Methodology for Constructing Form Ontology

U. Ungkawa, D. H. Widyantoro & B. Hendradjaya
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
ungkawa@gmail.com

Abstract—**Form ontology was built to complement the knowledge base of XReformer system, a system to generate web forms design automatically with case-based reasoning (CBR) approach. Case base is used to store cases of form design while the ontology is used to define forms and its elements and the relationship between them as well as between the elements itself. The ontology acts as a small-scale knowledge base that can grow to become a big one. The existing ontology development methodologies were too complicated and mature and they were feasible to apply on a large scale ontology. Certainly, it was not efficient to build a small-scale ontology with these highly-discipline methodologies. In this paper, we propose a simple ontology development methodology but covers all important aspects of the development of ontologies, as an alternative to the existing methodologies.**

Keywords—ontology development methodology; form ontology; case-based reasoning;.

I. INTRODUCTION

Our work concerns the reuse of design using case-based reasoning (CBR) approach to generate HTML form design automatically. We named our system as XReformer. As a reasoning paradigm, XReformer needs form ontology as a complement of the case based on the system as the knowledge base. The case based is used to store cases of form designs while the form ontology is used to define forms and the relationship between forms and its elements as well as between the elements itself. In this research, we built the prototype of a small-scale form ontology which could be expanded into a large one. Certainly, it is not efficient to build a small-scale ontology with a high discipline methodology, since it would require a great effort. In this paper, we propose a simple ontology development methodology, but it still covers all important aspects of the development of ontologies.

Some important issues underlying form ontology development in XReformer system are query normalization, forms design adaptation, and generation of an HTML file. The aim of the query normalization is to standardize form and field terms that correspond to the terms used internally and for semantic matching. The purpose of the adaptation of form design is a process to compose elements, encode grouping of form elements, fields ordering and laying out. While in the generation of HTML forms, the ontology is used to define fields type and layout. Thus, the form ontology plays important roles from handling the query to the physical forms generation.

The query normalization is a part of query processing. In many studies, we know query reformulation, query transformation, query rewriting and query expansion. This work only studies terms normalization in order to match the internal terms that the system uses. Many different terms with the same meaning and purpose would have the same internal term in the system.

The adaptation of form design requires an ontology to define groups of form elements, field ordering and laying out. In fields grouping, form elements should be classified in the same or adjacent class. The elements in the same/adjacent class will have a higher probability located in the same group than elements in the different class. The second adaptation process is field ordering that uses grouping results in order to not violate the ordering semantics. The elements in a group are ordered according to the relative position defined in the ontology. The third adaptation process is field laying out to determine the actual position of the element whether below (vertically) or right (horizontally) of another element. Up to this point, the new form design was completely defined. The next step is to generate a concrete form.

The form generation transforms the new form design encoded in the internal representation into the physical form. It also takes ontology as a source of knowledge to determine the type of elements such as text fields, dropboxes, and others.

This paper proposes a new method of ontology development from scratch. The next section will discuss briefly some of the ontology development methodologies. Then the following section discusses the analysis of some of the existing methodologies, and then proceed with the proposed methodology and the form ontology that the methodology produces.

II. RELATED WORKS

Since the 1990s, researchers have begun to build ontology development methodology. The ontology development is a creative activity that is naturally more an art than a science. The ontology development, as well as knowledge engineering, adopt methods in software engineering.

Some initial ontology development methods are IDEF5 [1], [2]. MENELAS [3], Plinius [4], Kactus project [5], Method of Uschold and King [6], Method in Tove project [7], PHYSSYS [8], SENSUS [9],

978-1-5386-0549-3/17/$31.00 ©2017 IEEE

Mikrokosmos [10], ONIONS [11], and METHONTOLOGY [12].

Some of the 2000 decades methodologies were On-To-Knowledge [13], the Development 101 [14], DOGMA [15], DILIGENT [16], [13], BORO [17], [18], UPON [19][20], SAMOD [21] and many more methods developed by researchers which are combination of several previous methods [22], [23]. Some methodologies are discussed briefly here.

