OntoGear – A Road to Funnotation

Sunao Takafuji, Yoshinobu Kitamura, and Riichiro Mizoguchi

The Institute of Scientific and Industrial Research Osaka University
8-1 Mihogaoka, Ibaraki, Osaka, 567-0047 Japan
takaj@sanken.osaka-u.ac.jp, {kita, miz}@ei.sanken.osaka-u.ac.jp

Abstract. Ontology-based modeling and metadata annotation is the important foundation for any semantic web applications. Our ontology and metadata annotation framework about functionality have provided both theoretical basis and practical contribution to engineering knowledge management. On the other hand, to minimize modeling and annotation cost for a knowledge practitioner is another key aspect to apply the ontology in real use; the technique or tools such as supporting modeling and capturing ontological knowledge from text is complementarily required. This paper focuses on such complementary topic, and proposes functional modeling and semi-automatic metadata annotation system to interoperably share and reuse engineering knowledge.

1 Introduction

Metadata play an important role in semantically managing web pages and corporate documents, so that there exist many annotation frameworks or tools such as OntoAnnotate[16], OntoMatAnnotizer[1], S-CREAM[5], etc. While those tools are useful for general purpose, they may need extensible capability in terms of representing and handling metadata for the specific use like engineering one. In engineering domain, functionality of an artifact is regarded as indispensable viewpoint in many pieces of research[6, 14, 17, 20], and designers contrive how to safely realize the best functionality for their artifacts, and describe a variety of engineering documents, e.g. specification sheets, FMEA (Fault Mode and Effects Analysis) sheets, and so on. Such documents include so-called design rationale[2, 13], which means designer's intention, but it can be implicit due to the natural language text in an ad hoc way. To reveal and share the tacit knowledge on engineering design, we believe that functional modeling and functional metadata annotation is necessary for engineering documents.

To clarify and share engineering knowledge from the viewpoint of functionality of an artifact, we have presented systematization framework of functional knowledge for years[8], and deployed the initial system based on the framework into an industry successfully[9]. As the application of the framework from the Semantic Web context, we also present functional annotation framework[10], which is called Funnotation, and renewed the above-mentioned system that is named OntoGear with extended functionality for modeling and Funnotating.

[1] http://annotation.semanticweb.org/ontomat/index.html

This paper proposes semi-automatic approach for Funnotation framework with OntoGear using natural language processing and text mining strategy, especially focusing on function of a device. First, we overview functional ontology and Funnotation framework. Next, we discuss the architecture of OntoGear and our approach of semi-automatic annotation. Finally, we introduce some case studies using OntoGear to be applied to our framework.

2 Knowledge Sharing Framework Based on Ontology Engineering

2.1 Functional Ontology and Modeling

Our definition of function is a role played by a behavior specified in a context (Fig. 1(a))[8]. For example, while intrinsic behavior of a heat exchanger is just only to transfer thermal energy, heating or cooling is the functional concept interpreted as thermal energy transferred from a medium carrying more energy to another carrying less energy, or vice versa. This viewpoint about behavior and function allows a designer to clearly identify the function of a device. Based on the consideration of function, we have built a functional concept ontology composed of about 220 concepts[8].

As any device has hierarchical structure of components and functions, we can describe functional structure of a device as a tree structure. Although functional decomposition[14] is known as an industrial design method, we have proposed an improved approach to separately describe function and way that means how to achieve a function. The separation of them contributes not only to clarifying the functional structure of an artifact, but also to enabling comparison between different ways to the same function.

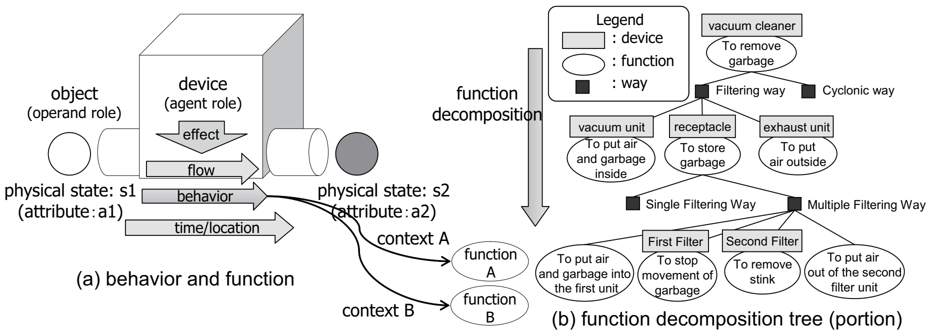


Fig. 1. Function Model and Function Decomposition Tree

Furthermore, we can organize way knowledge by means of generalizing an instance of combination of a function, a way and sub-function(s). The organized way knowledge database enables one to interoperably share the stored design knowledge. Fig.1(b) shows function decomposition tree (hereinafter FDT) that indicates the functional structure of a vacuum cleaner. The main function of a vacuum cleaner is *to remove garbage (from the floor)*, which is described as the root node of the tree. The root function has two ways (OR connection): Filtering and Cyclonic ways each of which has a sub functional structures (AND connection). Thus an FDT is decomposed from the top node of function to the bottom with the combination of ways and functions.

2.2 Annotation about Function for Semantic Application

Our ontology can also provide functional metadata schema, so-called Funnotation schema, in order to annotate metadata for engineering documents on the Internet or an intranet. In this section, we briefly introduce overview of the Funnotation framework.

