A Methodology for Acquisition of Software Component Attribute Ontology

Liu Jin, He Keqing, Li Bing, Chen Hao, Peng Liang
The State Key Laboratory of Software Engineering, Wuhan University, Wuhan, China, 430072
xzliu@whu.edu.cn

Abstract
Although value of ontologies is widely admitted in web-based information management, acquisition of ontologies is still the bottleneck in developing applications based on them. Rapid acquisition of high-quality software component attribute ontology (SCAO) is desired in some infrastructural platforms that intend to facilitate weaving Web-based information systems based on software component. After analyzing existing methodologies, we advance a new methodology for acquiring SCAO. The purpose of this methodology is that both rapid acquiring ontologies and quality control of ontologies can be achieved. The ontology acquisition process, the algorithm of metaproperty collision inspection, and a tool based on this algorithm are discussed in our methodology. The process of SCAO acquisition is hereby described in detail, which indicates that our methodology is practical to rapidly acquire high-quality ontologies.

1. Introduction
With the development of the Internet technique, in recent years applications based on ontologies have become common in Web-based information management. Gruber defines ontology as a specification of a conceptualization[1]. For their benefits to providing consistent semantics and sharing common conceptual knowledge in specific Universe of Discourse(UoD), ontologies do great help in many aspects in the Internet, such as overcoming semantic heterogeneity among Web-based information systems [2], promoting common understanding between people or electronic agents, and facilitating system integration[3].

Acquisition of SCAO plays a crucial role in developing WHCRP(WuHan Component Repository Platform)—a project in SKLSE(The State Key Laboratory of Software Engineering, Wuhan University). WHCRP aims to be an infrastructural platform for weaving information systems based on software component in the Internet. By providing service of registering and managing metadata of software components, and service of dynamic binding of software components, WHCRP intends to promoting rapid construction of Web-based software systems in the Internet. The ontological commitment of SCAO explains that in what terms should we think about reuse of software component. Reuse-oriented SCAO explicitly defines semantics of software component attributes and relationships between these attributes. Since query based on SCAO effectively restricts the size of retrieval result, the validity of query can obviously be improved.

Acquisition of ontologies is always the bottleneck in ontology applications. Not long ago many people still thought the process of acquiring ontologies for use in information systems an arcane art form [4]. The main reason for this puzzle is that most people don’t know exactly how to transform scrappy knowledge fragments in domain experts’ mind into formal conceptualization of domain knowledge.

The existing methodology of ontology acquisition can mainly be reduced to two categories. One is engineering methodology, which is mainly through refining competency questions to form concept hierarchical structure of ontologies [3,5,6]. The other category is OntoClean methodology advocated by Nicola Guarino and Christopher Welty [4,7,8,9,10], which is chiefly through assigning meta-properties to concept nodes of ontologies to validate taxonomy ontology by exposing inappropriate and inconsistent modeling decisions. Either methodology has its virtues and deficiencies.

The result of comparing and analyzing these two methodologies indicates that there are no evident
conflicts between them. The chief difference between these two methodologies is that they emphasize different aspects in the process of acquiring ontology. Therefore this paper brings forward a new practical methodology for acquiring SCAO by combining these two existing methodologies. With the support of corresponding tools, our methodology can be adopted to rapidly acquire SCAO. Apart from that, it also provides an effective technique to evaluate ontological decisions. Thus quality of SCAO can be guaranteed and its reusability can be improved. It is clear that our methodology is also practical in acquisition of other ontologies.

The rest of this paper is structured as follows: Section 2 advances our methodology for acquiring SCAO on the basis of analysis of two existing methodologies. The ontology acquisition process of this methodology is discussed in this section. Section 3 presents the algorithm of metaproperty collision inspection and a tool based on this algorithm. Then section 4 illustrates the process of acquiring SCAO with our methodology. Finally section 5 concludes this paper.

2. Methodology for Acquiring SCAO

2.1 Mergence of Methodologies

The prominent character of the engineering methodology for ontology acquisition is that it can rapidly capture informal domain knowledge and transform them to formal ontologies. The engineering methodology inherits its virtues from traditional methodologies in knowledge engineering domain because all of them adopt the technique of competency question. For this reason, the technique of competency question contributes to the achievement of this methodology. In spite of this powerful technique, the engineering methodology does not provide a means to strictly evaluate quality of ontologies. Since the function of a ontology is to illuminate semantics of information resources [2] and share a common understanding, a clear high-quality ontology is usually indispensable for its applications. It has proved that high-quality ontology can improve performance of applications based on it [11].

