Visualizing Task Breakdown: An Interactive Force-Directed Graph Approach to Task Management

A Technical Paper submitted to the Department of Computer Science
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 in Computer Science, School of Engineering

By

Lanah Pheng

May 9, 2025

Technical Project Team Members
Yanson Khuu

ADVISOR
N. Rich Nguyen, Department of Computer Science

Visualizing Task Breakdown: An Interactive Force-Directed Graph Approach to Task Management

TaskGraph

Lanah Pheng
UI/UX Programmer
University of Virginia
School of Engineering & Applied Science
dmt9xb@virginia.edu

ABSTRACT

Being confronted with a large, overwhelming task can be debilitating for some people. While traditional task management systems typically rely on linear lists that fail to encapsulate the complex relationships between interdependent tasks, this paper presents a novel approach that leverages graph visualization to enhance task breakdown and management to offer an alternative for the public who may need additional assistance in organizing tasks. In this web application, we introduce a system that represents tasks as interactive nodes through a dynamic network that enables users to visualize hierarchical relationships while manipulating task properties in real-time.

Our implementation utilizes React with ForceGraph2D for our main structure and incorporates a three-state status tracking mechanism with visual color indicators. 5 individuals participated in an in-depth user study that accessed the front-end features on general interface design as well as interactivity evaluation to test the effectiveness of the application. It revealed mixed results: while participants generally found the node selection, visual highlighting, and status color changes intuitive, they encountered challenges with discoverability of right-click functionality and descriptions panel toggles. Participants found the graph-based visualization helpful in conceptually breaking down complex tasks, but they mentioned that latency during graph recentering had the possibility of disrupting workflow and could be further improved. The clarity of the integrated chat interface itself received moderate ratings and participants had expressed an increase desire for more in-depth subtask structure once further AI assistance was implemented. Our findings indicate that force-directed graphs offer a promising alternative to traditional task management interfaces, particularly for visualizing task relationships, though refinements in interaction design and visual feedback are needed to address usability challenges identified during testing.

1 INTRODUCTION

Many people struggle with effective task management in both professional and personal contexts. Traditional approaches such as the creation of linear lists or hierarchical structures fail to capture the complexities between different relationships between tasks. In addition to clearer task visualization, the goal is to create an application that prevents "functional freeze," a term used to describe executive dysfunction, a common symptom of ADHD. By breaking down large, complex tasks into smaller, more manageable chunks, users can easily find which direction to begin a task and are more likely to finish it [1]. This serves to break down mental models through digital task representation to prevent the cognitive burden a generalized task may impose.

To combat this problem, the application TaskGraph was developed. It is a graph-based visual approach to task management that includes interactive nodes within a dynamic force-directed network that portrays tasks in a web-like manner. To use it, users input a task they would like broken down into a chatbot, which then breaks it down into structured nodes. These nodes are mapped in a way to improve spatial understanding and promote intuitive connections between each task.

There are four main objectives for the usage of TaskGraph: interactive node-based visualization, intuitive task relationship representation, real-time editing capabilities, and natural language chat interface. While the breakdown of tasks serves as a separate challenge, this study was designed to specifically focus on investigating the ideal ways users would like to visualize a task breakdown as well as additional features they would like to incorporate into a graph-based application.

The first objective of node-based visualization involves how users would like to envision the tasks as nodes within the graph—whether that be the direct manipulation paradigm and the shape of the node, or the visual selection highlighting when focusing on an individual task node. Another factor of this is the spatial task environment in which the node exists. It is crucial that the design invokes simplicity to ensure a clean, usable interface that does not distract or overwhelm the user. The application must be visually clear and encourage streamlined interaction with frictionless manipulation

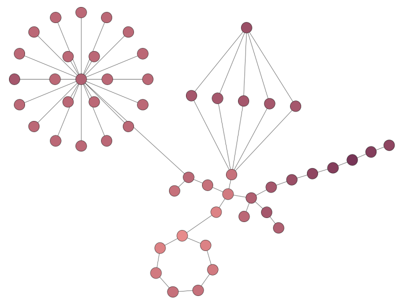


Figure 1: Force-directed graph layout

The second objective of the application is an intuitive task relationship representation. This is done by incorporating a force-directed graph layout shown in Figure 1, that connects nodes to other nodes to represent a meaningful connection between topics. The directional connections aim to show parent-child relationships to build contextual awareness and clarity between tasks. These connections, as well as the automatic proximity positioning of the nodes, add meaningful organization and organic structure.