Among the above development methods, the most influential and has laid the foundation in the next development method is a method of Uschold and King, and it is divided into four phases: first, identify purpose: detailing the reasons why the ontology needs to be built and the purpose of construction. Second, building the ontology consists of ontology capture, ontology coding, and integrating existing ontologies. Third evaluation: checking the correspondence between ontologies, software, and documentation on each reference framework. Fourth, documentation.

The other most widely referenced methodology is a methodology in the Tove project [7] which is based on first-order logic. The proposed stages are as shown in Figure 1:

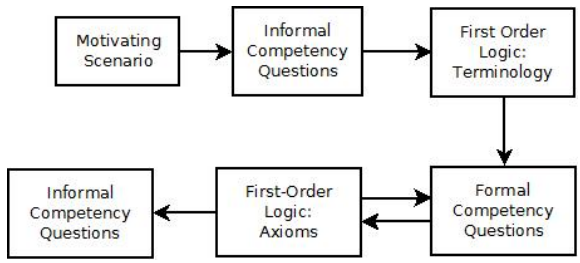

Figure 1. Ontology design and evaluation

A more mature method in the development of ontology is Methontology [12], it divides the step into three groups of tasks [24]:

- Project management activities cover planning, control, and quality assurance.
- Development-oriented activities include specification, conceptualization, formalization, implementation, and maintenance.
- Support activities include the acquisition of knowledge, evaluation, integration, documentation and configuration management.

III. ANALYSIS OF DEVELOPMENT METHODOLOGY

When for the first time the concept of ontology was involved in information technology, ones were trying to think about how to build an ontology. From that time, various methodologies built to give a way to how an ontology created. At the beginning of the development of methodology, ones just think how ontologies are made without thinking about other aspects such as maintenance and reuse of existing ontologies. Devising the methodology continues until eventually involve ontology evaluation stage. The next stage involved is reuse and configuration management [25].

However, a perfect methodology ultimately arises a problem of a complex and large effort. From this point, it happens a turning point for thinking about how to build an ontology quickly (Agile) [21]. But, the methodology is not yet escaping from the complexity of the process and is still hard to be applied, especially for novice ontology developers in a small scale ontology.

For that reason, in developing the form ontology, we develop a simple methodology, but it covers all important aspects. The important requirements of form ontology development method are:

- Incremental Iterative.
- Lightweight/agile.
- Documentation.
- Evaluation (application based).
- Maintenance
- Using ontology language OWL 2.
- Based on a complete tool: Protégé: builder, reasoner, and visualizer.

IV. FORM ONTOLOGY BUILDING METHODOLOGY

The characteristic of form ontology is a small scale ontology which bases on a wide variety of forms. The conceptualization is done manually by taking many sample forms, and then derive the relation from the form name and form fields. Next, from the collection of forms, find the semantic relationship between a form with its fields and among the form fields. Broadly speaking, the form ontology development method is as shown in Figure 2.

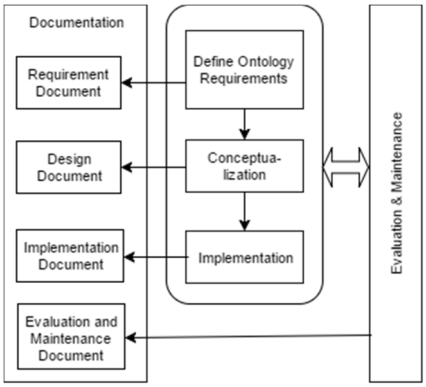

Figure 2. The Form Ontology Method

A. *Define Ontology Requirements*

In this stage, identify the user, the use of ontologies, the objectives and its scope, what should exist in the ontology and what should not [26]. In this stage also specify the functional requirements through some competency questions [7].

The form ontology users are application agents i.e. normalization, adaptation and generation modules. The ontology should include all forms derived from real HTML forms. The structure of the ontology shall be constructed in such a way in order to meet the requirements of the application. Structuring the ontology is carried out in the conceptualization stage.

In order to be able to evaluate how far the created ontology fulfills its function, it should be derived some competency question. Some competency questions are:

Questions relating to the normalization:

- Does every name/identifier of a field have a standard name (term)?

Questions relating to the composition of the adaptation:

- Which elements/fields are the most important in a form?
- Which elements/ fields are the most unlikely exist in a form?