Unlike the engineering methodology, OntoClean is an evaluation methodology. Several philosophical notions are introduced and defined as metaproperties in this methodology, i.e., Rigidity, Identity, Unity, Dependency, Identity Condition(IC), and Unity Condition(UC) [7,8,9,10]. According to these notions, metaproperties are defined, i.e., R for Rigidity, I for Identity, U for Unity, D for Dependency and O for supplying IC. Each metaproperty has different meanings when it is prefixed with symbols respectively, i.e., ‾, +, −, and −. By introducing this technique, an effective means of quality evaluation to ontologies is provided. Constraints and assumptions that exert on ontology are refined as follows [7,8,9,10].

\[
\Phi^R \text{ can't subsume } \Psi^R \quad (1)
\]
\[
\Phi^I \text{ can't subsume } \Psi^I \quad (2)
\]
\[
\Phi^U \text{ can't subsume } \Psi^U \quad (3)
\]
\[
\Phi^B \text{ can't subsume } \Psi^B \quad (4)
\]
\[
\Phi^B \text{ can't subsume } \Psi^B \quad (5)
\]

Properties with incompatible ICs/UCs are disjoint. (6)

The rest of this paper references them as estimation rules of metaproperty collision. In spite of that, this methodology does not explain how those initial important terms and the term hierarchy come from. More often than not, the difficulty is still how to obtain original domain knowledge.

Since there is no evident collision between these two preceding methodologies, it is highly possible to merge them into a new methodology. Gómez-Pérez has depicted the ontology life cycle [6], which indicates that acquisition and evaluation are two dominating activities. These two activities are mainly carried out in two phases, i.e., conceptualization and formalization. All of these indicate that either engineering methodology or OntoClean has grasped only one important aspect in the process of ontology acquisition, which makes for their half achievement. With competency questions, the engineering methodology can effectively capture domain knowledge and refine competency questions to form ontology. Compared with the former, OntoClean methodology emphasizes evaluating validity of ontological decisions. The target of our work is to merge these two methodologies into a new one so that both rapid capturing knowledge of software component attribute and effective estimation of the validity of SCAO can be achieved in our methodology. As a consequence, high-quality SCAO can be achieved.

2.2 Environment

Formal ontologies are described by ontology describing language. For different ontology languages, their ability of representing ontologies and reasoning varies. In the past few years, many ontology languages have been developed such as KIF,
Interlingua, OIL and so forth. Consider rapid development of ontology language and the open environment of WHCRP, the ontology language describing SCAO should readily accommodate to open circumstance in the Internet and comply with well-known international specifications for Web environment. On this account, our methodology adopts W3C’s Web Ontology Language (OWL) as ontology language to describe SCAO [12].

When it comes to tools that support the process of ontology acquisition, Protégé is a very influential ontology editor and knowledge-base editor developed by Stanford Medical Informatics (SMI) [13]. There are several reasons for our methodology to choose Protégé as ontology editor platform. To begin with, it is an open-source, Java tool that provides an extensible architecture for building customized knowledge-based applications. Thus it possible to add new features to protégé so that starting from scratch can be avoided for ontology acquisition. Apart from that, many innovational techniques of ontology application can be embodied in protégé in good time. OWL plug-in is not the unique example we have known.

2.3 Process of Ontology Acquisition

On the basis of the analyzing our methodology and its environment, the ontology acquisition process of this methodology is composed of five components, i.e., Competency Question List (CQL), Term Enumeration, Baseline Ontology, Metaproperty Validation, and Instance Definition. Figure 1 represents the whole process of ontology acquisition of our methodology.

The initial stage of the process is to produce CQL. Generally speaking, a competency question does not need to be a detailed one. The intention of a competency question is to make an effort to depict scrappy knowledge fragments from knowledge experts into informal questions. For a ontology, CQL ought to cover every aspect of its UoD, i.e., information that is deeply considered by domain experts. In order to facilitate the formation of competency questions, we introduce here the technique of Mind Maps [15]. Mind Maps enforces a spatial structure in the external representation, i.e., ideas are graphically connected via branches, which facilitates building an internal representation of an interest group’s shared space of ideas. As the artifact in this stage, CQL is refined in the following phases, i.e., Term Enumeration, Baseline Ontology, Metaproperty Validation, and Instance Definition.