Another objective is real-time editing capabilities of the nodes. To make it a realistic task managing application, TaskGraph incorporates a three-state tracking system (Not Started, In Progress, Completed) for users to keep track of their task habits. These features, as well as the ability to edit, add, and delete current nodes serves to make the application more visually coherent and usable.

Finally, the fourth objective is to have integration with the natural language chat interface. While currently the enhancement of the automatic task breakdown feature is still in development, ideally there will be task creation through conversational input that would provide the initial structure generation. The final version aims to have a hybrid integration model of natural language input and direct manipulation. This current study explores the direct manipulation portion.

TaskGraph was created using React and the react-force-graph library to render dynamic *force-directed graphs* [2]. The use of the react-force-graph library's built-in physics simulation allowed us to create an intuitive, 2D design with an interactive representation of connections between entities, enhancing user understanding of complex relationships. It is also highly customizable with varying node shapes, colors, forces, and link distances. The use of these technical design choices enables the application to have a responsive design as well as an adaptable interface.

TaskGraph is a novel approach to task visualization and management which aims to be a transformative approach in its unique way of reimagining workflow. The enhanced comprehension it embodies increases the potential for complex project management as it can hone in on task-specific interdependencies which traditional methods may lack. The purpose of this study is to gather feedback from the conducted design studies to outline future considerations to improve and enhance this tool in terms of its features and usability.

2 REVIEW

2.1 Traditional Task Management Approaches

The purpose of TaskGraph is to create a visual application that differentiates itself from traditional task manager methods and systems. A popular option for task management that people gravitate towards are linear list implementations, such as to-do applications, which simply is an ordered set of tasks to check off. Another method is kanban methodologies, an AGILE methodology that visualizes workflow and focuses on continuous improvement. It is often represented by a Kanban board, where tasks move through different stages and work is pulled into the process only when there is capacity. Jira, a software development tool created by Atlassian, is well-known for this [3]. It is a versatile platform primarily used for bug tracking, issue tracking, and agile project management.

While all the listed options are effective when tailored for a users' specific needs, they demonstrate a form of representation incongruence when it comes to mapping the complexity of tasks—where there is a mismatch or inconsistency between thought and behavior. In this case, traditional task management methods make it difficult to flexibly showcase the way the brain thinks about large tasks, creating a significant mental model misalignment [4]. The increased cognitive load that comes with complex projects cannot be represented in linear systems, requiring the creation of a new visualization method that allows for a way to portray tasks intuitively to allow more support for emergent task structures.

2.2 Emerging Visualization Approaches

There are existing mind mapping tools that incorporate visual approaches to task management. One of these is MindMeister, a web-based tool where each topic can have one parent node, limiting the representation of complex tasks relationships that often have multiple dependencies or connections [5]. XMind is another platform that offers slightly more flexibility than traditional mind mapping through its traditional "relationship" feature, but still primarily uses a hierarchical organization system [6]. Lastly, Miro is an infinite canvas board that provides freedom for spatial organization where users can manually position and rearrangement of elements as projects evolve [7]. TaskGraph intends to draw on the most notable features from these applications as well as provide improvements to create a cohesive design that reflects the task making process of the mind.