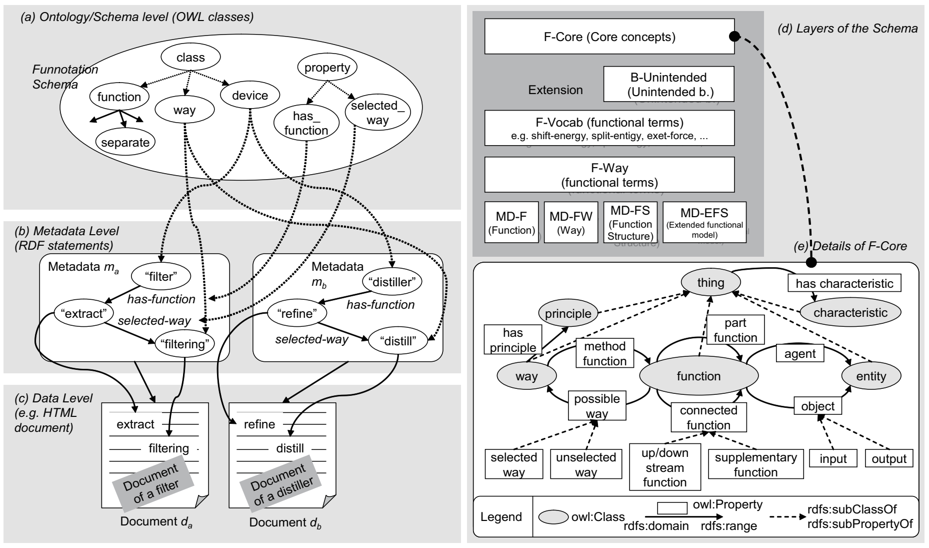


Fig. 2. Overview of the Funnotation framework

Funnotation framework is broken into three levels such as *Ontology/Schema Level*, *Metadata Level*, and *Data Level* (Fig. 2(a),(b),(c)). Ontology/Schema Level provides Funnotation Schema that consists of hierarchies of classes (types) in OWL (Web Ontology Language) corresponding to the elements of functional concept in our ontology. Ontologies at this level can be written by an ontology editor, e.g. Hozo[18]. The metadata based on the Funnotation schema are described as instances of those classes in RDF (Resource Description Framework) statements, which are actually described in engineering documents.

Funnotation schema consists of several layers: F-Core, B-Unintended, F-Vocab and F-Ways as shown in Fig. 2(d). F-Core layer defines primary concepts such as function, way, entity, extension of specific concept in terms of failure in B-Unintended. F-Vocab is a vocabulary set corresponding to the functional concept ontology. F-Ways layer defines generic function-achievement ways as a subclass of way class. Though Funnotation framework includes reference ontology in relation to other ontologies such as FB, this paper does not discuss such an advanced issue. Details of Funnotation framework and Funnotation search system as an application of the framework are discussed in [10].

2.3 Semi-Automatic Annotation Approach

Generally, annotating metadata onto vast amount of documents requires a lot of efforts to a knowledge author, so that semi-automatic and automatic semantic annotation research[5, 4, 21] is the primary topic on knowledge management research. While machine learning and text mining techniques are adapted to extraction or construction of ontological knowledge, they have limitations in terms of completeness. Therefore our semi-automatic approach adopts hybrid one that both manual annotation and automatic one are available according to the context of usage.

Our semi-automatic metadata annotation system has two sub-modules which consist of preparation and annotation as shown in Fig. 3. The former sub-module is for building resources that are functional concept dictionary and way knowledge database through functional concept mining environment and way knowledge mining environment, respectively. We minutely describe the preparation sub-module as Functional Knowledge Mining Framework in section 3.

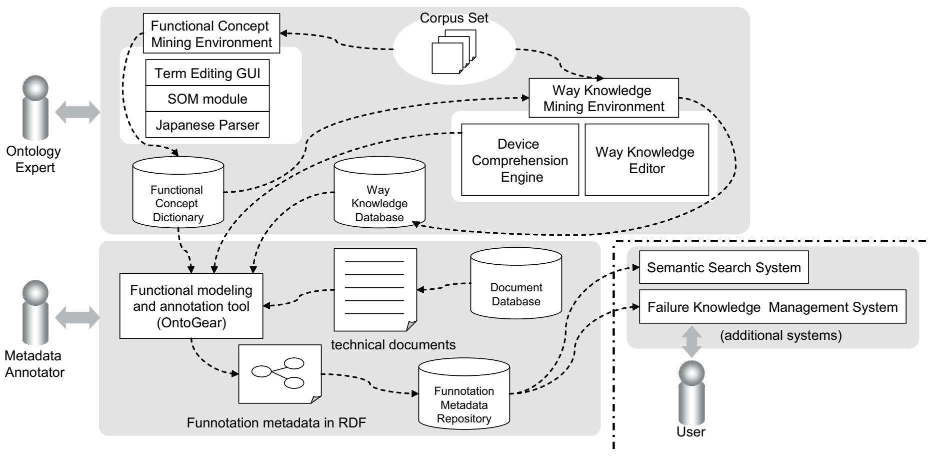

Functional Concept Mining Environment
Corpus Set
Term Editing GUI
SOM module
Japanese Parser
Way Knowledge Mining Environment
Ontology Expert
Device Comprehension Engine
Way Knowledge Editor
Functional Concept Dictionary
Way Knowledge Database
Functional modeling and annotation tool (OntoGear)
Document Database
Metadata Annotator
technical documents
Semantic Search System
Failure Knowledge Management System
(additional systems)
Funnotation Metadata Repository
Funnotation metadata in RDF
User

