International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V72I1P117

Volume 72 Issue 1, 104-108, January 2024


*Original Article*

# Enhancing Developer Experience by Reducing Cognitive Load: A Focus on Minimization Strategies

Srividhya Chandrasekaran

*Senior Product Manager, Bedford, MA, United States.*

*Corresponding Author : schandrasekaran@spotify.com*

Received: 05 December 2023 Revised: 09 January 2024 Accepted: 25 January 2024 Published: 31 January 2024

***Abstract*** *- This paper endeavors to scrutinize methodologies and approaches geared towards ameliorating the developer experience with a specific emphasis on strategies aimed at curtailing cognitive load. By delving into minimization strategies, the study seeks to unravel intricacies associated with mental effort reduction in software development tasks. The ultimate objective is to provide nuanced insights that can positively influence the cognitive dimensions impacting developers, ultimately enhancing productivity, decision-making, and the overall cognitive ergonomics of software development practices. The outcomes of this paper contribute substantively to the ongoing discourse surrounding optimization paradigms for developers, fostering a technologically advanced and cognitively attuned software development milieu.*

***Keywords*** *- Cognitive load, Platform engineering, Developer experience, Golden paths, Documentation.*

## 1. Introduction

Cognitive load [1] is the amount of information that our working memory capacity can hold at one time. According to Sweller J [2], who proposed the Cognitive Load Theory in the Late 1980s, the best learning occurs when the learning environment is aligned with human cognitive capacity. Sweller's theory emphasizes that our memory can only handle so much, so we need teaching methods that avoid overwhelming us with activities that do not help us learn directly. The cognitive load involved in a task is said to be the effort or amount of information processing required by a person to perform this task.

It is within this context that this study endeavors to address a discernible research gap. While existing literature elucidates the theoretical foundations of cognitive load and its implications for learning, an exploration of the practical application of strategies to mitigate cognitive load in specific corporate contexts is slightly limited. The goal of this paper is to bridge this gap by analyzing and proposing effective instructional methodologies that alleviate cognitive load and, in turn, foster enhanced learning experiences for software developers. By examining the interplay between cognitive load, memory processes, and instructional design, this study endeavors to contribute valuable insights to the optimization of learning environments to improve developer experience.

## 2. A Deep dive into the intricacies of memory processing

Three main processes characterize how memory works. They are encoding, storage, retrieval, or recall. Encoding refers to the process by which information is learned. The term "Storage" refers to the duration for which information is retained in memory. Retrieval is the process by which individuals access the store's information. While long-term memory in the human brain is unlimited, working memory is limited and subject to cognitive load. Humans intuitively know this, which is why they use a smartphone or a notebook to write down things that they do not want to forget.

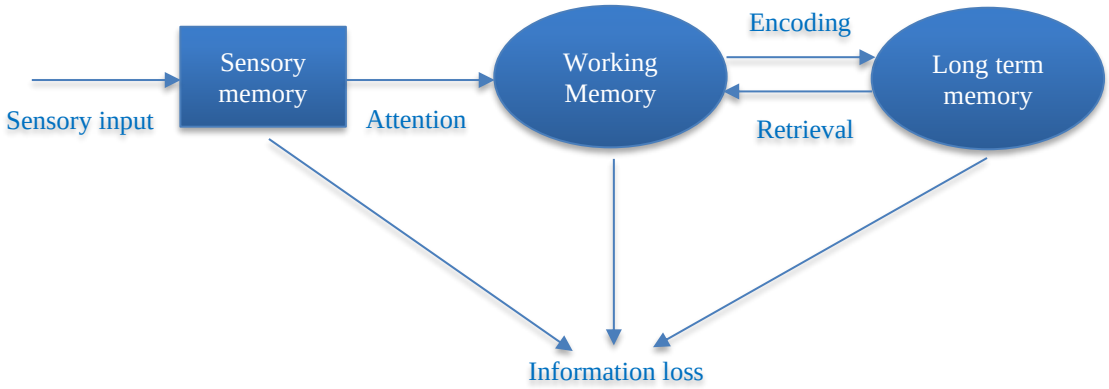


**Fig. 1 How memory works and how information is processed**

Similarly, there are three types of cognitive load [3].

