Notetaking Applications in Computer Science and their Connection with Different Notetaking Strategies and Theories of Learning in Computer Science Education

A Research Paper submitted to the Department of Engineering and Society

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Andrew Song

Spring 2023

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Advisor

Kent Wayland, Department of Engineering and Society

As the field of computer science continues to grow, the subfield of computer science education has grown along with it. Because computer science is a relatively young field of study, there are still many questions regarding the best practices when it comes to teaching and learning computer science. As with any other field of study, fostering proper education is essential is creating professionals that can contribute and provide value. Without a solid educational foundation, it is difficult for a worker to be a effective in their field. As such, researchers in computer science education seek ways to more effectively equip aspiring computer scientists with computational thinking skills, problem solving skills, and overall knowledge of computer science.

One of the many improvement areas deals with notetaking and finding ways for students to process and organize information more effectively. For example, researchers have theorized about different types of learning computer science. Additionally, because traditional notetaking methods are often not as effective for learning computer science subjects, researchers have looked to different notetaking strategies that are geared specifically for learning computer science subjects. To be able to observe how these strategies work in practice, one approach is to examine popular notetaking applications and analyze their connection with the strategies to see how the notetaking methods reinforce, or fail to reinforce, various theories of learning in computer science. Understanding these connections is important to know not only which notetaking methods are truly effective, but also to know how to develop and improve notetaking strategies when learning computer science subjects. Thus, this paper aims to find out some current notetaking strategies for computer science along with prevalent theories of learning in computer science education and analyze the ways in which popular notetaking applications for

computer science make use of certain notetaking methods to reinforce certain theories of learning while ignoring others.

Background

As digital technology has become a big part of classrooms today, there have been many new technologies in computer science education. For example, one notable technology is program animation. Animations are dynamic representations that evolve and show the behavior of algorithms and abstract concepts, show the execution of code and data in a program as a sequence of snapshots, and allow annotation of programs to generate visualizations (Jimenez et al., 2000). Technologies like these allow for new methods of notetaking in computer science, which can make use of various aspects of new technologies to create a notetaking application that fosters learning at a higher level. For example, Notion is a notetaking application making use of modern features. A developer of Notion, Sarah Lim, noted how they made it possible to flexibly represent and organize structured data in a way that does not scale well with other text editors and note-taking apps (Lim, 2020).

To explore how this learning is done, one approach is to look at various theories of learning to see which theories these notetaking strategies fit into. Much research and study in computer science education has led to a number of theories regarding how students learn computer science subjects effectively. One theory of learning that I explore in this paper is constructivism, the idea that students construct their own knowledge rather than purely receiving it. It suggests that knowledge is acquired recursively, meaning that data is combined with existing knowledge to create new cognitive structures, which are in turn the basis for more future construction. With constructivism, students form their own connections in tandem with the passive learning that is generally done in the classroom.

Cognitivism is another theory of learning in computer science. Cognitivism deals with the ability of a person to logically think through information in the learning process. It focuses on the learning of a concept through exploration of the underlying logic of that concept (Taylor et al., 2013). Related to the ideas of cognitivism, Berssanette, in a study on the Cognitive Load Theory, details the cognitive load theory and applies it to learning computer programming. The cognitive load theory states that learning is impaired when the total amount of processing requirement exceeds the working memory's capacity. The paper describes several evidences related to the cognitive load theory in learning programming; one of the most significant evidences was the worked example effect. This describes how students learned better when they were presented with a worked out example of the problem they were solving before attempting actual problems. These examples could be presented in various ways, but the important aspect was working through examples before attempting problems on their own. By working through these foundational examples that explicitly demonstrate the workings of a concept, students can lighten the amount of processing they have to do before building up to more abstract ideas.

These theories clash in some ways while mixing in other ways and are topics of great interest in computer science education. Interestingly, different groups of people tend to find different theories more relevant and applicable to education. For example, software engineers may be greater supporters of the idea that learning is more effective in practice, perhaps leaning towards constructivism, while researchers may be greater proponents of studying theory and abstract ideas.

Literature

A lot of research in computer science learning begins with the importance of visualization. There have been many efforts to explore the impacts of visualization in computer

science education. Visualization technology can be used to graphically illustrate various concepts in computer science. It is a method to try and promote active learning in CS, based on the idea that technology itself is of little value unless it engages the learner (Naps et al., 2002). Further, there has been research done in the thinking patterns when it comes to learning computer science. For example, computational thinking is a growing model of thinking in education for which educators aim to enhance and cultivate among students. Exploring these thinking patterns is a way for notetaking applications to implement specific features to promote learning.

