ISSN: 2320-8848 (Online)
ISSN: 2321-0362 (Print)

International Journal for Management Science And Technology (IJMST)

Volume 4; Issue 03
Manuscript- 1

"GENERATION OF SOURCE CODE SUMMARY BY AUTOMATIC IDENTIFICATION OF METHOD AND CLASS STEREOTYPES"

Srija R.
Computer Science and Engineering,
Anna University, Chennai,
India

Chitti babu K.
Computer Science and Engineering,
Anna University, Chennai,
India

Ramyaa B. C.
Computer Science and Engineering,
Anna University, Chennai,
India

www.ijmst.com

March, 2016

Abstract

Source Code summarization is being one of the serious and an analytic part of documentation is taken as the main objective of our paper work. Developers now a days find it difficult to analyze and study the entire source code to infer its purpose and several tools for document generation has also been evolving for this purpose. And Now, in our paper we propose a technique that will be helpful in generating a documentation for the source code provided with some tweaks and turns with some of the existing paper work with our work. The primary motive of our paper is to incorporate the stereotype templates and for this a tool that has been proposed already in the previous works (Jstereocode) is used that will be useful in analyzing the methods and class stereotypes depending on how they have been invoked and using which the summary is generated. The tool used here has the capability to infer the structural artifacts of the methods and classes using a presented taxonomy of the stereotypes that has been formulated earlier.

Key Words: Code stereotypes, Java, Program comprehension

1. Introduction

The system design and the roles of those methods and classes are interpreted which reveals the design intent of the source code.

The Stereotypes were used in software modeling During software maintenance, developers often cannot read and understand the entire source code of a system and rely on partial comprehension, focusing on the parts strictly related to their task at hand. Here the summaries are generated using predefined fill- in- the- blanks sentence phrases which we refer to as templates and we construct these templates specifically for different method stereotypes. In Object Oriented programs the methods and classes have their own purpose with assigned duties in, visualization, and comprehension, but only recently were these perspectives unified and characterized to provide a general taxonomy for classifying methods and classes depending on the intent of them those in the system. The stereotype templates are based on C/C++ code, but it can be extended to any OO programming language. Likewise, tool support for stereotype identification is restricted to C/C++ systems – the only such tools are reported in, and they are not currently available. Since stereotypes are a re-emerging concept, both situations limit their application in program understanding, and also in software engineering research. We constructed the templates specifically for different method stereotypes. Stereotypes reflect the basic meaning and behavior of a method and include

concepts such as predicate, set, and factory. In previous work by the authors on method stereotypes, a fully automated approach to assign stereotypes to methods was developed and evaluated. The summaries can improve source-code comprehension and speed up maintenance in several ways.

In order to provide support for stereotype discovery in other OO languages we have use JStereoCode, an Eclipse plug-in for the automated identification of the Java method and class stereotypes. The tool takes as input a Java compilation unit and its parent project, extracts structural attributes via static analysis, and according to several predefined rules, classifies each method and class within a stereotype, which is added to the documentation of the artifacts (i.e., to their Java doc comment). These reports can be easily used by other analysis tools.

First, the short description assists developers in understanding the main behaviour of the method and conclude its relevancy to their current task. Second, the documentation summary describes the main elements of a method (e.g., external objects, data members and parameters modified) in a structured format that can be easily mapped to the code. Third, because the main side effect of the method is often caused by one or multiple calls, including an abstract summary of calls (the stereotype of calls) provides additional information about their roles..

2. Related Works

A. Current Scenario

During software evolution, depending on the task at hand, developers need to understand relevant parts of the code. In consequence, developers often spend more time reading code [1] than writing it. Good leading comments help when reading code, by providing developers with at least a superficial understanding of the source code artifact that they describe. However, outdated or missing comments are very common and developers often must read more of the code or turn to external documentation in order to gain any understanding of the code relevant to their task. An obvious solution to this problem would be enforcing the creation and continuous update of internal documentation. While such a solution may work with new code, it will likely not work on existing, poorly-documented code. A more suitable approach is automatically generating summaries that describe the code. Such summaries can be used for re-documentation and will help developers in quickly understanding existing code.