1. Intrinsic cognitive load - This is the effort associated with a particular topic—for example, the effort associated with learning how Java classes are defined.
2. Extraneous cognitive load -This is the way information and tasks are presented. For example, when a developer is learning how to deploy a particular service.
3. Germane cognitive load - The work put in to understand the topic or domain area. For example: How does music get streamed on Spotify's app?

## 3. Cognitive Load in Software Development

Let's face it. Human working memory is limited. Researchers in this field have found that learning anything with multiple components that interact with each other is more complex than learning a system with minimal interaction elements. The former requires more cognitive capacity to process. Every time working memory capacity is exceeded, learning slows down, and cognitive load increases [4].

In the realm of software development, an elevation in cognitive load is directly linked to a decline in developer experience. Ever-increasing tools and technology choices in the last few years and changes in working practices such as DevOps have increased the cognitive load on software developers.

The below-shown Fig 2 is an image of the tools in the CNCF (Cloud Native Computing Foundation) landscape.

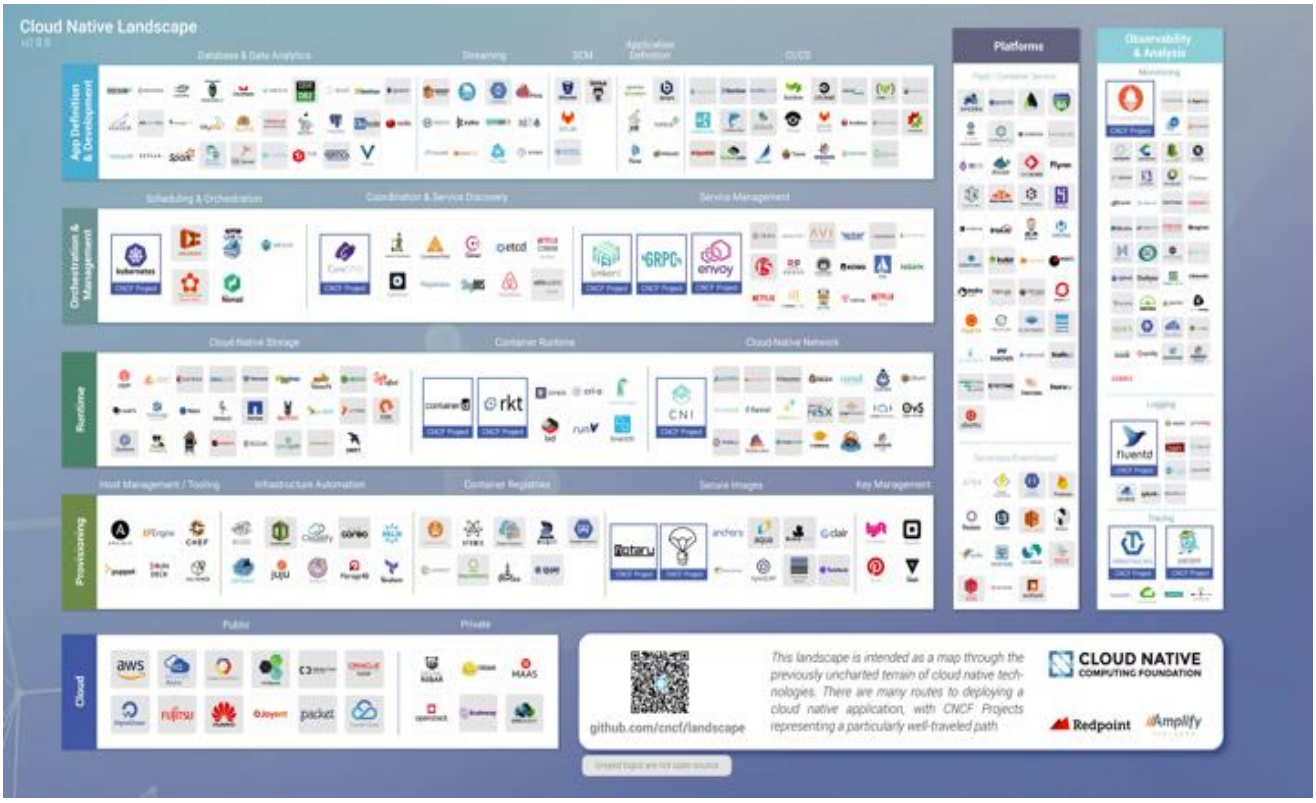


**Fig. 2 CNCF Cloud Native Landscape [5]**

There is a direct correlation between the rise in cognitive load for software developers and the increasing number of tools and complexity in technological landscapes. As the number of tools grows, developers are tasked with navigating and integrating diverse technologies, leading to a higher cognitive burden in managing and understanding the intricate relationships between these elements.

According to the book Team Topologies, the authors propose transforming cognitive load into intrinsic cognitive load by training and repetition. The idea is to eliminate unnecessary cognitive load, enabling individuals to maximize their cognitive capacity for essential, business-relevant aspects of their tasks. This approach helps people concentrate on the most critical aspects of their work.

This is where Developer experience or DevEx comes in. Developer Experience is about creating an environment where developers can do their best work [6]. This helps them improve speed and productivity and deliver value to their customers. In their day-to-day work, developers face points of friction multiple times. DevEx reduces this friction by improving productivity and helps with better outcomes by increasing product quality, thereby reducing attrition. A 2020 McKinsey study found that companies with better work environments for their developers achieved revenue growth four to five times greater than that of their competitors.[7]

