An Ontology based Method for Building Understandable Hierarchical Classification Structure for Software Assets Browsing

Ge Li, Zhi Jin
Software Institute, EECS, Peking University, P.R.China
Key Laboratory of High Confidence Software Technologies, Ministry of Education of P.R.China
{lige, zhijin}@sei.pku.edu.cn

Abstract—Software asset management (SAM) represents an important role for software development and maintenance. Many software companies have been using a SAM system to help to control costs and optimize software investments across their organization and throughout all stages of their life cycles. During all the process in SAM, how to classify the software assets reasonably and build a classification to help developers find their desired software asset effectively is an import part. However, how to build a reasonable classification is a dilemmatic problem for SAM system managers. Because it is difficult for them to find a Hierarchical Classification Structure (HCS) with good super-ordinate and sub-ordinate word relationships, at the same time, it is also difficult for them to build a Hierarchical Classification Structure (HCS) using the words selected by themselves. In this paper, we proposed an ontology based HCS modeling method for SAM system managers. With this method, the managers can build an understandable HCS for their users to support the browsing of software assets. We also present a case study to illustrate the availability of our method.

Keywords—software asset management; ontology; classification; browsing-based retrieval;

I. INTRODUCTION

Software asset management (SAM) represents an important role for software development and maintenance. Administered through an ongoing plan, SAM makes it easier to identify what you have, where it's running, and whether redundancy may exist [1]. Many software companies have been using a SAM system to help to control costs and optimize software investments across their organization and throughout all stages of their life cycles. A well-deployed SAM system can help companies cut costs, improve security and compliance, and anticipate future software needs. SAM system also helps organizations accurately capture the costs and benefits associated with IT projects that enable a competitive advantage.

During all the process in SAM, how to classify the software assets reasonably and build a Hierarchical Classification Structure (HCS) to help the developers to find their desired software asset effectively by browsing–based retrieval is an import part. Because, as we have known, the browsing–based retrieval is more instructive for developers and easier for them to find new assets, more and more developers prefer to find their desired assets by browsing the assets library by following a HCS.

However, how to build a reasonable HCS is a dilemmatic problem for SAM system managers. Because, strictly speaking, in a good HCS, the relationship between each super-ordinate word and sub-ordinate word should be an “is-a-kind-of” relation. However, it is usually very difficult for system managers to find a strict taxonomy with only this kind of relation to be used as a HCS directly. At the same time, it is also very difficult for them to build a HCS from scratch using the words selected by them, because they could not make sure whether the relationship among all the selected words is appropriate to be used in a HCS or not.

In this paper, we proposed an ontology based HCS modeling method for SAM system managers. In our method, an ontology is proposed for HCS building. This ontology confined the kinds of relationship between each super-ordinate word and sub-ordinate word in the HCS. Using the instances defined according to this ontology the system managers can build a HCS with controlled understandability (in another paper [2], we gave a HCS generating algorithm to support building the HCS automatically). For the relationships among the words in HCS are defined apparently, the understandability of the whole HCS can be controlled. Furthermore, it becomes easier for the system managers to make it clear that which part of HCS dragged down the understandability of the whole HCS, and then it becomes easier for them to make some improvements.

The rest of this paper is organized as follows: section II presents the motivation of our paper with a real using example; section III discussed the different levels of understandability and the corresponding relationships in HCS; Section IV presented our ontology and the HCS building process; In section V, we presented a case study to illustrate the availability of our method; Section VII draws the conclusion for our work.