Ahrens [8] demonstrated benefits of explicit relationship visualization when taking notes, that elevated the use of knowledge graphs to display information. Roam Research [9], first pioneered this concept of bidirectional, associative linking for the purpose of knowledge management. Another app is the widely used Obsidian [10], that uses a graph view with bidirectional linking to organize ideas. It maintains a strict separation between its visualization interface and content editing, requiring users to switch contexts between viewing relationships and managing contents. TaskGraph serves to take the structure from these well-known applications of knowledge graphs and transform it into one that focuses on the creation of actionable tasks, as well as encourage editing capabilities with dynamic properties.

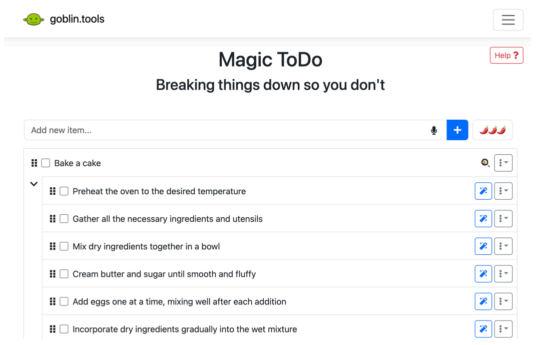
2.3 Force Directed Graph Advantages

A force-directed graph is a visual representation of a network where nodes (points) are arranged based on simulated forces, mimicking a physical stream. These connected nodes are attracted to one another, while all nodes repel, resulting in a layout that reveals the structure and relationships within the network. Di Battista et al. [11] demonstrated improved comprehension with force-directed layouts, leading to its incorporation within TaskGraph. The dynamic node positioning and relationship automate the spatial organization, reducing the cognitive load on the user. The selforganizing properties minimize the need for manual arrangement and the visual aesthetics increase engagement and comprehension. Using this encourages the scalability for more complex task networks.

Along with the graph force providing further comprehension to the layout, it also demonstrates superior pattern recognition in network visualizations according to Ware [12]. Relationship visibility enhances contextual understanding of the user, while its spatial memory utilization improves recall. Other components of force graphs such as the clustering of nodes and the dynamic adjustment reduce the cognitive load through the externalized relationships. These effects are reinforced by Gestalt principles—such as proximity, similarity, and continuity—which guide the user's perception of groupings and patterns within the graph.

2.4 AI-Driven Task Breakdown Approaches

The method of task decomposition through natural language processing techniques to improve usability and user experience is everchanging. Zhang et al. [13] explores the issue of granularity in AI task breakdown, highlighting the challenges of generating subtasks that are neither too broad nor too narrow. During TaskGraph's development, it was crucial to take note of the complexities of each task and recognize the structure it may impose. While further AI support is yet to be implemented, it is important to take note of the design at this stage to plan on what the user might want to see once it is further developed. As for identifying the task itself, semantic parsing has emerged as a promising approach for identifying meaningful subtasks, though its integration with visual representation systems remains limited. AI models have become more advanced, allowing for an adaptive decomposition strategy that respond to the complexity of a given task. Additionally, it is important to take into consideration personalized breakdown styles being developed to suit individual user preferences.



Magic ToDo

Breaking things down so you don't

Add new item...

- Bake a cake
  - Preheat the oven to the desired temperature
  - Gather all the necessary ingredients and utensils
  - Mix dry ingredients together in a bowl
  - Cream butter and sugar until smooth and fluffy
  - Add eggs one at a time, mixing well after each addition
  - Incorporate dry ingredients gradually into the wet mixture

Figure 2: Goblin Tools interface

The task breakdown aspect of the TaskGraph was inspired by Goblin Tools [14], an AI-assisted platform designed to help users manage everyday tasks through adaptive decomposition and cognitive scaffolding techniques. "Spiciness" level is one of the key features of the application, that allows users to adjust the granularity of task breakdowns to suit their preferences. The tool relies on a text-based interface, seen in Figure 2, which limits its ability to visually represent task relationships or maintain persistent spatial organization. Additionally, its contextual understanding across tasks is limited, affecting the coherence of multi-step planning. Despite these constraints, Goblin Tools supports flexible granularity adjustment and empowers users to structure tasks more manageably.

