ICIC Express Letters
Part B: Applications
Volume **9**, Number **8**, August **2018**

ICIC International ©2018 ISSN 2185-2766
pp. **805–811**

RESEARCH ON CODE KNOWLEDGE BASE ORIENTED TO SOFTWARE GENERATION

Aijing Jia, Yao Liu* and Yi Huang

Institute of Scientific and Technical Information of China
No. 15, Fuxing Road, Haidian District, Beijing 100038, P. R. China
*Corresponding author: liuy@istic.ac.cn

Received January 2018; accepted April 2018

Abstract. *In the era of Internet, automatic code generation is a kind of technical means commonly used in software reuse, which can greatly improve the efficiency of software production. Based on the study of the rules and methods of code marking, this paper uses natural language processing technology and machine learning method to reconstruct various types of code. On the basis of constructing code ontology, semantic annotation technology is used to annotate the code, to archive a structured and semantic code knowledge base. Experiments show that the code knowledge base can effectively improve the efficiency of software development.*

Keywords: Software generation, Code knowledge base, Semantic annotation, Natural language processing

1. **Introduction.** Into the Internet era, all kinds of scientific research are inseparable from a variety of software tools, and the preparation of the software requires a lot of code support, and the code occupies a very important position in the computer technology. However, the code is only tools, and we should not spend a lot of time in the preparation of the code. Code generation is a common means of the software reuse, and it can improve the efficiency of software production largely. Similar to the text generation, the code generation also needs a lot of corpus, and building a complete and practical code knowledge base is particularly important. Code knowledge base stored a large number of automatic or semi-automatic markup of the code corpus, using these corpora, according to the rules written code, code can be generated.

There is still little research on code knowledge base at present. In the generation of code, the research of the domestic and foreign mainly focuses on the specific programming language, framework and mode, using the UML based model [1], design pattern based [2], and the template engine based [3] and other methods, code generation customization and versatility is weak. Code hosting platform such as GitHub, Bitbucket, coding, CSDN, and code cloud, cannot structure storage code, also cannot represent the code on the next bit, the relationship between the code, so there is a certain distance on the code generation. In the construction of knowledge base, the Institute of Science and Technology Information of China has carried out research on key technologies including semantic reptile [4], ontology automatic construction [5], semantic annotation [6,7]. In view of this, this paper presents a code-generation code knowledge base for the construction method.

At present, the research of the software generated code knowledge base is less. This paper studies the construction of the code base of software generation, provides the corpus support for the later software generation, and improves the production efficiency of the software. The code in the code base is extracted, marked and so on, and the method of natural language processing and machine learning is used to reconstruct all kinds of code

DOI: 10.24507/icicelb.09.08.805

and realize the structure and semantics of the code resources, which has achieved good results.

2. **Design Idea.** The basic idea of this paper is to study the rules and methods of the code, and reconstruct and utilize all kinds of codes by ontology construction technology, achieving the structured and semantic synchronization of the code resources, combining the code marking rules to mark the code on the basis of constructing the code system framework, providing predictive support for code generation. The quantity and quality of the code in the code knowledge base will directly affect the quality of the code knowledge base. The accuracy of the code label directly affects the quality of the code generation. The main source is the source code and the software code of the CD. We assimilate the idea of automation into the design process of the code knowledge base, including the existing code of the automatic access, code cleaning and processing, automatic labeling of the code and other aspects. The construction of the code knowledge base is shown in the following figure.