Fig. 3. Semi-Automatic Metadata Annotation System

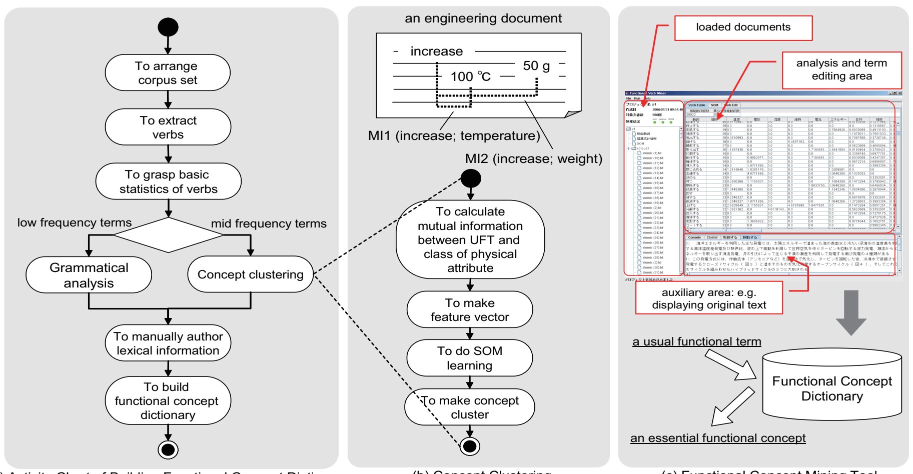
Then, the latter one is for Funnotation by using the resources, where functional modeling and annotation system bridges the gap between manual and full-automatic. We hereafter define the term *OntoGear system* is the system that realizes such semi-automatic metadata annotation system from broader viewpoint, and *OntoGear* is a functional modeling and annotation tool from narrower viewpoint in this paper.

3 Functional Knowledge Mining System

We here discuss how to build two prepared resources, i.e. functional concept dictionary and way knowledge database, to help annotators with less effort during annotation.

3.1 Mining Module of Functional Concept

Aiming at building language resources to map *usual functional terms* (hereinafter UFT) onto *essential functional concepts* (hereinafter EFC), we contrived the framework to effectively extract UFT from engineering documents by means of natural language processing and machine learning (Fig. 4).



To arrange corpus set
To extract verbs
To grasp basic statistics of verbs
low frequency terms
mid frequency terms
Grammatical analysis
Concept clustering
To manually author lexical information
To build functional concept dictionary
an engineering document
increase
50 g
100 ℃
MI1 (increase; temperature)
MI2 (increase; weight)
To calculate mutual information between UFT and class of physical attribute
To make feature vector
To do SOM learning
To make concept cluster
loaded documents
analysis and term editing area
auxiliary area: e.g. displaying original text
a usual functional term
Functional Concept Dictionary
an essential functional concept

(a) Activity Chart of Building Functional Concept Dictionary (b) Concept Clustering (c) Functional Concept Mining Tool

Fig. 4. Overview of Functional Concept Mining

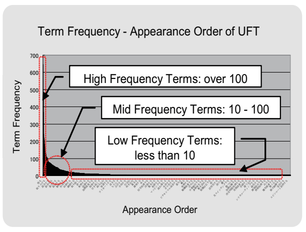
The basic flow of functional concept mining is shown in Fig. 4(a) as an activity chart in UML (Universal Modeling Language). According to this chart, the first step needs preparation of corpus, which means document collection to analyze for specific purpose, to extract UFT. After extracting UFT, we can grasp some statistical perspectives such as term frequency, term distribution, etc.

Our approach divide all terms into three groups: high-frequency terms that the number of occurrence are over 100, mid-frequency terms that those are over 10 and below 100, and low-frequency terms that those are below 10. We do not focus on high-frequency terms because of being too general verbs like *do* or *be*, but focus on mid-frequency and low-frequency terms in order to make clusters and analyze grammatical structure, respectively. After primary classification of UFTs by computer, they should be arranged manually, and stored into a functional concept dictionary.

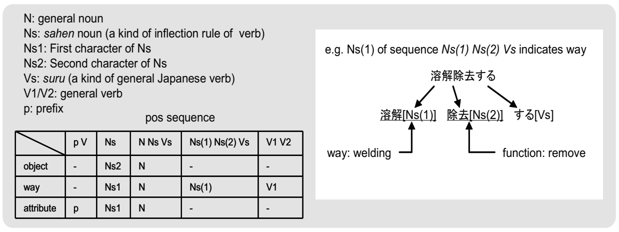
For realizing this mining framework, we created functional concept mining tool as shown in Fig. 4(c). The main functionalities of the tool are (1) loading of corpus, (2) statistical analysis, (3) grouping UFTs by SOM, and (4) the specific GUI for manual editing.

3.2 Evaluation of Functional Concept Mining

While omitting detail explanation because of not main issue of this paper, we briefly summarize our experiment of functional concept mining. First, we arranged corpus set, which involves about 1,700 documents, by collecting engineering documents from the web site that provides technical information such as *Standard Technology Collection Site* [2] presented by Japan Science and Technology Agency (hereinafter JST). As a result of statistical analysis of the corpus, the number of distinct functional verbs reaches a total of 2,097 (without redundancy). Among them, the number of mid-frequency verbs is 343, and that of low-frequency ones is 1,724. As for the term statistics, it is conformed that the distribution of UFT follows Zipf's law (Fig. 5(a))[22].



(a) Basic Statistics of Verbs


N: general noun
Ns: *sahen* noun (a kind of inflection rule of verb)
Ns1: First character of Ns
Ns2: Second character of Ns
Vs: *suru* (a kind of general Japanese verb)
V1/V2: general verb
p: prefix

pos sequence

	p V	Ns	N Ns Vs	Ns(1) Ns(2) Vs	V1 V2
object	-	Ns2	N	-	-
way	-	Ns1	N	Ns(1)	V1
attribute	p	Ns1	N	-	-

(b) Grammatical Analysis

Fig. 5. Result of Functional Concept Mining

To classify mid-frequency verbs, we applied SOM (Self-Organizing Map)[11] with setting feature vectors whose values are *mutual information* between a verb and physical concept such as temperature, weight, etc, around the verb (Fig. 4(b)). The experiment of clustering through SOM resulted in about 65 % which is the average percent of accuracy. The accuracy rate can be improved by preliminary reduction of noise terms, i.e. not functional terms. This result shows

[2] http://www.jpo.go.jp/shiryou/s_sonota/hyoujun_gijutsu.htm

effectiveness of this approach to reducing the cost of building language resource to map UFT to EFC, even if it is necessary for the manual authoring finally.