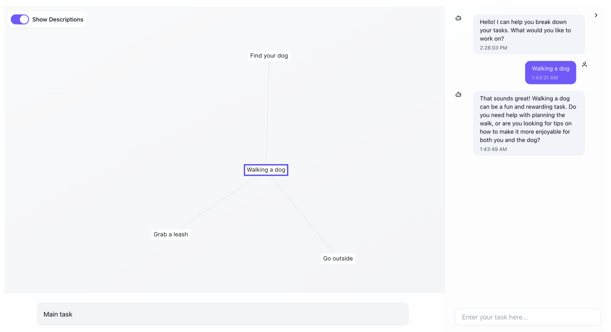
2.5 TaskGraph's Unique Positioning

TaskGraph works as a middle ground between traditional task management methods and powerful AI-driven systems by combining interactive visualization, dynamic decomposition, and cognition in a unified framework. Unlike traditional linear to-do lists or rigid kanban boards, TaskGraph utilizes a force-directed layout that automatically rearranges and organizes tasks based upon dependencies. This self-organization capability reduces manual reordering of tasks and yet maintains context and thus helps users to easily perceive change in status and the relationships between tasks. The interface supports immediate user interaction through tasks that can be easily edited visually while maintaining the relationships between tasks. TaskGraph alleviates the need to switch between separate applications or views in displaying the status, hierarchy, and dependencies simultaneously to simplify the task management process.

Cognitively, TaskGraph matches the way that individuals mentally organize complex tasks—highlighting relational rather than sequential arrangements. The graphical representation intensifies the perception of relationships and supplies visual cues that support prioritization of tasks. This capability allows users to maintain projects according to their inherent cognition processing style, minimizing the risk of cognitive overload and improving memory retention via spatial cognition. TaskGraph also supports various cognitive styles by displaying multiple task decompositions and different paths to depict task structure. By combining the technology benefits of scalable graphs with the cognition support provided by visual representations of tasks, TaskGraph is an innovative leap in task management technology that addresses users' changing needs comprehensively.

3 METHODOLOGY

3.1 System Architecture & Implementation



Hello! I can help you break down your tasks. What would you like to work on?
2:28:03 PM

Walking a dog
1:43:31 AM

That sounds great! Walking a dog can be a fun and rewarding task. Do you need help with planning the walk, or are you looking for tips on how to make it more enjoyable for both you and the dog?
1:43:49 AM

Figure 3. UI of TaskGraph

The TaskGraph application was developed using a component-based architecture centered around interactive visualization of task relationships. Our implementation leverages React as the primary framework, providing a modular approach to UI development through functional components with hooks-based state management. This architecture facilitates the separation of concerns between visualization, interaction, and data management layers. The application structure consists of four primary modules: the graph visualization engine, chat interface, task data manager, and state

For the graph visualization engine, we integrated react-force-graph-2d, which implements a physics-based layout system with customizable forces and interactive node manipulation. This library provides essential capabilities including automatic collision detection, smooth animations, and efficient canvas rendering that maintains performance even with dozens of interconnected nodes. The visualization engine was extended with custom node rendering to support rectangular task representations and status-based color indicators.

The styling system utilizes Tailwind CSS for consistent design language across components, with shadcn/UI providing accessible UI primitives that maintain visual coherence. This combination enables rapid iteration of the interface while ensuring responsive behavior across device sizes. State management is implemented through React's Context API with reducers to maintain application-wide state consistency. This approach ensures that task modifications propagate correctly throughout the system while preserving interaction history for undo/redo capabilities. The state management system explicitly tracks node positions and relationships, selection state for focused tasks, edit mode toggles for different node properties, view state for panels and modality shifts, and task status transitions across the three-stage workflow.

The architecture follows a decoupled design pattern where the graph visualization responds to state changes rather than directly manipulating the underlying data model. This interaction-driven design ensures state preservation across sessions while allowing for real-time collaborative extensions in future iterations. Critical to the system's effectiveness is the coordinate system transformation that maps abstract task relationships to visual space while maintaining spatial continuity during interactions. This transformation ensures tasks maintain meaningful relational positions even as users manipulate individual nodes or toggle between expanded and collapsed view states.

