DEVELOPING DICTIONARY FOR IDENTIFIERS USING NATURAL LANGUAGE PROCESSING

V. Angayarkanni, M.Tech student in Computer Science and Engineering
Dr.MGR Educational and Research Institute Chennai, Tamilnadu, INDIA
angayarkannics@gmail.com

Abstract

Now a day's software maintenance and reuse process is a tedious process. Meaningful method names are useful for readability and maintenance of the software .Large time and effort provided to software maintenance and reuse. This can be reduced by providing software engineers with software tools such as IDEs, which automatically provide information about the source code. Identifiers are the important element in the program, if the identifiers in the program has inappropriate name programmers find difficult to understand the program. In this paper proposes a method to for building dictionary for identifiers from source code written in object oriented language. Dictionary is built using the verb object relationship extracted from source code. This dictionary is helpful in software reengineering and maintenance.

Key Terms: reengineering, identifiers, maintenance, programmers, source code, verb object.

Introduction

Identifiers are the important element in the source code. Developers try to understand the source code using the roles of identifiers. If the identifiers in the program has inappropriate name it will take large time to program comprehension. Multiple identifiers are used to represent the role of the program. To make the software reengineering and maintenance process easier this dictionary is developed. Reengineering is the modification of the software system takes place after it has been reverse engineered generally to add new functionality or to correct errors. To make the reengineering process easy developers has to give appropriate name to identifiers. Unfortunately not all the developers are giving correct names to identifiers, so during the program comprehension it is tough understand the program elements. To reduce the work for developer's dictionary is developed. Software reengineering process is complex process. Programmers are provided with tools like IDEs to increase the productivity. IDEs are designed to maximize programmer's productivity by providing tightly-knit components by similar user interfaces. This should mean that the programmers has to do less mode switching versus using discrete development program Dictionary contains the good examples for identifiers. This dictionary is useful for naming classes and variables. This dictionary is built using the verb object relationship in the source code. Verb object relationship obtained using the method property. Verbs are extracted from the method name and class name and objects are extracted from the formal parameters of the method, relationship is obtained using the pattern matching system. This dictionary is developed using MVC2 architecture to increase the security. Model view controller architecture is a software architecture currently considered architecture pattern used in software engineering. This pattern isolates domain logic from the user interface permitting independent development, testing and maintenance of each. Model view controller (MVC) pattern creates application (input logic, business logic, and UI logic) while providing a loose coupling with this element. In complex computer applications that present a large amount of data to a user, a developer often wishes to separate data (model) and user interface (view) concerns, so that changes to the user interface will not affect the data handling, and that the data can be recognized without changing the user interface. The model view controller solves this problem by decoupling data access and business logic from data presentation and user interaction by introducing the intermediate component controller. It is common to split an application in to separate layers presentation (UI), domain logic, data access. In MVC2 the presentation layer is further separated in to view and controller. MVC2 encompasses more of the architecture of an application than is typical for a design pattern. MVC2 is often seen in web applications, where view is actual HTML page, the controller is the code that gathers dynamic data and gathers dynamic data and generates the content with in the HTML. Finally, the model is represented by the actual content, usually stored in database or XML files. The user interacts with the user interface in some way. The controller handles the input event from the user interface, often via a registered handler or callback. The controller access the model, possibly updating it in a way appropriate to the user's action. A view uses the model (indirectly) to generate appropriate user interface. The view gets its own data from the model. The model has no direct knowledge of the view. The user interface waits for further user interactions, which begins the cycle anew. By decoupling models and views, MVC2 helps to reduce the complexity in architectural design, and to increase flexibility and reuse. The model view controller design pattern, the model view controller design pattern, also known as model2 in J2EE application programming, is a well established design pattern for programming. The model1 and model2 architectures both separate content generation (business logic) from the content presentation (HTML formatting). Model2 differs from model1 in the location where the bulk of the request processing is performed by a controller rather than in the JSP pages. The main aim of the MVC2 to separate the business logic and application data from the presentation data to the user. And also MVC2 architecture is reusable when the problem recurs, there is no need to invent the solution just has to follow the pattern and adapt as it necessary. They are very expressive by using MVC2 our application becomes very expressive.