B. Similar Works

Automated summarization of source code has been a topic of recent interest for multiple researchers. Sridhara et al. proposed techniques to automatically generate natural language comments for Java methods, sequences of statements and formal parameter using NLP. McBurney and McMillan proposed generating documentation summaries for Java methods that include information about the context of methods using call graph information and NLP. Moreno et al. use method stereotypes and class stereotypes to generate natural-language summaries for Java classes. The approach uses method and class stereotypes to filter out irrelevant methods to the class stereotype. While we also use method stereotypes as applied in [, the main contribution of our paper is to generate documentation summaries for methods, not classes. Thus, we generate a much finer granularity of documentation then that for a class. Our technique uses method stereotypes to generate a standard summary for each method.

Similarly, TaxTOOL classifies C++ classes according to a very broad taxonomy that combines several OO characteristics, included but not limited to data types, accessibility, polymorphism, exception handling, and inheritance [2]. The catalogued classes are stored as entries in a repository to be used in other development tasks. More recent work describes a tool (namely, TaxTOOLJ) that extends the taxonomy to Java code [1]. It is important to notice that this class catalogue is not meant to capture the design intent of classes. Once again, neither of these tools is freely available.

3. Java Code Stereotypes

As a first step to implement a stereotype identifier for Java software, we adapted existing definitions for C++ method and class stereotypes.

A. Method Stereotypes

Method stereotypes provide the general responsibility of the methods within a class, which is derived by a set of rules referring their content. We adapted and refined the stereotype taxonomy proposed for C/C++ methods in order to satisfy the characteristics of the Java code. The taxonomy was extended from 15 to 17 method stereotypes, categorized as follows: structural, when the method returns information about the object's state (accessors), or modifies it (mutator); creational if the method creates or destroys objects; collaborational, when the method determines how objects interact or are controlled in the system; and degenerate, in any other case. A method has a primary stereotype from any category and can have an optional secondary stereotype from the collaborational category.

In a nutshell, the new stereotypes describe methods that provide control logic by invoking only local methods (local controller) and methods declared without an implementation (abstract). This latter stereotype differs from the empty one, which describes a method with no statements in its implementation. The two new stereotypes are part of the collaborational and degenerate categories, respectively.

A general description of the resulting method stereotypes is presented in Table 1

Table1. Method Stereotypes

Category		**Stereotype**	**General description**
Structural	*Accessor*	Get	Returns a local field directly
		Predicate	Returns a Boolean value that is not a local field
		Property	Returns information about local fields
		Void-accessor	Returns information about local fields through the parameters
	Mutator	Set	Changes only one local field
		Command	Changes more than one local fields
		Non-void command	Command whose return type is not void or Boolean

Creational	Constructor	Invoked when creating an object
	Destructor	Performs any necessary cleanups before the object is destroyed
	Copy-constructor	Creates a new object as a copy of the existing one
	Factory	Instantiates an object and returns it
Collaborational	Collaborator	Connects one object with other type of objects
	Controller	Provides control logic by invoking only external methods
	Local controller	Provides control logic by invoking only local methods
Degenerate	Abstract	Has no body
	Empty	Has no statements
	Incidental	Any other case

B. Class Stereotypes

Class Stereotypes represent the general intent of classes in the system's design. A class is recognized as Data Provider if it contains mostly accessor methods and has low control of classes. Quantitatively, these conditions are expressed as follows:

$$|accessors| > 2\ |mutators|$$
$$\wedge$$
$$|accessors| > 2\ |controller \cap factory|$$

factory methods in the class, and |accessors| and |mutators| represent the number of methods in the accessor and mutator categories, respectively.

We modified several of the original conditions for three reasons: (i) to consider the method stereotypes adaptations; (ii) relax some rules that resulted in unclassified classes; and (iii) remove the overlap between rules that caused classes with multiple stereotypes.