3.2 Features & Design Implementations

The final version of TaskGraph uses a force-directed graph approach to represent tasks and their relationships in a dynamic, interactive context. Nodes are laid out using a physical simulation that ensures automatic collision detection and optimum spacing to enable smooth transition and improve clarity as the task network evolves. Directed edges are used to graphically display task dependencies, thus augmenting the topological structure of the project. Every task is depicted as a rectangular node with an accompanying textual label. The spatial layout is fully editable, allowing manual changes where needed. Zoom and pan controls enable viewport navigation, and the interface is fully responsive to support a variety of screen sizes. The visualization framework is based on the principles of spatial cognition, providing a representation of tasks that is semantically complete and topologically correct.

TaskGraph supports the representation of hierarchical tasks with the use of parent-child relationships via directed edges and allows users to build multi-level hierarchical tasks. The visual hierarchy mirrors logical groupings that help users organize and monitor complex workflows effectively. The modification of tasks is supported via an intuitive and contextual interface. Users have the capacity to edit task headers directly, include rich text annotations, and re-order tasks via a drag-and-drop action.

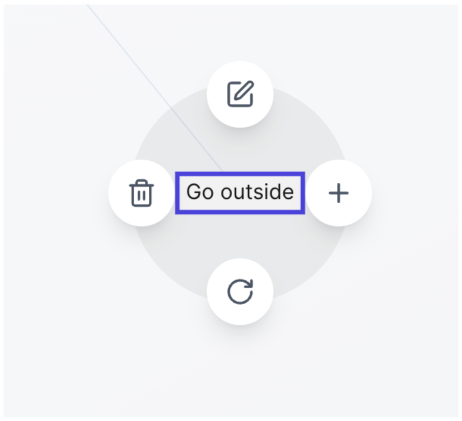


Figure 4. Node context menu

TaskGraph's interaction model centers on intuitive user engagement. Clicking a node selects it, automatically adjusting the zoom and centering the camera for a focused view. When a task is selected, it is highlighted by a purple border. Clicking on the background deselects the node, maintaining spatial continuity and clarity. A right-click on a node opens a context menu, displayed in Figure 4, that offers four primary choices: Add, Edit, Status Change, and Delete. The menu is always located relative to the node, thus enabling consistent and intuitive task management. Adding a task creates a pop-up with the option to enter a new task name and optionally enter a description along with it. Edit creates a similar pop-up to change the task name. When Status is clicked, the color of the node is changed and conveyed through its color-coding: white for not yet started, yellow for in progress, and green for completed. Task deletion will delete a task and warn the user if it has cascading effects to remove linked subtasks and thus preserve structure coherence.

Interaction with the chat panel involves enables expansion and collapse, with smooth transitions. The graphical interface is collapsable to ensure visibility and centricity with respect to the changing panels. This is a multi-layered design that follows the principles of affordances, direct manipulation, and spatial consistency throughout the interface.

3.3 AI Integration Framework

While this study focuses on the front-end portions of the application, AI still plays a role in understanding the way users foresee tasks being broken down after having a conversation with the chatbot. The architecture of integrating with AI is based on a natural language processing pipeline that transforms user input to systematically arranged visual representations of tasks. Main input is from the chat interface where users state tasks in natural language. User inputs are deciphered by a large language model (Llama3) that understands task components, their relationships and relative complexity levels. Within task decomposition, the system employs a hierarchical analysis to define top-level objectives and their respective sub-components. The analysis yields an initial graph structure that defines appropriate parent-child node relationships. Additionally, the system determines an adequate degree of granularity with respect to task complexity, thus providing users with an acceptable cognitive load.