This paper proposes a method as an example of an application, which could support the work of a company that must help with improvement of the company.

Related Work

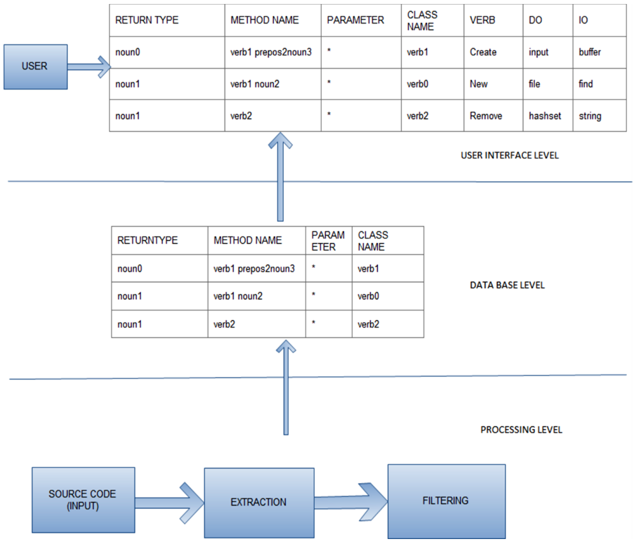
A wide variety of tools are required to programmers to develop an application, for that many tools and IDEs are developed. IDEs are very useful for the developers to save their time. To make the developers work easy many tools and systems are developed. So that developer's productivity will be increased. The first existing system developed was syntactic details for method names; it will give meaningful name for method property. The vast amount of software written in java defines an implicit convention for pairing names and implementations. This system is also used to mechanically check whether the method name is correct or not. The first existing system is used to extract the method names, which are used to identify the names bugs in the large software applications. The first existing system is also automatically suggesting suitable names for method. The software systems are growing continuously, locating code for maintenance and reuse tasks become very difficult. Code search techniques are using natural language query processing; in the second existing system they provide the system to automatically extract the code from source code and categorize the search results in hierarchy. This helps developers to quickly identify relevant program elements for investigation or to quickly identify relevant words. It will give little support to developers to search the relevant code, So that maintenance work will be easy and also developers can reuse the codes. The effort given by software developers in software maintenance can be reduced by providing tools and IDEs that give tedious and error prone tasks. The natural language processing improves the effectiveness of the software maintenance process and reuse. Tools such as IDEs automatically provide program information and automated support to developers. There is considerable place for improvement for existing software development tools. The third existing system is useful in improving tool for program comprehension; maintenance and reuse of object oriented code. This system automatically extracts the clues in the form of verb object pairs and also it uses the natural language processing. This system describes the extraction rules and an evaluation of the system for java is described. The above described existing systems are used in software maintenance and reuse process and the above described tools are based on the natural language processing and extraction of the method property. The existing system developed is domain specific dictionaries for identifiers. This is also based on the natural language processing and verb object relationship. This dictionary is also useful in software maintenance and reuse process. Dictionary is helpful in naming identifiers. Because some developers are giving inappropriate names for identifiers so when doing maintenance it is tough to understand the codes. Identifiers are the important key element in understanding the program elements. Verb object relationships are extracted from the method name and parameters of the method and the class name. Verb is extracted from the method name and objects are extracted from the method parameters. Method property is used for the verb object relationship. First the method name, parameters are extracted from the method property and it is classified as return type, verb, direct object and indirect object after that very frequent relationships are filtered and put it in the dictionary. Verb object relationship is obtained using the pattern matching system. For developing the system input given is the object oriented source code and output is the dictionary. This dictionary is useful in naming identifiers and also for the program comprehension. It will help the developers to increase their productivity and to reduce the work.