Table 2. Class Stereotype

Stereotype	General description
Entity	Encapsulates data and behavior. Keeper of data model and business logic.
Minimal entity	Trivial Entity that consists entirely of accessor and mutator methods.
Data provider	Entity that consists mostly of accessor methods.
Commander	Entity that consists mostly of mutator methods
Boundary	Communicator that has a large percentage of collaboration methods, a low percentage of controller, and not many factory methods.
Factory	Consists mostly of factory methods.

Controller	Controls external objects - the majority of its methods are controllers and factories.
Pure controller	Consists entirely of controller and factory methods.
Large class	Contains a high number of methods that combine multiple roles, i.e., it consists of accessors, mutators, collaborational, and factory methods.
Lazy class	Its functionality cannot be easily determined. It consists mostly of incidental, and get or set methods.
Degenerate	Very trivial class that does very little - it consists mostly of empty, and get or set methods.
Data class	Degenerate behavior - it has only get and set methods.
Pool	Consists mostly of class constants and a few or no methods.

4. Design And Implementation

JStereoCode is a plug-in for the Eclipse IDE. The following are the steps used : First input a compilation unit(labeled) that has the source code and its project ( )The compilation unit is parsed (1) to obtain its Abstract Syntax Tree (AST). For every class found in the AST, each one of its methods is examined to extract the structural attributes (2) required to identify the method's stereotypes (3). These stereotypes are used together with some code metrics (4) to

determine the stereotype of the class (5). Right after the analysis, the stereotypes identified for each artifact are added (6) to the corresponding Javadoc (@) and reports (), which represent the plug-in's outputs as shown in figure3

A. Core

The core component consists of the method of retrieving the method and class stereotypes with the help of AST walker analysis, which uses an extended version of the AST visitor provided by the Eclipse API. In order to determine the stereotype of a class, JStereoCode needs first to identify the stereotype of all its methods and (for Large Classes) extra information about the project, namely the average and standard deviation of methods by class (see Table 2).

B. GUI

The GUI Component handles the events fired by the user through the Eclipse GUI, Specifically the JStereocode options are displayed in the contextual menu of each compilation unit, package and project in the Project Explorer view. In consequence, the stereotype identification can be performed for a single compilation unit or for all the compilation units in a package or project. Multiple elements can be analyzed by using the stereotype option when selecting more than one compilation unit, a package, or the project

