

Figure 1: A Table

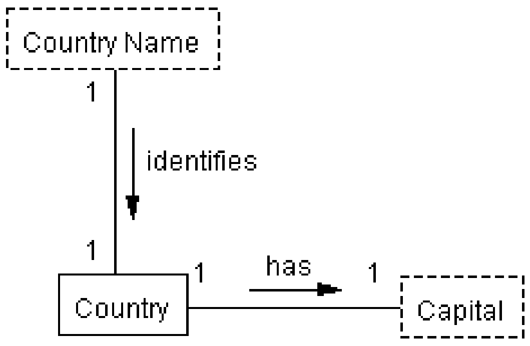


Figure 2: A Mini-ontology.

To make our ideas clear, we give an example. Figure 1 presents a table from countryreport.org [2]. Based on this table, the mini-ontology in Figure 2 can be generated. Assume a user already has the growing ontology in Figure 3. The mini-ontology consists of three object sets: *Country, Capital*, and *Country Name*; and two relationship sets: *Country Name identifies Country* and *Country has Capital.* For the example, we assume that the user already has the growing ontology in Figure 3. The growing ontology consists of four object sets: *Country, Name, City* and *Capital*;

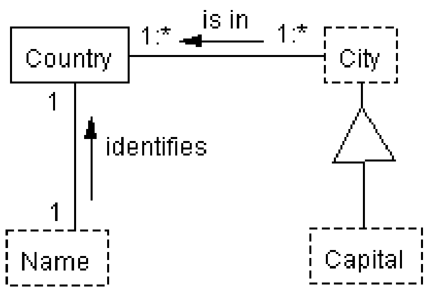


Figure 3: A Growing Ontology.

two relationship sets: *Name identifies Country* and *City is in Country*; and one generalization/specialization specification: *Capital isa City*.

Analyzing these two ontologies in Figures 2 and 3, the user can easily find several matches. The user specifies these matches by selecting the mapping component from the tool bar and dragging the mouse cursor from one to the other. A dash-dot line then appears (and may be bent to improve the appearance by adding anchor points). Figure 4 shows the result.

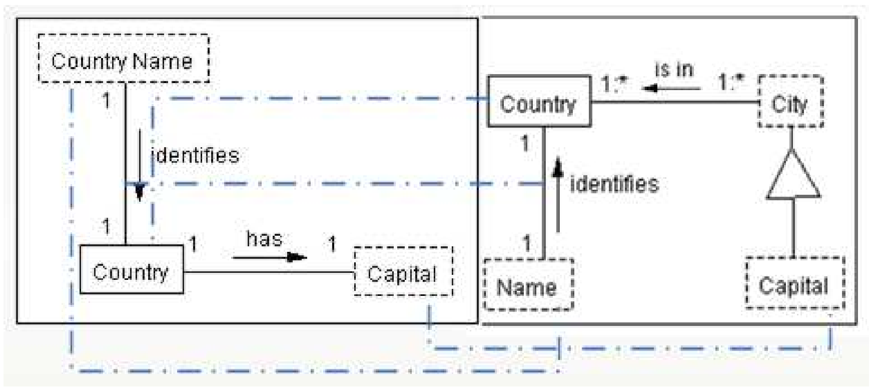
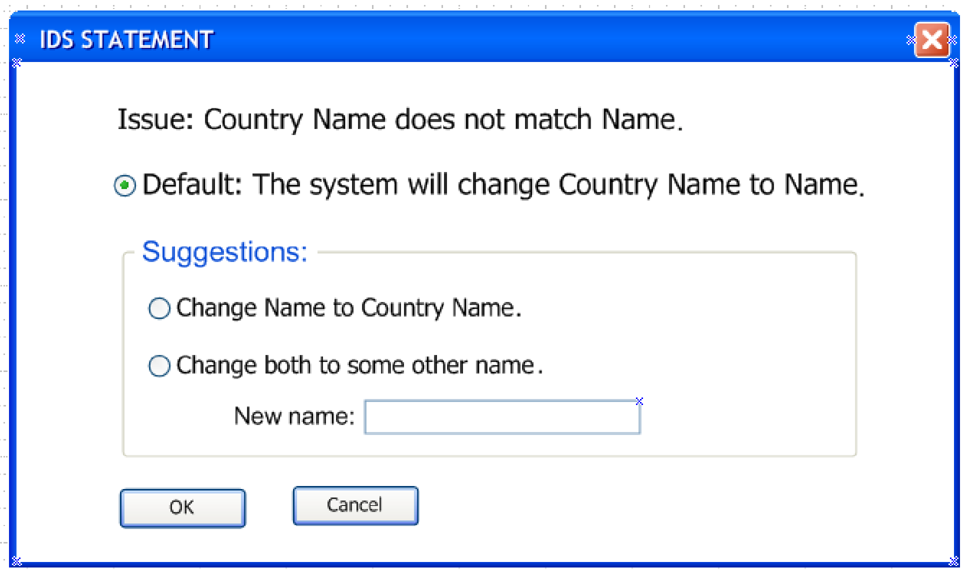


