Sindh Univ. Res. Jour. (Sci. Ser.) Vol.47 (3): 535-540 (2015)

SINDH UNIVERSITY RESEARCH JOURNAL (SCIENCE SERIES)

Identifying Software Features as Birthmark

S. NAZIR[++], S. SHAHZAD, Q.U.A. NIZAMANI**, R. AMIN* M. A. SHAH***, A. KEERIO**

Department of Computer Science, University of Peshawar, Pakistan

Received 1st April 2014 and Revised 2nd August 2015

Abstract: Software can be dissected into features under various categories, such as syntactic features and semantic features that contain all the information related to the construction and functionality of the software. These features have intrinsic connections in-between which uniquely identify their working in a particular piece of software. These characteristics of software are known as a birthmark. A lot of research has been conducted to identify different techniques defining software birthmarks. A software birthmark uniquely identifies software and hence used to detect software theft and piracy. This research aims at identifying as much software features as possible and proposes a software feature model based on features lying under different categories. Each software will poses a unique value set for features identified according to that feature model. These value sets can then be used to detect similarity among software programs to detect software theft and piracy.

Keywords: Software features, Birthmark, Piracy and theft detection

1. INTRODUCTION

Software features present are all the information related to a software system. Software features collectively define and support software system and its functionality. Some features are also the formal representation of a user centric organization of software or program functionality. These features are virtually interlinked performing different operations and due to these operations the software or program is considered to be a functional software system. The idea of identifying software features began from the source code theft detection (Aiken, 2005) One of the main problem with identifying software feature may be that the source code is always not available. (Tamada *et al.* 2014) considered java byte code set as a software feature rather than source code. This is because a software feature is the well-known functional and non-functional formation of attributes and unique user-visible characteristics of software. There are different categories of software features which can be defined for a software. A clear understanding of these features and their organization into logical categories helps to understand the code. This understanding is important to identifying similarities among instance or copies of presumably same software application.

This paper presents a feature oriented dissection and analysis of software or program. That dissection is then used as a birthmark for theft and piracy detection. This study provides a roadmap and benefits to software engineers for the purpose of detecting piracy throughout the life cycle of software systems. The main advantage of this study over existing is that software theft can easily be detected as we have defined more possible features of the software. For the purpose of clarity in this research we identify software program features as all possible features of a software system, defined under any category. Also we identify software program as whole software or a module which is to be tested for theft detection and piracy.

The organization of the paper is as follows. Section 2 provides a related work of existing theft detection techniques and birthmark. Section 3 provides a detailed description of software features and theft detection. The main contribution of this study is given in section 4. Section 5 and 6 provide a brief discussion and conclusion respectively.

2. RELATED WORK

With the development of counter-techniques like watermark removal and destruction the usefulness of watermarks has been compromised. Also the use of advanced techniques such as obfuscation and optimization for semantic transformation of watermark had completely made it illogical to use software watermarks for this purpose. The concept of software birthmark was then developed as a technique to detect software theft. Tamada *et al.* (2014) proposed the first birthmark which consisted four different birthmarks, namely; constant value in field variable (CVFV), sequence of method calls (SMC), inheritance structure (IS), and used classes (UC). This birthmark technique has been successfully used by the industry for software theft detection. Also, Zeng *et al. (*2012.) presented a semantic based abstract interpretation framework for

[++]Corresponding Author S. Nazir, email snshahnzr@gmail.com
*Department of Mathematics, University of Peshawar
** Institute of Mathematics and Computer Science, University of Sindh, Jamshoro
***Department of Electrical Engineering, Sukkur Institute of Business Administration, Sukkur

software birthmark. Myles and Collberg (2004) presented a technique of "Whole Program Path Birthmarking" that was based on complete control flow of the software program. Bai *et al. (*2009) presented a dynamic K-gram based software birthmark for identification of origin of program.

Kalaoja, (1997) emphasised on the feature modelling of embedded software systems. Several studies exists on identifying different software features to define birthmark, such as feature selection model for software defect prediction Wang, *et al., (*2012) identified of steganography software based on feature matching Zheng, *et al., (*2012), identified of software theft based on multi attributes features Guo, *et al.,(* 2014) The purpose of using different features is to consider these features as a software birthmark. A software can be dissected in multiple categories of different software elements which can be termed as features of a particular software. A collection of all the features of a program may help to provide a more close and authentic identification of a software program and then ultimately be used for theft detection.

