Analysis of Existing Software Cognitive Complexity Measures

Sanjay Misra, Covenant University, Ota, Nigeria
Adewole Adewumi, Covenant University, Ota, Nigeria
Robertas Damasevicius, Kaunas University of Technology, Kaunas, Lithuania
Rytis Maskeliunas, Kaunas University of Technology, Kaunas, Lithuania

ABSTRACT

In order to maintain the quality of software, it is important to measure it complexity. This provides an insight into the degree of comprehensibility and maintainability of the software. Measurement can be carried out using cognitive measures which are based on cognitive informatics. A number of such measures have been proposed in literature. The goal of this article is to identify the features and advantages of the existing measures. In addition, a comparative analysis is done based on some selected criteria. The results show that there is a similar trend in the output obtained from the different measures when they are applied to different examples. This makes it easy for adopting organisations to readily choose from the options based on the availability of tool support.

KEYWORDS

Cognitive Informatics, Cognitive Information Complexity Measure (CICM), Metrics, Software Complexity

INTRODUCTION

Software metrics play an essential role in the software development process, since; it aids the software developer in ensuring the quality of software. Developers use these metrics in the entire software development process so as to better comprehend and assess the quality of systems that are being built. According to Fenton (1994), “it is not possible to develop the absolute measure. Instead, software engineers attempt to derive a set of indirect measures that lead to metrics that provide an indication of quality of some representation of software”. Similarly, Sommerville (2015) asserts that, “the quality objectives such as performance, reliability, availability and maintainability are closely related to software complexity”.

IEEE (1990) defines software complexity as, “the degree to which a system or component has a design or implementation that is difficult to understand and verify”. Fenton (1997) identified two categories of software complexities which include computational and psychological complexity. According to Fenton (1997), “computational complexity is related to algorithm complexity and evaluates time and memory needed to execute a program while psychological complexity also referred to as cognitive complexity is concerned with evaluating the human effort needed to perform a software task”. The latter definition is also related to the comprehensibility of the software and
task and assesses the difficulty experienced in understanding and/or performing such a task. Several researchers have also provided alternative definitions of cognitive complexity. Henderson-Sellers (1996) defines cognitive complexity as, “those characteristics of software that affect the level of resources used by a person performing a given task on it.” Fenton (1997) defines it as, “a measure of the effort required to understand software”. Zuse’s (1998) definition states that, “software complexity is the difficulty to maintain, change and understand software”. It is worth noting at this point that metrics and measures are often used interchangeably in software engineering. It is because of the fact that both terms have approximately similar definitions. Pressman and Maxim (2015) explain the measure in software engineering context as “one that provides a quantitative indication of the extent, amount, dimension, capacity, or size of some attributes of a product or process while a metric is defined as a quantitative measure of the degree to which a system, component, or process possesses a given attribute”.

Cognitive informatics (CI), a multidisciplinary area of research is emerging. It includes research in the field of cognitive science, computer science, informatics, mathematics, neurobiology, psychology, and software engineering (Wang, 2002, 2004, 2005, 2006, 2007, 2009). The essence of the research in CI is due to the fact that, it tries to solve the common problems of two related area in a bi-directional and multidisciplinary approach (Wang, 2004). CI employs the computing technique in order to solve the problem of cognitive science, neurobiology, psychology, and physiology and on the other hand the theories of cognitive science, neurobiology, psychology, and physiology are used to investigate the issues and their solution in computing, informatics and software engineering. For instance, measurement in software engineering is a major issue, which is still in its nascent phase and needs a lot of efforts to standardize it (i.e. the measurement techniques for software engineering). In the last few years, a number of researchers have tried to address these challenges by combining the principles of cognitive science and measurement in software engineering. The growing number of proposals of cognitive complexity measures (Shao & Wang, 2003; Misra, 2006; Kushvaha & Misra, 2006; Misra, 2007; Benjapol & Limpiyakorn, 2008; 2009; Misra & Akman, 2010; Sharma & Kushwaha, 2010; De Silva et al., 2017) are the results of these efforts.

Cognitive Complexity refers to the human effort required to execute a task, or the difficulty experienced in comprehending the code or the information packed in it (Misra & Kushvaha, 2006). Understandability of the code is known as program comprehension and is a cognitive process and related to cognitive complexity. Briand et al. (2001) put it in this way, “cognitive complexity is the mental burden on the user (i.e. the developer, tester and maintenance staff) who deals with the code. It provides valuable information for the design of systems hence high cognitive complexity indicates poor design, which sometimes can be unmanageable”. In such cases, the maintenance effort increases drastically. In this respect, cognitive complexities are important in assessing the performance of the system. They refer to those characteristics of software, which affect the level of resources used by a person performing a given task on it (Zuse, 1998). The system with reduced cognitive complexity will not only improve the quality of the code but also reduce the future comprehension and therefore maintenance efforts.

In addition, to the aforementioned, complexity metrics can be used to improve software security (Moshtari et al. 2013). Initial studies by Shin & Williams (2008) showed a weak correlation between complexity metrics and security challenges in software. However, subsequent studies have been able to show how the count of source lines of code, code churn, developer activity metrics, coupling and cohesion metrics are able to predict the vulnerability in software with about 70% accuracy (Gegick et al., 2008; Shin et al., 2011; Chowdhury & Zulkernine, 2011; Shin & Williams, 2011; Shin & Williams, 2013).

In 2003, Shao & Wang (2003) proposed the first cognitive complexity measure based on the principles of cognitive informatics, which states that, “the nature of software is characterized by its informatics, behavioural, mathematical, and cognitive properties. Further, the complexity of software is the function of input, output and cognitive weights”. Following this approach, several proposals
Conceptual Model for Specialized Learning Systems within Organizations
Isabel Mendes, Henrique Santos and Celina Pinto Leão (2014). International Journal of Systems and Service-Oriented Engineering (pp. 19-34).
www.igi-global.com/article/conceptual-model-for-specialized-learning-systems-within-organizations/119657?camid=4v1a

Anticipating Requirements Changes-Using Futurology in Requirements Elicitation
www.igi-global.com/article/anticipating-requirements-changes-using-futurology/65563?camid=4v1a