Figure 4: Mappings between the Mini-ontology and the Growing ontology.

The next step is to merge the mini-ontology into the growing ontology. If there are no conflicts, the system simply coalesces the matching components and copies the non-matching mini-ontology components into the growing ontology. If there are conflicts, the system interacts with the users to resolve them.

To make the resolution process consistent, the interaction works based on IDS statements. An IDS statement raises an issue (I), specifies a default action (D) that it will carry out if the user does not intervene, and suggests (S) one or more alternative resolutions the user may take. In our example, when the user activates the merge, the system detects a conflict and displays the IDS statement in Figure 5. Radio buttons allow users to choose a resolution, and the text box allows a user to provide information to the system—the new name in this example. For the example, assume the user chooses the default. Figure 6 presents the merged diagram.


IDS STATEMENT

Issue: Country Name does not match Name.

[✓ Default: The system will change Country Name to Name.]

Suggestions:

Change Name to Country Name.

Change both to some other name.

New name:

Figure 5: An IDS Statement Dialog Window.

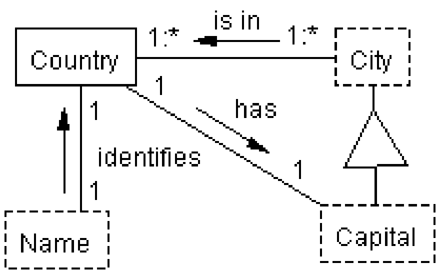
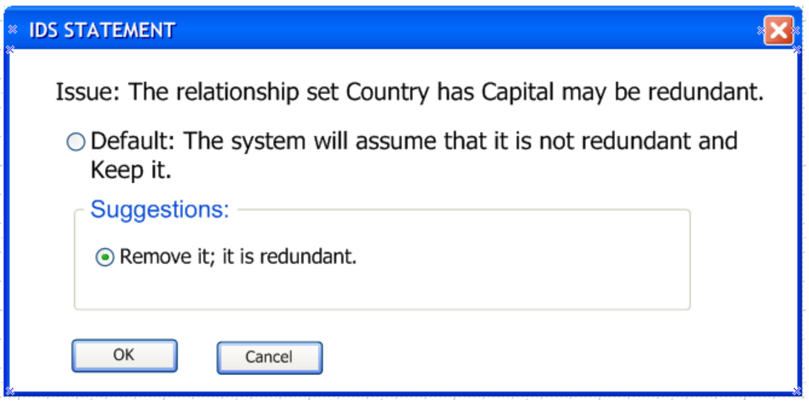


Figure 6: A Merged Growing Ontology.

We note here that as part of the mapping process the user has full control to edit either the mini-ontology or the growing ontology at any time. Typically mini-ontologies would not match perfectly, and the user would need to alter the mini-ontology before declaring the correspondence.

After doing the basic merge, the user may wish to clean up the resulting diagram. The user may remove unwanted relationship sets and object sets, alter constraints, or adjust the layout. Sometimes users have different opinions according their understanding. In our example, based on the relationship sets *City is in Country* and the generalization/specialization *Capital isa City*, it is possible to deduce the relationship set *Country has Capital*. Thus the relationship set *Country has Capital* may be not necessary. The user, however, may want to keep this relationship set, because it retains the one-to-one constraint between a country and its capital.


IDS STATEMENT

Issue: The relationship set Country has Capital may be redundant.

Default: The system will assume that it is not redundant and Keep it.

Suggestions:

[✓ Remove it; it is redundant.]

Figure 7: An IDS Statement Dialog Window.

