Programming Languages Popularity and Implications to Testing Programmable Logic Controllers

Eduard Paul Enoiu[1]

[1]**Mälardalen University, Sweden , eduard.paul.enoiu@mdh.se**

ABSTRACT

The popularity of domain-specific programming languages has implications on how we test software in these domain industries. For example for programmable logic controllers five standard languages were defined and used in practice. Detailed data on popularity of these languages should show some implications on what languages to target when testing. We suggest that massive new data sources resulting from programmers may offer a new perspective on how we test domain-specific languages. By analyzing Google query volumes for search terms related to programmable logic controllers languages, we find patterns that may be interpreted as signs of actual usage in practice. Comparing with the current testing approaches proposed by researchers our results illustrate both potentials and threats on what is needed in reality when testing programmable logic controllers.

Keywords: plc, fbd, software testing

1 INTRODUCTION

The increasing volumes of data reflecting various aspects of our software engineering activities represent a new opportunity for scientists to address fundamental questions about the world of software. Software testing is a prime target for such quantitative studies. Recently according to Choi and Varian (2012), the search engine Google has begun to provide information on the queries for different search terms and how these queries change over time.

In the present study we investigate the possibility of analyzing search query data from Google Trends to provide new insights into the popularity of domain specific programming languages for programmable logic controllers. These insights are mapped to current testing approaches as the ones proposed by Enoiu et al. (2014); Jee et al. (2014). Here we suggest that within the time period we investigate, Google Trends data not only can reflect the current popularity of programming programmable logic controllers using different languages but may help researchers to focus on developing test generation techniques that are relevant in practice.

2 PROGRAMMING PROGRAMMABLE LOGIC CONTROLLERS

A programmable logic controller (PLC) is a hard real-time system used for automation of industrial systems, such as control management of trains. Under the IEC 61131-3 standard as described in John and Tiegelkamp (2010), PLCs can be programmed using different programming languages. IEC 61131-3 defines five programming languages for programmable control systems: function block diagram (FBD), ladder diagram (LD), structured text (ST), instruction list (IL) and sequential function chart (SFC).

Specifically IEC 1131-3 languages are in practice the programming standard in many industrial software systems, such as in the railway domain John and Tiegelkamp (2010). Such systems typically require a certain degree of certification, such as some level of structural coverage which must be demonstrated on the developed software. Some work like the one by Enoiu et al. (2013); Jee et al. (2005) on automated test generation has been shown to be capable of generating test data for achieving high coverage in the railway domain for a domain-specific language like 1131-3. Nevertheless, these techniques are focused on specific languages programming PLCs and there is little evidence that these approaches are relevant in practice for the programming languages actually used.

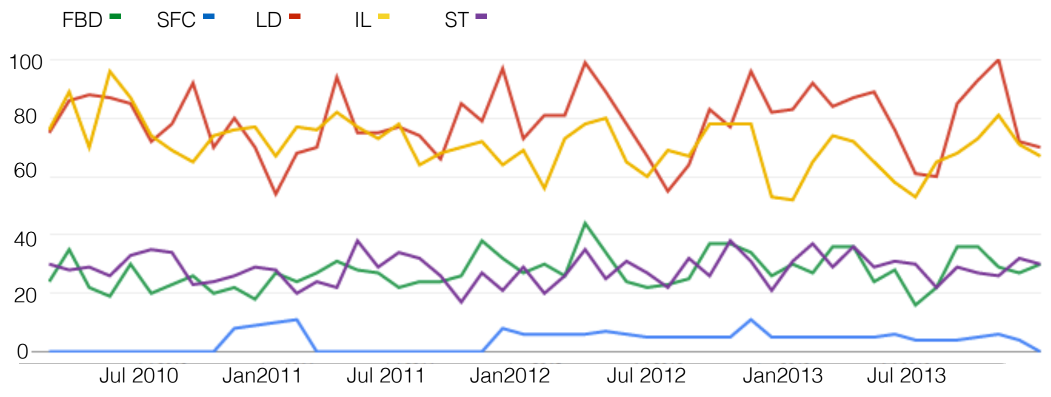

Figure 1. Graph showing the interest rate for different PLC programming languages between January 2010 and January 2014.

3 METHOD

To study the popularity of using these programming languages, we use publicly available historical Google search query data as a proxy for programming language usage, as has been done in some previous work (PYPL PopularitY of Programming Language index reference)[1]. The public nature of this data is an important feature of this method, as historical data on the usage of these languages is hard to find. The Google search query data is obtained from Google Trends tool and reports the relative number of Google search queries for a given search term. The use of search query data for the study of programming language popularity is useful because it is arguably representing a measure of user activity or interest related to the usage of the programming language.That being said, inactive users not using search engines for tutorials or advice are not counted in this approach. Nevertheless, popularity as measured by the search query data can be seen as representing a more meaningful metric of user activity or interest compared to registered programmers using programmable logic controllers.

