Chapter 23
Modeling Software Development Process Complexity

Vyron Damasiotis
Department of Accounting and Finance, University of Applied Sciences of Thessaly – TEI Thessaly, Larisa, Greece

Panos Fitsilis
Department of Business Administration, University of Applied Sciences of Thessaly – TEI Thessaly, Larisa, Greece

James F. O'Kane
Edinburgh Napier University, Business, Edinburgh, UK

ABSTRACT

Modern software systems are growing increasingly complex, requiring increased complexity of software and software development process (SDP). Most software complexity measurement approaches focus on software features such as code size, code defects, number of control paths, etc. However, software complexity measurement should not only focus on code features but on features that cover several aspects of SDP in order to have a more complete approach to software complexity. To implement this approach, an extensive literature review for identifying factors that contribute to the complexity of SDP was performed and seventeen complexity factors were identified. As there were indications that the identified factors were not independent from each other but there were interrelations between them, statistical methods for identifying the underlined relations and refining them were applied, resulting to the final set of measures used in the proposed model. Finally, the proposed model has been tested in five software projects and the results were evaluated.

INTRODUCTION

As information technology (IT) evolves and becomes part of every aspect of everyday life, the demand for more powerful and reliable software becomes a necessity. However, this leads to software applications becoming larger and more complex, in terms of development and maintenance. In the future, this

DOI: 10.4018/978-1-7998-3016-0.ch023

tendency is expected to continue to increase (Da-Wei, 2007). As a result, almost half of IT projects cannot fulfill their initial requirements in terms of time, cost and quality (Bolat, Kusdemir, Uslu, & Temur., 2017; Altahtooth & Emsley, 2017). The consequences of increased software complexity have been identified and studied early either from the aspect of cost e.g. COCOMO (Boehm, 1981) or in other aspects of software project development such as schedule delays, quality deficiencies and increased error rates (Banker, Datar, & Zweig, 1989).Most current software complexity measurement approaches are based on aspects of code such as Lines of Code (LOC) (Park, 1992), McCabe Cyclomatic Complexity (McCabe, 1976) and Halstead complexity measure (Halstead, 1977).

In this research, we approach software complexity from a wider point of view identifying complexity aspects within the whole software development process including technical software development parameters, software development environment and various properties of the software system being developed. Furthermore, this research focuses on identifying factors that can be assessed at the early stages of software development process permitting the early estimation of expected SDP complexity and allowing the timely planning of appropriate actions to cope with it. The structure of this paper is as follows. In section 2, the current software complexity measurement approaches and the identification of SDP complexity factors are presented. The methodological approach followed in this research is given in section 3. Next in section 4, the steps and results of the statistical analysis performed as well as the model definition are presented. In section 5 the proposed model was applied in two case studies and the results are evaluated. Finally, in sections 6 and 7 the conclusions and limitations of this research are given.

CURRENT COMPLEXITY APPROACHES

Software complexity relates to both software product and to SDP. Several approaches of software complexity have been proposed by researchers according to the domain where they originated from.

Zuse (1990) approached software complexity from a programmer's psychological perspective and defined it as the difficulty to analyze, maintain, test, design and modify software. Along the same lines, Kushwaha & Mishra (2006) defined software complexity as the degree of difficulty to understand and verify a system or a component. Keshavarz, Modiri, & Pedram, (2011) stated that although there were different approaches for defining software complexity, most of them comply with Zuse's approach. Ribbers & Schoo (2002) in their research for complex software implementation programs, examined complexity through the prism of implementation complexity and identified three complexity dimensions: variety, variability and integration. Variety is defined as the different states a system can take. Variability of a system is defined as the dynamics of its elements and the interrelations between them. Finally, integration is referred to as the planned changes during the implementation program including IT systems and business processes.

Software engineers approach software complexity by examining various properties and code characteristics such as code size, number of software defects, development cost and time, number of control paths and frequency of operators and operands within the software. However, the existence of more classes, control flows or modules in code does not necessarily mean that is more complex than another one with less of these characteristics and therefore a more rigorous approach is needed (Ghazarian, 2015). In addition, Khan, Mahmood, Amralla, & Mirza, (2016a) in their research compared several complexity measurement models based on code characteristics and identified that different models produce different results as they capture different aspects of software code. Other researchers are studying software project