II. MOTIVATION EXAMPLE

In SAM systems, the HCS usually help the developers to find their desired software asset by browsing–based retrieval
HCS is a hierarchical tree structure, in which each node is labeled with a word and represents a set of assets based which related to this word. Navigated by HCS, the browsing retrieval process becomes a process to follow a route in the HCS. In a certain browsing-based retrieval step, the developer is usually presented with a screen containing many word-links (e.g. a hyper-link which is labeled by a word in HCS). Each word-link represents a sub-node of a certain node in the HCS. After an overview of the screen, the developer selects a link that he or she thinks the desired assets should be contained, and clicks on the link. This will result in a step of retrieval, in which all the software assets that related to this word are returned as retrieval results. Then the developer can review all the returned assets to check whether there are the desired assets among them or not. If he or she locates all the desired assets, the retrieval process will end; if the returned results are not so confined that the developer cannot find the desired assets, he or she will continue the retrieval process. In the next step, some further links, which represent the sub-nodes, are displayed on the next screen. The developer will keep on the same clicking, retrieving and reviewing until the set of retrieved assets is so confined that the retriever can easily determine whether or not the desired assets as are contained in this retrieved asset set.

From the above analysis, we can see that during the browsing-based retrieval process, developers usually using the HCS to make it clear what assets are there in the SAM system. In each retrieval step, when developer selects a word-link from the screen, he or she always assumes that the word-link could lead him or she to find the desired assets, in other words, the desired assets should be “contained” in the link. For example, for a sample HCS, as Figure 1 shows, “Android” is-a-kind-of “Mobile Operating System”, so, by these two words developers could be navigated. However, it is really very difficult for SAM system managers to find practicable predefined taxonomy ontology to be used as a HCS.

At the same time, it is also very difficult for system managers to build a HCS from scratch using the words selected by themselves, because, in the literature, there still no method discuss about what kinds of relationships could make the retriever consider “the assets set related to the super-ordinate word should include the assets set related to the word sub-ordinate word? This is a dilemmatic problem for SAM system managers.

However, what kinds of semantic relationships between each super-ordinate word (e.g. “Mobile Operating System”) and sub-ordinate word (e.g. “Android”) in the HCS could make the retriever consider “the assets set related to the super-ordinate word should include the assets set related to the word sub-ordinate word? This is a dilemmatic problem for SAM system managers.

Strictly speaking, the “is-a-kind-of” is a good candidate kind of relationship for each super-ordinate and sub-ordinate word pair in the HCS, just like the words defined in some predefined taxonomy ontologies. For example, as Figure 1 shows “Android” is-a-kind-of “Mobile Operating System”, so, by these two words developers could be navigated. However, it is really very difficult for SAM system managers to find practicable predefined taxonomy ontology to be used as a HCS. Also, it is very difficult for system managers to build a HCS using only “is-a-kind-of” relationship. For, different from taxonomy ontology, HCS is a tool used to navigate the discovery of the software assets, but not used to present taxonomy knowledge.

At the same time, it is also very difficult for system managers to build a HCS from scratch using the words selected by themselves, because, in the literature, there still no method discuss about what kinds of relationships could make the retriever consider “the assets set related to the super-ordinate word should include the assets set related to the word sub-ordinate word. So, how to modeling the word relationships in HCS and how to build an understandable HCS is remained quite a problem for SAM system.
III. OUR APPROACH

In this paper, we proposed an ontology based HCS modeling method for SAM system managers. An upper level ontology is proposed to confine the kinds of relationship between each super-ordinate and sub-ordinate word pair in the HCS. Using the instances defined according to this ontology, the managers can build an understandable HCS for their users to support the browsing of software assets. In this section, we will discuss the different levels of understandability of HCS, and then present the upper level ontology used to modeling the HCS. From this section on, we will call the developers that are retrieving the SAM system as retrievers.

A. Retriever Assumptions and Relationships Analysis

As we have discussed in section II, during the retrieving process, the retrievers are always holding some assumption: the assets set related to the super-ordinate word should include the assets set related to the sub-ordinate word. And, because of holding this assumption, the developers could be navigated by the HCS to find their desired assets. In fact, besides of this assumption, there are several other assumptions holding by the developers during the retrieval process. According to the degrees of strictness of this assumption, we classified them into 4 different levels, named Retriever Assumption I-4:

Retriever Assumption I: During the retrieving process, the retrievers are always assuming that the assets set related to the super-ordinate word should include all the assets related to the sub-ordinate word.