Taking into low-frequency terms from the viewpoint of grammatical structure, almost such terms are compounded of smaller linguistic pieces that are semantically independent, and each of them implies whether it is a functional concept or not. For example, Japanese verb *youkai-jokyo-suru* consists of three pieces; *youkai* means *weld* which is the way to achieve removing in this context, *jokyo* means *remove* which is the main concept in this verb, and *suru* is a general verb like *do* in English (Fig. 5(b)).

Based on the result of this experiment, we could build a functional concept dictionary which involves about 1,000 UFTs and 150 EFCs. In this dictionary, we adopted proper representation for UFT and EFC, that is, feature structure based on HPSG (Head-Driven Phrase Structure Grammar)[15] as shown in Fig. 6. Such kind of structure enables us to semantically represent vocabulary, and to flexibly append any additional information.

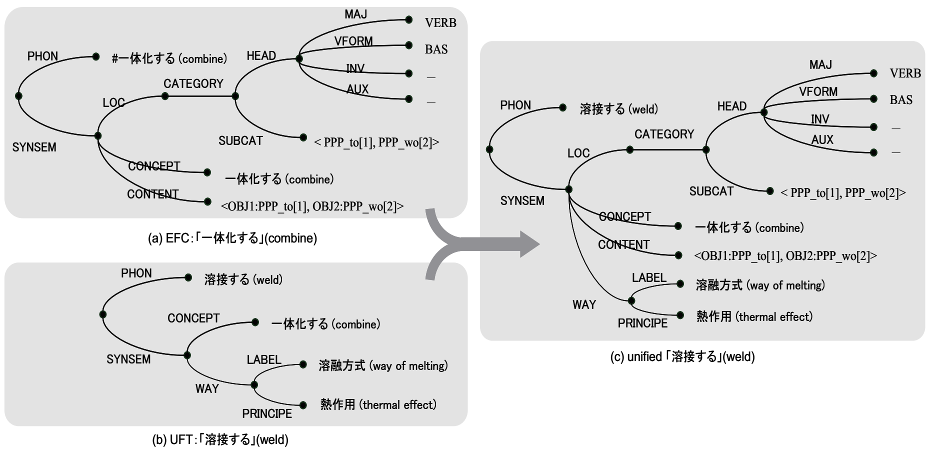


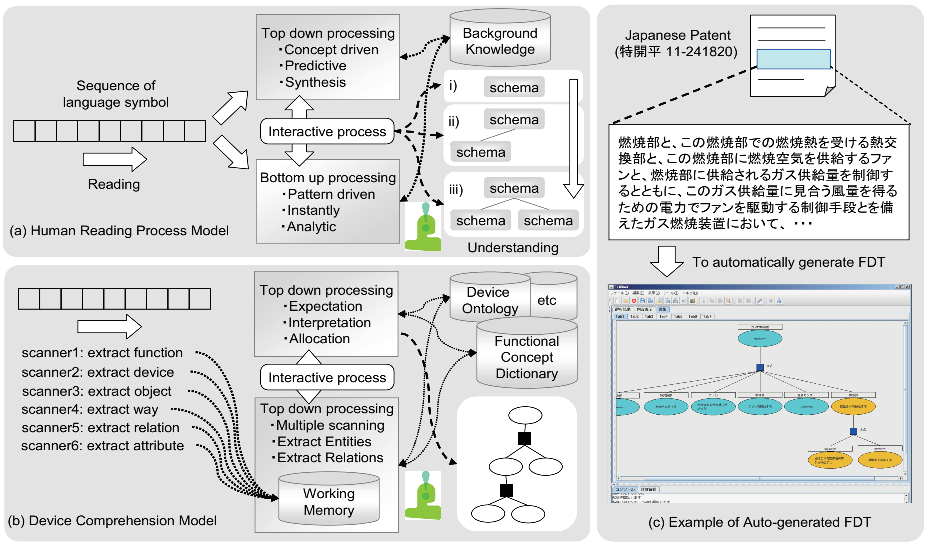
Fig. 6. Feature Structure of Functional Terms

Fig. 6(a) illustrates the feature structure of *combine* as a EFC in graph form. By contrast, Fig. 6(b) illustrates the feature structure of *weld* as a UFT in the same form, which includes way and principle structure: *way of fusion* and *thermal effect*. As a result of unification of *combine* and *weld*, the whole structure becomes like Fig. 6(c).

3.3 Mining Module of Way Knowledge

When an engineering document involves ways of function achievement as natural language description, a knowledge author can grasp the functional structure, and describe as a function decomposition tree. But it is not easy for a computer to do so in the same manner. Any traditional parsing technique does not have enough capability to reveal the functional structure of a device from texts. So we contrived the device comprehension model, which is called *DeCoMo* (Device Comprehension Model), that allows a computer to catch the functional structure of a device from texts, and build the function decomposition trees.

DeCoMo is designed based on theories of cognitive science for reading comprehension of a person, such as schema theory[1, 7]. Those theories indicate that when a person read a text, top-down and bottom-up process occur simultaneously and work interactively during the reading process. The top-down process is an activity such that the reader recalls the specific schema he/she has built in his/her minds ever before, and the bottom-up process is an activity such that the reader extracts linguistic pieces from the text, respectively (Fig. 7(a)).



燃焼部と、この燃焼部での燃焼熱を受ける熱交換部と、この燃焼部に燃焼空気を供給するファンと、燃焼部に供給されるガス供給量を制御するとともに、このガス供給量に見合う風量を得るための電力でファンを駆動する制御手段とを備えたガス燃焼装置において、・・・

Fig. 7. Overview of DeCoMo (Device Comprehension Model)

DeCoMo works like the schema theory mentioned above, that is, with interactivity between top-down and bottom-up process (Fig. 7(b)). The top-down process of DeCoMo consists of three tasks: expectation, interpretation, and allocation. What the expectation task expects are presumable entities corresponding to the concept of functional ontology such as agent, object, way, function, etc. The interpretation task interprets whether those are valid or not. The last task allocation put the confirmed element into working memory.