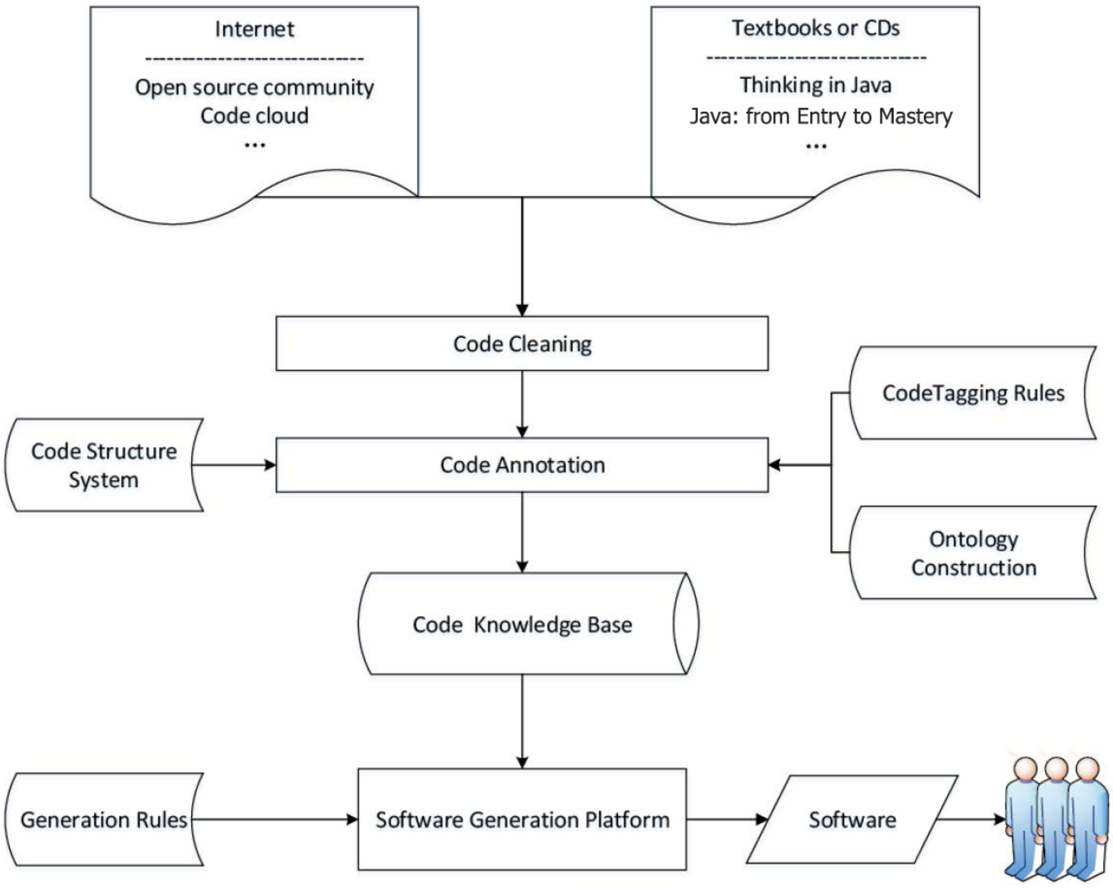


Figure 1. The process of the code knowledge base construction

After learning the code in the "21-day learning java 6" version 5, "Java language programming" and "Java programming thought: fourth edition", summarizing the elements of the Java code include the data type, the identifier, Keywords, operators, comments, in theory, Java code is the different combinations of these five elements. It forms the various layers of Java code, including variables, arrays, statements, methods, classes, objects, and interfaces, as shown in Figure 2. Analysis of these elements can reveal the regularity between the codes, build code model and lay the bedding for further research.

