PAPER Special Section on Knowledge-Based Software Engineering

# DODDLE-OWL: Interactive Domain Ontology Development with Open Source Software in Java

**Takeshi MORITA**[†], *Nonmember*, **Naoki FUKUTA**[††], **Noriaki IZUMI**[†††], *and* **Takahira YAMAGUCHI**[†a)], *Members*

**SUMMARY** In this paper, we propose an interactive domain ontology development environment called DODDLE-OWL. DODDLE-OWL refers to existing ontologies and supports the semi-automatic construction of taxonomic and other relationships in domain ontologies from documents. Integrating several modules, DODDLE-OWL is a practical and interactive domain ontology development environment. In order to evaluate the efficiency of DODDLE-OWL, we compared DODDLE-OWL with popular manual-building method. In order to evaluate the scalability of DODDLE-OWL, we constructed a large sized ontology over 34,000 concepts in the field of rocket operation using DODDLE-OWL. Through the above evaluation, we confirmed the efficiency and the scalability of DODDLE-OWL. Currently, DODDLE-OWL is open source software in Java and has 100 and more users from 20 and more countries.

***key words:*** *domain ontology, OWL, open software*

## 1. Introduction

Ever since the necessity of ontologies has been acknowledged to share common understandings between people and software agents [1], ontologies have become very popular and significant in many application areas. As the Semantic Web is the most attractive application filed of ontologies, many ontologies has been represented by the ontology description language, OWL (Web Ontology Language) [2]. However, as well as other application areas, it still takes many costs for users to develop and maintain domain ontologies. Furthermore, we do not have good environment to support users in constructing domain ontologies in Japanese.

Regarding domain ontology development support, many studies have been done with knowledge engineering, natural language processing and data mining techniques [3], [4] to make possible automatic domain ontology construction from existing information resources, such as texts and general ontologies. However, as the techniques are not yet mature to achieve the task and domain ontology structure depends on the aspects from human experts (users), full automatic process does not go well with the task. Instead of developing full automatic environment, it is more important to provide refined semi-automatic environment with integrated facilities to construct practical domain ontologies. Furthermore, as open software is easy to evolve developed software, it is significant to build up interactive domain ontology development environment with open software.

From the above consideration, in order to build up good interactive environment for domain ontology development support, we should focus on the following four keywords: standard (OWL), extension (to Japanese), open and scalability and so propose an interactive domain ontology development environment called DODDLE-OWL (a Domain Ontology rapiD DeveLopment Environment -OWL extension). The architecture of DODDLE-OWL is redesigned and extended with the four keywords, based on DODDLE-II [5] that is not open software, not related with the Semantic Web, does not have integrated facilities, and has been evaluated just with small size of case studies.

We have developed DODDLE-OWL as open software in Java, integrating several extended facilities, such as OWL exporting facility and EDR [6] reference facility to construct domain ontologies in Japanese. DODDLE-OWL has the following six modules: Ontology Selection Module, Input Module, Construction Module, Refinement Module, Visualization Module, and Translation Module. DODDLE-OWL refers existing ontologies such as WordNet and EDR as general ontologies to construct taxonomic relationships (defined as classes) and other relationships (defined as properties and their domains and ranges) for concepts. Especially, to realize the user-centered environment, DODDLE-OWL is mounted with user interactive functions in each module.

In order to evaluate the efficiency of DODDLE-OWL, we compared DODDLE-OWL with popular manual-building method. In order to evaluate the scalability of DODDLE-OWL, we constructed a large sized ontology over 34,000 concepts in the field of rocket operation using DODDLE-OWL.

Currently, DODDLE-OWL is open source software in Java and has 100 and more users from 20 and more countries.

This paper is structured as follows. In Sect. 2, we first present the design of DODDLE-OWL. In Sect. 3, we show the implementation of DODDLE-OWL. In Sect. 4, we present evaluation of DODDLE-OWL. In Sect. 5, we report related works. Finally in Sect. 6, we conclude this paper and point out future work.

Manuscript received July 2, 2007.
Manuscript revised October 17, 2007.

[†]The authors are with Keio University, Yokohama-shi, 223–8522 Japan.
[††]The author is with Shizuoka University, Hamamatsu-shi, 432–8011 Japan.
[†††]The author is with National Institute of Advanced Industrial Science and Technology, Chiyoda-ku, Tokyo, 101–0021 Japan.
a) E-mail: yamaguti@ae.keio.ac.jp

DOI: 10.1093/ietisy/e91–d.4.945

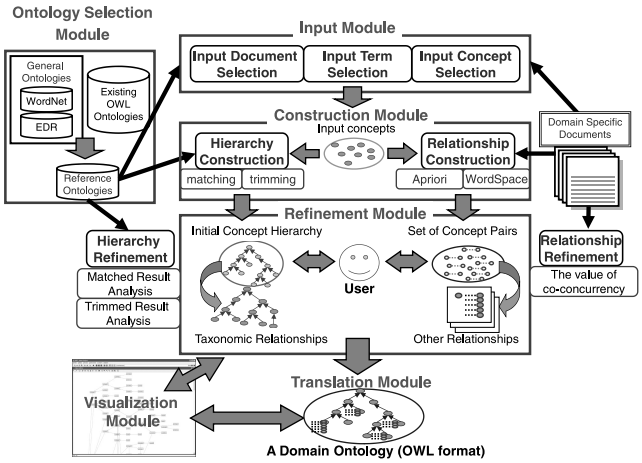


**Fig. 1** Overview of DODDLE-OWL.

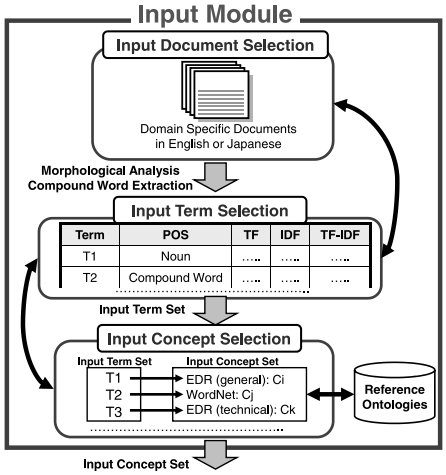


**Fig. 2** Flow of input module.

## 2. Design

### 2.1 Overview

Figure 1 shows an overview of DODDLE-OWL. DODDLE-OWL has the following six main modules: Ontology Selection Module, Input Module, Construction Module, Refinement Module, Visualization Module, and Translation Module.

Hierarchy Construction Module and Hierarchy Refinement Module were included in DODDLE-I [7] to support the user construct taxonomic relationships. Relationship Construction Module and Relationship Refinement Module, both added on to DODDLE-II, support the construction of taxonomic and other relationships. Ontology Selection Module, Input Module, Visualization Module, and Translation Module were additionally integrated in DODDLE-OWL to make possible an interactive domain ontology development environment.

In the rest of this section, we describe the six main modules in DODDLE-OWL.

### 2.2 Ontology Selection Module

In the Ontology Selection Module, a user selects reference ontologies. The reference ontologies are used in the other modules in DODDLE-OWL. WordNet [8] and EDR [6], which are general ontologies in English and Japanese, can be used as reference ontologies in DODDLE-OWL. Furthermore, DODDLE-OWL can use existing ontologies, which are described in OWL, as reference ontologies. It is considered that if the ontologies for a target domain exist on the web and can be reused, the cost of refining semi-automatically generated ontologies will be reduced. The ontologies constructed by DODDLE-OWL are described in OWL. Therefore, these ontologies can be reused as reference ontologies in DODDLE-OWL.