Proposed Work

In the proposed work we developed a dictionary for identifiers using MVC2 architecture and client server model. This dictionary can be accessed by multi user at the same time.MVC2 architecture is for better security and to change the window based application to the web application. Client-server model is a computing model that acts as distributed application which partitions tasks or workloads between the providers of a resource or service, called servers and service requesters, called clients. Often clients and servers communicate over a computer network on separate hardware. A server machine is a host that is running one or more server programs which share their resources with clients. A client do not share any of its resources, but request a servers content or service function. Clients therefore initiate communication sessions which await incoming requests. The client-server characteristic describes the relationship of cooperating programs in an application. The server component provides a function or service to one or more clients, which initiate requests for such services. Functions such as web application and data access are done on client server model.

User accessing the dictionary from their computer uses a web browser client to send a request to a web browser at any point. Those program in turn forward the request to its own database client program that sends a request to a database server, which in turn serves it back to the web browser client displaying the results to the user. Many business applications being written today use the client-server model.

MVC2 architecture is used to develop the dictionary structure.MVC2 architecture is for the better security. Model View Controller (MVC) applied to web applications. Hence the two terms can be used interchangeably in the web world. With MVC2 you can have as many controller servlets in web application. In fact we can have one controller servlet per module. However there are several advantages of having a single controller servlet for the entire web application. Web applications based on Model 2 architecture are easier to maintain and extend since the views do not refer to each other and there is no presentation logic in the views.



RETURN TYPE	METHOD NAME	PARAMETER	CLASS NAME	VERB	DO	IO
noun0	verb1 prepos2noun3	*	verb1	Create	input	buffer
noun1	verb1 noun2	*	verb0	New	file	find
noun1	verb2	*	verb2	Remove	hashset	string

RETURNTYPE	METHOD NAME	PARAMETER	CLASS NAME
noun0	verb1 prepos2noun3	*	verb1
noun1	verb1 noun2	*	verb0
noun1	verb2	*	verb2

Figure 1. Architecture of proposed system

A. Obtaining Identifiers Using Natural Language Processing

Identifiers are the names of variables, methods, classes, packages and interfaces. Unlike literals they are not the thing themselves, just ways of referring to them. Identifiers must be composed of letters, numbers, the underscore and the dollar sign. Identifiers may only begin with letter, the underscore or a dollar sign.

Each variable has a name by which it is identified in the program. It's a good idea to give your variables mnemonic names that are closely related to the values they hold. Variable names can include any alphabetic character or digit and the underscore. The main restriction on the names you can give your variables is that they cannot any white space. You cannot begin a variable name with a number. There is no limit to the length of a variable name. Identifier allows a programmer to refer to the item from other places in the program. To make the most out of the identifiers you choose make them meaningful and follow the standard java naming conventions.

The name of the method in object oriented program is called as verb. Mostly method name is verb or verb clause. Method parameters are noun or noun clause. In some cases the method name may be noun or adjective. In some cases method name contains no verb, Method name will be noun or adjective. In some cases method name contains no verbs. A function name and a set of named arguments which are defined by nested escaped identifiers. This allows active identifiers to be nested to an arbitrary depth making possibly for them to specify a full functional program within an identifier.

