Introduction

In the last 10 years, the rise in popularity of software development triggered an explosion in computer science related massive open online courses (MOOCs) and coding boot-camps.

Unsurprisingly, the availability of published research into how to learn and teach computer science effectively seems to have also increased in quantity.

Many of these papers discuss variations and interpretations of cognitive learning theory in the context of learning computer science (as distinct from the science of proxying human intelligence with computers).

It is possible that cognitive learning theory, which is only one of the foundational learning theories Merriam & Bierema wrote about in chapter 2 of "Adult Learning: Linking Theory and Practice", "Traditional Learning Theories" (Merriam & Bierema, 2014), is popular in computer science learning focused research because computer science itself is an input/output science. It fits naturally with the mental processes modelled metaphorically in the theory of cognitivism as 'input, throughput, and output' (Merriam and Bierema, 2014)(pg. 31).

In this report, we too elaborate on a cognitive learning theory, but focus on the **'spacing'** format to suggest that longer hours in a day studying one computer science subject or problem (interspersed with breaks), for fewer days in the week, is more effective than fewer hours in a day studying one computer science subject across more days in the week.

While we do not have the research facilities to test this theory on a population of students, we are confident we can posit the theory intelligently, given available research and our experience in the field.

Overview

Before we introduce reasoning to support our thesis, let's look at some characteristics of cognitive learning theory.

In general, cognitive theory speaks to the influence of internal and external factors on learning, with an emphasis on three phases of learning: **comprehension**, **memory**, and **application** (Ivan Andreev, 2022).

The more narrowly focused cognitive development theory, introduced by Jean Piaget, emphasizes cognitive processes that support **observing**, **classifying**, **categorizing**, **attention**, **perception**, **interactivity**, and **reasoning** (Your eLearning World, 2020).

Finally, Weinstein, Madan, and Sumeracki, in "Teaching the science of learning" (Weinstein, Madan, & Sumeracki, 2018), introduce six characteristic process of cognitive learning that are particularly relevant to curriculum design, and the theory we want to introduce in this paper, **"cognitive load theory"**.

Those six principles are:

1. Spaced Practice

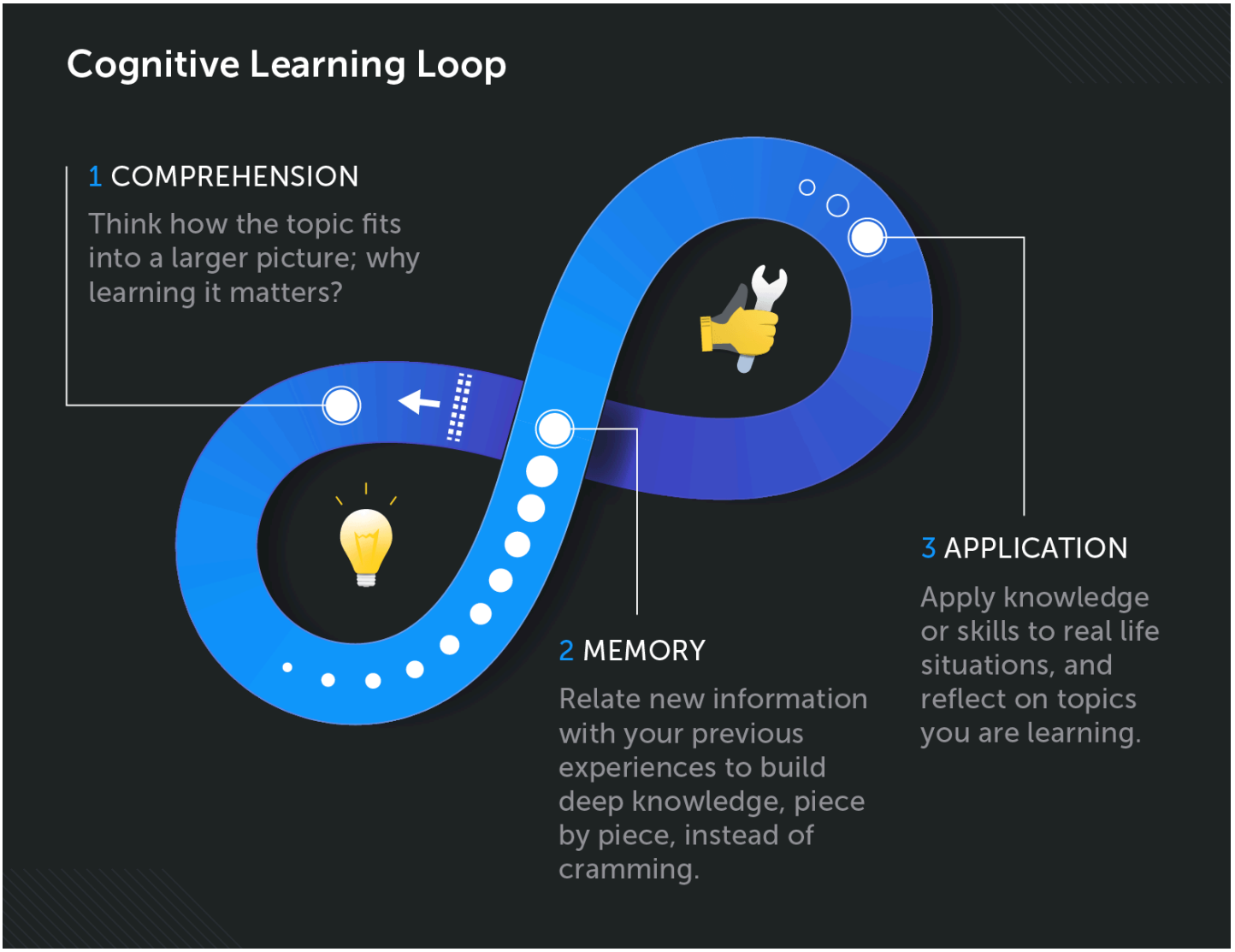
Revisiting material according to a pattern spaced out over time periods to maximize learning.