Assuming the existence of a plug-in module that checks for redundancy [12]. Our tool will offer a way for the plug-in module to issue and respond to IDS statements. For our example the IDS statement in Figure 7 would be presented to the user. We assume the user chooses the suggestion and does the deletion, the resulting growing ontology is in Figure 9.

In addition, as part of the tool, users may prefer to work with the textual version of OSM rather than the graphical version. OSM-L is a textual language, which is fully equivalent to the graphical version [12]. As an example, Figure 8 shows the textual equivalent of Figure 4.

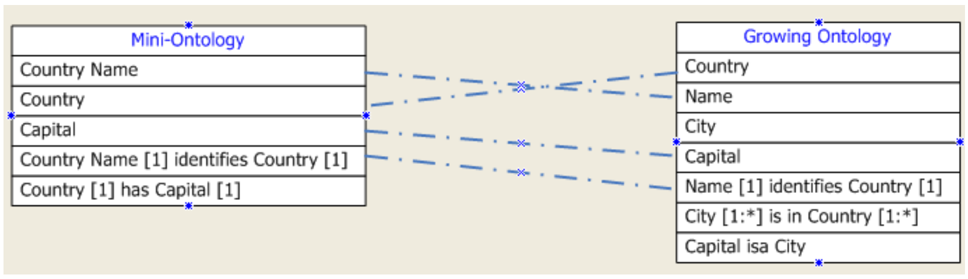

Mini-Ontology
Country Name
Country
Capital
Country Name [1] identifies Country [1]
Country [1] has Capital [1]
Growing Ontology
Country
Name
City
Capital
Name [1] identifies Country [1]
City [1:*] is in Country [1:*]
Capital isa City

Figure 8: A OSM-L Language based Mapping.

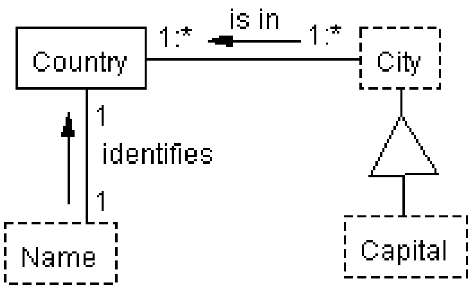


Figure 9: A New Growing Ontology

Finally, our tool allows and supports instance data integration. Issues include data canonicalization, object identity, value split and merge, and constraint resolution. Our tool will provide users with a set of capabilities to resolve these issues. These capabilities include the ability to manually integrate data, an API for facilitating automatic data integration, and a constraint conflict detector.

3.2 APIs for Automated Mapping and Merging Algorithms

From the above examples we see that there are two major steps: (1) edit the mini-ontology and growing ontology and provide mapping information between these two ontologies and (2) merge the ontologies according to the mappings and clean up the merged ontology. As explained in Section 3.1, the user can manually do these two steps. We intend, however, for these two steps to be automated to the extent possible. To provide for automation of these steps, our tool will allow various mapping and merging algorithms to replace manual operations as Figure 10 shows.

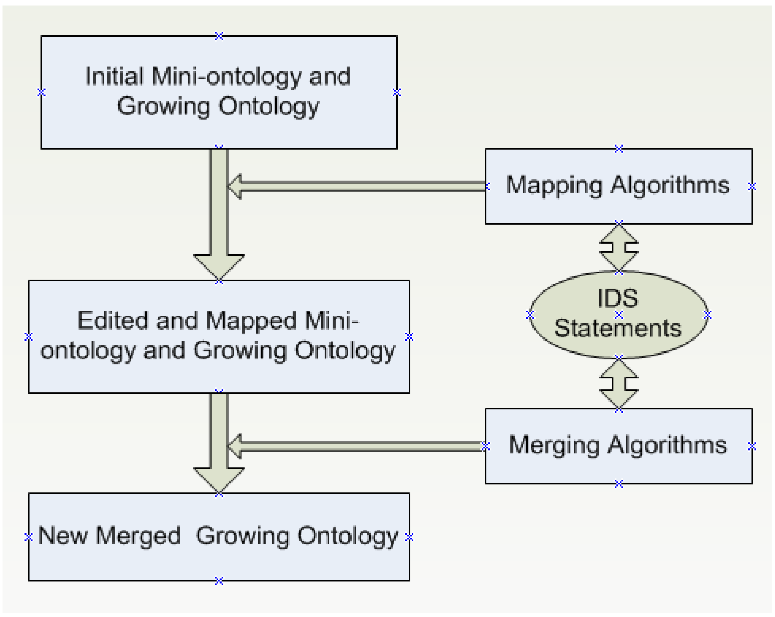

Initial Mini-ontology and Growing Ontology
Mapping Algorithms
IDS Statements
Edited and Mapped Mini-ontology and Growing Ontology
Merging Algorithms
New Merged Growing Ontology