3. SOFTWARE FEATURES AND THEFT DETECTION

A software program is a collection of different software features of certain types. A clear understanding of these features and their organization into logical categories is another step further in understanding the code. This understanding of a specific program code eventually helps in identifying similarities among more than one instance or copies of presumably same software application (that is, the program). The identification of similarities hence facilitates piracy and software theft detection. Section 2 identifies some frequently used techniques for defining software birthmark based on one, two or a small set of features. Some of the techniques are applicable on program source code while others are meant to be used only with byte-code.

There are different categories under which software features can be placed. For example, functional features, which relate to the functional requirements of the user, for example, calculating profit on sales; structural features, which relate to the inner structure of the software, for example, number of functions in the software program; quality features, which are associated with the quality requirements of the software, for example, ease of use and reliability, etc. (Guo, *et al.,* 2014) provide a categorization as input software features, self-software features and output software features. (Cesare and Xiang, 2012). categorized the features into syntactic and semantic features. Syntactic features deal with the structure of the program, while semantic features deal with the meaning of the program. According to (Kang *et al.,* 1990) different program features are processed at different phases of a software program. In this sense features can also be categorized and identified as compile time features, load time features and runtime features.

This fine grained categorization of software program features provides a metaphor to identify an exhaustive set feature for any software system. These software features provide a lot of important information about the software system that they present. At the same time the software features that uniquely identify a software system are used in critical operations, for example in software theft detection. Several techniques related to software theft are used by researchers and industry, for example, program identification for software theft (Myles, 2006) static API trace based detection (Park, *et al.,* 2008), behavior based software theft detection (Wang, *et al.,* 2009) static instruction trace based theft detection (Park, *et al.,* 2008) heap graph for software theft detection (Chan, *et al.,* 2013) and customizing k-gram based birthmark for software theft detection (Lim, 2013) etc. All these techniques are based on some unique program features termed as software birthmark.

4. DISSECTION AND ANALYSIS OF SOFTWARE FEATURES AS A BIRTHMARK

In this section a software feature model is presented which is based on an exhaustive list of features that can be identified in a software program. Taxonomy of the categories of software features is presented in **(Fig. 1).** The taxonomy is designed keeping in view the generic sequence which is followed to analyze a software program. This generic sequence also presents the inherent relationships among different feature categories.

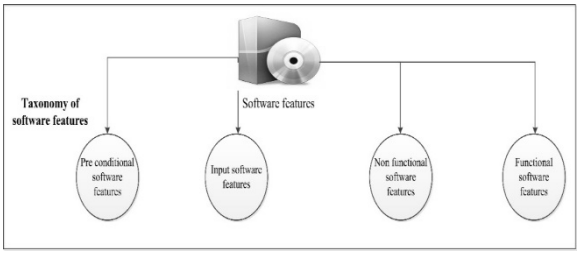


Fig.1. Representation of software features

(Fig,2) shows the software feature model. The model groups related software program features under four broad categories named as pre-conditional software features, input software features, non-functional software features and functional software features. A software program can be analyzed to identify and /or calculate specific values for these features. Each of these features may be assigned a quality value (for example, good, high, etc.) or a constant value (for

example 10, 25, etc.) depending upon the type of feature (either qualitative or quantitative). The resulting value set of a software program will be the birthmark of the software. The value sets of candidate software programs can then be compared to find out similarities and to detect software theft.

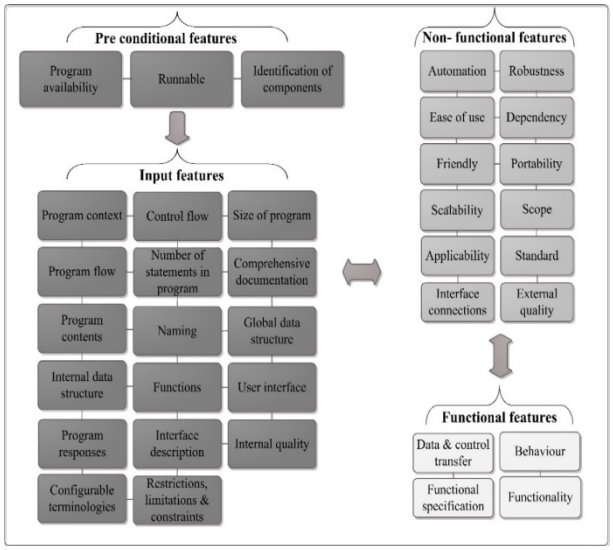


Fig. 2. Features as a birthmark

(Fig.3) shows the proposed process of comparing similarity of two programs. Process starts by analyzing software programs to extract features. Different techniques may be applied for feature extraction (depending upon the type of features). The techniques may be manual or automatic (description of these techniques and the resulting values is not in the context of this paper). After the extraction of features (values) the candidate programs will be compared. If the similarity among features is found above a predefined threshold value then the programs will be termed as similar otherwise dissimilar. In this way value sets (based on software feature model) of different software programs may be compared with the value set (based on software feature model) of the original copy of software program to detect piracy and theft.