By contrast, the bottom-up process of DeCoMo employs a specific mechanism that multiple scanners scan the same text stream in parallel, extracting entities and relationships, which rely on functional ontology, between them. Finally, DeCoMo outputs a FDT as long as it could recognize.

Implementing DeCoMo as way knowledge mining tool, it can automatically construct a function decomposition tree, or portion of it, even if the description lacks of enough information. Fig. 7(c) shows an example that the tool built the functional decomposition tree by reading a patent which describes an invention of gas burner.

3.4 Evaluation of Way Knowledge Mining

For preliminary test of way knowledge mining, we arbitrarily prepared 7,000 Japanese patent documents in XML format, and applied way knowledge mining tool to extract way knowledge in terms of selected 15 EFCs. The extracted way knowledge by the tool resulted in 73 way instances, that is, about 1 % of the whole documents. The extracted examples are shown in Fig. 8(a).

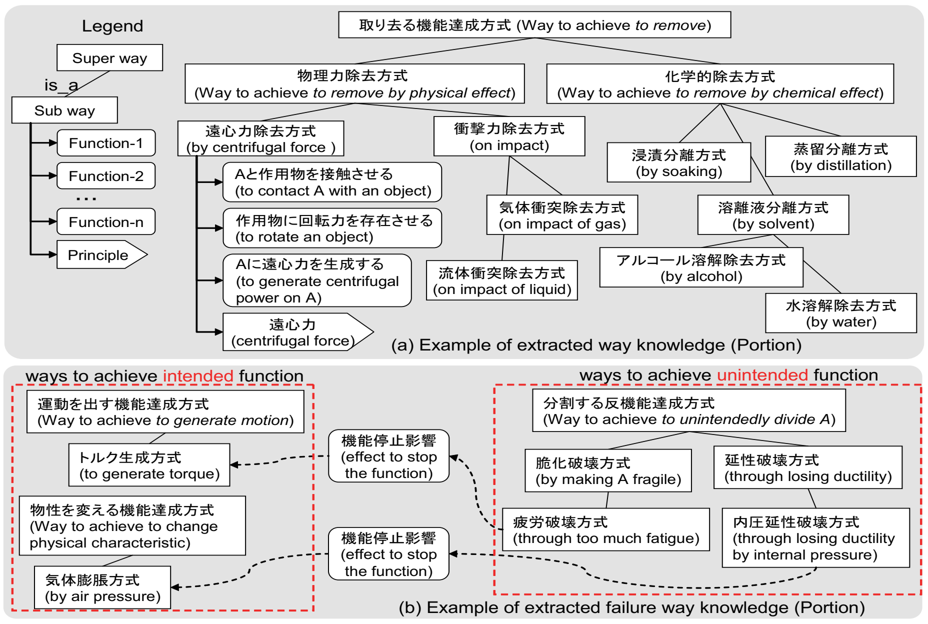

Legend
Super way
is_a
Sub way
Function-1
Function-2
Function-n
Principle
取り去る機能達成方式 (Way to achieve *to remove*)
物理力除去方式 (Way to achieve *to remove by physical effect*)
化学的除去方式 (Way to achieve *to remove by chemical effect*)
遠心力除去方式 (by centrifugal force)
衝撃力除去方式 (on impact)
浸漬分離方式 (by soaking)
蒸留分離方式 (by distillation)
Aと作用物を接触させる (to contact A with an object)
作用物に回転力を存在させる (to rotate an object)
気体衝突除去方式 (on impact of gas)
溶離液分離方式 (by solvent)
Aに遠心力を生成する (to generate centrifugal power on A)
流体衝突除去方式 (on impact of liquid)
アルコール溶解除去方式 (by alcohol)
水溶解除去方式 (by water)
遠心力 (centrifugal force)
(a) Example of extracted way knowledge (Portion)
ways to achieve intended function
ways to achieve unintended function
運動を出す機能達成方式 (Way to achieve *to generate motion*)
トルク生成方式 (to generate torque)
機能停止影響 (effect to stop the function)
分割する反機能達成方式 (Way to achieve *to unintendedly divide A*)
脆化破壊方式 (by making A fragile)
延性破壊方式 (through losing ductility)
物性を変える機能達成方式 (Way to achieve to change physical characteristic)
疲労破壊方式 (through too much fatigue)
内圧延性破壊方式 (through losing ductility by internal pressure)
機能停止影響 (effect to stop the function)
気体膨脹方式 (by air pressure)
(b) Example of extracted failure way knowledge (Portion)

Fig. 8. Example of Extracted Way Knowledge

The processing time was within 2 minutes for 1,000 documents, including the time of I/O (Input/Output) of XML files. After raw extraction of way instances, we edited them by way knowledge editor into complete form of way knowledge with generalization and *is_a* hierarchical positioning. The manual editing time for each of them was about 10 minutes in average.

Meanwhile, failure way knowledge that means the way to achieve *unintended function* can also be extracted by modification of DeCoMo engine. We also had preliminary test of extracting failure way knowledge from several home pages in *Failure Knowledge Database*[3] (by JST), and picked 18 ways. Some pieces of failure way knowledge are connected to corresponding ways of intended functions as shown in Fig. 8(b).

Seeing this result, the extracted samples seem to be very few, but we consider that it is significant enough because the experiment was under the restricted UFT and EFC, and extracting capability of the tool can be developed further by sophisticating the mechanism and resources. We will build over several hundreds of way knowledge in the near future.

4 Implementation of OntoGear

4.1 Architecture of OntoGear

OntoGear that embodies functional modeling methodology based on ontology engineering is implemented on xfy[4] [19] that JustSystems Corp.[5] developed, and Java[6] programming language as shown in Fig. 9 (a).

