Commentary

iMedPub Journals
www.imedpub.com

American Journal of Computer Science and Information Technology
ISSN 2349-3917

2022
Vol.10 No.8:002

Selecting the Best fitting Programming Language Ecosystem based on the Project requirements

Siamak Farshidi*

Department of Information and Computer Science at Utrecht University, Utrecht, The Netherlands

***Corresonding author:** Siamak Farshidi, Department of Information and Computer Science at Utrecht University, Utrecht, The Netherlands Email: Siamak432@yahoo.com

Received date: July 04, 2022, Manuscript No. IPACSIT-22-14383; **Editor assigned date:** July 07, 2022, PreQC No. IPACSIT-22-14383 (PQ); **Reviewed date:** July 20, 2022, QC No IPACSIT-22-14383; **Revised date:** July 27, 2022, Manuscript No. IPACSIT-22-14383(R); **Published date:** Aug 05, 2022, DOI: 10.36648/ 2349-3917.10.8.2

Citation: Farshidi S (2022) Selecting the best fitting programming language ecosystem based on the project requirements. Am J Compt Sci Inform Technol Vol. 10 Iss No.8:002

Description

Software development is a continuous decision-making process that mainly relies on the software engineer's experience and intuition. One of the essential decisions in the early stages of the process is selecting the best fitting programming language ecosystem based on the project requirements. As the selection of programming language ecosystems depends on the application to be developed and its environment, a decision model is required to analyze the selection problem using systematic identification and evaluation of potential alternatives for a development project. Software engineers make a sequence of design decisions while developing a software product. Each design decision can be analyzed as an episode of complex problem solving that relies on a substantial amount of knowledge and rationale. Design decisions in the software development lifecycle are significantly constrained by former decisions and lead to additional constraints on future decisions. Making informed design decisions in different phases of the software development lifecycle has critical impacts on the success of a software product. Over the last decades, thousands of programming languages belonging to several programming paradigms have been introduced. Despite the significant number of programming languages, only a few fundamental programming concepts and languages have survived for more than ten years. Some languages have grown into extensive software ecosystems, while others have failed to grow beyond their niche or disappeared altogether.

Selecting and Employing Multiple Programming Languages in One Development Project

No unique programming language is the *best option* for all potential scenarios. Judging the suitability of a programming language for a software product, as an application or a customized component, is a non-trivial task. For instance, a purely functional language like *Haskell* is the best fit for writing parallel programs that can, in principle, efficiently exploit huge parallel machines working on large data sets. However, while developing a dynamic website, a software engineer might consider *ASP.net* as the best alternative, and others might prefer using *PHP* or a similar scripting language. It is interesting to highlight that successful projects have been built with both: Stack Overflow is built-in *ASP.net*, whereas Wikipedia is built-in *PHP*. Furthermore, a software engineer might prefer particular criteria, such as scalability in enterprise applications, whereas other criteria, such as technology maturity level, might have lower priorities. Selecting and employing multiple programming languages in one development project is quite common. Furthermore, leading popular software available in the market is developed in multiple languages. Acquiring and expanding knowledge about programming languages is a highly complex process, as significant numbers of criteria and alternatives exist in the market. Various factors need to be taken into account, of which not all are obvious. Simultaneously, the choice of programming languages can have repercussions on the implementation cost, quality of the result, and maintenance cost of the application. Some of these consequences may not be felt for years after the initial programming language choice decision has been made.

Build a Decision Model to Capture Knowledge about Programming Languages and Concepts Systematically

Each programming language has different characteristics, communities, and ecosystems that should be considered. The selection process is mainly based on surrounding ecosystems and communities. Third-party libraries play an essential role as many software applications are built by gluing together plenty of existing libraries in the market, so such libraries increase language growth. Additionally, communities generate wikis, forums, and tutorials to improve the learnability and understandability of languages. Nowadays, the development of software products, systems, and services typically results in complex decision models and decision-making processes. Selecting the best fitting programming language ecosystem(s) for a software project can be modeled as a Multi-Criteria Decision-Making (MCDM) problem that deals with the evaluation of a set of alternatives and takes into account a set of decision criteria. Knowledge about programming languages is

scattered among a wide range of literature, documentation, and software engineers' experience. This study's main motive is to build a decision model to capture knowledge about programming languages and concepts systematically and make it available in a reusable and extendable format. Accordingly, we have followed our framework to build such a decision model for the programming language selection problem.