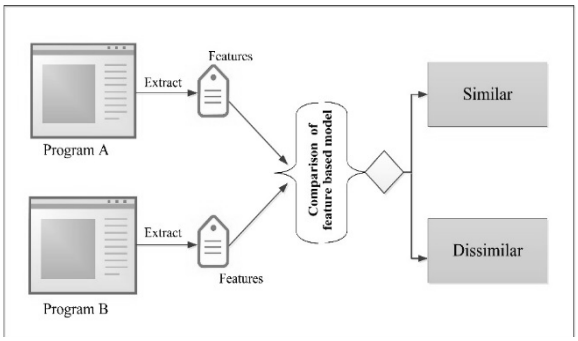


Fig.3. Program similarity checking

The success of the above mentioned process depends upon the identification of a considerable set of features which may be identified in an organized manner for a software program. The model is based on the collectiveness of features means the feature(s) independently cannot act as a birthmark. The following sections briefly define the identified software features under their specific category.

4.1 Pre conditional features

Details of the pre conditional features of a software program are given below.

4.1.1 Program availability

The first and the most important feature in detecting the similarity of software or programs is the availability of the original software program (and the candidate software which is to be analysed for theft detection). The program availability should be logically complete so as to be able to analyse it properly.

4.1.2 Runnable

When a program is to be checked for piracy, it should be runnable. An executable program is needed to properly analyse its relevant features for the purpose of checking piracy.

4.1.3 Identification of components

Different components inside the program are identified for feature extraction. A detailed description of each component should be contained within the structure of the program. Whole set of components and their interface should be clearly identified so that features can be traced in the program.

4.2 Input features

Input features category contains following features.

4.2.1 Program context

The context information and details of the program can be defined. It requires the primary inputs and outputs information about the program. The “big picture” of the program should be defined in order to clearly understand the program. The program is placed in the context of product and business (domain). Identification of the context also adds another attribute to detect similarity in software programs.

4.2.2 Program flow

Program flow refers to the order in which the program components execute. Flow of a software program can be identified with the help of data flow diagram, state transition diagram, and control flow diagram to check the similarity among software programs. A program’s sequence of flow can be compared with the other program sequence of flow that can present level of piracy.

4.2.3 Program contents

The contents of a program set of statements and functions. It includes the operations of organizing structure of the program, modification, deletion, insertion, etc., in the program. Comparison of programs on the basis of their contents also shows similarity among programs.

4.2.4 Internal data structure

Internal data structure of a program is passed among different components or modules of the program. The internal data structure also refers to the flow of data in a proper sequence. Two programs having same internal data structure also shows the similarity among programs.

4.2.5 Program responses

The similarity can also be detected on the basis of the output being generated by programs. Exactly same output also shows a degree of similarity among software programs.

4.2.6 Configurable terminologies

Configurable terminologies are the relevant terms associated with a program. It includes what types of inputs will be passed through a program, what operations will be performed by the program, and what results will be displayed by the program. The terminologies of a program can be compared with the terminologies of another program to check for piracy.

4.2.7 Control flow

The control flow of a program refers to the control order in which the program executes. Control flow breaks up the flow of execution of a program by employing decision making regarding the aspect for which the control is imposed. The control flow of the programs can be checked for finding similarity in the programs.

4.2.8 Number of statements in program

Static characteristics can be identified using static analysis. Lines-of-code is a most commonly used static metric which will also help in identifying similarity.

4.2.9 Naming

The naming (variable name, functions names, etc.) used in one program can be compared with the naming of another program to check the extent of piracy. This naming differs from the configurable terminologies where configuration of terminologies is involved.

4.2.10 Functions

A function is module of code which performs a specific task. Functions usually take data, process it and give result(s). A function of one program can be compared with another program to check its behaviour against piracy.

4.2.11 Interface description

The interface is the way of communication between components in a program. The configured interfaces of a program can be checked and compared with another program to show the level of piracy between the programs.