### 2.3 Input Module

Figure 2 shows the flow of the Input Module. In the Input Module, a user selects input concepts which are significant concepts in a domain. Input Module consists of the following three sub-modules: Input Document Selection Module, Input Term Selection Module, and Input Concept Selection Module.

#### 2.3.1 Input Document Selection Module

First, in the Input Document Selection Module, the user selects domain specific documents described in English or Japanese. At this step, the user can select part of speech (POS) for extraction of words from the documents. Input Document Selection Module automatically distinguishes one sentence from another referring to the period punctuation. However, when the input document consists of sentences with no period punctuation marks, Input Document Selection Module cannot distinguish where to punctuate the sentence. These input documents cause the decrease in the accuracy of other relationships constructed by using association rule learner in the Relationship Construction Module. Considering such a case, the user can edit manually the punctuation of one sentence in the documents using Input Document Selection Module.

#### 2.3.2 Input Term Selection Module

Second, the Input Term Selection Module shows a list of extracted terms including compound words, POS, Term Frequency (TF), Inverse Document Frequency (IDF), and TF-IDF in the documents. Domain specific documents contain many significant compound words. Therefore, accurate extraction of compound words is necessary to construct domain ontologies. At this step, while considering POS, TF, and so on, the user selects input terms which are significant terms for the domain. For certain domains, important terms

do not occur in the documents. In such a case, Input Term Selection Module has a function allowing the manual addition of important terms as input terms by the user. In order to prevent the leakage of the selection of input terms from the documents, Input Term Selection Module maintains the relationships between the extracted terms and the terms in the documents.

#### 2.3.3 Input Concept Selection Module

Finally, in the Input Concept Selection Module, the user identifies the word sense of input terms to map those terms to the concepts in the reference ontologies selected with the Ontology Selection Module. A particular single term may have many word senses. Therefore, there may be many concepts that correspond to the word. Input Concept Selection Module shows the input terms and the concepts that correspond to the input terms. While considering the domain, the user selects the most appropriate concept for the term from the list of concepts. In order to decrease the cost for input concepts selection, Input Concept Selection Module has a function enabling automatic word disambiguation (input concept selection). This function shows the list of concepts, which is ordered by some criteria, corresponding to the selected input term.

Input Concept Selection Module uses `perfectly matching` and `partially matching` to disambiguate input terms. Though, labels of most concepts do not contain compound words. Therefore, it is difficult to select the appropriate concept for compound words. To deal with this, `partially matching` is used to disambiguate most of the compound words of the input terms. `Perfectly matching` and `partially matching` means an input term perfectly or partially corresponds to labels of a concept. The priority of `perfectly matching` is higher than that of `partially matching`. If an input term does not correspond perfectly to any labels of concepts in the reference ontologies, Input Concept Selection Module analyzes the morphemes of the input term. The input term can be considered to be a list of the morphemes. Input Concept Selection Module tries to correspond the sub lists (example shown below) to the concepts of the reference ontologies. Of the matched concepts corresponding to the sub lists, the longest concept is selected as the concept of the input term, and the input term becomes the sub concept of the concept.

For example, the input term "rocket delivery system" does not perfectly correspond to the labels of concepts in the reference ontologies. Input Concept Selection Module analyzes morphemes of "rocket delivery system". "Rocket delivery system" is resolved to "rocket", "delivery", and "system". The sub lists for this input term becomes "delivery system" and "system". First, Input Concept Selection Module disambiguates "delivery system." Then, Input Concept Selection Module disambiguates "system." In this example, "delivery system" does not correspond to the labels of concepts in the reference ontologies. On the other hand, "system" corresponds to the labels of concepts in the reference ontologies. Consequently, in order to disambiguate "rocket delivery system", Input Concept Selection Module shows the concepts which have "system" as their label.

Input terms which do not correspond to the labels of concepts in the reference ontologies are `undefined terms`. The input terms are also undefined terms if the concept exists but there are no appropriate concepts in the reference ontologies. The user defines the undefined terms manually in the Refinement Module.

### 2.4 Construction Module

The Construction Module automatically generates the basis of an ontology, an initial concept hierarchy and set of concept pairs, by referring to reference ontologies and documents. An initial concept hierarchy is constructed as taxonomic relationships. Set of concept pairs are extracted by using co-occurrency based statistic methods. These pairs are considered to be closely related and that they will be used as candidates to refine and add other relations. The user identifies some relationships between concepts in the pairs. The methods for generating an initial concept hierarchy and sets of concept pairs are described in our former study [5].

### 2.5 Refinement Module

In the Refinement Module, the initial ontology generated by Construction Module is refined by the user through interactive support of Visualization Module. In order to refine the initial ontology, we manage `concept drifts` and evaluate the sets of concept pairs.

If the initial concept hierarchy is constructed from a general ontology, adjustment of the initial concept hierarchy to the specific domain considering an issue called `concept drift` is required. `Concept drift` implies that the positions of particular concepts may change depending on the domain. For concept drift management, DODDLE-OWL applies two strategies: `matched result analysis` and `trimmed result analysis`. In `matched result analysis`, DODDLE-OWL divides the taxonomy into PABs (PAths including only Best-matched concepts) and STMs (SubTrees that includes best-matched concepts and other concepts and so can be Moved) and indicates on the screen. PABs are paths that include only best-matched concepts that have senses suitable for the given domain. Best-matched concepts means the input concepts. STMs are subtrees of which root is an internal concept of the reference ontology and its subordinates are all best-matched concepts. Since the sense of an internal concept has not been identified by a user yet, STMs may be moved to other places for the concept adjustment to the domain. In addition, for Trimmed Result Analysis, DODDLE-OWL counts the number of internal concepts when the part was trimmed. By considering this number as the original distance between those two concepts, DODDLE-OWL indicates to move the lower concept to other places. The detail of these strategies are described in our former study [5].

At the phase of concept specification template construction, criteria to evaluate significant concept pairs are necessary; the significant concept pairs are from the sets of concept pairs generated by the Construction Module. In [5], two statistics based methods are investigated: the value of context similarity by the WordSpace method [9] and the value of confidence by the association rule learner [10]. These methods and values based on co-occurrency of concepts work well in terms of wide use (do not depend on some particular domains).

### 2.6 Visualization Module

In order to visually support the refinement of the semi-automatically constructed domain ontology, DODDLE-OWL is integrated with the Visualization Module. DODDLE-OWL uses $MR^3$ : Meta-Model Management based on RDFs Revision Reflection [11] as the Visualization Module. $MR^3$ is a graphical RDF and RDFS editor for managing relationships between RDF and RDFS descriptions. DODDLE-OWL can interchange an OWL ontology with $MR^3$ using a plug-in function of $MR^3$ .

Visualization Module has two main roles for supporting domain ontology construction. One is the visualization function for concept drift management in the Refinement Module. Visualization Module displays the initial concept hierarchy generated in the Construction Module. Then, the user can visually refine candidates of concept drifts which are suggested by the Refinement Module. The other role is the externalization of the domain ontology. The externalization of the domain ontology means visualizing the whole taxonomic relationships and other relationships in the domain ontology. Taxonomic relationships and other relationships are constructed separately in the Hierarchy Construction Module and the Relationship Construction Module. By the externalization of the domain ontology, the user can refine the domain ontology while regarding the balance of the taxonomic relationships and other relationships.