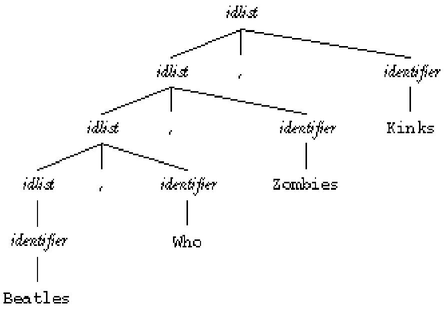


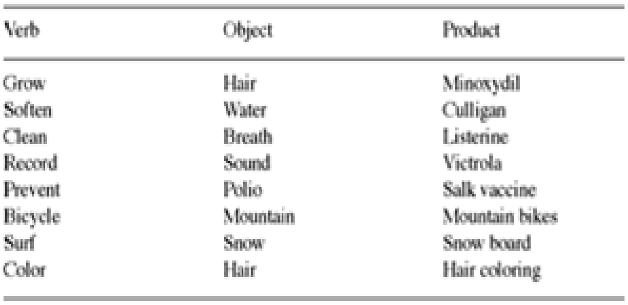
Fig .2. Nested Identifiers

B. Obtaining Verb And Object From Method Property Using Pattern Matching Technique

Verb and object are obtained from the method property. Verb is extracted from the method name and object is extracted from the method parameters, the above process is done by the natural language processing and pattern matching technique. The verb object relationship is obtained by the pattern matching system. The input to the method is source files written in object oriented language, and the output is a dictionary consisting of Return type, verb, direct object and indirect object.

In source code, there are many pairs of verbs and objects that appear in natural language.

TABLE 1: Verb object combinations



Verb	Object	Product
Grow	Hair	Minoxydil
Soften	Water	Culligan
Clean	Breath	Listerine
Record	Sound	Victrola
Prevent	Polio	Salk vaccine
Bicycle	Mountain	Mountain bikes
Surf	Snow	Snow board
Color	Hair	Hair coloring

C. Pattern Matching Technique

Pattern matching addresses issues of searching and matching strings and more complicated patterns such as trees, graphs, regular expressions and arrays. The goal is to derive non-trivial combinatorial properties for such structures and then to exploit these properties in order to achieve improved performance for the corresponding computational problem.

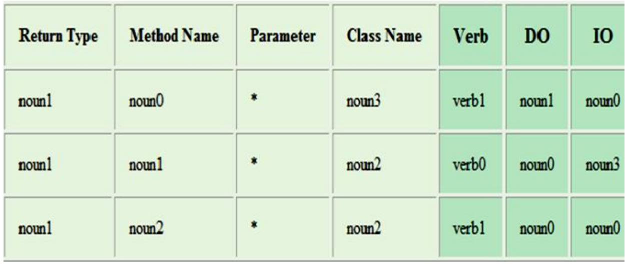
Pattern recognition algorithm generally aims to provide a reasonable answer for all possible inputs and to do “fuzzy” matching of inputs. This is opposed to pattern matching algorithms, which look for exact, matches in the input with pre-existing matches.

A common example of a pattern matching algorithm is regular expression matching, which looks for pattern of a given sort in textual data and is included in the search capabilities.

D. Extraction Of Verb Object Relationship From Source Code

To extract the verb object relationship from the source code, identify the method property from the source code. After identifying the method property retrieve all method declarations from the source code. Then for each method declaration obtain identifiers for each declaration. Obtain words for each identifier. This information is expressed in the form of tuple V, DO, IO. Select the tuple that appear frequently in many software products. Source files are parsed using pattern matching to get the verb object relationship.

When extracting the method, Return type, method name, names and type of the formal parameters, Name of the class are extracted and stored as the tuples. Obtain the words from the identifiers and part of speech of each word and form the words in VO relationship, and store it in a dictionary.



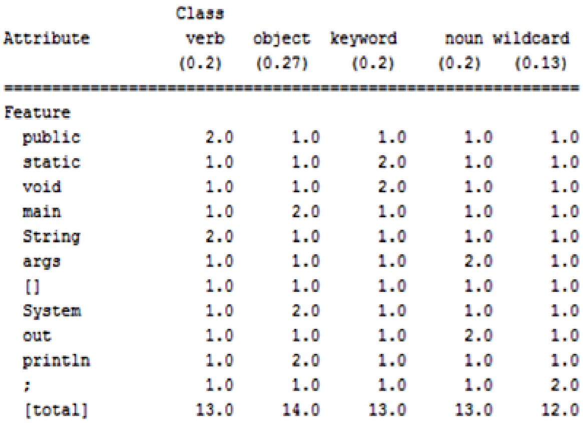
Return Type	Method Name	Parameter	Class Name	Verb	DO	IO
noun1	noun0	*	noun3	verb1	noun1	noun0
noun1	noun1	*	noun2	verb0	noun0	noun3
noun1	noun2	*	noun2	verb1	noun0	noun0

Figure 3. Extracted Dictionary

In the extracted dictionary the return type may be non void type, if it is non void it is treated as noun. Sometimes indirect object may be empty.