![Process of ontology acquisition](image)

Figure 1. Process of ontology acquisition

The following phase is Term Enumeration. According to information provided by CQL, important terms are enumerated so that baseline ontology can be constructed. Existing ontologies also help to raise CQL. These ontologies relevant to the ontology to be built are analyzed to collect available information. Many important terms may be gained in this activity. A term is usually relevant to one or more competency questions, which indicates that information provided by this term contributes to answering relevant competency questions. A term classification box is induced into this methodology because terms collected from various sources need to be estimated until their roles are identified in a ontology.
After that, the third phase is to build baseline ontology. Based on information provided by CQL, enumerated terms are organized by OWL into a formal baseline ontology. In this phase, classes are defined on the basis of terms representing concepts. Concepts and subsumption relationship of these concepts make up of class hierarchy of baseline ontology. Following the definition of classes, properties of these classes and restrictions on these properties are also defined. In protégé OWL Plug-in, properties are named as slots and restrictions on these properties are named as facets.

Metaproperty validation is the fourth phase in the process and is crucial for ontology evaluation. Since it is not convenient to assign metaproperties to classes in protégé OWL plug-in, a specific instance of each class is defined to hold metaproperties assigned to this class. On the basis of the metaproperty collision rules, an automatic inspection tool is provided to validate OWL files labeled with metaproperties. The algorithm that is implemented in this tool will be discussed later in this section. First, a subset of the baseline ontology is automatically evaluated, i.e., classes with Rigidity metaproperty and the subsumption relationship between them. Then according to the result, domain experts can reconsider their design and manually rectify positions of these backbone classes and metaproperties of these classes in class hierarchy to form backbone ontology. After that, other classes are joined in, validated, and rectified in this manner. As a result, a clean ontology is established.

The last phase in the process is to define instances of classes. In general, classes define types for instances and instances are things or facts that exists in real world. CQL also provides necessary information for definition of instances.

In practice, the whole process is an iterative process so that corresponding artifact will be redressed manually by domain experts if any discordance is inspected in the process. This is especially true when it comes to CQL because it contains necessary original information for decision-making in other phases.

3. Algorithm of Metaproperty Collision Inspection

For the tool of metaproperty collision inspection mentioned above, the function of the algorithm employed in this tool is to automatically inspect metaproperty collision according to metaproperty collision rules. Since OWL and protégé OWL plug-in are involved in the process of our methodology, the algorithm ought to extract necessary information from OWL files for inspecting metaproperty collision. What we take into consider is whether the metaproperties of a class collide with those of other classes that subsume it. The definition of elements in a XML document has a style with strictly closed pairs. Thus it is easy to get such information by reading XML document in a consequent manner.

Our algorithm chooses the technique of Simple API for XML (SAX) to process OWL files, which works on a stream of data adopting event-based model [14]. In this way, document analysis can get started immediately, rather than having to wait for loading all the data and the hierarchy of a document. This is a huge advantage when it comes to exceptionally large OWL documents. If processing XML files in a similar way like DOM or Jena, parsing and loading the entire document may be slow and resource-intensive.

![Figure 2. Information extraction for metaproperty collision inspection](image)
In order to extract information for metaproperty collision inspection, it is necessary to identify features of document elements. In this algorithm, a instance of each class is intentionally used to hold metaproperties of this class. Particularly for OWL files produced by protégé OWL Plug-in, OWL document elements that hold metaproperties of classes may like this:

```
<j.0:B1 rdf:id="TRY_Instance_1"
    j.0:MetaProperty="+O+U-D+R"/>
```

And a typical OWL document element that represents a class and its superclass is shown as follows:

```
<owl:Class rdf:id="B1">
    <rdfs:subClassOf>
        <owl:Class rdf:about="#A"/>
    </rdfs:subClassOf>
</owl:Class>
```

Figure 2 depicts the process of information extraction that is necessary for metaproperty collision inspection of an OWL file.

A 4-tuple \((C, ISC, MC, MISC)\) is introduced here to represent information extracted from an OWL file, where \(C\) is a Class, \(ISC\) is the immediate superclass of \(C\), \(MC\) is the metaproperties of \(C\), and \(MISC\) is the metaproperties of \(MC\). The 4-tuple \(T\) deals with all classes in a ontology. For the root node of a ontology, the value of \(ISC\) is represented by \(NULL\) which indicates that there is not immediate superclass for a root node in this ontology.

The algorithm of Metaproperty Collision Inspection is defined as follows.

Algorithm: Given a ontology with metaproperties, inspect metaproperty collision on the basis of metaproperty collision rule(1)–(5).