### 2.7 Translation Module

Translation Module exports the taxonomic relationships and other relationships described in OWL. Taxonomic relationships are defined using `owl:Class` class and `rdfs:subClassOf` property[†]. Other relationships are defined using `owl:ObjectProperty` class, `rdfs:domain` property, and `rdfs:range` property.

Figure 3 shows an example of exporting taxonomic relationships and other relationships in OWL. The upper part of Fig. 3 shows that `goods` class is a subclass of `artifact` class. The lower part of Fig. 3 shows that `attribute` relationships is defined between an individual of `goods` class and an individual of `quality` class.

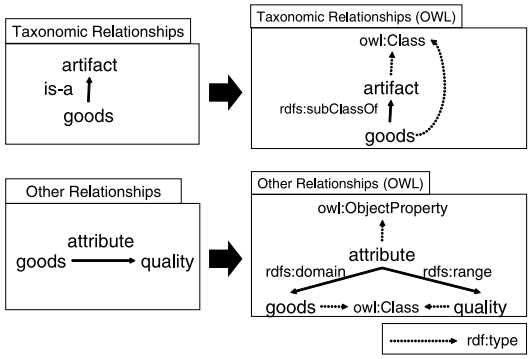


**Fig. 3** An example of exporting taxonomic relationships and other relationships in OWL.

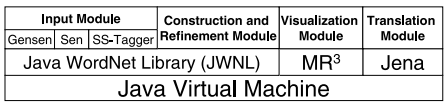


**Fig. 4** Implementation architecture of DODDLE-OWL.

## 3. Implementation

### 3.1 Implementation Architecture

Figure 4 shows the implementation architecture of DODDLE-OWL. DODDLE-OWL is implemented in Java language. Input Module, Construction Module, and Refinement Module use Java WordNet Library (JWNL) [12] to access WordNet. Input Module uses Gensen [13], Sen [14], and SS-Tagger [15]. Gensen is used for extracting Japanese and English compound words. Sen is a Japanese morphological analyzer implemented in Java language. Sen is used for extracting Japanese words and its POS from documents. SS-Tagger is an English POS tagger. SS-Tagger is used for extracting English words and its POS from documents. $MR^3$ [11] is used as the Visualization Module. $MR^3$ is an RDF(S) graphical editor with meta-model management facilities such as the consistency checking of the set of classes and a model, in which the classes are used as the type of instances in the model. Translation Module uses Jena: a Semantic Web framework for Java [16] to export constructed domain ontology described in OWL.

### 3.2 Typical Usage

Figure 5 shows a typical usage of DODDLE-OWL. DODDLE-OWL's user interface consists of "Ontology Selection Panel", "Document Selection Panel", "Input Term Selection Panel", "Input Concept Selection Panel", "Construction and Refinement for Classes Panel", "Construction and Refinement for Properties Panel", "Visualization Panel", and "Construction and Refinement for Relationships

[†]owl is a prefix of http://www.w3.org/2002/07/owl#
rdfs is a prefix of http://www.w3.org/2000/01/rdf-schema#.

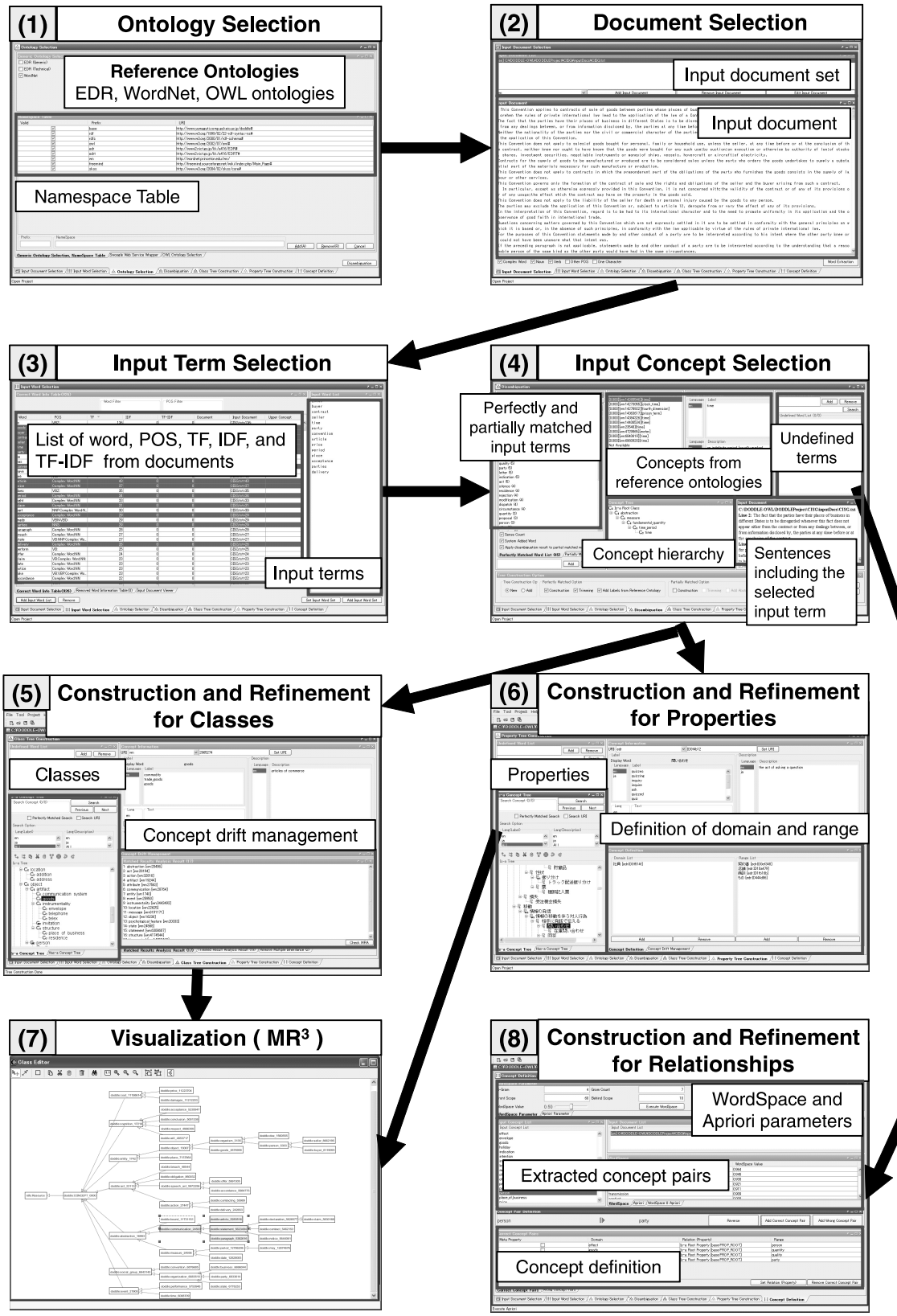


**Fig. 5** A typical usage of DODDLE-OWL.

Panel". First, the user selects reference ontologies from WordNet, EDR, and existing OWL ontologies in the "Ontology Selection Panel ((1) in Fig. 5)". Second, in the "Document Selection Panel ((2) in Fig. 5)", the user opens domain specific documents. In this "Document Selection Panel", words in the documents are extracted. Third, in the "Input Term Selection Panel ((3) in Fig. 5)", the user selects input terms which are significant terms for the domain. The user can sort the extracted terms based on POS, TF, IDF, and TF-IDF in this panel. Fourth, in the "Input Concept Selection Panel ((4) in Fig. 5)", the user associates the input terms with concepts by referring to the reference ontologies which were selected in the "Ontology Selection Panel". After mapping input terms to corresponding concepts, an initial class and property hierarchy are generated. Also set of concept pairs are extracted by co-occurrency based statistic methods such as WordSpace method and the association rule learner by default parameters. (5) and (6) of Fig. 5 shows the "Construction and Refinement for Classes Panel" and the "Construction and Refinement for Properties Panel". These panels indicate some groups of concepts in the taxonomy that might cause concept drifts, so that the user can decide which group to refine. (7) of Fig. 5 shows the display of concept drift management in the "Visualization Module". (8) of Fig. 5 shows the "Construction and Refinement Panel for Relationships". This panel is used for setting parameters used in the WordSpace method and the association rule learner to apply these to the documents in order to generate significantly related concept pairs. In WordSpace method, there are parameters such as the gram number, minimum N-gram count, front scope, and behind scope in the documents. In the association rule learner, minimum confidence and minimum support are set by the user. Finally, the user can export through the Translation Module a constructed domain ontology described in OWL.

