Commentary
Volume 11:2, 2022

Accounting & Marketing

ISSN: 2168-9601 Open Access

Insights on Identifying Key Classes in Software Systems Using Entropy-based Metrics

Steve Oakes*

Department of Marketing, University of Liverpool Management, UK

Description

The functions of the software are increasing as the software system is constantly updated, and the structure is becoming increasingly complicated. In general, the high performance and reliability of a specific software are inextricably linked to its ongoing maintenance. However, because developers are constantly being replaced, new developers who want to maintain software must be familiar with the structure and function of software, which adds a significant time overhead to software maintenance. As a result, the field of software engineering has gradually focused on how to maintain software efficiently. When maintaining unfamiliar software, developers should first understand the structure and main functions of the software, which can take up to 60% of the total time

Software development documents are the traditional way for developers to understand software. However, the abilities of developers in a team vary, and development documents may not be easily understood by all members. As a result, development documents alone are insufficient for developers to gain a quick understanding of the software. Many software entities, such as attributes, methods, classes, and packages, are included in object-oriented (OO) software [1,2]. The interaction between these entities completes the software's operation. Classes serve as the foundation for information encapsulation in object-oriented software and are critical components of the software system. Although there are many classes in a software system, only a few classes perform critical software functions and are referred to as key classes.

In recent years, software has frequently been mapped into a complex network, with the software entities (attributes, methods, classes, packages, etc.) serving as network nodes and the coupling relationships between the entities serving as network links. A network model extracted from software is commonly known as a software network. As a result, the problem of identifying key classes in software can be recast as the problem of extracting key nodes in software networks. Researchers proposed various approaches for identifying key classes, with the majority of them identifying key classes by constructing static dependency networks. The network links represent the coupling relationships that may exist between all entities in the software. These coupling relationships include redundant relationships that may or may not exist when the software is executed. (ii) Ignoring the number of node interactions: The weights in the static dependency network constructed by analysing the software's source code are calculated based on the complexity of the modules and the number of method calls; however, such a quantitative relationship cannot objectively reflect the true coupling strength between nodes when the software runs [3-5].

To obtain the coupling relationship of the classes generated by method calls while the software is running, we must insert a marker in each method of each class to record the call path. Javassist, a byte code rewriting tool, can assist us in accomplishing this goal. Before we get into how we get class-level call information, we need to define a few terms. Instrumentation in Java: Instrumentation refers to a separate agent programme that can be used to monitor and assist the application programme running on the JVM. The "javaagent" parameter specifies the agent programme in the "premain" method of instrumentation. We can obtain the class-level call information generated by calling all methods while the software is running by performing the aforementioned operations. To execute all of the program's functions, we use automatic execution rather than manually clicking the button. To begin, the programme must be launched in order to display the Java GUI software's window. Then, using the windows, we obtain the component list and insert the event components from the component list into the event sequences. Some components may contain sub-components; therefore, we must use recursive methods to obtain all components. Finally, we execute the event sequences based on the event type to obtain the class's dynamic call graph.

Conflict of Interest

None.

References

1. Francis, Bill B., Iftekhar Hasan and Qiang Wu. "Are female CFOs less tax aggressive? Evidence from tax aggressiveness." *J Am Tax Assoc* 36 (2014): 171-202.
2. Ittner, Christopher D., and David F. Larcker. "Quality strategy, strategic control systems, and organizational performance." *Account Organ Soc* 22 (1997): 293-314.
3. Koester, Allison, Terry Shevlin and Daniel Wangerin. "The role of managerial ability in corporate tax avoidance." *Manag Sci* 63 (2017): 3285-3310.
4. Demerjian, Peter R., Baruch Lev and Melissa F. Lewis. "Managerial ability and earnings quality." *Acc Rev* 88 (2013): 463-498.
5. Van Delden S. H, M. SharathKumar, M. Butturini and L.J.A. Graamans, et al. "Current status and future challenges in implementing and upscaling vertical farming systems." *Nat Food* 2 (2021): 944-956.

***Address for Correspondence:** Steve Oakes, Department of Marketing, University of Liverpool Management, UK, E-mail: SteveOakes105@gmail.com.*

***Copyright:** © 2022 Oakes S. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.*

Received: 02-Feb-2022, Manuscript No. jamk-22-65614; **Editor assigned:** 04-Feb-2022, Pre QC No. P-65614; **Reviewed:** 18-Feb-2022, QC No. Q-65614; **Revised:** 23-Feb-2022, Manuscript No. R-65614; **Published:** 02-Mar-2022, DOI: 10.37421/2168-9601.2022.11.372.

How to cite this article: Oakes, Steve. "Insights on Identifying Key Classes in Software Systems Using Entropy-based Metrics." *J Account Mark* 11 (2022): 372.