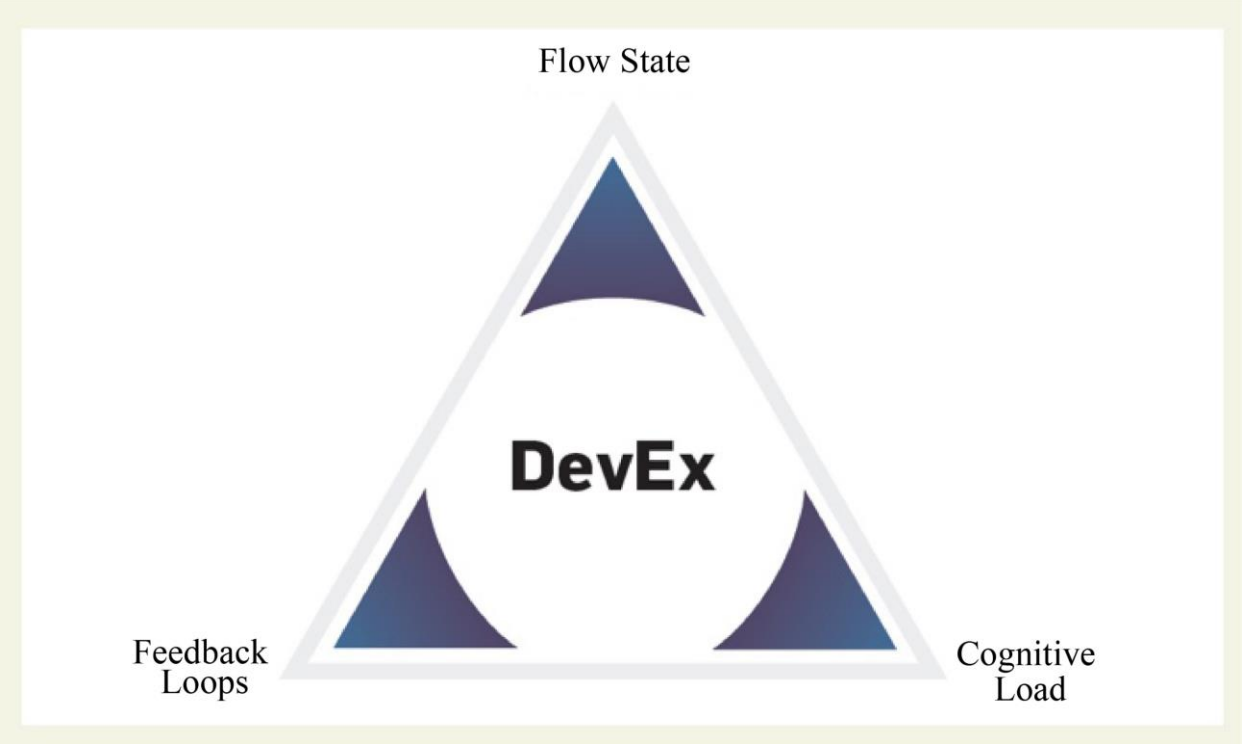


**Fig. 3 Three core dimensions of developer experience [6]**

*Source: https://dl.acm.org/doi/fullHtml/10.1145/3610285*

## 4. Delving extensively into a facet of the developer experience: Cognitive load

In other words, cognitive load is the effort that a software developer needs to put in to complete a task.