### 3.3 DODDLE-OWL as an Open Source

In order to open our technology to communities concerning ontology construction such as knowledge system, Semantic Web, and so on, DODDLE-OWL and its source code are provided via our Web site (Fig. 6). Currently, DODDLE-OWL is open source software in Java and has 100 and more users from 20 and more countries. Some users have provided comments about DODDLE-OWL. We would like to reflect these comments in our future work.

## 4. Evaluation

### 4.1 Comparison between DODDLE-OWL and Manual-Building Method

#### 4.1.1 Conditions and Process of Experiment

In order to evaluate the efficiency of DODDLE-OWL, we compared the results for constructing ontologies using


**Fig. 6** Web site of DODDLE-OWL.
(URL: http://www.yamaguti.comp.ae.keio.ac.jp/mmm/doddle/)

DODDLE-OWL with the results of the popular manual-building method. In this experiment, manual-building method indicates the method for constructing ontologies using the currently most popular ontology editor, Protégé. During the experiment, when the users constructed ontologies with the manual-building method, we permitted them to refer to general ontologies such as WordNet and EDR. In this experiment, ontology points is-a relationships (taxonomy).

Users A and B, each with some experience in ontology-construction, are the experimental subjects for this experiment. The target domains are financial accounting (FA) and human affairs (HA). The users are not domain experts but have abilities to read domain specific documents and understand the contents. In this experiment, the domain specific documents are written in Japanese, which is the users' native language. For this reason, DODDLE-OWL referred to EDR, a Japanese general ontology, as a reference ontology in this experiment.

Figure 7 shows the ontology construction process using the manual-building method in the experiment. First, a user selects input terms from the input document set. Then, the user constructs is-a relationships (taxonomy) from the input terms using Protégé referring to EDR.

Figure 8 shows the ontology construction process using DODDLE-OWL in the experiment. This process consists of the following five sub processes. First, a user extracts terms from the input document set using the Input Document Selection Module. Second, in order to select input terms, the user removes unnecessary terms from the extracted terms and adds input terms from the input document set which were not automatically extracted using the Input Term Selection Module. Third, the user selects input concepts from the input terms using the Input Concept Selection Module. Fourth, an initial taxonomy is automatically constructed from the input concepts by the Hierarchy Con-

struction Module. Fifth, the user refines the initial taxonomy using the Hierarchy Refinement Module.

The following two methods are viewed as the main methods to compare DODDLE-OWL with the manual-building method. In method 1, each user constructs both FA and HA domain ontologies with DODDLE-OWL and the manual-building method as shown in Table 1, and the construction times are compared. In method 2, one user constructs a FA domain ontology with DODDLE-OWL and the other user constructs a HA domain ontology with the manual-building method, and vice versa, as shown in Table 2, and the construction times are compared.

In method 1, the same user constructs four ontologies with DODDLE-OWL and the manual-building method from the same document sets of the same domains. When the influence of the proficiency in ontology construction of a particular domain is small, method 1 can be used to perform an accurate evaluation. However, when the influence is big, the experience of the first ontology construction leads to the speed-up of the next ontology construction. For this reason, depending on the order of ontology construction with DODDLE-OWL and the manual-building method, each ontology construction's time will not be reliable if used as it is. Therefore, in order to use method 1, it is required to examine beforehand the influence of each user's proficiency on the ontology construction time of each domain.

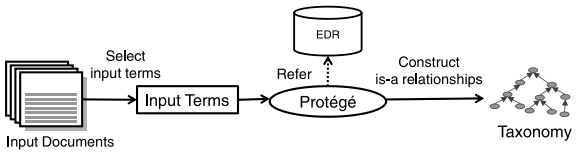


**Fig. 7** Ontology construction process using the manual-building method in the experiment.

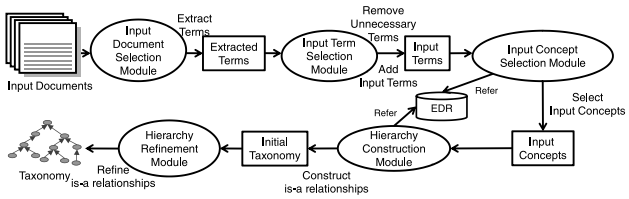


**Fig. 8** Ontology construction process using DODDLE-OWL in the experiment.

**Table 1** Domains of the ontologies constructed by each user in method 1.

| | DODDLE-OWL | Manual-Building Method |
|---|---|---|
| User A | FA | FA |
| | HA | HA |
| User B | FA | FA |
| | HA | HA |

**Table 2** Domains of the ontologies constructed by each user in method 2.

| | DODDLE-OWL | Manual-Building Method |
|---|---|---|
| User A | FA | HA |
| User B | HA | FA |

In method 2, each user constructs ontologies from the document set of each domain by using DODDLE-OWL or the manual-building method. This way, the time result in method 2 will not be influenced by the proficiency in the ontology construction of a particular domain as in method 1. However, if the knowledge levels of users A and B for domain ontology construction are not at the same level, it is not clear whether the difference in the ontology construction time is due to DODDLE-OWL or because of the difference in the knowledge level between the 2 users. Here the difference in the knowledge level indicates the difference in the ontology construction time coming from the domain knowledge level, the experience of ontology construction, the proficiency of tools, and etc. Therefore, it is difficult to directly compare the ontology construction times where one user uses DODDLE-OWL and the other user uses the manual-building method. In order to use method 2, it is required to examine the difference in the knowledge level between users A and B in the ontology construction of each domain beforehand. When the difference in the knowledge levels between the two users is examined, the expected ontology construction time of the manual-building method by the user using DODDLE-OWL can be obtained from the ontology construction time of the manual-building method by the other user. Note that since the ontology construction time might change depending on the content of the document set even if the ontology is constructed from the document set of the same domain and the same size, the expected ontology construction time of the manual-building method cannot be accurately measured. However, the expected time will become a guide. If the ontology construction time of DODDLE-OWL by one user is greatly shortened compared with the time of the manual-building method by the other user, it can be said that DODDLE-OWL has efficiency.