E. Filtering Verb Object Dictionary

This step filters the verb object dictionary that obtained from the extraction part. It filters the frequently appeared verb object relationship. In the filtering process it calculates the occurrences of the each words and it filters the frequently appearing words. The words that appear in the certain number software are included in the dictionary. Frequencies of the each word are calculated in the filtering process. After the frequencies are calculated the words are filtered from the extracted dictionary. In the filtering and extracting process Naïve Bayes classifier is used to extract and filter the large amount of data.



```
                     Class
Attribute             verb    object  keyword     noun wildcard
                     (0.2)    (0.27)    (0.2)    (0.2)   (0.13)
=============================================================
Feature
  public               2.0       1.0      1.0      1.0      1.0
  static               1.0       1.0      2.0      1.0      1.0
  void                 1.0       1.0      2.0      1.0      1.0
  main                 1.0       2.0      1.0      1.0      1.0
  String               2.0       1.0      1.0      1.0      1.0
  args                 1.0       1.0      1.0      2.0      1.0
  []                   1.0       1.0      1.0      1.0      1.0
  System               1.0       2.0      1.0      1.0      1.0
  out                  1.0       1.0      1.0      2.0      1.0
  println              1.0       2.0      1.0      1.0      1.0
  ;                    1.0       1.0      1.0      1.0      2.0
  [total]             13.0      14.0     13.0     13.0     12.0
```

Figure 4. Frequency Calculation of Words

F. Support Vector Machine

Support vector machines (SVM) are offers one of the most robust and accurate methods among all well-known algorithms. It has a sound theoretical foundation, requires only a dozen examples for training, and is insensitive to the number of dimensions. In addition, efficient methods for training SVM are also being developed at a fast pace.

In a two-class learning task, the aim of SVM is to find the best classification function to distinguish between members of the two classes in the training data. The metric for the concept of the "best" classification function can be realized geometrically.

For a linearly separable dataset, a linear classification function corresponds to a separating hyper plane *f (x)* that passes through the middle of the two classes, separating the two. Once this function is determined, new data instance *xn* can be classified by simply testing the sign of the function *f (xn)*; *xn* belongs to the positive class if *f (xn)* > 0.

SVM additionally guarantee is that the best such function is found by maximizing the margin between the two classes. Intuitively, the margin is defined as the amount of space, or separation between the two classes as defined by the hyper plane. Geometrically, the margin corresponds to the shortest distance between the closest data points to a point on the hyper plane. Having this geometric definition Allow us to explore how to maximize the margin, so that even though there are an infinite number of hyper planes, only a few qualify as the solution to SVM.

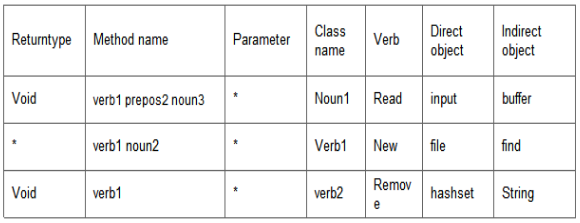
The reason why SVM insists on finding the maximum margin hyper planes is that it offers the best generalization ability. It allows not only the best classification performance (e.g., accuracy) on the training data, but also leaves much room for the correct classification of the future data. To ensure that the maximum margin hyper planes are actually found, an SVM classifier attempts to maximize the following function with respect to w and *b*:

$$L_P = \frac{1}{2} \| \vec{w} \| - \sum_{i=1}^{t} \alpha_i \, y_i \, ((\overrightarrow{w.x}) + b) \sum_{i=1}^{t} \alpha i$$

Where *t* is the number of training examples, and *αi*, *i* = 1, . . . , *t*, are non-negative numbers such that the derivatives of *L P* with respect to *αi* are zero. *αi* are the Lagrange multipliers and *L P* is called the Lagrangian. In this equation, the vectors _w and constant *b* define the hyper plane.