Questions relating to the grouping in adaptation:

- Are the adjacent form elements grouped in the same class or in the same super class (adjacent class)?

Questions relating to the ordering in the adaptation:

- What elements precede the other elements?

Questions relating to the lay out in adaptation:

- What elements have a position in a form above or below another element?

Questions relating to the generation of the HTML form:

- Does each element/field have a field type?
- Does each element have an attribute?

B. Conceptualization

As mention above, in this stage, the domain knowledge is being structured. The conceptualization is similar to the design stage in software engineering. The stage includes: build a glossary of terms, classification tree of concepts and descriptions of concepts [12], [24]. In the development of form ontology, the conceptualization stems from a set of forms of various web sites and referring the above competency questions. The stage seeks to answer the above questions.

1) Classification

The structure of form ontology follows the structure of the real form, consisting of a form name and form elements. The form elements are categorized into an input, output, and control elements. Each form is given a name and is grouped in each category of forms. A category of forms becomes a class (concept) in the ontology.

The form names used as an identifier. The set of identifiers turns out to be a dictionary of terms in the ontology. Some identifiers with the same meaning should be represented by only one name, and it is taken as a standard name in the system, which is also used in the case base. The same meaning identifiers should be stated explicitly in the ontology. For example, a *reservation* form is the same as a *booking* form. If not, both forms are considered as different individuals.

The next step in the conceptualization stage is a classification of form field names. The classification also follows the form. For example, the *address* class consists of *city, street, zip code* and more. So these terms are semantically unified in a group. The application will take advantage of this class structure to determine the semantic similarity. *Street* and *city* semantically closer than *street* and *time*.

After one finished defining the name of classes (concepts), he/she should name the collected individuals (instances). The names of the individual within the same class is likely to have the same meaning. For that end, the name of the individuals that have the same meaning should be declared as the same entity. Protégé gives the option "*Individual Same As*" in the description of the individual. Then, from a set of the same names (terms), choose one to be the standard term of normalization, i.e. it used as a standard term of the system in the case base for that meaning. The selection of this term is free but must be consistent with the system. If the name is unique in the ontology, then the unique name is automatically taken as the standard term.

2) Form Relation

A relation (object property) of an ontology has a main role in answering competency questions of corresponding functional requirements of the ontology. So, relationships are built to be able to answer the questions. For example, for the query normalization, it should be a standard term for each form element. There must be a relation for defining the standard term, such as *hasNorName*. This object property is used for e.g. *emailaddress hasNorName email*, as well as *email_address hasNorName email*.

In finding the relationship between form and its elements (fields), first, it must be determined whether any element is required (mandatory) in the form or maybe the field should not be in the form. If it required, it must be defined a property that relates it to the mandatory field(s). It is necessary to answer a question like "what is the most important element in the form?", and for a question "what element is most unlikely in a form?".

The next step is to determine the relationship between a field with other fields in terms of fields grouping, ordering, and lay outing. Since the relationship of grouping determined by the adjacency of the element in the structure of the ontology, it does not need to be defined any property which states explicitly that any element is in a group with another element. This way is preferred because it does not have to define a new relationship that is redundant in the knowledge base.

3) Naming Convention

The form ontology naming convention follows the Horridge advice [27], that is *Camelback* notation rules such as *isTypeOf*. The class name begins with a capital letter without any spaces. The property name begins with lowercase and *has-* or *is-* prefixed word. Name of individual begins with a small letter but does not use *Camelback* notation. To separate the two words in the name of an individual, it uses underscore such as *first_name*.

4) Structuring Form Ontology

To simplify the breakdown of concepts (classes) and to help better understanding of the built ontology, it required drawing the basic structure of the ontology. For form domain, first, we create a *Form* concept that consists of form elements (*FormElement* concept) and a form name. The form element has attributes and types and consists of input, output and control elements. The *FormElement* concept is built to accommodate the same features i.e. types and attributes (higher level abstraction).

The form ontology must accommodate the application needs for query normalization, adaptation, and form generation. Therefore, the next step is to define properties required for those processes. Normalization defines the same term semantically so that the property for normalization must relate the same class of elements: the same input, output, and control. Similarly for the properties of grouping in the adaptation. But, the properties of ordering and lay out will connect different classes of elements.