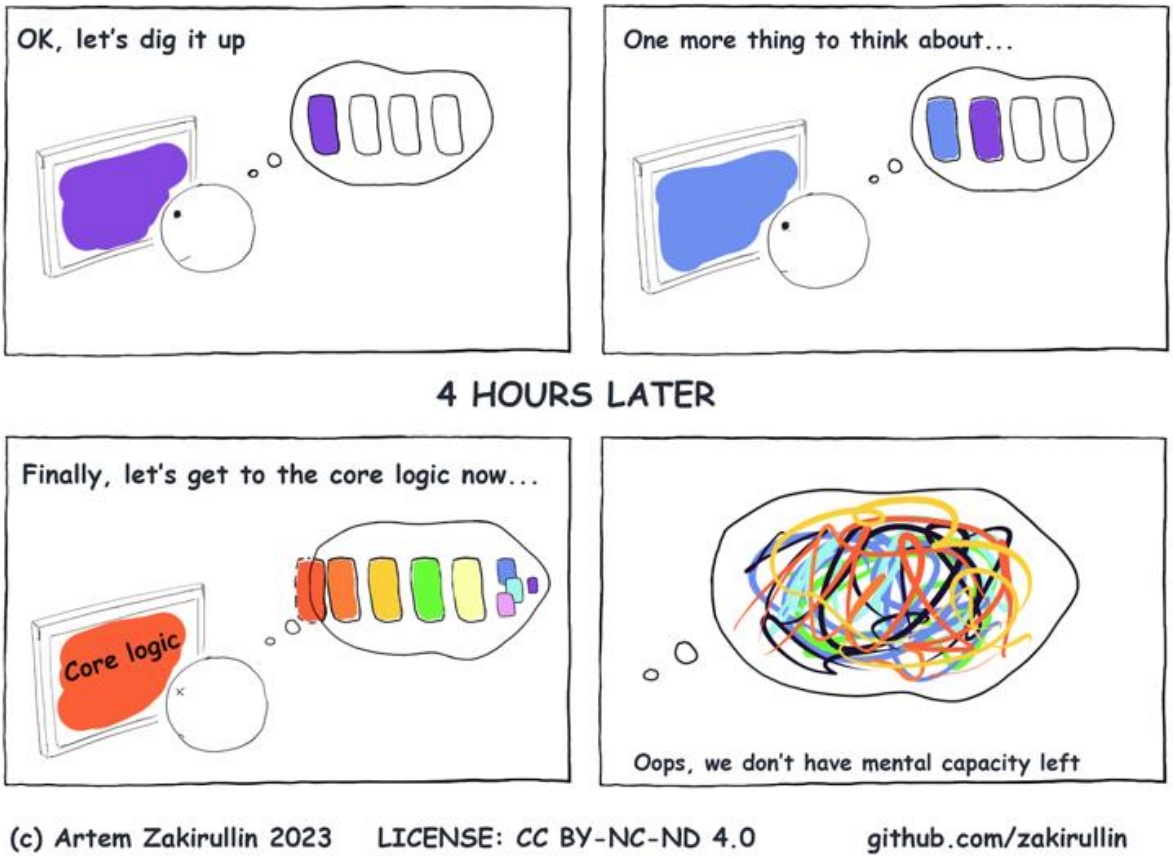


**Fig. 4 The 3 core Dimensions of developer experience [8]**

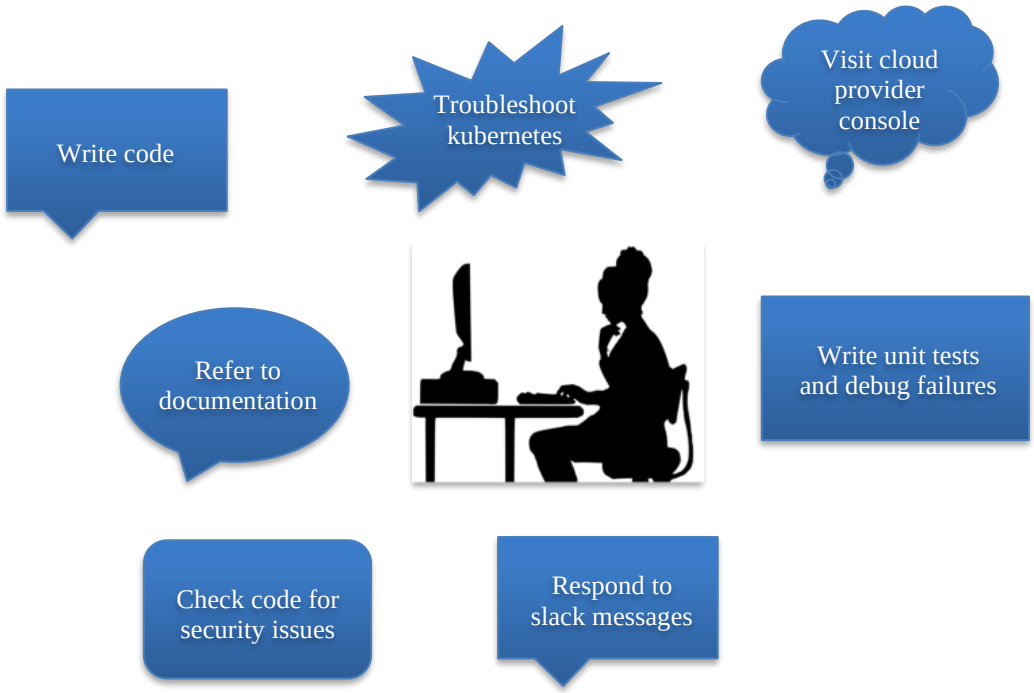


**Fig. 5 A Standard Day in the Life of a Software Developer**

Upon commencing a daily task, a developer is confronted with the necessity to consult diverse documentation, strategically deploy breakpoints within the codebase, and navigate intricate code structures to grasp the existing context before initiating modifications. For those newly acquainted with the domain, product, or organizational intricacies, the ensuing cognitive load tends to surpass that of their counterparts well-versed in the codebase. This augmented cognitive demand poses a discernible risk, potentially culminating in escalated error frequencies, protracted development durations, and an attenuated overall yield of work output.