In order to resolve the above issues, we first performed a preliminary experiment, and then performed the main experiment regarding the results of the preliminary experiment. For the preliminary experiment, we prepared document sets of FA and HA domains which are different from the document set used in the main experiment but having almost the same size of the document sets used in the main experiment. Table 3 shows the number of documents and the number of words in each document sets of FA and HA

**Table 3** # of documents and # of words in each document sets of FA and HA domains using the preliminary experiment and the main experiment.

| | Domain | # of Documents | # of Words |
|---|---|---|---|
| Preliminary Experiment | FA (Procurement) | 19 | 20,041 |
| | HA (Acceptance) | 25 | 20,035 |
| Main Experiment | FA (Payment) | 19 | 20,388 |
| | HA (Payroll) | 29 | 20,286 |

**Table 4** Preliminary Experiment: construction times for the 2 times FA (procurement) domain ontology was constructed with the manual-building method by user A and B.

| | User A | | User B | |
|---|---|---|---|---|
| | 1st time | 2nd time | 1st time | 2nd time |
| Time for Input Term Extraction (min) | 65 | 43 | 42 | 34 |
| Time for is-a Relationship Construction (min) | 116 | 76 | 104 | 75 |
| Total Time (min) | 181 | 119 | 146 | 109 |

**Table 5** Preliminary Experiment: # of input terms and # of concepts of the FA domain ontologies described in Table 4.

| | User A | | User B | |
|---|---|---|---|---|
| | 1st time | 2nd time | 1st time | 2nd time |
| # of Input Terms | 210 | 208 | 207 | 173 |
| # of Concepts | 258 | 256 | 235 | 232 |

domains used in the preliminary experiment and the main experiment.

In the preliminary experiment, users A and B constructed ontologies by the manual-building method twice in each domain. There are two reasons to perform the preliminary experiment.

The first reason is to examine the difference in the knowledge level between users A and B in the FA and HA domain ontologies construction. Since the difference of the knowledge level between users A and B may differ from domain to domain, the users constructed the ontologies for each domain.

The second reason is to examine the influence of each user's proficiency in the FA and HA domain ontologies construction. In order to examine this influence, the users constructed the ontologies twice in each domain.

We decided to select whether to use method 1 or method 2 for the main experiment regarding the result of the preliminary experiment. If the influence of the proficiency in the ontology construction is small, we would use method 1 for the main experiment. On the other hand, if the influence of the proficiency is big, it is difficult to use method 1 to compare DODDLE-OWL and the manual-building method. In such case, we use method 2 referring to the difference in the knowledge level between user A and B obtained from the preliminary experiment.

#### 4.1.2 Results of Preliminary Experiment

Table 4 and Table 5 show the results of the preliminary experiment for FA (procurement) domain. Table 4 shows the construction times for the 2 times FA (procurement) domain ontology was constructed with the manual-building method by users A and B. Table 5 shows the number of input terms and the number of concepts of the FA domain ontologies described in Table 4.

Table 6 and Table 7 show the results of the preliminary experiment for HA (acceptance) domain. Table 6 shows the construction times for the 2 times HA (acceptance) domain ontology was constructed with the manual-building method by users A and B. Table 7 shows the number of input terms and the number of concepts of the HA domain ontologies described in Table 6.

#### 4.1.3 Discussion on Preliminary Experiment

According to Table 4 and Table 6, the times for the second ontology construction using the manual-building method by users A and B were approximately 25 minutes to 60 minutes shorter than the first ontology construction. Since the influence of the proficiency largely reduced the ontology construction time, we selected method 2, shown in Table 2, as the method for the main experiment.

According to Table 5 and Table 7, when users A and B constructed FA (procurement) and HA (acceptance) domain ontologies with the manual-building method, the size of the ontologies (the number of input terms and concepts) were approximately the same. Therefore, the difference in the knowledge level between users A and B in each domain is able to be roughly measured by the ratio of the first ontology construction time of both users. Table 8 and Table 9 show the ratio in the knowledge level between users A and B in each domain.

#### 4.1.4 Results of Main Experiment

In the main experiment, user A constructed the FA (payment) domain ontology with DODDLE-OWL and the HA (payroll) domain ontology with the manual-building method. User B constructed the FA (payment) domain ontology with the manual-building method and the HA (payroll) domain ontology with DODDLE-OWL.

We evaluated the main experiment by the following procedures. First, referring to the knowledge level between the users shown in Table 8 and Table 9, we calculated for a particular domain the estimated ontology construction time of the manual-building method for a user which performed ontology construction using DODDLE-OWL for the domain, from the ontology construction time of the manual-building method by the other user. Second, we compared each user's estimated ontology construction time for the manual-building method to the actual ontology construction time for DODDLE-OWL.

Table 10 and Table 11 show the results of the main experiment for the FA (payment) domain. Table 10 shows user A's FA (payment) domain ontology construction times for DODDLE-OWL and the manual-building method (estimate), and user B's FA (payment) domain ontology construction time for the manual-building method. Table 11 shows the number of input terms and the number of concepts in the FA (payment) domain ontologies constructed by

**Table 6** Preliminary Experiment: construction times for the 2 times for HA (acceptance) domain ontology was constructed with the manual-building method by user A and B.

| | User A | | User B | |
|---|---|---|---|---|
| | 1st time | 2nd time | 1st time | 2nd time |
| Time for Input Term Extraction (min) | 50 | 34 | 41 | 39 |
| Time for is-a Relationship Construction (min) | 87 | 70 | 108 | 84 |
| Total Time (min) | 137 | 104 | 149 | 123 |

**Table 7** Preliminary Experiment: # of input terms and # of concepts of the HA domain ontologies described in Table 6.

| | User A | | User B | |
|---|---|---|---|---|
| | 1st time | 2nd time | 1st time | 2nd time |
| # of input Terms | 191 | 205 | 198 | 205 |
| # of Concepts | 228 | 252 | 245 | 235 |

**Table 8** Preliminary Experiment: the ratio in the knowledge level between user A and B for FA domain ontology construction.

| | User A | User B |
|---|---|---|
| Input Term Extraction | 1.548 | 1.000 |
| is-a Relationship Construction | 1.115 | 1.000 |

**Table 9** Preliminary Experiment: the ratio in the knowledge level between user A and B for HA domain ontology construction.

| | User A | User B |
|---|---|---|
| Input Term Extraction | 1.000 | 0.820 |
| is-a Relationship Construction | 1.000 | 1.241 |

user A with DODDLE-OWL and user B with the manual-building method. Table 12 shows the number of extracted terms by DODDLE-OWL, the number of input terms added by user A, the number of input terms, precision, and recall in the FA (payment) domain ontologies by user A with DODDLE-OWL.

Table 13 and Table 14 show the results of the main experiment for the HA (payroll) domain. Table 13 shows user B's HA (payroll) domain ontology construction time for DODDLE-OWL and the manual-building method (estimate), and user A's HA (payroll) domain ontology construction time for the manual-building method. Table 14 shows the number of input terms and the number of concepts in the HA (payroll) domain ontologies constructed by user B with DODDLE-OWL and user A with the manual-building method. Table 15 shows the number of extracted terms by DODDLE-OWL, the number of input terms added by user B, the number of input terms, precision, and recall in the HA (payroll) domain ontologies by user B with DODDLE-OWL.

$$SI = I - UI \tag{1}$$

$$Precision = \frac{SI}{ST} \tag{2}$$

$$Recall = \frac{SI}{I} \tag{3}$$

The precision and the recall in Table 12 and Table 15 were calculated using Formula 1 to 3. In Formula 1 to 3, $SI$ is the number of extracted input terms by DODDLE-OWL, $I$ is the number of input terms, $UI$ is the number of input terms added by a user, and $ST$ is the number of extracted terms by DODDLE-OWL.