Formally, the form ontology structure can be described in 3 tuples: *C, I, P.*

where *C*:= set of classes.
I := set of instances (individuals).
P:= set of properties.

A typical form (an instance of form) has sets of

I={fn, a, b, ..., k}
P={P1, P2, ...)
C={FC, FEC, FA, FT}

where *FC* : a class of form
FEC: a class of form elements
FA: a class of field attributes
FT: a class of field types

Generally, *P* can be stated as a relation between classes:

$$P: R \rightarrow C \times C$$

Thus, the structure of form ontology depicts the relationship between the top classes and its parts of the top classes (meronymy) as shown in Figure 3. From the figure, form constructed from five first-level classes: *Forms*, *FormNames, FieldElements, FieldAttributes,* and *FieldTypes*. The arrows indicate the object properties.

C. Implementation

We used Protégé version 5.0 [1] for building the form ontology. Protégé is an open source software for developing (Integrated Development Environment, IDE) OWL 2-based ontology (Web Ontology Language 2)[2]. OWL is a W3C (World Wide Web Consortium) recommended ontology language. The system uses Pellet[3] reasoner, for both developments that are used to examine the consistency of the ontology as well as in applications i.e. for query normalization. To depict the class hierarchy of ontology and ontology navigation, we use OWLViz [4] and OntoGraf [5] as Protégé plugin. OWLViz is used to see the class hierarchy and to navigate to each (sub) class. It can also be used to compare the class of an explicit (asserted) class and a class derived through an inference (inferred class). OntoGraf can be used to see the relationship of the ontology: class, object properties, individuals, and equivalence.

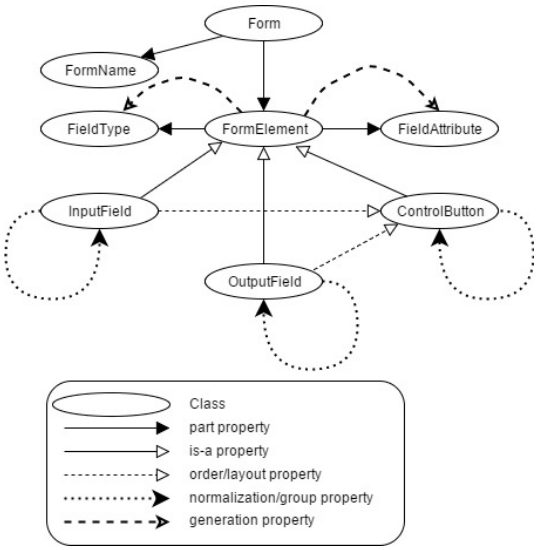

Figure 3. The Form Ontology Structure.

The structure of the first two top-level of the implemented form ontology (prototype) built with Protégé is shown in Figure 4:

D. Evaluation and Maintenance

The evaluation phase aims to predict the success of the resulted ontology [28]. In other words, an evaluation is an act of measuring the quality of the ontology [29]. Here, the evaluation is intended to ensure that it can meet the (functional) requirements of the form ontology as mentioned above, i.e. for normalization, form design adaptation and form generation. Thus, the evaluation is done through three application modules. Each module evaluation concerns one of the three main functions.

[1] http://protege.stanford.edu
[2] http://www.w3.org/2001/sw/wiki/OWL and http://www.w3.org/TR/owl2-primer/
[3] http://clarkparsia.com/pellet and http://pellet.owldl.com/
[4] http://protegewiki.stanford.edu/wiki/OWLViz. This Plugin requires GraphViz of AT&T (www.**graphviz**.org)
[5] http://protegewiki.stanford.edu/wiki/OntoGraf

Vrandecic mentions eight evaluation criteria: accuracy, adaptability, clarity, completeness, conciseness, consistency, computational efficiency and organizational fitness [29]. While Al-Debei mentions six criteria: clarity, coherence, conciseness, preciseness, completeness, and customizability [28]. In building the form ontology, the initial evaluation takes the criteria of consistency, utilizing the Pellet reasoner. Secondly, the evaluation utilizing the application or application-based evaluation [30].