Cognitive load is linked to the psychological well-being of software developers. Prolonged exposure to high cognitive load may contribute to stress, burnout, and job dissatisfaction, highlighting the need for a balanced and supportive work environment. Studies also recognize that cognitive load is experienced differently by individual developers. Factors such as expertise, experience, and cognitive abilities play a role in shaping how developers perceive and manage cognitive demands.

## 5. Recommendations: Ways to improve developer experience

### *5.1. Platform Engineering*

Platform Engineering Streamlines development projects by employing a uniform set of tools and frameworks. This enables developers to work with Internal Developer portals to carry out repetitive tasks instead of manually overseeing them to completion. This, in turn, allows developers to focus on innovation and minimize delays. Platform Engineering's automation capability also promotes cross-team collaboration and reusable components to speed up product delivery.

Applying a product mindset to building platform tools also helps improve developer experience. What this means is that business needs and outcomes are prioritized over timelines and estimates. This helps engineers look at a product from the customer's perspective and helps build the platform as a product right from day one.

### *5.2. Improved Documentation*

README's, getting started, and 'How to' guides and one place to access these docs for developers reduce cognitive load and improve the individual developer's experience.

### *5.3. One place for All Tools for Developers to Access*

An Internal Developer portal is a self-service tool/application that provides one place to access all APIs, Documentation, Software Templates, and other platform capabilities, including reducing context switching for engineers, improving software quality, and improving their flow and focus.

### *5.4. Standardize using Golden Paths*

About 9-10 years ago, [during hack week], eight top engineers at Spotify gathered their forces and created a tutorial on the recommended way of using our services; it was named "The Golden Path" [9]. The Golden Path is the way Spotify supports an easy and streamlined way of working. It is an 'opinionated' and 'supported' path to build something.

## 6. Conclusion

While it might currently seem impossible to eliminate cognitive load for software developers, it is definitely possible to use the above ways to improve an individual developer's experience, thereby reducing stress, burnout, and turnover. The use of technology to facilitate learning has

become a necessity but the ultimate goal must not be to overwhelm the learner but to help them understand and apply their learnings in their everyday work. This must be done with an understanding of the limitations of working memory and by using strategies that fit the working styles of software developers with the intent of making their lives easier and helping them in ways that they learn best.

## References


[1] Paul Main, Cognitive Load Theory: A Teacher's Guide, Structural Learning, 2022. [Online]. Available: https://www.structural-learning.com/post/cognitive-load-theory-a-teachers-guide

[2] John Sweller, "Cognitive Load Theory and E-Learning," *International Conference on Artificial Intelligence in Education, AIED 2011*, pp. 5-6, 2011. [CrossRef] [Publisher Link]

[3] Paula Kennedy, Whose Cognitive Load is it Anyway?. [Online]. Available: https://platformengineering.org/blog/cognitive-load

[4] De Jong, "Cognitive Load Theory, Educational Research, and Instructional Design: Some Food for Thought," *Instructional Science*, vol. 38, no. 2, pp. 105–134, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[5] Cloud Native Landscape. [Online]. Available: https://landscape.cncf.io/?license=open-source

[6] Abi Noda et al., "DevEx: What Actually Drives Productivity: The Developer-Centric Approach to Measuring and Improving Productivity," *Queue*, vol. 21, no. 2, pp. 35-53, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[7] Shivam Srivastava et al., "Developer Velocity: How Software Excellence Fuels Business Performance," *McKinsey & Company*, pp. 1-11, 2020. [Google Scholar] [Publisher Link]

[8] zakirullin, Cognitive Load in Software Development, 2024. [Online]. Available: https://github.com/zakirullin/cognitive-load

[9] Gary Niemen, How We Use Golden Paths to Solve Fragmentation in Our Software Ecosystem, Spotify Engineering, 2020. [Online]. Available: https://engineering.atspotify.com/2020/08/how-we-use-golden-paths-to-solve-fragmentation-in-our-software-ecosystem/