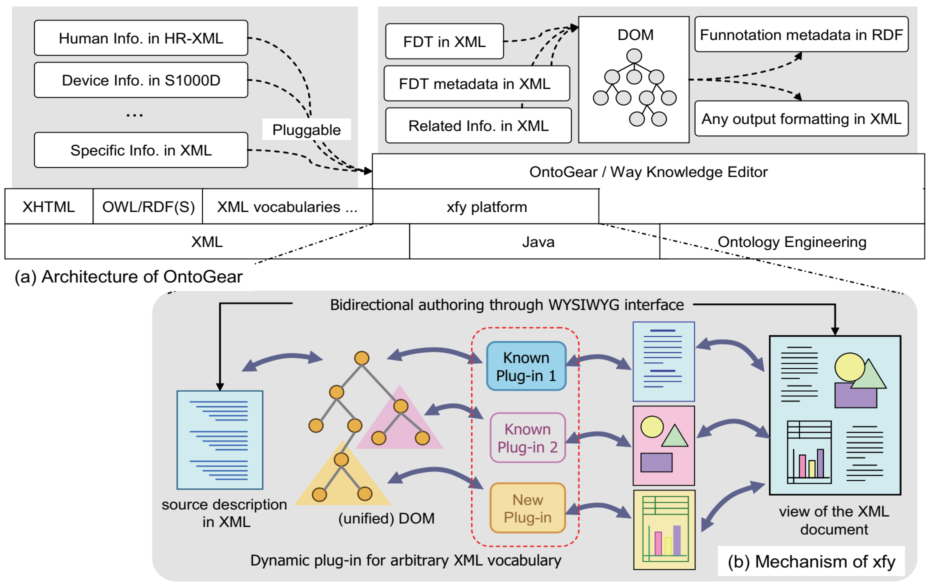


Fig. 9. Architecture of OntoGear and Mechanism of xfy

Referring to xfy, it is an XML (eXtensible Markup Language) application platform that has three main features: extensibility, flexibility and user-friendliness.

[3] http://shippai.jst.go.jp/en/Search
[4] http://www.xfy.com
[5] http://www.justsystems.com
[6] http://www.sun.com/java/

The extensibility is featured through aid for combining any set of custom XML vocabularies, the flexibility is through arbitrary composition of intrinsically independent DOM trees, and the user-friendliness is through GUI (Graphical User Interface) based interactive editing of XML documents (Fig. 9 (b)).

Because OntoGear inherits the characteristic from xfy, it can be extended flexibly according to application, and interlinked easily to other XML centric systems. Way knowledge editor also has as same architecture and features as OntoGear.

4.2 Prototype System

We developed OntoGear according to the architecture described in the previous section. OntoGear is composed of some XML vocabularies that specify each components of FDT and its related information, and written in the specific script language *xvcd*, which is similar to XSLT[7] (XML Stylesheet Language Transformations) but not same due to some additional functionalities, that xfy platform supplies. If there needs more complex processing rather than xvcd can handle, it can be written in Java as a plug-in module, and plugged into OntoGear. The module of managing functional concept dictionary is one of the plug-ins such like that.

For evaluation use in any companies and universities, we are distributing OntoGear as a prototype system from 2007 in Japan. Although the current distribution of OntoGear does not include the functionality for Funnotation, it develops practical enough to author FDTs and share them with reflecting new theory of ontology engineering, e.g. integration model of function and failure function[12]. While OntoGear is not deployed into real world engineering activity, we gain some positive comments; According to a patent manager of our university, it is helpful to clarify the invention's contents of patents by FDT modeling; According to an architecture company, it becomes easy to handle a lot of failure reports by FDT modeling for them rather than plain description in natural language.

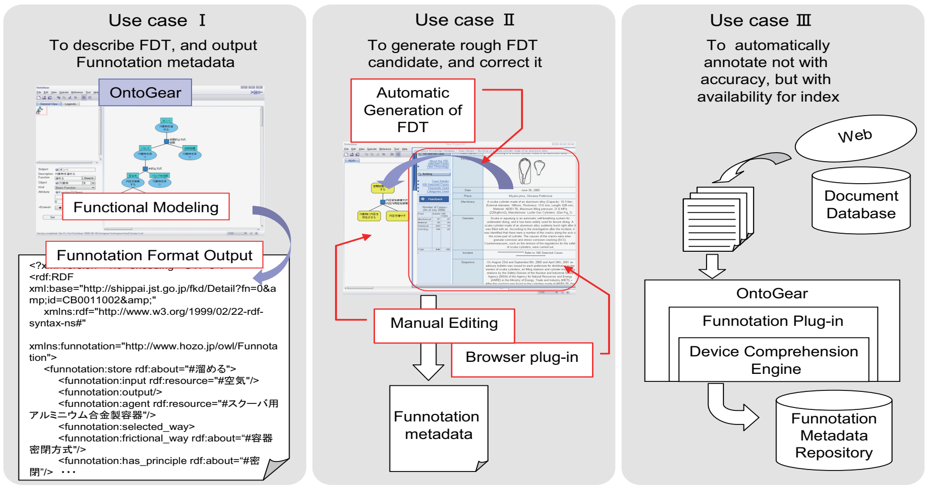
For Funnotation task, we created a special version of OntoGear with several experimental plug-ins, and will show how to use it in the following section.

5 Use Cases Using OntoGear

5.1 Funnotation Task by OntoGear

Funnotation framework by OntoGear enables annotators to attach Funnotation metadata for engineering documents according to the use cases: 1) when describing an FDT in the design phase, 2) when annotating while reading an engineering document, 3) when automatically creating annotation for vast amount of documents, as shown in Fig. 10.