The graph generation algorithm creates a dynamic visualization using the react-force-graph library with physics-informed node positioning that successfully communicates relational hierarchies without compromising on clarity. Every node represents a unique task with appropriate visual cues to represent status and importance. The continuous improvement mechanism employs user feedback and interaction data to improve the task segmentation process and employs insights from successfully completed tasks and adapts to user tendencies so that it continuously evolves with tailored task breakdown templates that mirror users' unique work patterns and working style preferences.

3.4 User Study Methodology

Assessment of our task graph application was performed with mixed-methods usability research involving five participants over a comprehensive protocol with guided tasks and free exploration that was employed to investigate both guided and spontaneous interaction behaviors. Quantitative measures of task completion time, Likert-scale satisfaction ratings from 1 to 5, and qualitative information collected through think-aloud protocols and open-ended commentaries were collected in each of the 30- to 40-minute-long sessions. Following a systematic breakdown of the protocol into progressive phases of initiating the generation of tasks via the chat interface, determining node selection and navigation, task info edit functionality, status change function capability, interaction with panels, and personal task generation, we measured completion times per phase with user confusion instances and collected focused comments on specific interface aspects and interaction modalities.

The information collected using the tools utilized consisted of standardized measures of task time, records of confusion, feature-specific satisfaction ratings using 5-point Likert scales, and standardized interviews after task completion. The analytical methodological framework focused on the detection of interaction difficulties, accessibility problems with regards to features, and the root causes of user satisfaction. Major measures examined included the time to complete benchmark tasks, feature discoverability ratings, transition fluency ratings, and thematic analyses of qualitative feedback to identify trends in participant feedback. This evaluative approach provided rich insights into usability problems and successful interaction models.

4 RESULTS

4.1 Participant Overview

Five participants, aged between 18 to 22, took part in the user study, providing varied perspectives on task management. In general, everyone reported frequent use of task management tools, with 80% of them reporting weekly or more frequent use. The most common tools used included traditional methods of paper lists and planners (100%), calendar programs (80%), and electronic to-do lists (60%). Participants reported moderate familiarity with visual task management systems, with a mean rating of 3.2 out of 5; however, no one reported high levels of experience with graph-based visualizations in particular.

Four of the participants had previous experience with AI-supported tools, primarily ChatGPT, that gave them a contextual background for understanding AI-based task decomposition. Their typical task management needs covered various areas, including academic projects (100%), personal projects (100%), work (80%), and daily chores or tasks (60%). This diverse experience allowed the participants to evaluate the task graph app from various usage perspectives.

The user test results revealed significant trends in the users' interaction with the task graph application. Task time to completion, usability ratings, and the percentage of successful completions were collected from the five users under normal use conditions.

4.2 Task Creation Workflow

Participants required on average 33.7 seconds to discover and use the chat panel, with significant spread observed between individuals (ranging from 16.2 to 53 seconds). The main challenge was caused by the chat panel's location, with one participant stating that they "wouldn't think that was the first button you'd see" in their experience with chat facilities as a newcomer. Users gave a clarity score of 3.8 out of 5 to the process of creating tasks through chat, reflecting moderate satisfaction but also the possibility of improvement. Strong conflict was observed between user expectations and the way the system works. Most participants expected the chat interface to return a conversational response rather than display a graph visualization. As described by one participant: "Wasn't expecting the visual to come along with the prompt." In addition, participants universally mentioned the lack of loading indicators during processing time, which was on average 17 seconds long. One participant suggested the use of "3 dots" to inform users that the system is in a processing state.

4.2 Task Editing and Manipulation

The behavior of right-clicking to open editing options received a variety of discoverability ratings, averaging 3.4 out of 5. The interaction pattern showed that most users first tried double-clicking before learning about the right-click menu. One participant suggested that the editing functionality "should appear when you hover on it" to make it more discoverable. By contrast, description editing was highly intuitive when found, with a rating of 4.2 out of 5, and users were pleased with the hover behavior. One participant said that "hovering was pretty clear," while another characterized the editing process as "chill." However, some participants expected the description editing controls to be part of the node context menu, instead of being placed separately at the bottom of the screen.

