**Dinh Trung Hieu Le, M.Sc. Software System Engineering**
Hasso Plattner Institute, Potsdam, Germany
E-Mail: dinhtrunghieu.le@student.hpi.de

**Lecture Series on HPI Research**
Lecture in Focus:
Connected Healthcare, Prof. Bert Arnrich

Hasso Plattner Institut
Digital Engineering · Universität Potsdam

# Cognitive Load Assessment

# In Programming Languages and Paradigms

## Abstract

In contemporary software development, the selection of programming languages and paradigms is influenced by factors such as popularity, task suitability, unique features (Rust's borrow checker for safety or Erlang's BEAM for concurrency), performance, execution speed, and most recently energy efficiency [1]. However, despite an acknowledgment of the importance of cognitive load—evidenced by Python's emphasis on readability and Go's simplicity—there remains a significant gap in empirical research and metrics to evaluate how cognitive load impacts software development. Current methods primarily rely on subjective user feedback and evaluations of technique efficacy.

This research proposes an evaluation to assess cognitive load in programming languages using wearable devices , which will aid companies in making informed decisions when cognitive load is a concern. By leveraging cognitive load theory, which describes cognitive load as the total amount of information that the working memory can handle at one time, and measureing them, this study aims to illuminate the effects of different programming languages on cognitive efficiency and overall developer performance. This comprehensive approach seeks to establish clearer correlations and provide actionable insights that can influence future language and tool adoption.

## Problem

The discussion around how cognitive load in programming languages affects software development often hinges on key debates, yet it suffers from a lack of empirical research and specific metrics for measurement. These debates include:

1. Dynamic vs. Static Typing: Advocates for static typing argue that it reduces cognitive load by providing more explicit information to the developer, enhancing features like autocomplete and autosuggest. In contrast, proponents of dynamic typing suggest it simplifies coding and accelerates task completion by reducing upfront complexity.

2. Compiled vs. Interpreted Languages: Compiled languages handle errors during compilation, which might decrease cognitive load by catching errors early and reducing the need for runtime debugging. Besides , interpreted languages require developers to maintain awareness of issues typically managed by a compiler, thus increasing the cognitive burden as developers act as "human compilers."

3. Programming Paradigms: The impact of programming paradigms varies; imperative languages (e.g., C, Go, Rust) with composition, object-oriented languages (e.g., Java, C++, Python) focus on structuring programs around objects and inheritance, and finally, functional languages like Haskell or Erlang, which may reduce the need to track system states or side effects, potentially lowers cognitive overhead, allowing developers to change code confidently.

4. Languages Complexity: Languages such as C++ and Rust are known for their extensive features and powerful abstractions, which can potentially reduce cognitive load by providing sophisticated tools to address complex problems effectively. However, the complexity of their syntax and advanced features like borrow checking and lifetimes can conversely increase cognitive load. C++ and Haskell are notable for offering multiple ways to accomplish tasks, including various methods to initialize variables and create objects, which can overwhelm developers with choices and increase cognitive effort. In contrast, simpler languages like Python or Go, while easier to write and understand, can become verbose in expressing complex abstractions, leading to a different kind of cognitive load as developers manage larger codebases.

Despite numerous discussions on the topic, most current opinions about the cognitive load associated with programming languages are based on theories, personal feelings, or self-assessments rather than empirical research. Additionally, various language features and paradigms may alleviate certain types of cognitive load, such as intrinsic and extraneous loads, but could exacerbate others, such as germane load. This variability makes it challenging to definitively assess the overall impact of programming languages on cognitive load, underscoring the need for more structured and scientific studies to gain a clearer understanding.

## Goal

The primary objective of this research is to develop and refine metrics that accurately measure the cognitive load imposed by various programming languages and paradigms, using wearable devices for data collection. The aim is to deepen our understanding of how programming languages influence cognitive load and to assess how this impacts productivity, learning, reading efficiency, and overall usability. Additionally, this research will investigate the broader implications of cognitive load on usability costs, comparing it with other critical factors like performance and energy efficiency. By leveraging wearable technology to monitor physiological and behavioral responses, this study hopes to offer actionable insights that could lead to more efficient and user-friendly programming environments.

## Solution

The research will focus on developing metrics to quantify cognitive load in programming, primarily using wearable devices for data collection:

- Review of Existing Work: We will examine current theoretical and empirical research on cognitive load in programming to establish a solid base for our study.
- Metric Formulation: We'll develop indicators for assessing cognitive load, drawing on productivity measures and physiological data captured by wearable sensors, as outlined in Fabian's paper [2].
- Data Collection: Real-time physiological and behavioral data will be gathered from participants as they use various programming languages, employing wearable technology for accurate monitoring.
- Analysis of Data: We will analyze the data to find correlations between physiological responses, behavioral actions, and the perceived complexity of tasks to validate our metrics.
- Drawing Conclusions and Making Recommendations: The findings will be synthesized to offer practical recommendations for selecting programming languages based on their cognitive load impact, aiming to enhance the efficiency of software development processes.

This thorough evaluation is designed to develop validated metrics for measuring cognitive load in programming languages. It will offer insights into how various language features and paradigms influence cognitive load and developer productivity. By taking cognitive load into account when choosing programming languages, companies can improve their development processes and enhance the quality of their software. The analysis will consider results at various levels of detail, utilizing previous studies on cognitive load and physiology to ensure a comprehensive understanding of the effects.

Assessing Cognitive Load in Software Development with Wearable Sensors
Fabian Stolp

**References:**

[1] Pereira, R., Couto, M., Ribeiro, F., Rua, R., Cunha, J., Fernandes, J.P., & Saraiva, J.D. (2017). Energy efficiency across programming languages: how do energy, time, and memory relate? Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering.
[2] F. Stolp, "Assessing Cognitive Load in Software Development with Wearable Sensors," 2023 IEEE/ACM 45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), Melbourne, Australia, 2023, pp. 227-229, doi: 10.1109/ICSE-Companion58688.2023.00062.