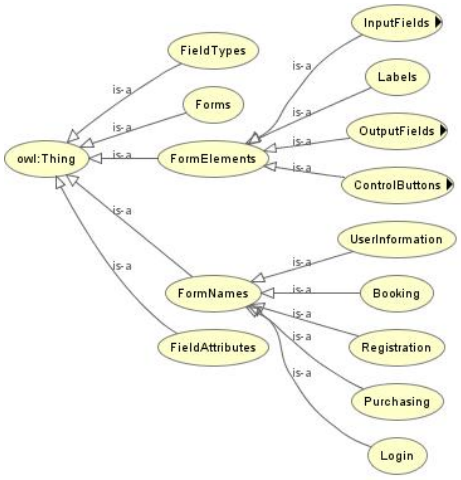

Figure 4. Implemented Form Ontology Structure.

The consistency evaluation is performed after the classification. The reasoner will show the class hierarchy (inferred hierarchy) as a result of its reasoning process [27]. The inconsistent class name is, for example, highlighted with a different color. The manual inconsistency evaluation was also conducted to ascertain whether the term used consistently in accordance with the standard terms in the system. If there any inconsistency, it is necessary to evaluate the ontology development from the first stage to the last.

The application based evaluation is done after the ontology was free from any inconsistency. The ontology is used alternately for each agent. The evaluation should ensure whether ontology was able to fulfill its function. If not, it may require to adding any property. Usually, this step is also affecting the application. To meet the functional requirement, the applications must also be able to handle the latest version of the ontology because there must be any new property to be dealt with the application.

The maintenance phase is performed hand in hand with the evaluation. Once the problem found in the evaluation, it should be carried out any revision of the application, the ontology, and its documentation. In the ontology, maybe there should be any restructuring or refinement actions [31].

E. Documentation

All activities in each phase in ontology development should be documented. The requirement specification stage produces a Requirements Ontology document containing the purpose and objective of the ontology development, and scope and language used. It also mentions end users: human or application. It also describes usefulness, functional and non-functional requirements, competency questions and a glossary of terms.

A conceptual ontology document should cover all concepts/classes, relationships/properties, and all constraints. While the implementation document depicts the real structure and the language used. Whenever any problem arises in the evaluation it should be analyzed and the solution should be written in the evaluation and maintenance document. This document contains a chronology of problems and how to find a solution that can be used in the future.

V. CONCLUSION.

In this paper, we have presented a methodology for ontology development from scratch. This method has been successfully applied in building a form ontology as a case study. This methodology is derived from a variety of methods previously by also considering the development of a software development methodology. The advantage of this methodology is its simplicity which allows someone to adopt easily, especially for novice developers. This method is not intended as a substitute for the established or more mature methods but as an alternative, as a shortcut when someone wants to make a simple ontology.

In the future, it should be considered how to evaluate this methodology, especially against other methodologies.

VI. REFERENCES

[1] C. P. Menzel and R. J. Mayer, “IDEF5 Ontology Description Capture Method : Concepts and Formal Foundations,” 1992.

[2] P. C. et al. Benjamin, “IDEF5 Method Report,” 1994.

[3] J. Bouaud, B. Bachimont, J. Charlet, and P. Zweigenbaum, “Acquisition and structuring of an ontology within conceptual graphs,” *ICCS’94 Work. Knowl. Acquis. using Concept. Graph Theory*, pp. 1–25, 1994.

[4] P. E. Van Der Vet, P.-H. Speel, and N. J. I. Mars, “The Plinius ontology of ceramic materials,” no. April, pp. 1–24, 1995.

[5] G. Schreiber, B. Wielinga, and W. Janswijer, “The KACTUS View on the ’ O ’ Word,” no. 8145, 1995.

[6] M. Uschold and M. King, “Towards a Methodology for Building Ontologies,” *Methodology*, vol. 80, no. July, pp. 275–280, 1995.

[7] M. Gruninger, M. S. Fox, and others, “Methodology for the Design and Evaluation of Ontologies,” *Proc. Work. Basic Ontol. Issues Knowl. Sharing, IJCAI*, vol. 95, pp. 1–10, 1995.

[8] P. Borst, H. Akkermans, and A. Pos, “The PhysSys ontology for physical systems,” 1995.