When it comes to learning, there have been many different theories on how students in computer science may learn best. Constructivism is one of the most popular theories in computer science education. At the highest level, the main idea behind this theory is the idea that "students construct knowledge rather than merely receiving and storing knowledge transmitted by the teacher" (Ben-Ari, 2001). Another popular theory is cognitivism. This theory focuses on the idea that cognitive processes like induction, deduction, pattern recognition, etc. are the keys for learning and understanding (Taylor et al., 2013). Other theories focus on the social aspects of education. For example, social construction is a theory that focuses on "how the individual interacts with a community in developing their understanding" (Machanick, 2007). Similarly, the situated learning theory argues that a "community of practice" is needed for deep learning of material (Ben-Ari, 2004).

Methods

I intended for the majority of my sources to be secondary literature. This included journals, surveys, and research papers from various databases. These were selected based on the computer science education technologies, notetaking strategies, and theories of learning that they introduce and detail. Further, they detailed the state of the sociotechnical system which my

research looks into, which is the balance of educators, students, and professionals in the field. Surveys of students were also used to gather student opinions of notetaking methods and learning theory. An example of this was looking into online forums where students discussed what kinds of notetaking applications that they used. In addition, I looked at videos created by various content creators who talk about what kind of notetaking apps they use as developers/computer science students and analyze what features they highlight of a particular app.

Along with these secondary sources, I also examined a number of primary sources. One primary source that was used in my research were reviews for various notetaking applications. Reviews give an idea of what various groups of people are thinking, and what they want in notetaking applications. In addition, I looked at direct statements from developers themselves. I have gathered interviews of the developers of Obsidian and Notion, for example.

Results

Notetaking applications in computer science make use of many different strategies to aid the learning of computer science concepts. One of the most popular apps that we can analyze is Obsidian. In an interview with Erica Xu, a co-founder of Obsidian, Xu highlighted that a key feature of Obsidian was its extensibility. Obsidian supports many plugins that offer various functionalities for computer science students. Among others, Xu mentioned Mind Map, Advanced Tables, and Sliding Panes as some of the most popular plugins in Obsidian (Cunff, 2022). The graph view is an especially useful plugin for creating notes for computer science subjects. For example, in the figure below, we can see how a developer has taken notes on Svelte, a front end javascript framework, and used the graph view to link together notes on styling, state management, and routing in Svelte (Julien, 2021). These could also link to more general notes on any of these topics.

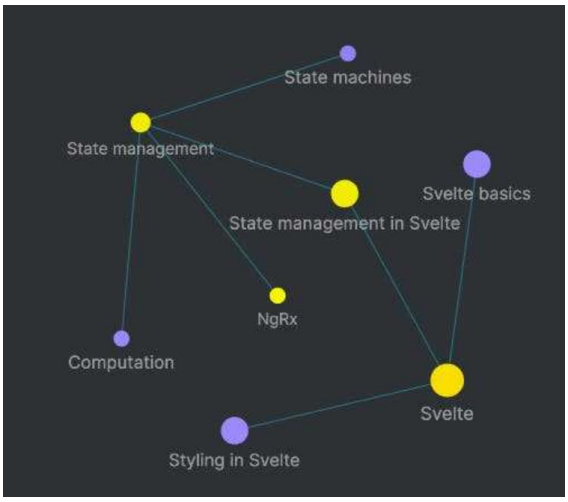

Figure 1: A Graph View of Svelte

Additionally, Obsidian supports Markdown and code blocks in many different languages which allows coders to build code snippets inside of their notes. This can be used in tandem with the Obsidian plugin, Editor Syntax Highlight, so that users can follow syntax more easily. Plugins like Obsidian Git allow users to automatically back up their vaults into git. Altogether, plugins like these provide the student with an environment that emulates a development environment in many ways.

Notion is another popular notetaking app among computer science students. Notion's founder, Sarah Lim, describes how her personal favorite feature of Notion was the flexibility of databases. She noted how they made it possible to flexibly represent and organize structured data in a way that does not scale well with other text editors and note-taking apps. She described how she personally liked to use Notion to track all the topics that she was learning. When she had a new concept, technology, or idea that she wanted to investigate further, she would create a new database entry and assign a level of familiarity. Later during further study, she could add an entry

to the corresponding topic page, and as she gained familiarity with a topic, she'd move cards to the next column—a cool and satisfying way to visualize the accretion of knowledge.