#### 4.1.5 Discussion on Main Experiment

According to Table 10 and Table 13, the times for the ontology construction by both users A and B using DODDLE-OWL were approximately 1 hour shorter than the estimated ontology construction times using the manual-building method. From these results cannot be shown the correlation between the size of the ontology and the ontology construction time able to be shortened by DODDLE-OWL. However, it can be said that the construction cost for the is-a relationships including hundreds of concepts construction cost can be reduced by using DODDLE-OWL.

We analyzed where DODDLE-OWL was effective in the ontology construction processes.

In the input term extraction, the extraction time by user A using DODDLE-OWL was shorter than the estimated extraction time for the manual-building method for user A. However, the extraction time by user B using DODDLE-OWL was longer than the estimated extraction time for the manual-building method for user B. According to Table 3, the number of words in the document set of FA (payment) domain is approximately the same as the number of words in the document set of HA (payroll) domain. However, the precision and the recall of input term extraction using DODDLE-OWL in each domain were different. According to Table 12 and Table 15, the precision and the recall of input term extraction using DODDLE-OWL by user A in FA (payment) domain were higher than the precision and the recall of input term extraction using DODDLE-OWL by user B in HA (payroll) domain. Therefore, the time for the removal of unnecessary terms from the automatically extracted terms and the addition of input terms from the input document set, which could not be automatically extracted, by user B using DODDLE-OWL took longer than user A doing the same work using DODDLE-OWL. From the above results, it can be said that the input term extraction time might not be able to be shortened by using DODDLE-OWL when the precision and recall of input term extraction using DODDLE-OWL were low depending on domains, documents, and etc. However, according to Table 11 and Table 14, a user using DODDLE-OWL could extract input terms more than the other user using the manual-building method. One of the reasons is that when the users extract input terms with the manual-building method, they often overlook input terms that look alike. For this reason, it

**Table 10** Main Experiment: user A's FA (payment) domain ontology construction times for DODDLE-OWL and the manual-building method (estimate), and user B's FA (payment) domain ontology construction time for the manual-building method.

| | User A DODDLE-OWL | User A Manual-Building Method (Estimate) | User B Manual-Building Method |
|---|---|---|---|
| Time for Input Term Extraction (min) | 22 | 63 | 41 |
| Time for Concept Selection (min) | 8 | 0 | 0 |
| Time for is-a Relationship Construction (min) | 35 | 91 | 82 |
| Total Time (min) | 65 | 154 | 123 |

**Table 11** Main Experiment: # of input terms and # of concepts in the FA (payment) domain ontologies constructed by user A with DODDLE-OWL and user B with the manual-building method.

| | User A DODDLE-OWL | User B Manual-Building Method |
|---|---|---|
| # of Input Terms | 275 | 177 |
| # of Concepts | 326 | 215 |

**Table 12** Main Experiment: # of extracted terms by DODDLE-OWL, # of input terms added by user A, # of input terms, precision, and recall in the FA (payment) domain ontologies by user A with DODDLE-OWL.

| | |
|---|---|
| # of Extracted Terms by DODDLE-OWL | 537 |
| # of Input Terms Added by User A | 18 |
| # of Input Terms | 275 |
| Precision | 0.48 |
| Recall | 0.93 |

is considered that DODDLE-OWL prevents the leakage of the selection of input terms from the documents. This explains the fact that the number of concepts in the ontologies constructed by DODDLE-OWL was larger than the number of concepts in the ontologies constructed by the manual-building method.

Since the times for concept selection using DODDLE-OWL for both users were both approximately 8 minutes, it can be said that this cost is not so high when looking at the entire ontology construction time.

Since the times for the is-a relationships construction using DODDLE-OWL for both users were both approximately 1 hour shorter than the estimated times of the manual-building method, it can be said that DODDLE-OWL was effective for is-a relationships construction.

The users had an impression that when they constructed ontologies using the manual-building method, they took much time for the nonessential parts of ontology construction such as extracting input terms from the input documents and inputting the input terms as classes into Protégé. On the other hand, the users had an impression that when they constructed ontologies using DODDLE-OWL, they were able to concentrate on essential parts of the ontology construction such as refining the ontology since the nonessential parts were supported by DODDLE-OWL. From these impressions of the users, it may be said that high quality ontologies could be constructed by using DODDLE-OWL.

## 4.2 Case Studies

### 4.2.1 Specification of Case Studies

In order to evaluate the scalability of DODDLE-OWL, we constructed a large sized ontology over 34,000 concepts in the field of rocket operation using DODDLE-OWL. Since ontologies used for searching documents usually includes enormous concepts to cover the documents, we constructed an ontology usable for searching documents. In these case studies, we show that DODDLE-OWL can reduce the cost for construction of a large sized ontology.

In order to evaluate the large sized ontology, we installed the ontology to a search engine and performed document retrieval with the cooperation of Galaxy Express Corporation (GALEX). We used GXFinder [17] developed by GALEX for the search engine. GXFinder is able to install a domain ontology described in OWL and perform ontology based search using concept hierarchy in the domain ontology. A rocket operation expert evaluated the top 10 and 20 search results. Since it was difficult for the expert to make test collection (pairs of keywords and documents corresponding to these keywords) without assuming the search context, we showed the expert the concept hierarchy of the rocket operation ontology to have the expert makes the test collection. Then, the expert performed keyword search and ontology based search.

In the followings, we describe ontology based search, construction of the rocket operation ontology using DODDLE-OWL, comparison between keyword search and ontology based search, and discussion on these case studies.

### 4.2.2 Ontology Based Search

The ontology based search consists of `specialized search` and `generalized search`.

When there are huge amounts of search results (more than 50 for these case studies), `specialized search` helps the user to find appropriate documents. The procedures for the specialized search are as follows. First, the user inputs search keywords to GXFinder. Second, GXFinder finds from the installed domain ontology the concepts having the search keywords as their labels. Third, GXFinder extracts the labels of sub concepts of the concepts extracted in the second step. Finally, GXFinder finds documents which include the labels extracted in the third step.

**Table 13** Main Experiment: user B's HA (payroll) domain ontology construction time for DODDLE-OWL and the manual-building method (estimate), and user A's HA (payroll) domain ontology construction time for the manual-building method.

| | User B DODDLE-OWL | User B Manual-Building Method (Estimate) | User A Manual-Building Method |
|---|---|---|---|
| Time for Input Term Extraction (min) | 40 | 35 | 43 |
| Time for Concept Selection (min) | 8 | 0 | 0 |
| Time for is-a Relationship Construction (min) | 38 | 109 | 88 |
| Total Time (min) | 86 | 144 | 131 |

**Table 14** Main Experiment: # of input terms and # of concepts in the HA (payroll) domain ontologies constructed by user B with DODDLE-OWL and user A with the manual-building method.

| | User B DODDLE-OWL | User A Manual-Building Method |
|---|---|---|
| # of Input Terms | 300 | 212 |
| # of Concepts | 393 | 272 |

**Table 15** Main Experiment: # of extracted terms by DODDLE-OWL, # of input terms added by user B, # of input terms, precision, and recall in the HA (payroll) domain ontologies by user B with DODDLE-OWL.

| | |
|---|---|
| # of Extracted Terms by DODDLE-OWL | 783 |
| # of Input Terms added by User B | 63 |
| # of Input Terms | 300 |
| Precision | 0.38 |
| Recall | 0.79 |

When there are few or no search results (less than 10 for these case studies), `generalized search` helps the user to find appropriate documents. The procedures for the generalized search are as follows. The first two steps are the same as the first two steps in the specialized search procedures. In the third step, GXFinder extracts labels of the sibling concepts and the super-concepts of the concepts extracted in the second step. Finally, GXFinder finds documents which include the labels extracted in the third step.