In our method, in order to modeling the right semantic that meets Retriever Assumption I, we defined three kinds of relationships for each super-ordinate and sub ordinate word pair:

IsaKindof: for word $W_1$ and word $W_2$, if the concept represented by $W_1$ is a specialization of concept represented by $W_2$, and the concept represented by $W_2$ is a generalization of concept represented by $W_1$, then we have $W_1 \text{ IsaKindof } W_2$.

For example, in Figure1, “Android” is a kind of “Mobile Operating System”. According to the definition of IsaKindof relationship, we have: “Android” IsaKindof “Mobile Operating System”. During the retrieving process, if the retrieval result had been confined to the assets set related to “Desktop Operating System”, then the words “File Management Subsystem” means the assets set that related to the “Desktop Operating System in Desktop Operating System”, this set is included in the assets set related to the super-ordinate word (“Desktop Operating System”), so we can see, the semantic of “File Management Subsystem” IsaPartof “Desktop Operating System” can meet the definition of Retriever Assumption I.

[Attribute]Is: for word $W_1$, attribute $A_1$, and word $W_2$, if $A_1$ is an attribute of $W_1$ (denoted as: $W_1.A_1$), and if the word $W_2$ is a kind of $W_1.A_1$, in other words, if $W_2$ IsaKindof $W_1.A_1$ then we have $W_1.A_1$Is $W_2$.

For example, in Figure1, “Manufacturer” is an attribute of “Workstation Operating System” (although the words “Manufacturer” is not appeared in the HCS), and “IBM” is a kind of “manufacturer” of “Workstation Operating System”. According to the definition of [Attribute]Is relationship, we have: “Workstation Operating System” [Manufacturer]Is “IBM”. Here the word “IBM” means the workstation operating systems that were manufactured by IBM. During the retrieving process, if the retrieval result have been confined to “Workstation Operating System”, then the words “IBM” means the assets set that related to the “Workstation Operating System that manufactured by IBM”, this set is included in the assets set related to the super-ordinate word (“Workstation Operating System”), so we can see, the semantic of “Workstation Operating System” [Manufacturer]Is “IBM” can meet the definition of Retriever Assumption I.

DisjointWith: for word $W_1$ and word $W_2$, if there isn’t any common instance for the concept represented by $W_1$ and the concept represented by $W_2$, then we have $W_1 \text{ DisjointWith } W_2$. To facilitate representation,
for a words set \( W_5 = \{W_i \in \{1, 2, \ldots, n\}\} \), if for any two words \( W_i \) and \( W_j \) in the set, we have \( W_i \triangleq \text{DisjunctWith} W_j \), then we can donate the relationship as: \( \text{DisjunctWith}[W_i \in \{1, 2, \ldots, n\}] \) or \( \text{DisjunctWith} W_j \).

For example, for the words “Symbian”, “Android” and “Windows Mobile” in Figure 1, if there haven’t any common instance for the concept represented by each of them. According to the above definition, we have: \( \text{DisjunctWith} \{“Symbian”, “Android”, “Windows Mobile”\} \). Which means the assets sets related to word “Symbian”, “Android” and “Windows Mobile” is isolated to each other; there isn’t any intersection between each pair of them. In this case, during the retrieving process, if a retriever selected to search from the Operating System aspect, and confined the retrieval result to the assets related to “Mobile Operating System”, he or she could make a clear choice of which should be selected. So, we can see that the semantic of \( \text{DisjunctWith} \{“Symbian”, “Android”, “Windows Mobile”\} \) can help us to build HCS that meets the definition of Retriever Assumption II.

**Retriever Assumption III:** During the retrieval process in one aspect (e.g. the Operating System aspect), on each retrieving step, if the current word-link include many sub-ordinate word-links, the retrievers are always assuming that the desired assets must related to at least one of them, but not exist as a isolated leaf.