4.2.12 Restrictions, limitations and constraints

Restriction, limitation and constraints are the principles which limit the extent of something. These principles applied on one program can be compared with another program to check whether the second program is the copy one or not.

4.2.13 Size of program

The size of one program on a disk can also be easily calculated. The size of a program can also be measured in lines of code (LOC), kilo lines of code (KLOC) and so on.

4.2.14 Comprehensive documentation

Comprehensive documentation provides a detailed description about the program in which all the relevant information is provided. It presents precise and usable documents which helps one to identify the similarity between the programs.

4.2.15 Global data structure

Global data structure is the structure of data which is available in the major parts of the program. It can be compared with another copy or with the same program for checking the similarity.

4.2.16 User interface

The user interface is the way of communication of program with the user. The goal of the interface is to make the interaction easy, efficient and user friendly. Similarity among programs can also be detected by analysing the similarity in UI components of two software programs being checked for similarity.

4.2.17 Internal quality

Internal quality deals with the internal structure of the program and is about the design of the software. Internal quality is used to facilitate the process of a good and reliable product. It includes cohesion, low coupling, simplicity, generality and clarity. These features of a program can be compared with the same features of another program to show the level of similarity.

4.3 Non-functional software features

The non-functional software or program features are those features which are indirectly provided by a software program. These features sometime depend upon the input features of a software program. Details of the non-functional software or program features are given below.

4.3.1 Automation

Automation is the use of control for operating the system. The control of original and pirated (copy)

program can be checked for the purpose of checking duplication in the program.

4.3.2 Ease of use

The ease of use of a program shows how easily a program can be used. Ease of use also refers to the term usability, which is considered to be useful in customer's point of view. Both the original and copy programs can be checked to check the similarity in programs.

4.3.3 User friendly

A good software or program is the one which is user friendly and user can easily use it. User friendliness depends upon different characteristics. The evaluation of these characteristics demonstrates user friendliness of a program. Two pieces of code (programs) can also be compared on the basis of their defined user-friendliness.

4.3.4 Scalability

The scalability is the ability of a program to continuously function in order to meet customer requirements. The scalability of a program can be compared with the scalability of another program to check the extent of piracy.

4.3.5 Applicability

The programs should be checked in the context of application domain for which they are developed. If one of them is different from another, then both the programs are different.

4.3.6 Interface connection

The interface connection is a specific connection in which the interfaces of the program are connected with each other forming a completed software or program. A connection represents a specific session with interfaces. An interface connection is able to provide information describing the connection. Interface connection of both the original and copy (same) programs should be checked to show its piracy

4.3.7 Robustness

The robustness of a program is the ability to handle errors during execution. Robustness can also be measured empirically and automatically.

4.3.8 Dependency

The dependencies can be formed through the analysis of source code or byte code. The present research has focused on a single dependency type that can be syntactic or logical for the relationship of failure proneness to the dependencies of software. Further the research has also focused on the human and organizational factors that are based on quality for the failure proneness of the software.

4.3.9 Portability

The portability of a program is the usability of the program in different environments. When a program or software of the same functionality is adopted or produced in more than one place. The portability of a program can be compared with the portability of another program to check whether the program is pirated or not.

4.3.10 Scope

The scope of the software or program is the activity which limits the program by defining behaviour. The scope documentation lists explicit program goals, deliverables, tasks and deadlines. The scope of both the original and copy or same program can be compared to checking the piracy in the programs.

4.3.11 Standard

The software or program standard enables interoperability's among the different programs developed by different developers. Software or program standards having definite terms, concepts, data, formats, styles of documents and different techniques.

4.3.12 External quality

The external quality is the property of a software or program that a user experiences it. It includes conformity, reliability, accuracy, easy to use, and robustness, etc.

4.4 Functional software features

4.4.1 Data and control transfer

The data inside the program can be transferred from one part of the program to another part by calling some set of statements. On the other hand, control transfer defines the flow of execution.

4.4.2 Functional specification

The functional specification is the essential requirements regarding program. The functionality of both the programs is compared to check the extent of piracy in programs.

4.4.3 Behaviour

The behaviour is the action of a software or program. The behaviour of programs can be compared to find the similarity in software.

4.4.4 Functionality

The functionality of a program is the aspects of what a program or software can do for a user or customer.

