A Metrics-Based Model for Programming Skill

Oluwatoyin Adelakun
Department of Computer Science
University of Ibadan
Ibadan, Nigeria
toyin@sure-impact.com

ABSTRACT

Programming is a core skill in computing. Much of computing education and work in industry and research has programming skill is an important factor. Human subjects have varying levels of programming experience which affects their output. The inability to quantify programming skill threatens the validity of research results. Previous research has attempted to model programming skill and experience by finding a correlation between programmer characteristics and number of correct answers in a set of programming tasks. However, the model was not validated. This study is therefore designed to model the skill of programmers and to validate the model using experimental proof approach.

Undergraduate students were presented with programming tasks and a questionnaire to elicit programming experience and demographic data. Using Stepwise regression, a linear model was extracted which showed two variables: *lengthSchoolJava* and *schoolHours6Months* contributed most to the number of correct answers. Furthermore, factor analysis extracted two factors: *Java knowledge* and *Practical Java experience,* that explained the number of correct answers. For the future direction, we want to analyse programming error logs to improve our quality metric and confirm the extracted model by collecting more qualitative data.

CCS CONCEPTS

•General and reference~Cross-computing tools and techniques~Empirical studies

KEYWORDS

Programming Skill; Measurement; Metrics; Self-efficacy; Cause-effect.

1 RESEARCH MOTIVATION

The skill of subjects in a software engineering experiment is an important confounding element which still has no standard means of control [1] [2]. Programming skill is defined as "the ability to use one's knowledge effectively and readily in execution or performance of programming tasks" [3]. There is as yet no method for measuring programming skill independent of programming language. In empirical software engineering research, the effect of the skill of subjects has over time been inconsistently accounted for [2]. Many researchers have either ignored it or used proxy measurements such as years of experience. None the less, programming skill remains an important confounding factor to be in empirical software engineering [1] [4]. Not accounting for the skill of the subjects, impacts negatively on the external validity of such research [2].

Further, several software effort and quality measurement models historically suffer from inaccuracy. Despite years of research [5], the best solutions still combine two or more methods or, revert to expert judgement. The human element is a possible suspect in the failure of these models till date.

Developing a means of measuring the skill of programmers is hence an important research topic; the success of which, has far reaching benefits on software engineering teaching, research and practice.

ICER '20, August 10–12, 2020, Virtual Event, New Zealand

ACM ISBN 978-1-4503-7092-9/20/08
https://doi.org/10.1145/3372782.3407106

2 LITERATURE REVIEW

Research into measurement of programming skill is sparse. Two notable works - [2] and [3] describe an empirical approach using students and professionals. Similar approaches are used in both experiments. A questionnaire is used to collect demographic data and data relating to experience with programming. This data is then correlated to results from programming tasks presented to the subjects. After applying regression and factor analysis to the data, Siegmund [2] proposed a 5-factor model for measuring skill, while Bergersen developed a tool that can be used to measure the programming skill of JAVA programmers. These works are the most influential for this research. [6] and [7], further demonstrate the possibility of measuring and modeling skill by observing and recording specific behaviors.

2.1 Theoretical Background

This work seeks to conceptualize programming skill by relating observable (and measurable) programmer characteristics with the quality of work (program artefact) produced. This is based on the following theories:

1. **The Power Law of Skill Acquisition Theory:** This law states that as practice increases, numbers of errors and time to completion reduce. [8]

2. **Theory of Self-efficacy:** This is a self-reported proficiency. The literature is replete with assertions to the fact that self-efficacy correlates with skill [1] [2] [3].
3. **Theory of Measurement:** According to [9], "measurement is the assignment of numerals to objects events based on rules". The rules, mathematical properties and statistical operations only need to be made explicit.

2.2 Conceptual Model

The proposal is to apply the cause-effect model. We postulate that software quality is 'caused' by programmer(s) skill, which itself is 'caused' by innate programmer characteristics such as previous knowledge and experience. Our proposal fits the criteria for establishing a cause-effect model as detailed in [10].