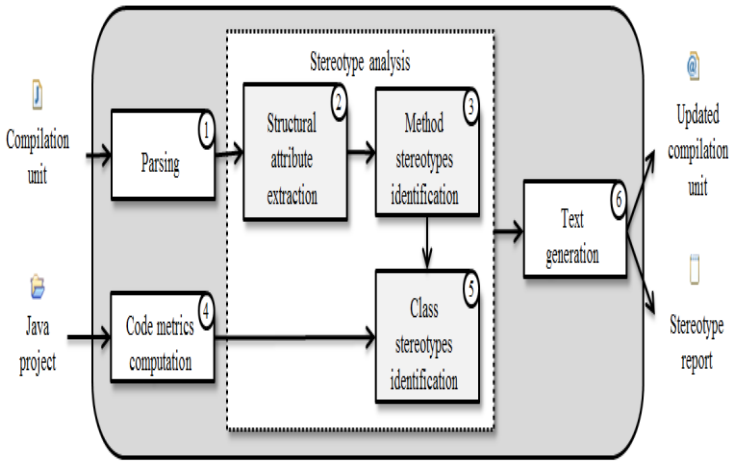


Figure 3. Stereotype identification in JStereoCode

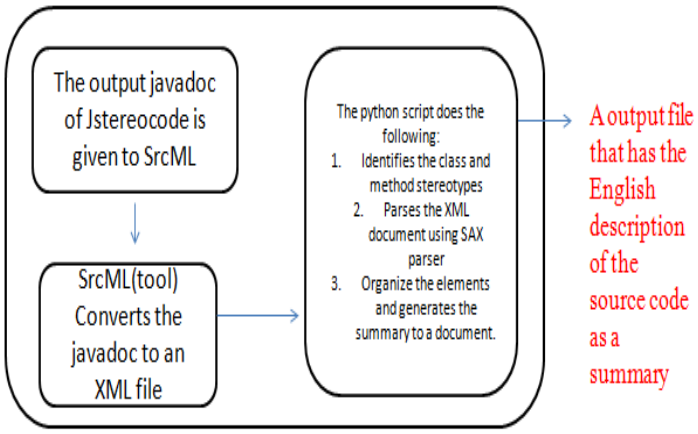
C. Text

The text component has two main functions: update the Javadoc comment of the methods and classes analyzed, and generate reports about the stereotypes identified.

The stereotypes for each artifact are added to the respective Javadoc, under the tag @stereotype. If the Javadoc does not exist, it is created. If the @stereotype tag already exists, there are two options: replacing its value with new stereotype, or creating another tag that reflects the most recent stereotype. To differentiate the tags, a date stamp can be added each time the analysis is run. JStereoCode also provides the option of removing the stereotype tags from the Javadoc comments. This option is useful if one wants to study if the stereotype of the code elements changes over time, as the code evolves

5. The Approach

In our approach the output of the Jstereocode which is a JavaDoc file is generated and given as an inout to the srcML tool which is an XML representation for source code, where the markup tags identify elements of the abstract syntax for the language. The srcml program is a command line application for the conversion source code to srcML, an interface for the exploration, analysis, and manipulation of source code in this form, and the conversion of srcML back to source code. The current parsing technologies supports C/C++, C#, and Java



As a result the converted XML formatted file of the source code is given as an input to the python script which uses the SAX parser and keeps track of the class and method stereotypes using the XML elements of the source file and our script which has a specific format to generate the summary is generated successfully and it is stored as a document and the work flow summary generation is shown in the figure4.

To automatically construct a summarization, two issues must be addressed. The first is determining what information should be included in the summary. The next is to present this information efficiently. Previous studies on source-code summarization have investigated the former issue in depth and this information is used as