[7] http://www.w3.org/TR/xslt


Use case Ⅰ
To describe FDT, and output Funnotation metadata
OntoGear
Functional Modeling
Funnotation Format Output
Use case Ⅱ
To generate rough FDT candidate, and correct it
Automatic Generation of FDT
Manual Editing
Browser plug-in
Funnotation metadata
Use case Ⅲ
To automatically annotate not with accuracy, but with availability for index
Web
Document Database
OntoGear
Funnotation Plug-in
Device Comprehension Engine
Funnotation Metadata Repository

```
<?x
<rdf:RDF
xml:base="http://shippai.jst.go.jp/fkd/Detail?fn=0&a
mp;id=CB0011002&amp;"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#"

xmlns:funnotation="http://www.hozo.jp/owl/Funnota
tion">
    <funnotation:store rdf:about="#溜める">
        <funnotation:input rdf:resource="#空気"/>
        <funnotation:output/>
        <funnotation:agent rdf:resource="#スクーバ用
アルミニウム合金製容器"/>
        <funnotation:selected_way>
        <funnotation:frictional_way rdf:about="#容器
密閉方式"/>
        <funnotation:has_principle rdf:about="#密
閉"/>  ・・・
```

Fig. 10. Use Cases of Funnotation by OntoGear

The first use case might occur in concept design phase. In this case, OntoGear can support designers to describe the FDT of an artifact by designating functional concept and way knowledge through the functional concept dictionary and way knowledge database, and to automatically output Funnotation metadata of the FDT, which can be attached to any related documents.

In the second use case, extended OntoGear with DeCoMo engine and browser plug-in helps users to edit the FDT as Funnotation on browsing an engineering document by automatic generation of FDT from the document.

The last one is as same as the second one except for manual editing, and may be used for metadata indexing for semantic search engine. We will investigate that case in the near future.

5.2 OntoGear/FPoD: A Failure Prevention System

A function is implicitly regarded as a conceptualization of how a device *normally* works, whereas a failure can be regarded as *unintended function* according to our functional ontology[12]. All designers want to avoid failure of devices that they design in the design phase as much as possible. While prevention of failure is becoming more important for them, the circumstance of today's complex development urges them to sophisticatedly handle failure knowledge. To support well-handling of failure, we developed a prototype of failure prevention system, so-called OntoGear/FPoD (Failure Prevention on Demand), which enables designers to be aware of failures before occurring.

Left illustration of Fig. 11 shows an overview of OntoGear/FPoD system that is based on our Funnotation metadata framework. This system needs preparation of failure way knowledge database and failure Funnotation metadata repository that are related to failure reports in failure knowledge database (i.e.

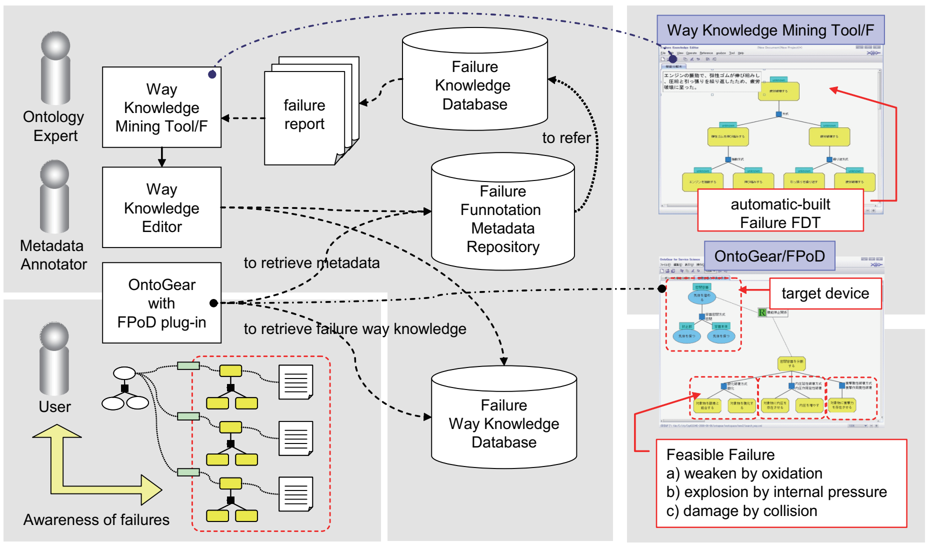

Ontology Expert
Metadata Annotator
User
Way Knowledge Mining Tool/F
Way Knowledge Editor
OntoGear with FPoD plug-in
failure report
Failure Knowledge Database
to refer
Failure Funnotation Metadata Repository
to retrieve metadata
to retrieve failure way knowledge
Failure Way Knowledge Database
Awareness of failures
Way Knowledge Mining Tool/F
automatic-built Failure FDT
OntoGear/FPoD
target device
Feasible Failure
a) weaken by oxidation
b) explosion by internal pressure
c) damage by collision

Fig. 11. Overview of OntoGear/FPoD system

http://shippai.jst.go.jp) by using our annotation tools. After completion of the database and repository, users can access to them through OntoGear/FPoD, and gain retrieved results that indicate possible failures interlinked to the original incidents.

Two sample snapshots are shown in right-hand side of Fig. 11. The uppermost picture is way knowledge mining tool that is customized to have extended functionality to automatically build *failure FDT* as shown in the picture. The bottom one is OntoGear/FPoD, displaying some predictive failure cases of explosion of an airtight container due to oxidization, gas expansion, and collision of an object.

6 Related Work

Annotation tools for Semantic Web vary from a simple editor to an advanced tool with automatic annotation engine. Amaya[8] provided by W3C as open source software is a Web editor/browser which includes a collaborative annotation functionality, and it can attach general metadata annotation such as comments, notes, remarks, etc, to Web documents. Our interest is not in general use without ontology, but in engineering one with ontology.

Some tools, e.g. OntoMat-Annotizer or Swoogle[3], claim that their annotation capabilities rely on ontology. OntoMat enables annotators to create OWL-markups to his/her webpage, but the current version does not include annotation support functionality such as information extraction. On the other hand, Swoogle

[8] http://www.w3c.org/Amaya

is rather comprehensive system for semantic web search with over 10,000 ontologies, containing metadata engine. As mentioned in section 3, OntoGear system can help annotators through semi-automatic annotation capability, and develop to the specific application such as FPoD for engineering use.