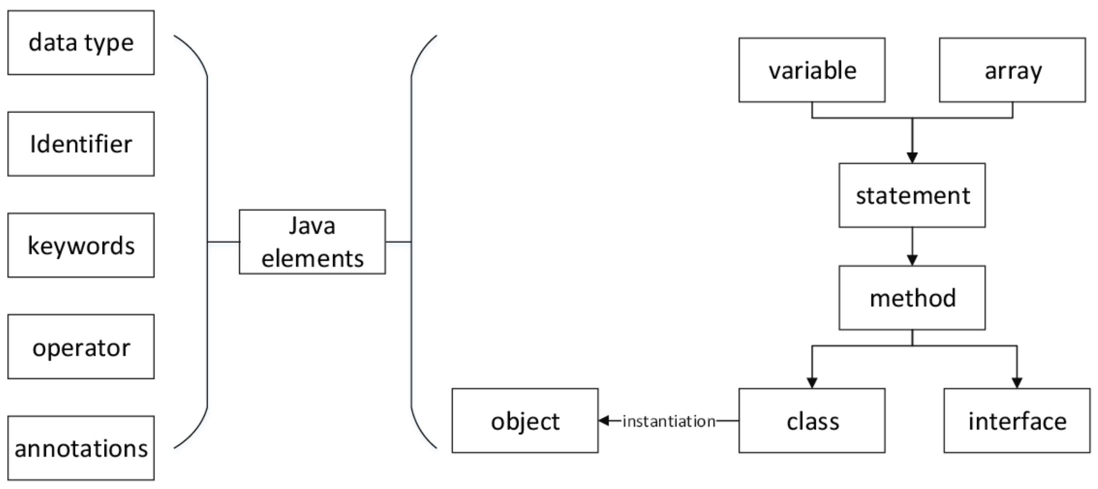


Figure 2. Code composition

3. Research on Key Technology.

3.1. **Code tagging rules building.** It is helpful to standardize the code software obtained from different sources, and also provide the basis for the analysis and annotation of the code, which provides structural support for the construction of code ontology. By summing up the various keywords and the related concepts, we divided the java code into 8 top-categories, objects, modifiers, variables, statements, comments, data types, behavior and operators.

The secondary classification of the object is the class. The class contains four classifications: String, array, VO, and set. The array contains two classifications: the int array and the String array. The set includes four classifications: List, ArrayList, HashMap and HashSet.

The modifiers include the public modifiers, the class modifiers, the member variable modifiers, and the method modifiers. The public modifier contains public, private, protected, friendly, and static modifiers, but we can only use one of them when the class, method or the variables are modified. The modifiers with static can modify static variables, methods, and classes, the modifier without static can modify general methods. The abstract modifier can modify the classes, the transient and volatile modifiers can modify the abstract classes and the member variable. The synchronized and native modifiers can modify the method. The variables contain constants, expressed separately with different identifiers.

The secondary classification of the statements includes assignment statements, flow control statements, and interrupt control statements. The flow control statement includes two classifications: the conditional statement and the loop statement. The conditional statement includes the if conditional statement and the if else conditional statement. The loop statement includes a while loop, a for loop, a switch statement, a case statement, and a for each statement. The interrupt control statements include break statements, return statements and continue statements.

Comments include three types, namely "//", "/ *" and "/ **".

The data types include integer, floating-point, char, and Boolean. Integer is divided into four classifications: int type, short type, long type and byte type. Floating point includes float and double. The char type includes several special characters, "/b", "/t", "/n", "r", and "/u".

The behavior only considers the object behavior, including methods and interfaces. The main method is the entrance of the program, so we labeled the main method.

Table 1. The rules of code annotation

first classification	second classification	third classification	forth classification	label
object				/obj
	class			/cla
		String		/Str
		array	int array	/int[]
			String array	/Str[]
		VO		/VO
		set	List	/Lis
			ArrayList	/Arr
			HashMap	/HasM
			HashSet	/HasS
			Map	/Map
modifier	public modifier	public		/mod1
		private		/mod2
		protected		/mod3
		friendly		/mod4
		final		/fin
		static		/sta
		non-static		
	class modifier	abstract		/abs
	member variable modifier	transient		/tra
		volatile		/vol
	method modifier	synchronize		/syn
		native		/nat
variable				/var
	constant			/quan
Annotations	//			/com
	/*			
	/**			
datatype	integer	int		/int
		short		/sho
		long		/lon
		byte		/byt
	float	float		/flo
		double		/dou
	char			/cha
		special character	\b	
			\t	
			\n	
			\r	
			\u	
	boolean			/boo
	void			/voi

first classification	second classification	third classification	forth classification	label
action	object action			
		method		/meth
			main method	/main
		interface		/intf
Operators	arithmetic operator	addition	+	/+
		subtraction	−	/−
		multiplication	*	/*
		division	/	//
		modulo	%	/%
		increment	++	/++
		decrement	—	/—
	relational operator	equal to	==	/==
		not equal to	! =	/! =
		greater than	>	/>
		greater than or equal to	>=	/>=
		less than	<	/<
		less than or equal to	<=	/<=
	bitwise operator	and	&	/&
		or	|	/|
		not	~	/~
		xor	^	/^
	logical operator	and	&&	/&&
		or	||	/||
		not	!	/!
statement	Assignment statement			/assi
	Flow control statement	conditional	if	/if
			if else	/ifel
		loop	while	/while
			for	/for
			switch	/swi
			case	/case
			for each	/fore
	Interrupt control statement	break		/bre
		return		/ret
		continue		/cont