2. Interleaving

Introducing different ideas or problems in the same study session, rather than repeatedly attempting the same problem in different formats.

3. Retrieval

Recollecting memory by bringing long-term memories out of 'storage' into short-term memory



Source: (Ivan Andreev, 2022)[https://www.valamis.com/hub/cognitive-learning]

4. Elaboration

Adding features to an existing memory, (Weinstein, Madan, & Sumeracki, 2018)

5. Concrete Examples

Anchoring abstract ideas to more easily conceptualized problems or scenarios

6. Dual Coding

Introducing the same material in multiple media formats, such using visual aids, like images, and physical models, along with audio aids, as well as text formats, like websites, slides, and reading assignments.

These principles are elaborated on in “Teaching the science of learning” (Weinstein, Madan, & Sumeracki, 2018), and we recommend reading up on them.

> *As we can see from above, cognitive learning theory examines characteristics of the human mind and the learning environment to arrive at techniques and practices that enhance learning.*

There is another theory, called **"cognitive load theory"**, we want to explore in the context of our proposition (thesis), *that longer hours and shorter work weeks improve computer science learning outcomes*.

Cognitive Load Theory

Cognitive load theory is an instructional design framework that focuses on how the human brain processes, stores, and retrieves information.

It was introduced by John Sweller in the late 1980s and has since gained significant recognition for its implications on learning outcomes and instructional design.

Cognitive load theory emphasizes that the human brain has limited memory resources, particularly within working memory, which can be easily overwhelmed when processing large amounts of new or complex information, leading to reduced comprehension and retention.

The theory identifies three types of cognitive load: intrinsic, which is inherent to the task; extraneous, which is caused by unnecessary information or activities; and germane, which supports the construction of schemas that facilitate learning.

Understanding these types of cognitive load helps educators optimize teaching practices to enhance learning efficiency.

The paper on cognitive load theory we suggest reading was written by Shaffer et al., in 2003, and is titled, "Applying Cognitive Load Theory to Computer Science Education" (Shaffer et al., 2003).

It's necessary when looking at the Shaffer et al. to distinguish between 'cognitive load' and 'cognitive loading'.

'Cognitive load' describes the 'load' a problem space imposes on cognitive working memory, while 'cognitive loading' describes the process of retrieval, wherein information is consumed and stored in short-term memory and long-term memories are brought out of storage and simultaneously stored in working memory.