Figure 10: A Work Flow for Ontology Mapping and Merging Process with a set of Mapping and Merging Algorithms

To incorporate these algorithms into our tool, we need to provide an application program interface (API). It provides access to various types of data as possible input information for the algorithms, and they standardize outputs.

Basically, the API must provide three major types of data as input information for the algorithms. First, it should offer all the information it has about the components in both the mini-ontology and growing ontology. This includes information about the object sets, relationship sets, generalization/specialization hierarchies, aggregation hierarchies, and all constraints over and among these components. Second, it should offer the instance values for all object and relationship sets in both the mini-ontology and the growing ontology. Third, once obtained, it should offer the mapping information that it receives either from mapping algorithms or user specifications. The mapping information is an injective function from mini-ontology constructs to growing ontology constructs with the restriction that the type of constructs must be the same.

The API must also provide three types of output. It must let algorithms (1) update mini-ontologies; (2) update growing ontologies; and (3) specify mappings between mini-ontologies and growing ontologies.

Finally, the API must also provide an interface to standardize IDS statements that may arise from mapping or merging algorithms. An algorithm must provide an I-statement string, a D-statement string, and a list of S-statement strings. Further, for any input it expects the algorithm must provide a list of S-statement input pairs. Each S-statement input pair (x, y) consists of an S-statement number x, so that it can be associated with an S-statement in the list of S-statements, and a string y that labels the input text box to be filled in with a value. In Figure 5, the I-statement is “Country Name does not match Name.”; the D-statement is “The system will change Country Name to Name.”; and the list of S-statement strings is “Change Name to Country Name.” followed by “Change both to some other name”. There is only one input text field; its (x, y) pair is “(2,‘New name’)”.

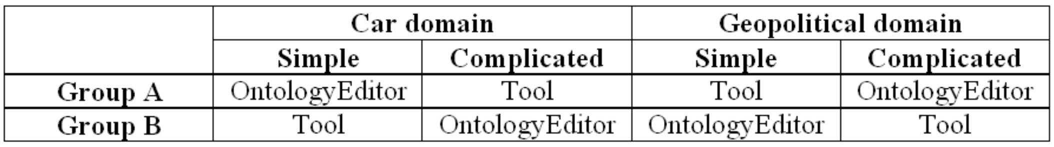
Based on the API, different plug-in algorithms can obtain and exchange enough information to enable them to do their job. Thus once the user also can modify the mapping and merging results through our tool, the tool can be run automatically, semi-automatically, or manually.

3.3 Validation

The evaluation of this tool is consist of two parts. The first part is to test the usability of the tool. The second part is to test whether the tool is extendable with different mapping and merging algorithms.

For the validation of the first part, we mainly focus on whether this tool can used for the experiments planned for the TANGO project. Although we cannot have groups of users build ontologies from sets of input tables with and without the full suite of TANGO tools, we can simulate part of the experiment.

Figure 11 presents our testing plan. We plan to have two groups of users (Group A and Group B) who are familiar with both the ontology editor and our mapping and merging tool. We will assign each group the 4 tasks shown in Figure 11. For each task



	Car domain		Geopolitical domain	
	Simple	**Complicated**	**Simple**	**Complicated**
Group A	OntologyEditor	Tool	Tool	OntologyEditor
Group B	Tool	OntologyEditor	OntologyEditor	Tool

Figure 11: The Testing Plan

we will provide a certain number of mini-ontologies which are from the car domain or from the geopolitical domain. We classify tasks as "simple" and "complicated". A simple task contains three mini-ontologies and a complicated task, relatively larger, contains seven mini-ontologies. Our tool and the OntologyEditor will track all user interactions and model changes, time-stamp them, and record them in log files. Finally, we will evaluate the correctness of the resulting growing ontologies. To summarize the results, we will compute a work efficiency measure, dividing the work accomplished by the time to complete the work.