Operators include arithmetic operators, relational operators, bitwise operators, logical operators, mathematical functions, and expressions. The arithmetic operator contains the addition, subtraction, multiplication, division, modulo operators, and the increment and decrement operators. Relational operators contain equal to, not equal, greater than, greater than or equal to, less than and less than equal to six three categories. The bitwise operators include four classifications: and, or, not, xor. Logical operators include three classifications: and, or, not.

We make the following code according to the preparation of java code.

Usually we use the first three letters of the word as the keyword, for example, the integer variables int are labeled as int, the short integer variables are labeled as sho. We relabeled some repeated keywords, using the first 4 letters or the first 3 letters of the first word and the first one letter of the second word as the keyword, for example, the hashMap is labeled as HasM, the HashSet is labeled as HasS. The for is labeled as for, the for each is labeled as fore. We labeled the four letter words as itself, such as main. In addition, public, private, protected and friendly 4 modifiers cannot appear in a class or method or variable, so we labeled them as mod1, mod2, mod3 and mod4, and set the four variables can only appear in one in each class, method or variable of the modifier.

Once you have established the above code tagging rules, you can automate the annotation of the acquired code. Analyzing the results obtained by the label and code structure, which can construct a simple code ontology, describe the upper and lower relations between the elements in the code, attribute relations, etc., can provide important structural guidance for code generation.

3.2. **Code knowledge base construction.** The main function modules of the code repository include model building, code addition, code editing and element editing modules.

3.2.1. *Model building.* Model building is the framework of code knowledge base. The model building module is based on the first-level classification of the code marking rules and the logical relationship of the programming language, constructs the code system framework, describes the structure of the code classification at all levels and the relationship between them.

3.2.2. *Code add.* The code add module is a module that adds the new code to the knowledge base. For a new code, you need to decode the code and add it to the repository. The process of adding modules is that, first of all, you need to use a variety of static code review tool to review the new code to ensure that it is running correctly, and then label and deconstruction of the code in accordance with the structure. Finally mark function and then further deconstruct the code to the element granularity and storage.

3.2.3. *Code edit.* The code in the code knowledge base maybe needs to be edited for some reasons, for example, the writer decided to optimize the code or the code is failure. We can edit the code. After editing the code they need to add the process through the code. First, review whether the edited code can run, whether it conforms to the code specification, and then deconstruct it and store.

3.2.4. *Element edit.* The code is constantly evolving, the elements of the description of the code is increasing. In the construction of the knowledge base, if no corresponding elements or an element of the repeated phenomenon, you need to edit the elements. That is, edit the elements of the description of the code in order to describe the code better.

4. **Implementation and Application.** Based on the structure of the programming language and the logical relationship, the project team constructs the framework of the code system. The code ontology has 67 concepts and 12 attributes. It includes most of the Java 49 keywords, plus some non-keywords but in the Java code has an important role in the concept. The concept is divided into two categories: objects and modifiers. Objects in the object include: Object, class, String, array, int [], String [], Collection, ListList, LinkedList, Set, HashSet, LinkedHashSet, SortedSet, TreeSet, Map, SortedMap, TreeMap, HashMap, interface. The modifiers include 47 concepts, as shown in Table 2.

Using the above 67 concepts, it can represent most of the code types in Java. There are 12 description concepts of the property. The 6 attributes of the objects are nickname, English name, description, public, private and protected. There are six attributes of the modifier, including: nickname, English name, definition, public, private and protected.

Using the weather class in the "21 days learning of java 6 (fifth editon)" as an example, the main function of this class is to automatically determine whether the input data is in degrees Celsius or Fahrenheit and converted to another temperature representation.