5 RESULTS AND DISCUSSION

The present study is an endeavor towards identifying diverse software program features which can be used collectively as birthmark. Software program features can be categorized into four types of features that include pre conditional features; input features; non-functional features and functional features. Different aspects of a program or software features can be checked for showing the similarity in programs or checking the extent of piracy in programs. These feature categories of software program are further divided into

different features. Identification of a large set of features ultimately leads toward better detection of similarity with other software programs. In this study three feature of a program are considered as pre conditional features, seventeen features are considered as input features, twelve features are considered as non-functional features and four features are considered as functional features.

6 CONCLUSION

Program similarity detection can be greatly enhanced by identifying varying number of features which provide identification to software programs. Features of two different programs are compared to check whether the programs are copy of each other and are the same or not. The model is based on the collectiveness of features and the features independently are not considered as a birthmark. This approach can be used with different types of birthmarks. Also, in future the presented software feature model may be extended to add more features into the defined categories. Identification of specific techniques (qualitative and quantitative) to detect software features is also a task to further formalize the model.

REFERENCES:

Aiken, A. (2005). "Moss: A system for detecting software plagiarism," Univ.of California–Berkeley. http://www.cs.berkeley.edu/aiken/moss.html, accessed on March 2014.

Analysis (FODA) Feasibility Study," (2000) Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213.

Bai, Y., X. Sun, G. Sun, X. Deng, and X. Zhou, (2009) "Dynamic K-gram based Software Birthmark," in *19th Australian Conference on Software Engineering*, 644-649. Perth, WA,

Chan, P. P. F., L. C. K. Hui, and S. M. Yiu, (2013) "Heap Graph Based Software Theft Detection," *IEEE Transactions on Information Forensics And Security*, vol. 8, 101-110.

Cesare S. and Y. Xiang, (2012). *Software Similarity and Classification*. New York Dordrecht: Springer London Heidelberg,

Guo, Y., M. Wang, and Y. Luo, (2014) "Identifying Software Theft Based on Classification of Multi-Attribute Features," *Journal of Software*, vol. 9, 1401-1411.

Kalaoja, J., E. Niemela, and H. Perunka, (1997) "Feature Modelling of Component-Based Embedded Software," in *Eighth IEEE International Workshop on incorporating Computer Aided Software Engineering*, 444-451.

Kang, K. C., S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, (1990) "K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213.

Lim, H. I. (2013) "Customizing k-Gram Based Birthmark through Partial Matching in Detecting Software Thefts," in *IEEE 37th AnnualComputer Software and Applications Conference Workshops (COMPSACW)*, 1-4.

Myles, G. M. (2006) "Software theft detection through program identification," in *Department of Computer Science*. vol. Doctor of Philosophy: The University of Arizona, 1-351.

Myles G. and C. Collberg, (2004) "Detecting Software Theft via Whole Program Path Birthmarks," in *Information Security*. vol. 3225: Springer Berlin Heidelberg, 404-415.

Park, H., S. Choi, H.I. LIM, and T. Han, (2008), "Detecting Java Theft Based on Static API Trace Birthmark," in *Advances in Information and Computer Security*. vol. 5312: Springer Berlin Heidelberg, 121-135.

Park, H., S. Choi, H.I. Lim, and T. Han, (2008) "Detecting code theft via a static instruction trace birthmark for Java methods," in *6th IEEE International Conference on Industrial Informatics*, 551-556.

Tamada, H., M. Nakamura, and A. Monden, (2004) "Design and evaluation of birthmarks for detecting theft of Java programs," in *Proceedings of IASTED International Conference on Software Engineering*, Innsbruck, Austria, 569-575.

Wang, X., Y. C. Jhi, S. Zhu, and P. Liu, (2009) "Behavior based software theft detection," in *Proceedings of the 16th ACM conference on Computer and communications security* Chicago, Illinois, USA: ACM, 1-11.

Wang, P., C. Jin, and S.-W. Jin, (2012) "Software Defect Prediction Scheme Based on Feature Selection," in *2012 Fourth International Symposium on Information Science and Engineering*, Shanghai, 477-480.

Zeng, Y., F. Liu, X. Luo, and S. Lian, (2012.) "Abstract interpretation-based semantic framework for software birthmark," *Computers & Security*, vol. 31, 377-390.

Zheng, Y., F. Liu, X. Luo, and C. Yang, (2012) "A Method Based on Feature Matching to Identify steganography software," in *2012 Fourth International Conference on Multimedia Information Networking and Security*, Nanjing , 989-994. Nanjing ,