For the second part of the evaluation, we will implement a naive mapping algorithm and a naive merging algorithm, and also sophisticated algorithms from among those described in [7] and [18]. By doing this, we can gain confidence that the API can support different mapping and merging algorithms. We will also ensure that the API is well documented and complete. To ensure these desirable properties, we will provide an API specification document, and we will provide methods to expose all properties of every object set, every relationship set, every constraint, and all stored data.

4 Thesis Schedule

Literature Search and Reading: January 2006 – July 2006

Design and Coding: September 2006 – November 2006

Experiments: November 2006 – December 2006

Write Thesis: December 2006 – March 2007

Thesis Revision & Defense April 2007

References

[1] www.w3.org/2001/11/IsaViz/, 2004.

> This tool provides Graph Stylesheets (GSS) used to define schemas for rendering languages based on RDF. This is a reference for a manual ontology generation tool.

[2] COUNTRYREPORTS: A table about the the countries and their capitals. www.countryreports.org, 2006.

> This is the source for the mini-onotology in Figure 2.

[3] ezOWL. iweb.etri.re.kr/ezowl/, 2006.

> ezOWL is a visual ontology editor tool that can construct an OWL ontology. It provides friendly diagram-based visualizing and authoring techniques. This is a reference for a manual ontology generation tool.

[4] GKB Editor. www.ai.sri.com/ gkb/, 2006.

> The GKB-Editor (Generic Knowledge Base Editor) is a tool for graphically browsing and editing knowledge bases across multiple Frame Representation Systems (FRSs) in a uniform manner. It offers an intuitive user interface, in which objects and data items are represented as nodes in a graph, with the relationships between them forming the edges. Users edit a KB through direct pictorial manipulation, using a mouse or pen. This is a reference for a manual ontology generation tool.

[5] TOPKAT - The Open Practical Knowledge Acquisition Toolkit. www.aiai.ed.ac.uk/ jkk/topkat.html, 2006.

TOPKAT (The Open Practical Knowledge Acquisition Toolkit) is a hypertext and diagram-based toolkit, developed at the Artificial Intelligence Applications Institute (AIAI), which integrates knowledge elicitation techniques with the CommonKADS approach to knowledge modeling. This tool provides a simple natural language parser that can identify possible concepts and property values in a protocol transcript. It is an example of an automatic ontology generation tool.

[6] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. OilEd: A reasonable ontology editor for the semantic web. *Lecture Notes in Computer Science*, 2174:396–405, 2001.

The authors present OilEd, a ontology editor that provides a frame-based user interface. This tool can use reasoning to check class consistency and infer subsumption relationships. This is a reference for manual ontology generation tool.

[7] J. Biskup and D.W. Embley. Extracting information from heterogeneous information sources using ontologically specified target views. *Information Systems*, 28(3):169–212(44), May 2003.

In this paper, the authors propose a framework that models both the target and all the sources in the same modeling languanges. The authors also presents an Issue/Default/Suggestion (IDS) triples to solve the conflict raised in ontology integration process. They also summarize the issues and provide default and suggested solutions for model integration. This is a reference for our issue resolution procedure using IDS statements.

[8] P. Cimiano and J. Völker. Text2Onto - a framework for ontology learning and data-driven change discovery. In *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB)*, pages 227–238, Alicante, Spain, June 2005.

In this paper, the authors presents Text2Onto, a framework for ontology learning from textual resources. Text2Onto combines machine learning approaches with basic linguistic processing such as tokenization or lemmatizing and shallow parsing. It can be used to detect concepts, and various of relations such as subclass-of relations and instance-of relations. This is an example of a semi-automatic ontology generation tool.

[9] M. Denny. Ontology tools survey, revisited. www.xml.com/pub/a/2004/07/14/onto.html, 2004.

This reference presents an ontology tool survey, which provides categorical descriptions of 94 ontology editors.

[10] J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubzy, H. Eriksson, N. F. Noy, and S. W. Tu. The evolution of Protege: An environment for knowledge-based systems development. *International Journal of Human Computer Studies*, 58(1):89–123, 2003.

This paper gives the history of the knowledge-based system called Protege. The authors present the functions and features for each generation of Protege. This is a reference for manual ontology generation tool.