For example, in Figure 1, if a retriever selected to search desired assets from the “Operating System” aspect, and had selected “Desktop Operating System” in the first step, and then faces with three sub-ordinate words (“File Management Subsystem”, “Memory Management Subsystem” and “Process Management Subsystem”), the retriever may hold a assumption: all the assets related to “Desktop Operating System” had been divided into three subsets, which related to three sub-ordinate words separately. And, no isolated outside of these three subsets.

Therefore, we defined the fifth kind of relationship which used to modeling the right semantic that meets Retriever Assumption III:

**CoveredBy:** for word \( W_i \) and a words set \( W_5 = \{W_j \in \{1, 2, \ldots, n\}\} \), if for any instance of the concept represented by \( W_i \) must belongs to at least one instances set of the concept represented by some \( W_j \) in \( W_5 \), then we have \( W_i \) is covered by \( W_5 \), denoted as: \( W_i \text{CoveredBy} W_5 \).

For example, in Figure 1, if in the current SAM system, all the assets related to “Desktop Operating System” are divided into three subsets, and each subset related to three sub-ordinate words (“File Management Subsystem”, “Memory Management Subsystem” and “Process Management Subsystem”) separately. And there isn’t any isolated asset outside of these three subsets. Then we have: “Desktop Operating System” \( \text{CoveredBy} \{“File Management Subsystem”, “Memory Management Subsystem”, “Process Management Subsystem”\} \).

Besides the three assumptions mentioned above, we proposed a supplementary retriever assumption for retriever, which could be referenced in HCS building:

**Retriever Assumption IV:** During the retrieval process in one aspect (e.g. the Operating System aspect), in any consecutive two-step retrieving, the retrievers may assume that the two step’s retrieving are navigated by the same kind of relationships.

For example, in Figure 1, if a retriever selected to search desired assets from the “Operating System” aspect, and had selected “Mobile Operating System”. For the relationship between “Mobile Operating System” and “Operating System” is \( \text{IsaKindOf} \), so the retriever maybe assume that, the relationship in the next step is as same as the relationship in the last step.

We proposed 4 retriever assumptions above, as we can see that, different retriever assumptions could be superimposed together in one HCS. And, the more assumptions the HCS meets, the more understandable it is. So, in order to help the SAM system managers to build more understandable HCS, based on the relationships we mentioned above, we proposed an upper level ontology to guide the HCS modeling.

**B. Upper Level Ontology for HCS Modeling**

According to the above analysis, we proposed an upper level ontology for HCS modeling.

![Figure 2. Upper Level Ontology for HCS Modeling.](Image)

As Figure 2 shows, the upper level ontology consists of 4 elements: Words, Attributes, Relations and Axioms. In which:

- **Words** represents a set of words that will be used in HCS; there are 3 kinds of words (Super-ordinate word, sub-ordinate word, co-ordinate word).
- **Attributes** is a collection of some small attribute sets, and each small set in the collection includes all the attributes of one word;
- **Relations** represents a set of relationships among different kinds of words, only 5 kinds of relationships (as mentioned in above) can be used in the instance
level ontology to represent the relationships between each words pair;

Beside the above 5 kinds of relationships, in order to facilitate the representation of attribute, we defined IsanAttributeOf as a supplementary relationship:

\[
\text{Wp IsanAttributeOf Wq: to represent that the word Wp is an attribute of the word Wq.}
\]

e.g. “Manufacturer” IsanAttributeOf “Workstation Operating System”;

- **Axioms** represents a set of axioms; each of them is a constraint on the words or relationships (e.g. about the transferability, symmetry, etc). Each constraint can be expressed in Prolog-like rule [11].