High Cognitive Load

Shaffer et al. are concerned with high cognitive load. They optimize for learning in high-cognitive load environments by partitioning learning into two tenants, schema acquisition and automated procedural knowledge.

They map these tenants to basic tenants in cognitive learning so that schema acquisition is associated with retrieval, and automated procedural knowledge with application (labeled in the cognitive learning loop diagram above as 'memory' and 'application', respectively).

Then they partition skills into 'non-recurrent' and 'recurrent' skills, where non-recurrent skills vary from problem to problem — like skills required in structured decomposition, when business logic is broken down into complex programmatic workflows — and recurrent skills have minimal variation between problems.

In web development, for example, there are only so many methods that can be used to arrange child divs in a parent element, the primary examples being 'display:grid' and 'display:block'. For a mid-level engineer, structuring a hierarchy of divs would be considered a recurrent skill.

Next, they examine strategies at the problem attack level to minimize a variety of overlaps between recall and procedural skills development processes, and, in general, to minimize intelligently cognitive load.

While we don't elaborate on their solutions, we do present them in the section below 'Optimizing for Cognitive Load Theory in the Classroom'.

Again, we also recommend reading: "Applying Cognitive Load Theory to Computer Science Education" (Shaffer et al., 2003), especially if you're interested in the union between learning and computer science.

What we care about is one memory learning characteristic they highlight, which is that 'recall' is an expensive operation in terms of intense energy use.

High Cognitive Loading

Shaffer et al. suggest "the extent to which curriculum materials impose a load on working memory" may be related to "the number of elements that must be processed simultaneously in working memory." (Shaffer et al., 2003).

They state that working memory "has two limitations." (Shaffer et al., 2003).

"Firstly, it is short-lived. Secondly, it is only capable of processing up to approximately seven things at a time (Miller, 1956). If this capacity is exceeded, then some or possibly all of the information is lost." (Shaffer et al., 2003).

In our experience, the constraint they describe is binding on the atypical programming problem space, which demands from the average human mind an excessive amount of cognitive power to store and manipulate often many multiples of 'seven' variables.

An example of a such a problem space could be a project with a large code base with many complicated scripts and operational pipelines.

Just loading the problem into working memory when the day begins, not to mention setting up the problem space in a software development environment (loading and configuring SDKs, setting up servers and editors), is energy and focus intensive.

By the time the problem solving workspace 'set-up' is complete (the dev shop is primed), and the student is situated to enter into an effective programming flow, the bell rings and the class is over, or the student becomes distracted by other tasks, commitments, and people.

A significant amount of time and energy was expended loading the problem space into cognitive working memory, and the gains are lost. When the student returns, some hours later, or in the evening, the 'dev shop' must be primed again.

It's this problem of inefficient time and energy management we want to solve.

And the way we solve it, is relatively simple and straightforward.

In our view, one method for maximizing learning outcomes in high cognitive load environments, especially for novice developers, is to elongate the duration of the skills acquisition phase, after completing (and minimizing) the energy intensive declarative 'load phase'.

This is the key insight that leads to the formation of our thesis statement, so let's repeat it.

> *An additional method to maximize learning outcomes in high cognitive load environments is to elongate the duration of the skills acquisition phase, after completing (and minimizing) the energy intensive declarative 'load-phase'.*

Minimizing Cognitive Loading in the Context of Adult Learning: Linking Theory And Practice

The Role of the Learner in Cognitive Load Theory

As we've seen, cognitive load theory suggests there are physiological limitations that are unique to the application of computer science in programming.

While the suggested solutions to those limitations are behaviorist more than humanist, in that they're implemented in course design and delivery more than through the self-learning process, the role of learners is certainly impacted.

If the motivated learner is aware of the cognitive load associated with effective learning in programming, and the solutions suggested in the literature, he or she is more likely to find ways to implement those solutions to achieve his or her objective.

Nonetheless, it is still the learner who takes programming principles – and today suggestions from artificial intelligence – and uses them to construct knowledge and etch retrievable skills into cognitive memory.

The Role of the Instructor in Cognitive Load Theory

The instructor in cognitive load theory can still behave as the quintessential 'humanist' instructor, acting as a guide to the learning process, rather than an expert demanding responses from the student that 'measure up' to a preset definition of 'grade A'.