From the viewpoint of language resources related to ontology, there exist Dublin Core[9], OMV (Ontology Metadata Vocabulary)[10], etc. Those metadata cover general documents; The Dublin Core Metadata provides simple but effective metadata vocabularies such as title, author, date, etc, while OMV refers to vocabularies for a metadata standard, distinguishing between Ontology Base and Ontology Document; concepturization and realization. OntoGear has a functional concept dictionary to map UFT to EFC for engineering documents. Moreover, OntoGear plans to equip larger way knowledge database rather than a sample one.

7 Conclusions

In this paper, we proposed semi-automatic annotation framework for Funnotation, and developed some systems to embody the framework: a) functional concept mining tool to build a functional concept dictionary in advance, b) way knowledge mining tool to effectively extract way knowledge from technical documents, c) Way Knowledge Editor to organize way knowledge, and d) OntoGear for functional modeling and Funnotating.

Using functional concept mining tool, we built a functional concept dictionary which stores about 1,000 UFTs and 150 EFCs efficiently. Applying way knowledge mining tool to patent documents and failure reports, we achieved to build small way knowledge database. Based on the resources, we showed how OntoGear works to annotate functional metadata, and how OntoGear/FPoD supports designers to be aware of possible failure.

Current OntoGear system is in Japanese version. However, our approach mentioned in this paper does not depend on language except for natural language processing. We plan to investigate English version of OntoGear system.

References

[1] Carrel, P.L. & Joan C. Eisterhold: Schema theory and ESL Reading pedagogy. TESOL Quarterly. Vol.17 No.4 (1983).

[2] Chandrasekaran, B., Goel, A. K., Iwasaki, Y.: "Functional representation as design rationale" *Computer*, 26(1), pp.48-56 (1993).

[3] Ding, L., Finin, T., Joshi A., Pan R., Cost, R.S., Peng Y., Reddivari P., Doshi V., Sachs J., Swoogle: a search and metadata engine for the semantic web, *Proc. of the thirteenth ACM international conference on Information and knowledge management*, (2004).

[9] http://www.ietf.org/rfc/rfc5013

[10] http://ontoware.org/projects/omv

[4] M. Erdmann, A. Maedche, H. P. Schnurr, and Steffen Staab. From Manual to Semi-automatic Semantic Annotation: About Ontology-based Text Annotation Tools. In P. Buitelaar: Proc. of the COLING 2000 Workshop on Semantic Annotation and Intelligent Content (2000).

[5] Siegfried Handschuh , Steffen Staab, Fabio Ciravegna: S-CREAM – Semi-automatic CREAtion of Metadata, Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web, Springer Berlin/Heidelberg, pp.165-184 (2002).

[6] Hubka, V., Eder, W.E.: *Theory of Technical Systems*, Springer Berlin (1988).

[7] Kintsch, W.: The role of knowledge in discourse comprehension construction-integration model. Psychological Review, pp.163-182 (1988).

[8] Kitamura, Y., & Mizoguchi, R.: Ontology-based systematization of functional knowledge. *Journal of Engineering Design, 15(4)*, pp.327-351 (2004a).

[9] Kitamura, Y., Kashiwase, M., Fuse, M., Mizoguchi, R.: "Deployment of an Ontological Framework of Functional Design Knowledge", *Advanced Engineering Informatics*, 18(2), pp.115-127 (2004).

[10] Kitamura, Y., Washio, N., Koji, Y., Sasajima, M., Takafuji, S., Mizoguchi, R.: An Ontology-Based Annotation Framework for Representing the Functionality of Engineering Devices, In *Proc. of ASME IDETC CIE* (2006).

[11] T. Kohonen, Self-Organizing Maps, Springer Series in Information Sciences, vol. 30, Springer, Heidelberg, 1st ed., 1995; 2nd., 1997.

[12] Koji, Y., Kitamura, Y., Mizoguchi, R.: Towards Modeling Design Rationale of Supplementary Functions in Conceptual Design, In *Proc. of Tools and Methods of Competitive Engineering - TMCE 2004*, pp.117-130 (2004).

[13] Lee, J.: Design Rationale Systems: Understanding the Issues., *IEEE Expert*, Vol. 12, No. 3, pp. 78-85 (1997).

[14] Pahl, G., and Beitz, W.: *Engineering Design - a Systematic Approach*, The Design Council (1998).

[15] Pollard, C., Sag, I. A.: *Head-Driven Phrase Structure Grammar*, The University of Chicago Press (1994)

[16] S. Staab, A. Maedche, and S. Handschuh. An annotation framework for the semantic web. In *Proc. of the First Workshop on Multimedia Annotation*, (2001)

[17] Stone, R. B., Chakrabarti, (eds.): Special Issues: Engineering applications of representations of function, *Journal of Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM), 19(2 and 3)* (2005).

[18] Sunagawa, E., Kozaki, K., Kitamura, Y., Mizoguchi, R.: "A Framework for Organizing Role Concepts in Ontology Development Tool: Hozo", Papers from the AAAI Fall Symposium "Roles, an Interdisciplinary Perspective" AAAI TR FS-05-08, pp.136-143 (2005).

[19] Sunao Takafuji: Semantic Document Processing with xfy Technology, Workshop on the Semantic Computing Initiative (SeC2005), WWW2005, CD-ROM (2005)

[20] Umeda, Y., Ishii, M., Yoshioka, M. Shimmura, Y., and Tomiyama, T.: Supporting conceptual design based on the function-behavior-state modeler, *Artificial Intelligence for Engineering Design, Analysis and manufacturing* 10, pp.275-288 (1996).

[21] Maria Vargas-Vera , Enrico Motta , John Domingue , Mattia Lanzoni , Arthur Stutt , Fabio Ciravegna, MnM: Ontology Driven Semi-automatic and Automatic Support for Semantic Markup, Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, p.379-391 (2002)

[22] Zipf, G.K.: Selected studies of the principle of relative frequency in language. Cambridge, Mass.: Harvard University Press. (1932).