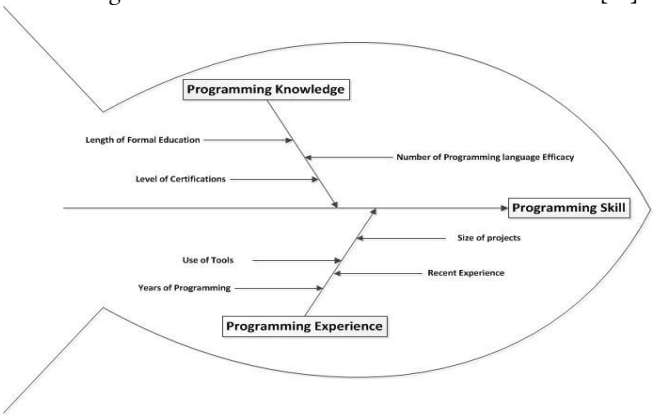

Figure 1: Cause-Effect Diagram for programming skill

3 METHODOLOGY

1. **Systematic Literature Review:** comprehensive search of literature in the programming skill measurement domain to discover the state of the art within the body of research.
2. **Questionnaires:** Short questionnaires are used to collect demographic and historical data about individual subjects' knowledge and experience of programming.
3. **Controlled Experiment:** Subjects are given a set of representative programming tasks to solve. Tasks includes timed coding and comprehension tasks. Source codes are saved on disk. Comprehension results are recorded in the printed task sheet. Subjects are monitored and data collated at the end of the experiment.
4. **Analysis:** Questionnaire is coded and Tasks are graded (Coding tasks: 0- code does not run, 1- code runs with bugs, 2- code runs without bugs and comprehension tasks: 0 - incorrect,1- correct). Data collected is analyzed to extract correlation models. Predictive variables will be extracted from this model and then plugged into a predictive model such a multiple linear regression and Bayesian network.

4 RESEARCH PROGRESS AND EXPECTED CONTRIBUTIONS

This work has established some relationships between observable characteristics of programmers and code produced. Codes were graded based correctness and this correlated with the subject's length of training and recent practical experience. These two variables were highlighted in the result of a stepwise linear regression which produced the model:

SumCorrect = lengthSchoolJava * .822 + schoolHours6Months * .489

A two-factor model was also extracted consisting of the following factors: *knowledge of JAVA* and *practical JAVA experience.*

4.1 Expected Contribution to Knowledge

Understanding what constitutes programming skill is important for several aspects of computing. It will contribute to improved teaching methods for computer programming, improved empirical software engineering research, potentially improve software estimation.

4.2 Expected Benefit from Doctoral Consortium

Some interim results were obtained from my initial data collection and analysis. Prosed next step is to collect further data to validate the interim model. We are also propose including programming error quotient and McCabe's complexity as metrics. At the doctoral consortium, I hope to receive comments around the methodology and design of the study.

REFERENCES

[1] S. Kleinschmager and S. Hanenberg. 2011. How to Rate Programming Skills in Programming Experiments: A Preliminary , Exploratory Study based on University Marks, Proceedings of the 3rd ACM SIGPLAN, pp. 15-24.

[2] J. Siegmund, C. Kastner, L. Jorg, S. Apel and S. Hanenberg. 2014. Measuring and Modeling Programming Experience. Journal of Empirical. Software Engineering.

[3] G. Bergersen. 2014. Measuring Programming Skill: Construction and validation of an Instrument for evaluating Java Developers. University of Oslo.

[4] J. Carver, L. Hochstein and J. Oslin. 2009. Identifying Programmer Ability Using Peer Evaluation: An Exploratory Study. OOPSLA.

[5] J. Zivadinovic, Z. Medic, D. Maksimovic, A. Damnjanovic and S. Vujcic. 2011. Methods of Effort Estimation in Software. Int. Symp. Eng. Manag. Compet. 2011, p. 417–422.

[6] C. Abuah, D. Schilder, M. Sherman and F. Martin. 2018. The Tablet Game: An Embedded Assessment For Measuring Students' Programming Skill In App Inventor. Journal of Computing in Schools and Colleges, vol. 33, no. 6, pp. 9-21.

[7] A. E. Tew and M. Guzdial. 2011. The FCS1: A Language Independent Assessment of CS1. In Proceedings of SIGCSE, Dallas.

[8] C. Speelman and K. Kirsner. 2005. Skill Acquisition: History, questions, and theories. In Beyond the Learning Curve: The construction of mind, Oxford University Press Online, 2005, pp. pp.27-66.

[9] S. S. Stevens. 1946. On the theory of Scales of Measurement. SCIENCE, vol. 103, no. 2684, pp. 677 -680.

[10] S. Khan. 2011. "Software Quality Metrics Overview," in Metrics and Models in Software Quality Engineering, Boston, Addison-Wesley Longman Publishing, p. 85–126..