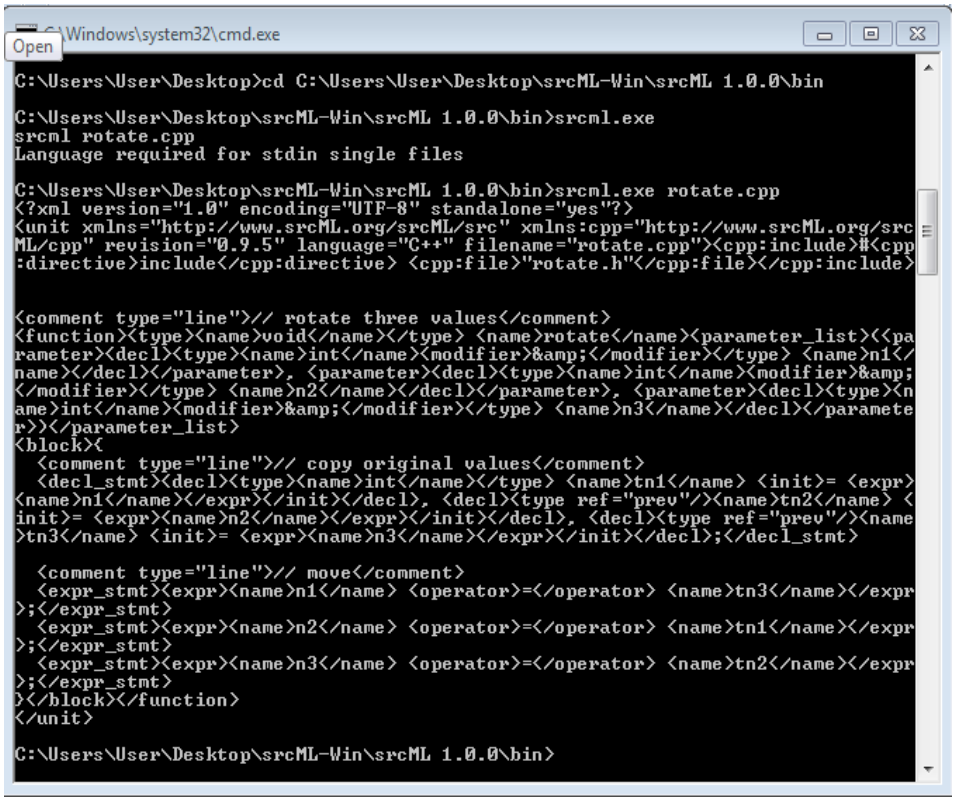


Figure 5. Output of srcML

a set of guidelines for building our automatic source-code summarization tool.

The documentation summaries are generated through two steps. First, the stereotype for all methods are identified and augmented to the source code and finally the given source code with the method stereotypes is given to the srcML tool which converts the java file into an equivalent xml file which is then parsed using the xml SAX parser(python script).

5. Acknowledgment

This paper reflects some of the works and implementations of the Jstereocode and Jsummarizer by L Moreno, A Marcus with the design architecture that has been implemented in this paper. Any opinions, findings regarding the stereotype implementation in this paper are those of these authors and do not necessarily reflect our approach in this paper.

6. Conclusion

This paper proposes an approach to generate summaries for java. This approach is highly scalable and can be generalized to other OO programming languages. We believe that the summaries can support comprehension during maintenance. We expect that JStereoCode will serve as an aid in software development and software engineering research. Since our tool

automatically identifies code stereotypes that reflect the design intent of artifacts, its main benefit is design/architecture recovery. We expect that JStereoCode will facilitate research regarding the application of stereotypes in other software engineering tasks. Unexplored and interesting areas include feature location, defect prediction, refactoring, traceability link recovery, defect prediction, etc. Then by parsing the xml file a summary about that classes and methods are displayed.

We also planned to incorporate the same method for other languages. Moreover we will work on these to make it more customizable regarding the identification rules and report generation

References

- Documentation Generation via Source code Summarization of Method Context",ICPC, 2014, pp. 279-290(2014)
- Dragan, N., Collard, M.L., Maletic, J.I. "Reverse Engineering
- Method Stereotypes", ICSM, 2006, pp. 24 - 34(2016)
- Abid, N., Dragan, N., Collard, M.L., Maletic, J.I., "Using Stereotypes in the Automatic Generation of Natural Language Summaries for C++ Methods" in the Proceedings of 31st IEEE International Conference on Software Maintenance and Evolution (ICSME'15), Bremen, Germany, Sept 29 - Oct 1
- Maletic, J.I., Collard, M.L., "Exploration, Analysis, and Manipulation of Source Code using srcML", in the Proceedings of 37th International Conference on Software Engineering (ICSE '15), Technical Briefing, Florence, Italy, May 19, 2015
- Collard, M.L., Decker, M. Maletic, J.I., "srcML: An Infrastructure for the Exploration, Analysis, and Manipulation of Source Code", in the Proceedings of the 29th IEEE International Conference on Software Maintenance (ICSM'13) Tool Demonstration Track, Eindhoven, The Netherlands, Sept. 22-28, 2013, pp.
- Jstereocode: Automatically identifying class and method stereotypes of java code.L Moreno, A Marcus ,2012
- Dragan, N., Collard, M.L., Maletic, J. I., "Automatic Identification of Class Stereotypes", in the Proceedings of the IEEE 26th IEEE International Conference on Software Maintenance (ICSM'10), Timisoara, Romania, Sept 12 - 18, 2010
- Dragan, N., Collard, M. L., Hammad, M., and Maletic, J. I.,"Using stereotypes to help characterize commits", in Proceedings of 27th IEEE International Conference on Software Maintenance (ICSM'11), Williamsburg, VA, 2011