[9] B. Swartout, R. Patil, K. Knight, and T. Russ,

"Toward Distributed Use of Large-Scale Ontologies," *Proc. Tenth Work. Knowl. Acquis. Knowledge-Based Syst.*, pp. 138–148, 1996.

[10] K. Mahesh, "Ontology development for machine translation: Ideology and methodology," *Development*, p. 87, 1996.

[11] A. Gangemi, G. Steve, and F. Giacomelli, "ONIONS : An Ontological Methodology for Taxonomic Knowledge Integration," no. March, 1996.

[12] M. Fernández-López, A. Gómez-Pérez, and N. Juristo, "METHONTOLOGY: From Ontological Art Towards Ontological Engineering," *AAAI-97 Spring Symp. Ser.*, vol. SS-97-06, pp. 33–40, 1997.

[13] Y. Sure, S. Staab, and R. Struder, "On-To-Knowledge Methodology," *Handb. Ontol.*, pp. 117–132, 2004.

[14] N. F. Noy and D. L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology," *Stanford Knowl. Syst. Lab.*, p. 25, 2001.

[15] M. J. Meersman, "Formal Ontology Engineering in The DOGMA Approach," *Adv. Web Semant. I*, no. ODBase 02, pp. 1238–1254, 2002.

[16] H. S. Pinto, S. Staab, and C. Tempich, "DILIGENT : Towards a fine-grained methodology for DIstributed , Loosely-controlled and evolvInG Engineering of oNTologies," *16Th Eur. Conf. Artif. Intell. - Ecai*, pp. 393–397, 2004.

[17] C. Partridge, "Business Objects: Re-Engineering for Re-Use," *Drug News Perspect.*, vol. 14, 2005.

[18] M. Xie and X. Zhou, "Analyzing the usability of BORO method for semantic interoperability in the military context," 2012.

[19] A. De Nicola and M. Missikoff, "Methodology for Rapid Ontology Engineering," *Commun. ACM*, pp. 79–86, 2016.

[20] M. Banek, B. Vrdoljak, and A. Tjoa, "A proposal for a Unified Process for Ontology building: UPON," *Database Expert Syst. Appl.*, vol. 5181, no. August, pp. 65–72, 2008.

[21] S. Peroni, "SAMOD : an agile methodology for the development of ontologies," pp. 1–14, 2016.

[22] M. C. Suarez-Figueroa, A. Gómez-Pérez, and M. Fernández-López, "The NeOn Methodology for Ontology Engineering," in *Ontology Engineering in a Networked World*, 2012.

[23] B. Stadlhofer, P. Salhofer, and A. Durlacher, "An overview of ontology engineering methodologies in the context of public administration," *Seventh Int. Conf. Adv. Semant. Process.*, no. c, pp. 36–42, 2013.

[24] Fernández-Lopez, "Overview Of Methodologies For Building Ontologies," *Proc. IJCAI99 Work. Ontol. Probl. Methods Lessons Learn. Futur. Trends CEUR Publ.*, vol. 1999, no. 2, pp. 1–13, 1999.

[25] H. S. Pinto and J. P. Martins, "Ontologies: How can They be Built?," *Knowl. Inf. Syst.*, 2004.

[26] A. Öhgren and K. Sandkuhl, "Towards a Methodology for Ontology Development in Small and Medium-Sized Enterprises," *JADIS Int. Conf. Appl. Comput. 2005*, pp. 369–376, 2005.

[27] M. Horridge, "A Practical Guide To Building OWL Ontologies Using Prot´ ege 4 and CO-ODE Tools Edition 1.3," pp. 0–108, 2011.

[28] M. M. Al-debei and G. Fitzgerald, "OntoEng: A Design Method for Ontology Engineering in Information Systems," *Acm Oopsla*, no. October, pp. 1–25, 2009.

[29] D. Vrandecic, "Ontology Evaluation," Karlsruher Instituts f¨ ur Technologie, 2010.

[30] J. Brank, M. Grobelnik, and D. Mladenić, "A survey of ontology evaluation techniques," *Proc. Conf. Data Min. Data Warehouses*, pp. 166–170, 2005.

[31] D. D. Kehagias, I. Papadimitriou, J. Hois, D. Tzovaras, and J. Bateman, "A Methodological Approach for Ontology Evaluation and Refinement," pp. 1–13, 2009.