Weather class is divided into four methods, namely main (String args []), fToC (int fahIn), cToF (int celIn) and getType (String temp []), where main (String args []), Main (String args []) is as the main method. The function of this method is to read the data entered by the console and initialize the Scanner object, and call each method in order to complete the conversion between degrees Celsius and Fahrenheit. First call the getType (String temp []) method, pass the variables passed by the console into the getType (String temp []) method, and determine whether the variable is Celsius or Fahrenheit. If it is a

TABLE 2. The concept of 47 modifiers in the ontology

implements	switch	short
abstract	case	long
new	break	byte
extends	continue	float
final	return	double
static	default	boolean
strictfp	instanceof	char
native	catch	super
synchronized	finally	this
transient	throw	void
volatile	throws	goto
if	try	const
else	assert	main
for	import	out
do	package	
while	int	

Celsius temperature, call the cToF (int celIn) method to convert the Celsius temperature to Fahrenheit, return to cel, and print the variable. If the temperature is Fahrenheit, call the fToC (int fahIn) method to convert the Fahrenheit temperature to Celsius, return fah, and print the variable.

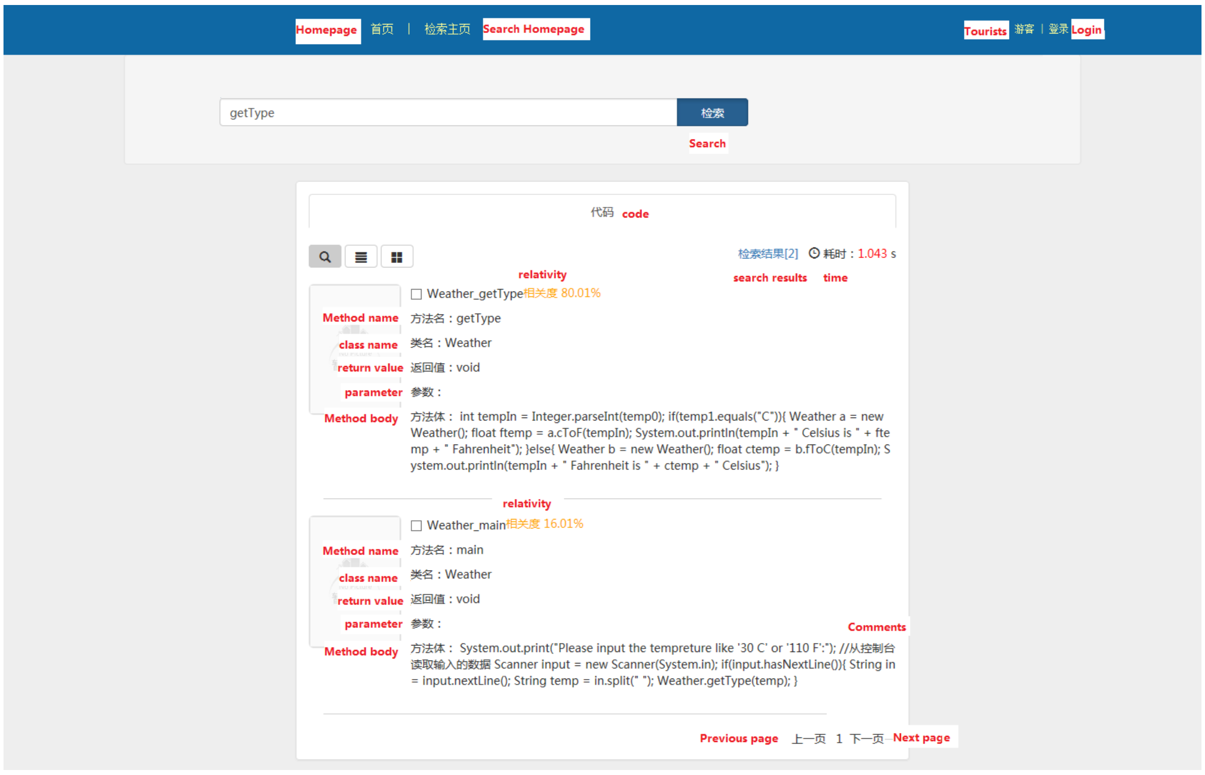
The system uses the code architecture framework to code the getType (String temp []) method code and label the results.

```
public/mod1 static/sta void/voi getType/meth (String/Str temp[]/Str[]){
    int/int tempIn/var = Integer.parseInt(temp[0]) /exp;
    if(temp[1].equals("C")/exp)/if {
        Weather/cls a/obj = new Weather();
        float/flo ftemp/var = a.cToF(tempIn)/exp;
        System.out.println/exp (tempIn/var + "Celsius is" + ftemp/var +
"Fahrenheit");
    }else/if{
        Weather/cls b/obj = new Weather();
        Float/flo ctemp/var = b.fToC(tempIn)/exp;
        System.out.println/exp (tempIn/var + "Fahrenheit is" + ctemp/var +
"Celsius");
    }
}
```