Open an OWL file;

While not end of OWL document {
    Identify an OWL document element \(E\);
    IF Identify \(E\) is a Class {
        Substitute \(E\) for \(C\) and record name of \(C\);
        Identify immediate superclass of \(C\) and record its name;
    } Else
    IF Identify \(E\) is a Class with metaproperties {
        Substitute \(E\) for \(K\) and record name of \(K\);
        Identify and record metaproperties of \(K\):
    }
}

Synthesize information into the 4-tuples \(T(C, ISC, MC, MISC)\)

While not every member of \(T\) is inspected {
    select a member \(t\) of \(T\);
    For \((I=1; I<=5; I++)\){
        IF \(t\) (MC) and \(t\) (MISC) violate Rule\((i)\)
            Print \(t\) and violation information of Rule\((i)\);
    }
}

Figure 3 shows the tool for metaproperty collision inspection with this algorithm.
4. Acquisition of SCAO

With our methodology, SCAO has been acquired and is now employed in WHCRP. According to the process of this methodology, the first step of acquiring SCAO is to make the scope and domain of SCAO explicit. In WHCRP, the role of SCAO is to capture essential information elements for depicting software components and promoting management and usage of metadata of software components. For WHCRP, query mechanism based on SCAO will be established to help reduce the size of fetched software component set. Software components that are needed may ultimately be chosen from this refined set.

Following that, competency questions that comprise more detailed information are put forward. Each competency question strives to capture one side of UoD of SCAO. These competency questions come together to form CQL and try to cover all main aspects of SCAO. A Competency questions are usually informal and not necessary to be detailed ones. A typical competency question about SCAO may like this: “What’s the running environment of this software component?”

The next step is to enumerate important terms that are used to build baseline SCAO. CLQ as a criterion provides indispensable information for term enumeration. Take the competency question mentioned above as an instance, enumerated terms based on it may be OS, Required Hardware, Distributed Object Standard, DB, Pre-requisite Software and so forth. At this time, terms are simply classified and deposited in Term Classification Box because roles and corresponding competency questions of these terms can not be determined at once. Since there are few existing ontologies relevant to software component attribute, reuse existing ontology is not considered in this acquisition process.

Form then on, terms are organized into baseline SCAO by hand on the basis of information of CQL. Subsumption relationship between classes is the primary relationship in class hierarchy of taxonomy ontology like SCAO. Thereafter metaproperties are assigned to each class and our automatic validating tool is used to validate backbone classes. As an illustration, term System Environment subsumes term Structure in the initial class hierarchy. The result of inspection indicates that there is a metaproperty collision between System Environment with metaproperties (+O+U+D+R) and Structure with metaproperties (+O~U+D+R). According to rule (3), classes with metaproperty Unity (+U) cannot subsume classes with metaproperty non-Unity (-U). Hence we rectify the position of Structure and make Structure subsumed by root node Software Component Attribute. In this manner, positions of backbone classes are reorganized to form backbone SCAO. Following backbone SCAO, other classes are added to backbone SCAO and validated with this tool. Their position may also be rectified according to inspection result.

![Diagram of Software Component Attribute Ontology (SCAO)](image)

Figure 4. A Segment of Software Component Attribute Ontology (SCAO)
The final phase of this process is to define instances of classes. For SCAO, end nodes of the class hierarchy comprise all necessary features that describe software component. Instances of end node classes are values of these features. In practice, features of a software component may not cover all features that SCAO provides but may be enough to tell how to identify this software component in WHCRP.

Figure 4 depicts a segment of the final SCAO. In practice, acquisition of SCAO is an iterative process. Discordances and fallacies may be found with the help of our tool and rectified by domain experts during this process. It is especially true when it comes to competency questions. While competency questions are modified, corresponding artifices should also be rectified.

5. Conclusion

Following analysis of existing methodology of ontology acquisition and ontology life cycle, this paper puts forward a new methodology for acquiring SCAO. The SCAO that is built in accordance with our methodology has been employed in WHCRP, which intends to provide service for facilitating rapid weaving information systems based on software component in the Internet environment. The ontology acquisition process and the algorithm of metaproperty collision inspection are piths in our methodology. The process of SCAO acquisition itself illustrates that both rapid acquiring SCAO and quality control of SCAO can be achieved with this methodology. That’s because fragments of domain knowledge can be captured effectively with CQL while technique of metaproperty provides effectual support for ontology evaluation. Although there are some proposals of automatic synthesis of existing ontologies into target ontologies [16], our methodology is more practical for rapidly acquiring high-quality ontology owing to scarcity of existing ontology resources.

References