The status change functionality showed high comprehension, with a clarity rating of 4.4 out of 5, but mixed reception to its implementation. All participants correctly identified that the color white meant "not started," yellow meant "in progress," and green meant "completed." However, several participants had accessibility concerns, with one suggesting, "Need label [and] strikethrough cross to make it work," to provide better visual distinction, particularly for color-blind users. Another participant said that using colors to indicate status could conflict with color-coding for categorization, commenting, "People could use colors to color code so using colors to represent status wouldn't be that effective."

4.4 Panel and Layout Interactions

The toggle of the description panel was one of the hardest to find, with search times ranging from 5.48 seconds to over 50 seconds. This wide spectrum of search times reflects significant usability problems pertaining to visibility and position of the feature. In addition to that, the function of the toggle also scored relatively low with an average score of 2.5 out of 5. Participants questioned the need for a dedicated toggle and stated that the same result may be attained by clicking outside of other regions of the interface: "Would never be turning off the description from up there, would be clicking outside the thing instead." One participant stated that the description panel "doesn't give that much meaning/real estate" and mentioned that it "doesn't preserve line breaks and things so it is smooshed in one thing."

Smoothness of transitions between opening and closure of the chat panel received mixed assessments with a mean score of 3.6/5 based largely on issues with the observed latency between the panel closure and the recenter of the graph. A number of participants called the delay "jarring" or "disorienting." One participant stated, "The latency in the moving is jarring. It is nice in the way that it moves but it takes too long." Participants suggested that the graph should instantly react or move together with the panel: "The graph should do it as I close it."

4.5 Task Breakdown Structure Analysis

The original structure with a main task and three subtasks was subject to multiple interpretations. An overwhelming majority of participants perceived a time-based sequence since three out of five participants mentioned the same upon being asked explicitly to list them in sequence with the main task. Participants rated the clarity of the three-node structure at 3 out of 5 upon being asked to design their own assignments from scratch. Participants viewed the subtask structure as largely helpful in general with one participant noting that it "helped me to split it into pieces, made it less overwhelming." However, participants had divergent opinions with regards to the optimal number of subtasks with most arguing that that should be context-dependent: "Would depend on the task. If it is autogenerated, then it should know how many subtasks there are."

Participants identified multiple visualization problems in impeding their understanding of task relationships. One of these problems was related to node spacing with one participant insisting that "the nodes should be more spread out to clarify relationships." Many of the participants described being unable to separate main tasks from subtasks in the visual hierarchy. An ongoing technical issue was in creating multiple subnodes in a task causing node overlap that created visual ambiguity. One participant stated that "the creation overlaps a previous node," which shows a lack of space distribution in the force-directed layout.

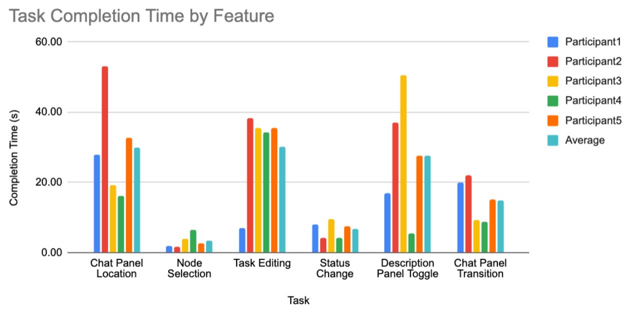
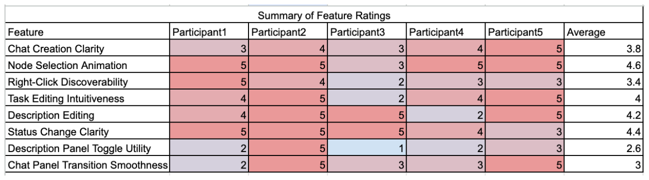