Resulting Dictionary

The output of the dictionary is a web application built using MVC architecture and client server model. The dictionary consists of the Tuple return type, method name, parameters, class name, verb, direct object, Indirect object. The entries in the dictionary, i.e. tuples consisting of <V, DO, IO>, are extracted from identifiers related to a method.



Returntype	Method name	Parameter	Class name	Verb	Direct object	Indirect object
Void	verb1 prepos2 noun3	*	Noun1	Read	input	buffer
*	verb1 noun2	*	Verb1	New	file	find
Void	verb1	*	verb2	Remove	hashset	String

Figure 5. Output of the dictionary

After extraction and filtering the output of the dictionary consists of the frequent words occurred in the source code. Here source code of the java is taken.

Conclusion

The main advantage of the proposed system is high security and dictionary can be accessed by multiple clients at the same time. Developer often wishes to separate data (model) and user interface (view) concerns, so that changes to the user interface will not affect data handling, and that the data can be reorganized without changing the user interface. By providing dictionary developers can increase their rate of work. Software maintenance and reuse process is complex process, because of this kind of tools maintenance process will be easy and productivity can be increased.

References

[1] Yasuhiro Hayase, Yu Kashima, Katsuro Inoue,Yuki Manabe., "Building Domain Specific Dictionaries of Verb-Object Relation from Source Code",2011 15th European Conference on Software Maintenance and Reengineering

[2] Eisenbarth, T., Koschke, R., and Simon, D., "Locating Features in Source Code", *IEEE Transactions on Software Engineering*, vol. 29, no.3, March 2003.

[3] Enslen, E., Hill, E., Pollock, L., and Vijay-Shanker, K., "Mining Source Code to Automatically Split Identifiers for Software Analysis", in Proc. of 6th IEEE MSR'09, Vancouver, Canada May 16-17 2009.

[4] Grant, S., Cordy, J. R., and Skillicorn, D. B., "Automated Concept Location Using Independent Component Analysis ", in Proc. of 15th WCRE'08, Antwerp, Belgium, October 15-18 2008.

[5] Haiduc, S. and Marcus, A., "On the Use of Domain Terms in Source Code", in Proc. of 16th IEEE ICPC'08, Amsterdam, The Netherlands, June 10-13 2008.

[6] Hill, E., Pollock, L., and Vijay-Shanker,K., "Automatically Capturing Source Code Context of NL-Queries for Software Maintenance and Reuse", in Proc. of 31st IEEE/ACM ICSE'09, May 16-24 2009.

[7] Lawrie, D., Morrell, C., Feild, H., and Binkley, D., "Effective Identifier Names for Comprehension and Memory", *Innovations in Systems and Software Engineering*, vol. 3, no. 4, 2007.

[8] Maletic, J. I. and Marcus, A., "Supporting Program Comprehension Using Semantic and Structural Information", in Proc. of 23rd ICSE'01, Toronto, Ontario, Canada, May 12-19 2001.

[9] Takang, A., Grubb, P., and Macredie, R., "The Effects of Comments and Identifier Names on Program Comprehensibility: An Experimental Investigation", *Journal of Programming Languages*, vol. 4, no. 3, 1996.

[10] Shepherd, D., Fry, Z., Gibson, E., Pollock, L., and Vijay-Shanker, K., "Using Natural Language Program Analysis to Locate and Understand Action-Oriented Concerns", in Proc. of 6th AOSD'07, 2007.

[11] Von Mayrhauser, A. and Vans, A. M., "Program Comprehension During Software Maintenance and Evolution", *Computer*, vol. 28, no. 8,1995.