#### 4.2.3 Constructing a Rocket Operation Ontology

A user who is not a domain expert constructed the rocket operation ontology using DODDLE-OWL spending about 30 hours. The user constructed the ontology following the procedures. 2,484 Japanese documents made by GALEX Toyosu Branch were used as input documents. First, nouns, verbs, and compound words were automatically extracted from the input documents by the Input Document Selection Module. Second, the user removed unnecessary terms and selected input terms using the Input Term Selection Module. Third, the user selected input concepts using the Input Concept Selection Module. Finally, DODDLE-OWL automatically constructed taxonomic relationships.

Table 16 shows the number of automatically extracted terms, input terms, perfectly matched terms, partially matched terms, undefined terms, and total concepts of the rocket operation ontology.

In a typical situation, a semi-automatically constructed taxonomy should be refined using the two strategies mentioned in Sect. 2. However, in the case studies, the user could not refine the taxonomy because of the enormous amount of concepts (about 34,000) causing the amount of concepts (about 2,000) to be refined by the refinement strategies to be enormous too. Therefore, the above-mentioned ontology construction time (about 30 hours) mainly includes the input term selection time and the concept selection time, and does not include the ontology refinement time.

**Table 16** # of automatically extracted terms, # of Input terms, # of perfectly matched terms, # of partially matched terms, # of undefined terms, and # of total concepts about rocket operation ontology.

| | |
|---|---|
| # of Automatically Extracted Terms | 41,806 |
| # of Input Terms | 32,814 |
| # of Perfectly Matched Terms | 4,982 |
| # of Partially Matched Terms | 26,835 |
| # of Undefined Terms | 997 |
| # of Total Concepts | 34,451 |

In the experiments in Sect. 4.1, the users with some experiences in ontology-construction took about 2 hours to construct ontologies including about 200 concepts with the manual-building method. Although it is difficult to estimate the construction time of the ontology including about 34,000, we can say that it will take at least hundreds of hours with the manual-building method. Therefore, it can be said that about 30 hours for constructing the ontology including about 34,000 concepts is extremely fast.

#### 4.2.4 Results of Case Studies

In order to evaluate the large sized ontology, we show two case studies as follows. Each describe a case in which a user wants to search documents concerning a specific keyword, but was not able to come up with the appropriate search keyword. Case study 1 show a case in which the user wants to find documents on "launching control table", but the search was performed with the keyword "control table". Case study 2 show the case in which the user wants to find documents on "terminal countdown sequence", but the search was performs with the keyword "countdown sequence". Figure 9 shows the concept hierarchy for "control table" and "countdown sequence". Table 17 shows the search result results for case studies 1 and 2. Table 18 and Table 19 show the recall, precision, and f-measure in each case studies.

#### 4.2.5 Discussion on Case Studies

According to Table 18 and Table 19, specialized search had

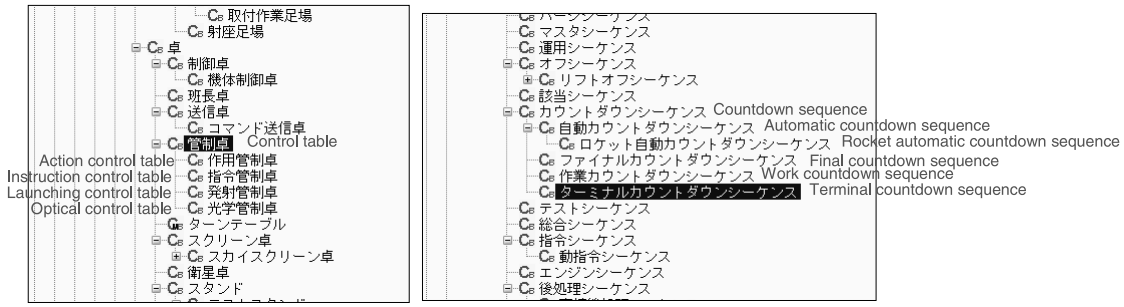


**Fig. 9** Concept hierarchy for "control table" and "countdown sequence".

**Table 17** Search results in case study 1 and 2.

| | Case Study 1 | Case Study 2 |
|---|---|---|
| # of total positive documents | 4 | 22 |
| # of hit of keyword search | 86 | 82 |
| # of positive documents in top 10 keyword search results | 1 | 3 |
| # of positive documents in top 20 keyword search results | 3 | 3 |
| # of hit of specialized search | 44 | 46 |
| # of positive documents in top 10 specialized search results | 3 | 8 |
| # of positive documents in top 20 specialized search results | 4 | 11 |

**Table 18** Recall, Precision, and F-Measure in case study 1.

| Search Type | Recall | Precision | F-Measure |
|---|---|---|---|
| top 10 keyword search | 0.250 | 0.100 | 0.143 |
| top 10 specialized search | 0.750 | 0.300 | 0.429 |
| top 20 keyword search | 0.750 | 0.150 | 0.250 |
| top 20 specialized search | 1.000 | 0.200 | 0.333 |

**Table 19** Recall, Precision, and F-Measure in case study 2.

| Search Type | Recall | Precision | F-Measure |
|---|---|---|---|
| top 10 keyword search | 0.136 | 0.300 | 0.188 |
| top 10 specialized search | 0.364 | 0.800 | 0.500 |
| top 20 keyword search | 0.136 | 0.150 | 0.143 |
| top 20 specialized search | 0.500 | 0.550 | 0.524 |

better search results than keyword search. It is considered that specialized search works well when the user cannot imagine clearly the search keywords. In the case studies, generalized search did not work well. One of the reasons was that there were many sibling concepts of the concepts which had the keyword as their labels. Therefore, there were too many search results with generalized search. In order to utilize the generalized search, domain ontology refinement (especially sibling concepts reduction) is necessary. Although DODDLE-OWL has the functions to refine domain ontologies, the functions did not work well in the case studies because of the enormous amount of concepts in the domain ontology. The development of a new ontology refinement function dealing with the great number of concepts is for future works.

When the domain expert checked the rocket operation ontology, the domain expert indicated some incorrect parts of the ontology. If the initial domain ontology was not constructed, we could not have such discussion with the domain expert.

Though the initial rocket operation ontology has several issues mentioned above, these case studies show us that DODDLE-OWL deals well with developing large sized ontology.

## 5. Related Work

Navigli, et al. proposed OntoLearn [18] which supports domain ontology construction by using existing ontologies and natural language processing techniques. In their approach, existing concepts from WordNet are enriched and pruned to fit the domain concepts by using NLP (Natural Language Processing) techniques. They argue that the automatically constructed ontologies are practically usable in the case study of a terminology translation application. However, they did not show any evaluations of the generated ontologies themselves that might be done by domain experts. Although much useful information is in the MRDs (Machine Readable Dictionaries) and documents in the application domain, some essential concepts and knowledge are still in the minds of domain experts. In our study, the ontologies were not generated automatically, but we suggest relevant alternatives to the human experts interactively while the experts construct domain ontologies. In our previous work [19], it was shown that even if the concepts are in the MRD, they may not be sufficient for use. In the case study, "concept drifts", in which some parts of hierarchical relations are counterchanged between the generic ontology (WordNet) and the domain ontology, were seen. Presenting an automatically generated ontology containing concept drifts may cause confusion in domain experts. To deal with this, it was argued that the initiative should be kept not on the machine, but on the hand of the domain experts at the domain ontology construction phase. This is the difference between our approach and Navigli's; our human-centered ap-

proach enabled tight cooperation between DODDLE-OWL and human experts.