As an example, the following is an instance ontology for the HCS in operating system aspect showed in Figure1:

**TABLE I. AN INSTANCE ONTOLOGY FOR OPERATING SYSTEM ASPECT**

<table>
<thead>
<tr>
<th>Operating System = {Words; Attributes; Relations; Axioms}</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Words</strong> = {</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td><strong>Attributes</strong> = {</td>
</tr>
<tr>
<td>“Manufacturer” IsanAttributeOf “Workstation Operating System”;</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td><strong>Relations</strong> = {</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

In practice, this ontology can be represented in any ontology representation language, such as the RDF(S), the DAML+OIL and so on.

Using the proposed upper level ontology can help the SAM system managers to define the relationships among words clearly. As we have discussed, in order to build more understandable HCS, we should try to make every superordinate and sub-ordinate word pair meets more retriever assumptions. In the following section we will use a case study to illustrate the using of the proposed ontology.

**IV. CASE STUDY**

This section uses a case study to illustrate the availability of our ontology. In this case study, we will present how the HCS in the Figure1 are built up step by step, in this process, as we can see that, with more retriever assumption been met, the understandability of the HCS will be improved step by step.

In the first step, we build a HCS that only meets retriever assumption I. During the modeling of this HCS, we only used 3 kinds of relationships: IsaKindOf, IsaPartOf, [Manufacturer]Is. We can see that, though the HCS is usable, and meets retriever assumption I, it is still not easy to be understood, for the sub-ordinate words of “Operating System” are so diversiform that confusing the retrievers.

![Figure 3. HCS Meet Retriever Assumption I](image)

In the second step, we try to add the relationship of DisjointWith to the model of HCS, and try to make the HCS to meet retriever assumption II. As Figure 4 shows, the
understandability is better than Figure 3, for in each retrieving step, the retriever only need to select only one sub-ordinate word-link, but do not need to make trade-offs among several possible word-links, so it becomes more clear and straight for retrievers.

In the third step, we try to add the relationship of CoveredBy to the model of HCS, and try to make the HCS to meet Retriever Assumption III, and the HCS becomes to as Figure 1 shows. We can see that, the understandability is better than Figure 4, for in each retrieving step, the retriever only need to select the next sub-ordinate word-link, but do not need to care about whether the desired asset is an isolated leaf, and isn’t related to any sub-ordinate word-link.

However, for the HCS in Figure 1, there is still room for improvement, because it doesn’t meet the Retriever Assumption IV. In order to make the HCS to meet Retriever Assumption IV, we could delete the words “IBM” and “Sun” from the HCS, just as Figure 5 shows.

So, we can see that, different retriever assumptions could be superimposed together, the more assumptions the HCS meets, the more understandable it is. By using more kinds of relationships, we can improve the understandability step by step.

V. DISCUSSION AND CONCLUSION

Software asset management (SAM) represents an important role for software development and maintenance. In SAM, how to classify the software assets reasonably and build a Hierarchical Classification Structure (HCS) and help the developers to find their desired software asset effectively by browsing–based retrieval is an important part. However, how to build a reasonable HCS is a dilemmatic problem for SAM system managers.

In this paper, we analyzed the different level of understandability of HCS, proposed 4 retriever assumptions, and discussed the kinds of relationship between each super-ordinate and sub-ordinate word in the HCS. We proposed an ontology based HCS modeling method for SAM system managers, which an ontology is proposed for HCS building. This ontology confined the kinds of relationship between each super-ordinate and sub-ordinate word in the HCS. Using the instances defined according to this ontology, the system managers can build a HCS with controlled understandability. And, if more kinds of relationships are used in the modeling, more assumptions in the HCS will be met, and more understandable the HCS is. Also, this ontology can be combined with the HCS generating method (we published in 2007[2]), to build a HCS automatically. Besides these, our method make it easier for the system managers to make it clear that which part of HCS dragged down the understandability of the whole HCS, and then it becomes easier for them to make some improvements.

VI. ACKNOWLEDGMENTS

This research was sponsored by the National Grand Fundamental Research 973 Program of China under Grant No. 2011CB302704, and the National Natural Science Foundation of China under Grant No. 60803010.