The designs of Obsidian and Notion were both heavily inspired by the developer community. The sliding pane plugin in Obsidian mimicked the work of Andy Matuschak, a software engineer who Xu said inspired the design of Obsidian and whose work is highly regarded in the Obsidian community. Further, the idea of plugins and extensions in Obsidian was something that is particularly popular among programmers and was inspired by the ability to add extensions to code editors like Visual Studio. Lim also mentioned that her inspiration for building Obsidian was seeing the number of note-taking posts regularly appearing on Hacker News, a news site for programmers (Lim, 2020).

A third notetaking app of interest is Evernote. Evernote was not an app that was heavily influenced by the developer community. It was created by Stephan Pachikov, a computer entrepreneur who conducted research in fuzzy logic, obtaining his PhD from the USSR Academy of Sciences. A standout feature of Evernote, according to researcher Dr. Robert George, is its diversity. It can support many types of notes on its platform, from handwritten notes to photos to audio notes. For example, it uses Optical Character Recognition (OCR) on all documents so that words from handwritten notes can be included in its search feature when the user wishes to search for keywords in their notes. Evernote also supports tagging. By adding keyword tags to notes, batches of notes can be more easily filtered without having to rely on keywords that may be inconsistent within the actual text of the notes. Dr. George also noted Web Clipping as an important feature for his research. When researching web pages, the Web Clipper add-in could preset both tags and the notebook before starting and assigning them when clipping the material.

He described how using presets for the tags made researching online much quicker as files could be tagged automatically (George, 2021).

Discussion

Analysis of the notetaking apps' features show that many of its features align with the various theories of learning. One feature that was common to all the apps that I explored was the incorporation of Markdown language into the application's functionality. By giving students a tool serving as a built-in method to write small examples of code into their notes that are formatted in a proper programming language rather than a typical writing format, the app supports learning through small, worked-out examples. For example, a student who is taking notes on the subject of Sorting Algorithms may find it useful to use Markdown to put a small snippet of a sample run of merge sort. With Markdown formatting, the text can both be easily understood along while also being easily run inside development environments. By supporting the creation of small code snippets, the apps allow students to work out smaller, foundational code examples inside of their notes, which is one the basic ideas of the cognitive load theory that students learn better when working through smaller examples before being exposed to bigger tasks that may come in the form of projects, coding assessments, etc.

Additionally, many of the apps' features aligned with the theory of constructivism. Obsidian demonstrates elements of constructivism through its graph view plugin. By placing the notes and concepts into a visual map that explicitly makes connections between various topics and concepts, it allows the user to better construct the connections in their head, laying the foundation to recursively add to this map structure to create more and more connections. This aligns with the constructivist ideas that learning occurs recursively, as a map makes it easier to visualize and add on to the existing knowledge. Further, Notion also has several features that

seem to align with constructivism. By focusing on the organization and structure of how notes are laid out in relation to each other, Notion also puts an emphasis on developing connections between various topics and concepts in the notes, aligning with the idea in constructivism that forming one's own connections between ideas and concepts is key for learning.

Cognitivism is another theory of learning that the apps seemed to align with. Evernote, through its design which allows users to better compile research into a single platform, is convenient for learning through cognitivism. The ability to swiftly incorporate various web links and pages into the user's notes means that the user can more quickly research and explore a given topic. Further, its support of various media of notes optimizes the user experience when it comes to compiling notes and ideas. As a result, through this functionality and interface, Evernote supports a way to flesh out and dive into topics and ideas better than other apps do. This is an important part of cognitivism, which stresses an importance on the learning of a concept through exploration of the underlying logic of that concept.

In exploring the reasonings behind why certain apps fit into some learning theories better than others, we can start by observing that, in the developer community, constructivism is a large part of how development practices are learned and taught. Object-oriented programming, for example, is a core concept of software development. It involves a degree of abstraction such that developers learn how to strategically manipulate object abstractions before understanding and learning the concrete object itself (Ben-Ari, 2001). By first constructing connections at the higher level of how objects can work in tandem with each other to solve given problems, developers can, initially, ignore much of the underlying logic that goes into each part of the solution by abstracting it.