However, the act of guiding should include imparting solutions conceptualized in this and other papers about cognitive load theory.

Moreover, to the extent the instructor designs the curricula, there is a lot he or she can do to overcome human cognitive limitations in the context of high-cognitive load material.

Optimizing for Cognitive Load Theory in the Classroom

Shaffer et al. provide a comprehensive list of techniques that address the problem of high-cognitive load. As you can see below, their solutions apply mostly to curriculum design, rather than teaching practice.

We list them below, but don't go into detail describing them.

Once again, reference "Applying Cognitive Load Theory to Computer Science Education" (Shaffer et al., 2003), for a deeper dive.

Spacing for High Cognitive Load

One recommendation that comes out of reasoning about our thesis, is that instructors and administrators of computer programming school programs could ensure collaboration between instructors across courses to keep students focused on the same problem (or at least subject) during a school day.

> *Example 1: offering a design lab in the morning for 2 hours, then a break, followed by a front-end framework focused hands-on class in the early afternoon, followed by a database-focused back-end class in the early evening.*

This structure would keep the student focused on one project during the day, minimizing energy and effort expended during cognitive loading and maximizing skills-acquisition.

Example 2: Another example could include an online learning environment, where the curriculum prompts the student to focus on different aspects of the same problem space, while encouraging the student to switch subjects across days, with only mild cross-day interleaving to optimize retention gains from revisiting material.

While online learning programs are more and more promoting self-learning, which leaves the student free to decide what to focus on, suggestions could be made to students in introduction videos and pre-course reading material.

Example 3: In cases where curricula are bound by fixed subject selection and schedules, such as in high-school, the same effect could "be achieved with no great costs if teachers set aside a few minutes per class to review information from previous lessons." (Weinstein, Madan, Sumeracki, 2018).

In such a case, it is assumed interleaving is mandatory, in which case the best an instructor can do is remind students of key points in previous lessons, and teach them to minimize cognitive load effort, through intelligent tool design and use (SDK).

In each of the cases, it's important to remember the guiding principles set out in cognitive learning theory.

Solutions From Shaffer Et Al.

> - partitioning of information into small segments, them simultaneous presentation of concepts and procedures in demonstrations
> - presentation of heuristic approaches and strategies, independent of domain reference
> - organization of presentation of non-recurrent skills into knowledge structures
> - presentation – repeated example and practice – broad practice
> - broad practice using partially-completed examples
> - gradual withdrawal of supporting information from practice tasks
> - decomposition of practice tasks into small steps then gradual integration into a complete skill
> - goal-free problem solving
> - careful integration of text and graphics
> - visual and aural sources for learning

[Shaffer et al., 2003]

Conclusion

In this paper we outlined the theory of cognitive learning, introduced principles of cognitive learning theory, and introduced principles of cognitive load theory.

Cognitive load theory, suggests learning includes components of knowledge retrieval as well as skills acquisition, or procedural learning.

We showed research that knowledge retrieval is a remarkably energy intensive activity, which supports what we've experience in our programming journey.

We then introduced our own thesis, *that longer hours and shorter work weeks improve computer science learning outcomes*, which elaborates on solutions provided Shaffer et al., in their 2003 paper, to solve the problem inherent in cognitive load theory, which is that cognitive loading is energy intensive and time consuming.

Finally, we provided some context of how our thesis impacts learners and instructors, and gave some examples of how our thesis could be applied in the classroom.

References

Weinstein, Y., Madan, C.R. & Sumeracki, M.A. Teaching the science of learning. Cogn. Research 3, 2 (2018). Retrieve from https://doi.org/10.1186/s41235-017-0087-y

Shaffer, D., Doube, W., & Tuovinen, J, (2003). Applying Cognitive Load Theory to Computer Science Education. Retrieved from: (https://www.researchgate.net/publication/250790986_Applying_Cognitive_Load_Theory_to_Computer_Science_Education)

Andreev, I. 2022. Cognitive learning. Valamis Group. Retrieved From: (https://www.valamis.com/hub/cognitive-learning)

Your eLearning World, 2020. The Learning Theory Of Cognitive Development In ELearning. Retrieved From: (https://yourelearningworld.com/the-learning-theory-of-cognitive-development-in-elearning/)