## 6. Conclusion

In this paper, we proposed an interactive domain ontology development environment called DODDLE-OWL. In order to build up good interactive environment for domain ontology development support, we focused on the following four keywords: standard (OWL), extension (to Japanese), open and scalability. By integrating several modules, DODDLE-OWL became a practical and interactive domain ontology development environment. Currently, DODDLE-OWL is open source software in Java and has 100 and more users from 20 and more countries.

In order to evaluate the efficiency of DODDLE-OWL, we compared DODDLE-OWL with the popular manual-building method. The times for the ontology construction by both users A and B using DODDLE-OWL were approximately 1 hour shorter than the estimated ontology construction times using the manual-building method. From these results cannot be shown the correlation between the size of the ontology and the ontology construction time able to be shortened by DODDLE-OWL. However, it can be said that the construction cost for the is-a relationships including hundreds of concepts construction cost can be reduced by using DODDLE-OWL.

In order to evaluate the scalability of DODDLE-OWL, we constructed a large sized ontology over 34,000 concepts in the field of rocket operation using DODDLE-OWL. In order to evaluate the large sized ontology, we installed the ontology to a search engine and performed document retrieval with the cooperation of Galaxy Express Corporation (GALEX). Through the case studies, DODDLE-OWL deals well with developing large sized ontologies.

Through the above evaluation, we confirmed the efficiency and the scalability of DODDLE-OWL.

As future work, we will try to develop a new ontology refinement function to deal with the case in which there is an enormous amount of concepts and is intractable. In order to construct domain ontologies using existing domain ontologies described in OWL, we will try to integrate DODDOE-OWL and an ontology search engine such as Swoogle [20].

## Acknowledgements

We appreciate Mr. Seiji Koide, Mr. Masanori Kawamura, Mr. Yunki Hong, and Mr. Yutaka Ono have worked with case studies.

This work was supported by Grant-in-Aid for JSPS Fellows (19-9818).

### References

[1] Y. Ding and S. Foo, "Ontology research and development, part 1 – A review of onlotogy," Journal of Information Science, vol.28, no.2, pp.123–136, 2002.

[2] M.K. Smith, C. Welty, and D.L. McGuinness, "OWL Web ontology language guide," 2004. http://www.w3.org/TR/owl-guide/

[3] A. Maedche and S. Staab, "Discovering conceptual relations from text," Proc. 14th European Conference on Artificial Intelligence, pp.321–325, 2000.

[4] D. Faure and C. Nedellec, "Knowledge acquisition of predicate argument structures from technical texts," Proc. International Conference on Knowledge Engineering and Knowledge Management, pp.329–334, 1999.

[5] M. Kurematsu, T. Iwade, N. Nakaya, and T. Yamaguchi, "DODDLE II: A domain ontology development environment using a MRD and text corpus," IEICE Trans. Inf. & Syst., vol.E87-D, no.4, pp.908–916, April 2004.

[6] National Institute of Information and Communications Technology, EDR Electronic Dictionary Technical Guide. http://www2.nict.go.jp/r/r312/EDR/index.html

[7] R. Sekiuchi, C. Aoki, M. Kurematsu, and T. Yamaguchi, "DODDLE: A domain ontology rapid development environment," Proc. 5th Pacific Rim International Conference on Artificial Intelligence, Lecture Notes in Computer Science, vol.1531, pp.194–204, Springer, 1998.

[8] G.A. Miller, "WordNet: A lexical database for English," ACM, vol.38, no.11, pp.39–41, 1995.

[9] H.S. Marti and A. Hearst, "Customizing a lexicon to better suit a computational task," Corpus Processing for Lexical Acquisition, pp.77–96, 1996.

[10] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large database," Proc. 20th International Conference on Very Large Data Bases, pp.487–499, Morgan Kaufmann, 1994.

[11] T. Morita, N. Izumi, N. Fukuta, and T. Yamaguchi, "A graphical RDF-based meta-model management tool," IEICE Trans. Inf. & Syst., vol.E89-D, no.4, pp.1368–1377, April 2006.

[12] "Java wordnet library." http://jwordnet.sourceforge.net/

[13] H. Nakagawa and T. Mori, "A simple but powerful automatic term extraction method," Computerm2: 2nd International Workshop on Computational Terminology, COLING-2002 WORKSHOP, pp.29–35, 2002.

[14] "Sen." http://ultimania.org/sen/

[15] Y. Tsuruoka and J. Tsujii, "Bidirectional inference with the easiest-first strategy for tagging sequence data," Proc. HLT/EMNLP, pp.467–474, 2005.

[16] HP Labs, "Jena: A semantic Web framework for Java," http://jena.sourceforge.net/

[17] S. Koide, M. Kawamura, T. Morita, T. Yamaguchi, and H. Takeda, "Semantic search: An implementation, deployments, and lessons learned," Proc. 1st Asian Semantic Web Conference Workshop on Web Search Technology, Beijing, China, 2006.

[18] R. Navigli and P. Velardi, "Automatic adaptation of WordNet to domains," Proc. International Workshop on Ontologies and Lexical Knowledge Bases, 2002.

[19] T. Yamaguchi, "Constructing domain ontologies based on concept drift analysis," IJCAI Workshop on Ontologies and Problem-Solving Methods, pp.13-1–13-7, 1999.

[20] L. Ding, R. Pan, T. Finin, A. Joshi, Y. Peng, and P. Kolari, "Finding and ranking knowledge on the semantic web," Proc. 4th International Semantic Web Conference, LNCS 3729, pp.156–170, 2005. http://swoogle.umbc.edu/

**Takeshi Morita** is a graduate student at the school of Science and Technology at Keio University. He received B.E. and M.E. degrees in computer science from Shizuoka University in 2003 and 2005, respectively. He has been a Research Fellow of the Japan Society for the Promotion of Science since April 2007. His research interests include Ontology Engineering and Semantic Web. He is a student member of JSAI and ISSJ.

**Naoki Fukuta** is an assistant professor at the Faculty of Informatics at Shizuoka University. He received B.E., M.E., and Ph.D degrees from Nagoya Institute of Technology in 1997, 1999, and 2002, respectively. His research interests include Software Engineering, Semantic Web, and WWW-based Intelligent Systems. He is a member of IEEE-CS, ACM, JSAI, IPSJ, and JSSST.

**Noriaki Izumi** is a researcher at the National Institute of Advanced Industrial Science and Technology. He received B.E. and M.E. degrees from Osaka Prefecture University in 1992 and 1994, respectively. He received a Ph.D degree from Keio University. His research interests include Intelligent Systems, Knowledge Modeling, Semantic Web, and Web Services. He is a member of JSSST, IPSJ, and JSAI.

**Takahira Yamaguchi** is a professor at the Faculty of Science and Technology at Keio University. He received his B.E., M.E., and Ph.D. degrees in telecommunication engineering from Osaka University in 1979, 1981, and 1984, respectively. His research interests include Ontology Engineering, KBSE, Advanced Knowledge Systems, and Machine Learning. He is a member of IPSJ, JSAI, JSFTS, JCSS, ISSJ, AAAI, IEEE-CS, and ACM.