Analyzed by the system, save the java file to the set java entity class List <CodeB-ean> and save the parsing results to the mongodb database. The final system builds the index, and the user can retrieve the relevant code, such as "get type" or "get type" as shown in the "Get type method" and "input box". The search results are shown in Figure 3.

5. Conclusion.

Aiming at the problem of code generation at this stage, this paper proposes a method of constructing code knowledge base for software generation. By studying the rules and methods of code tagging, this paper uses natural language processing technology and machine learning method to reconstruct and utilize all kinds of codes. On the basis of constructing the code system framework, combine the code marking rules and semantic annotation technology to semantic annotation code, to achieve the structure of the code resources, semantic synchronization. Experiments show that the code knowledge



检索结果[2] 耗时：1.043 s

☐ Weather_getType相关度 80.01%

方法名：getType

类名：Weather

返回值：void

参数：

方法体： int tempIn = Integer.parseInt(temp0); if(temp1.equals("C")){ Weather a = new Weather(); float ftemp = a.cToF(tempIn); System.out.println(tempIn + " Celsius is " + ftemp + " Fahrenheit"); }else{ Weather b = new Weather(); float ctemp = b.fToC(tempIn); System.out.println(tempIn + " Fahrenheit is " + ctemp + " Celsius"); }

☐ Weather_main相关度 16.01%

方法名：main

类名：Weather

返回值：void

参数：

方法体： System.out.print("Please input the tempreture like '30 C' or '110 F':"); //从控制台读取输入的数据 Scanner input = new Scanner(System.in); if(input.hasNextLine()){ String in = input.nextLine(); String temp = in.split(" "); Weather.getType(temp); }

FIGURE 3. Retrieve "getType"

base constructed by this method can improve the efficiency of software development, and also provide some useful reference for the customization of code generation.

This paper is only a preliminary study of software generation. After the construction of the code knowledge base is completed, the code model and software logic are set up, and then the software is used to build the code and the software logic. Logic knows the code model to generate the software that has reached the purpose of software generation.

REFERENCES

[1] X. Wang and H. Qian, Research on automatically generating C++ code from UML class and sequence diagrams, *Computer Applications and Software*, vol.30, no.1, pp.190-195, 2013.

[2] L. Ding and S. Xu, Automatic generation of java code based on SSH framework, *Computer Applications and Software*, vol.23, no.9, pp.72-77, 2014.

[3] D. Kong, F. Luo, W. Lin, L. Ge and M. Liu, Research on a velocity-based automatic code generation technology, *Computer Applications and Software*, vol.31, no.10, pp.20-23, 2014.

[4] X. Gong and Y. Liu, Research on construction of integrated semantic crawler, *ICIC Express Letters, Part B: Applications*, vol.7, no.7, pp.1591-1598, 2016.

[5] Y. Liu, X. Chen, Z. Sui, Y. Hu and Q. Zhao, Research on semantic method of library resources' organizing, *ICIC Express Letters*, vol.5, no.4(A), pp.1011-1017, 2011.

[6] Y. Liu, H. Shi, D. Zheng and Y. Huang, Study on semantic annotation for professional literature, *ICIC Express Letters, Part B: Applications*, vol.5, no.5, pp.1383-1389, 2014.

[7] Y. Liu, Z. Zhang and Y. Huang, Research and development of semantic annotation platform for scientific literature, *ICIC Express Letters*, vol.10, no.7, pp.1787-1794, 2016.

[8] Y. Liu, Y. Huang and Y. Wang, Research on the key technologies of Pyrios knowledge service platform, *ICIC Express Letters, Part B: Applications*, vol.6, no.5, pp.1323-1328, 2015.