[11] S. Lamparter, M. Ehrig, and C. Tempich. Knowledge extraction from classification schemas. In *Proceedings of CoopIS/DOA/ODBASE*, pages 618–636, Agia Napa, Cyprus, 2004.

This paper presents a method which allows semi-automatic knowledge extraction from underlying classification schemas such as folder structures or web directories. The authors introduce the five main steps and finally evaluated these processes by using a prototypical implementation and a set of real world folder structures. This is an example of a semi-automatic ontology generation tool.

[12] S. W. Liddle, D. W. Embley, and S. N. Woodfield. *An active, object-oriented model-equivalent programming language*, pages 333–361. MIT Press, 2000.

> The authors explain the difficulties encounted in making models and languages equilvlentin databases, and they intruduce a particular language and model, OSM, which can resolve these difficulties.

[13] T. Liebig and O. Noppens. OntoTrack: Fast browsing and easy editing of large ontologies. In *Proceedings of The Second International Workshop on Evaluation of Ontology-based Tools (EON2003)*, Sanibel Island, Florida, October 2003.

> The authors present an ontology authoring tool, OntoTrack, which combines a sophisticated graphical layout with mouse enabled editing features optimized for efficient navigation and manipulation of ontologies. It supports functions such as ontology graphical browsing, editing, and searching. This is a reference for manual ontology generation tool.

[14] A. Maedche and S. Staab. Ontology learning. In *Handbook on Ontologies*, pages 173–190. 2004.

> In this paper, the authors present a notion of ontology learning. The authors suggest that the ontology learning should proceed through ontology import, extraction, pruning and refinement, giving ontology engineer a wealth of coordinated tools for ontology modeling. The authors also show some exemplary techniques in the ontology learning cycle, and they present their ontology learning tool, called KAON Text-To-Onto. This is an example of a semi-automatic ontology generation tool.

[15] G. A. Modica, A. Gal, and H. M. Jamil. The use of machine-generated ontologies in dynamic information seeking. In *Proceedings of the 9th International Conference on Cooperative Information Systems (CoopIS01)*, pages 433–448, London, England, September 2001.

In this paper, the authors present a methodology for automating the evolution of ontologies. This methodology contains two sessions: a training session which is in charge of creating ontologies, and an additional session which is in charge of ontology merging. Although the merging is automatic, the tool only works on very narrow data sources.

[16] Y. A. Tijerino, D. W. Embley, D. W. Lonsdale, Y. Ding, and G. Nagy. Toward ontology generation from tables. *World Wide Web: Internet and Web Information Systems*, 8(3):251–285, September 2004.

The authors introduce an approach to generating ontologies based on table analysis. They thus call their approach TANGO (Table ANalysis for Generating Ontologies). Based on conceptual modeling extraction techniques, TANGO attempts to (i) understand a table's structure and conceptual content; (ii) discover the constraints that hold between concepts extracted from the table; (iii) match the recognized concepts with ones from a more general specification of related concepts; and (iv) merge the resulting structure with other similar knowledge representations. TANGO is thus a proposed, formalized method of processing the format and content of tables that can serve to incrementally build a relevant reusable conceptual ontology. This is a reference for the TANGO project.

[17] A. Wessman, S.W. Liddle, and D.W. Embley. A generalized framework for an ontology-based data-extraction system. In *Proceedings of the 4th International Conference on Information Systems Technology and its Applications (ISTA05))*, pages 239–253, Palmerston North, New Zealand, 2005.

In this paper, the authors report on the implementation of an ontology-based data-extraction system. The framework allows new algorithms and ideas to be incorporated into a data extraction system without requiring wholesale rewrites of a large part of the system's source code. It allows researchers to focus their attention on parts of the system relevant to their research without having to worry about introducing incompatibilities with the remaining components. They demonstrate the value of the framework by providing an implementation that exhibits appropriate characteristics. This is a reference for the OntologyEditor.

[18] L. Xu and D.W. Embley. Automating schema mapping for data integration. *Information Systems.* To appear.

In this paper, the authors present a framework for generating a source-to-target mapping containing direct as well as many indirect matches between a source schema and a target schema. They introduce four basic techniques for schema mapping: (1) terminological relationships, (2) data-value characteristics, (3) domain-specific, regular-expression matches, and (4) stucture. This is a reference for mapping algorithms.