The search query data obtained from Google Trends is presented in units of weekly search query frequency which are normalized such that the maximum data point in the selected time period corresponds to a maximum value of 100. We export the data from Google Trends in CSV format and import it in R for analysis. We store the interest rate, from FBD to ST, for each week. The data being normalized yields the share of interest, or popularity.

4 RESULTS

We show in Figure 1 search query data for the terms: function block diagram (FBD), ladder diagram (LD), structured text (ST), instruction list (IL) and sequential function chart (SFC). The data is normalized to the same scale in order to compare the relative number of search queries for all five languages. Whilst both ST and the FBD receive a modest number of searches, the LD and IL have a large growing interest. SFC language shows very low search queries for the entire period.

5 DISCUSSIONS

If our case study is a good indicator of how popular a programming language for programmable logic controllers is, the proposed index can help us to design better tools for developing and testing programs written in these languages. Testing software written in 1131-3 languages relies on both functional testing (i.e., tests are derived from requirements) and structural testing (i.e., tests are derived from the structure

[1]More details on PYPL can be found at the following link: http://pypl.github.io/PYPL.html

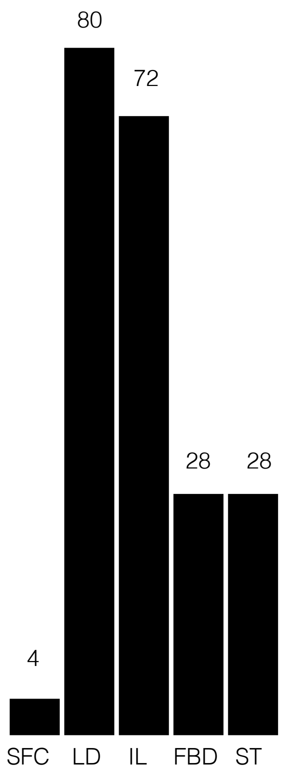

Figure 2. Average interest rate for different PLC programming languages.

Table 1. Automated Test Generation Approaches to Programmable Logic Controllers

Test Generation Approach	Targeted Language	Structural Measure	Year
CompleteTest (Enoiu et al. (2014))	FBD	Logic Coverage	2013
FBDTester (Jee et al. (2014))	FBD	Data-Flow Coverage	2013
FPCCTestGen (Wu and Fan (2014))	FBD	Data-Flow Coverage	2014

of the code in order to ensure some type of coverage). Functional testing and structural testing are complementary to each other and both are mandated and recommended to be applied to safety critical software John and Tiegelkamp (2010). In order to directly map the results in this study with the current research on 1131-3 languages, we are focusing only on structural testing and the attention each language got from the research community compared to the user interest as calculated by using search queries between 2010 and 2014.

There have been little research on structural testing for FBD programs. We reviewed three approaches to automated test generation for programmable logic controllers by searching Google Scholar with the following query: "structural testing AND IEC 1131-3". We identified three approaches published in peer reviewed journals between 2010 and 2014 that are targeting programmable logic controllers. In Table 1 we show these approaches by stating the targeted language, the coverage criteria used and the publishing year. From our results we can see that even if FBD has a lower average interest/popularity rate compared to LD and IL, is the only language targeted by the reviewed testing approaches. If this is true, and we show some evidence to this, we as researchers need to be more involved in an empirical investigation of programming languages for programmable logic controllers used in practice. The article's contribution to our knowledge of these IEC 1131-3 languages is at best modest. It is based on relatively incomplete data sources, rendering its quantitative conclusions dubious.

Nevertheless, we hope that, by the virtue of these inadequacies, this paper will spark further work on this topic. One of its conclusions that users are interested in other programming languages than the ones researchers are focusing on, seems to follow the record.

REFERENCES

Choi, H. and Varian, H. (2012). Predicting the present with google trends. *Economic Record*, 88(s1):2–9.

Enoiu, E. P., Čaušević, A., Ostrand, T. J., Weyuker, E. J., Sundmark, D., and Pettersson, P. (2014). Automated test generation using model checking: an industrial evaluation. *International Journal on Software Tools for Technology Transfer*, pages 1–19.

Enoiu, E. P., Sundmark, D., and Pettersson, P. (2013). Model-based test suite generation for function block diagrams using the uppaal model checker. In *Software Testing, Verification and Validation Workshops (ICSTW), 2013 IEEE Sixth International Conference on*, pages 158–167. IEEE.

Jee, E., Shin, D., Cha, S., Lee, J.-S., and Bae, D.-H. (2014). Automated test case generation for fbd programs implementing reactor protection system software. *Software Testing, Verification and Reliability*, 24(8):608–628.

Jee, E., Yoo, J., and Cha, S. (2005). Control and data flow testing on function block diagrams. In *Computer Safety, Reliability, and Security*, pages 67–80. Springer.

John, K.-H. and Tiegelkamp, M. (2010). *IEC 61131-3: programming industrial automation systems: concepts and programming languages, requirements for programming systems, decision-making aids*. Springer Science & Business Media.

Wu, Y.-C. and Fan, C.-F. (2014). Automatic test case generation for structural testing of function block diagrams. *Information and Software Technology*, 56(10):1360–1376.