These constructivism ideals of forming connections at the higher level are seen inside the Obsidian and Notion applications and seem to be a result of the apps being heavily influenced by the programming and developer community. In Obsidian, the idea of having a graph that serves as a sort of mind map is a method of forming connections between topics at a high level. It abstracts the concepts such that all that is needed to construct an understanding of how concepts relate to each other is essentially high-level knowledge. As such, this design, to the degree that it reinforces the ideals of constructivism, also seems to ignore aspects of cognitivism by allowing the user to avoid linear notetaking by delving into one concept, and then another, and so on.

Conclusion

Notetaking methods today have become quite sophisticated due to the possibilities that come with digital notetaking. As a result, they have become the go-to method for many students in the classroom to store their notes and thoughts. However, with the powers of these notetaking apps come with it complexities and features that require a deeper look to understand why they exist, and exactly what function they perform.

In this paper, we have examined some of the most popular notetaking apps among computer science students, in specific, Obsidian, Notion, and Evernote. By examining who the app creators were, and interviews with the creators, we could see that these features were influenced by who created the apps and who the creators took into account during development. Certain populations influenced the designs of the app to support their own methods and ideals of learning. Further, in analyzing the features of each of these notetaking applications, certain theories of learning could be seen in many of the features. Altogether, the applications show that notetaking is a technology that varies in its alignment with certain learning theories, in part due to the influences of the community around it. Certain notetaking apps better align with the ideals

of constructivism, while others better align with the ideals of cognitivism and the cognitive load theory.

Ultimately, this work helps gain a clearer idea of how notetaking applications differ in how they help students learn. Computer science is a subject that requires the connection of different ideas and concepts to go along with heavy computational thinking. With the goal of creating notetaking applications that can better cater towards the needs of computer science students in mind, future work can build upon this work by exploring more learning theories and how they are incorporated into notetaking apps, as notetaking apps are not one-dimensional in how they support learning. Further, by exploring more learning theories in computer science, new ways to support those theories can be found.

References

Ben-Ari, M. (2001). Constructivism in Computer Science Education. Journal of Computers in Mathematics and Science Teaching, 20(1), 45-73. Association for the Advancement of Computing in Education (AACE). https://doi.org/10.1145/274790.274308

Berssanette, J., de Francisco, A. (2022). Cognitive Load Theory in the Context of Teaching and Learning Computer Programming: A Systematic Literature Review. *IEEE Transactions on Education*, *65*(3), 440-449. https://doi.org/10.1109/TE.2021.3127215.

Breed M., Hauman I., Homann A., Taylor (2013). Choosing Learning Methods Suitable for Teaching and Learning Computer Science. International Association for the Development of the Information Society (IADIS). https://files.eric.ed.gov/fulltext/ED562318.pdf

Cunff, A. (2022). Exploring the power of note-making with the co-founder of Obsidian [blog]. *Ness Labs.* Retrieved from https://nesslabs.com/obsidian-featured-tool

George, R. (2021). *Evernote: More than a note-taking app*. National University. Retrieved from https://www.ncu.edu/blog/evernote-more-note-taking-app#gref

Jiménez-Peris, R., Pareja-Flores, C., Patiño-Martínez, M., & Velázquez-Iturbide, J. Á. (2000, January). New Technologies in Computer Science Education. ResearchGate. Retrieved October 8, 2022, from https://www.researchgate.net/profile/Cristobal-Flores/publication/242324742_New_Technologies_in_Computer_Science_Education/links/0f31752ed04f510939000000/New-Technologies-in-Computer-Science-Education.pdf?origin=publication_detail

Julien, S. (2021). Get Started with Obsidian as a Developer [web log]. Retrieved from https://www.samjulien.com/get-started-with-obsidian-as-a-developer.

Lim, S. [NotionSlim]. (2020, February 10). *I'm the engineer at Notion who built search. AMA!* [Online forum post]. Reddit. Retrieved from https://www.reddit.com/r/Notion/comments/f0lb23/im_the_engineer_at_notion_who_rebuilt_search_ama/

Machanick, P. (2007) A Social Construction Approach to Computer Science Education, Computer Science Education, 17(1), 1-20, DOI: 10.1080/08993400600971067

Thomas L. Naps, Guido Rößling, Vicki Almstrum, Wanda Dann, Rudolf Fleischer, Chris Hundhausen, Ari Korhonen, Lauri Malmi, Myles McNally, Susan Rodger, and J. Ángel Velázquez-Iturbide. 2002. Exploring the role of visualization and engagement in computer science education. In Working group reports from ITiCSE on Innovation and technology in computer science education (ITiCSE-WGR '02). Association for Computing Machinery, New York, NY, USA, 131–152. https://doi.org/10.1145/960568.782998