Figure 5. Summary of task completion time by feature



Summary of Feature Ratings						
Feature	Participant1	Participant2	Participant3	Participant4	Participant5	Average
Chat Creation Clarity	3	4	3	4	5	3.8
Node Selection Animation	5	5	3	5	5	4.6
Right-Click Discoverability	5	4	2	3	3	3.4
Task Editing Intuitiveness	4	5	2	4	5	4
Description Editing	4	5	5	2	5	4.2
Status Change Clarity	5	5	5	4	3	4.4
Description Panel Toggle Utility	2	5	1	2	3	2.6
Chat Panel Transition Smoothness	2	5	3	3	5	3

Figure 6. Summary of participants' feature ratings

5 DISCUSSION

The empirical study of the TaskGraph application produced significant insights in relation to user interaction with systems that utilize graph-based task visualizations. Participants had the ability to understand the general concept of representing tasks in the form of a networked set of nodes where the graphical organization allowed them to perceive the complex dependencies between tasks. In the words of one of the participants, the graph visualization "allowed me to divide it into pieces, made it less overwhelming," which indicates the enabling nature of spatialization.

The selection mechanisms used and the highlighting were positively reviewed since the visual cues of the selected nodes were perceived to be visible and understandable by the participants. Animation of the selected nodes was met with high ratings, with a mean score of 4.6 out of 5 and hence ranked among the most important traits of the application. These results suggest that feedback in the form of dynamic visualizations contributes to maintaining user interest and improving spatial perception of the graph.

However, the study demonstrated substantial differences between user expectations and the system's operational performance. Most users had expected the chat interface to communicate in a conversational style instead of generating a graphical representation. This discrepancy in expectations identifies the need for clearer information about the application's functions and better guidance for novice users.

5.1 System Effectiveness Evaluation

The force-directed graph showed mixed success in its ability to visualize tasks. While participants appreciated the visualization of relationships between tasks, they experienced spatial and visual hierarchy problems. The automatic layout of nodes sometimes was counterintuitive, as several participants noted that nodes would overlap when creating several subtasks from one parent task. One participant remarked particularly that "the nodes should be more spread out to clarify relationships," which indicated that the current implementation of the force simulation may require additional improvement. The color-coding system for task status—where white is not started, yellow is in progress, and green is complete—was fully understood but was questioned in terms of its accessibility and usability. Several participants expressed concerns related to its suitability for color blindness; one suggested that task status should be indicated by both "label [and] strikethrough" in addition to the use of color. In addition, another participant raised a fundamental issue by noting that "people could use colors to color code so using colors to represent status wouldn't be that effective," thus highlighting a potential conflict between the indication of status and the categorization organization.

The task management and editing features showed varying degrees of effectiveness. Task action context menu, available via right-click, had moderate discoverability problems since participants first resorted to other interaction patterns. Most users tried double-clicking before recognizing the right-click menu, and it was shown that the application may be improved with use of multiple interaction patterns to cater to different expectations from users. Description edit was positively rated for being intuitive once discovered, with participants commending the use of hover controls to edit. However, there was confusion over where to place these controls since most users expected them placed in the node context menu and not in a distinct panel at the screen's bottom end. This difference between where users expect and where the controls were placed is a significant usability issue. Three-state task status was understandable to participants, even though it had implementation problems. While all participants correctly interpreted what was being represented with changing colors, some complained about clarity and access to visual presentation of the approach in question. Status changes were rapidly performed with little time for interaction to become oriented to them, yet lacked enough visual differentiation to be immediately identified and understood by all users.

5.2 System Integration

The integration of the chat interface with the graph visualization provided many points of improvement opportunity. Participants took 33.7 seconds on average to find the chat panel, implying poor visibility or placement. Users talked about uncertainty over the purpose of the chat in the context of creating tasks since most expected a conversational response and not automated production of graphs. Transition from chat entry to graph visualization lacked adequate feedback. Participants consistently commented on the lack of loading indicators over the processing time of 17 seconds on average; one of them suggested the use of "3 dots" to signal that the system was working. This lack of feedback caused confusion over the correct operation of the system that could undermine user confidence. On opening and closing the chat panel, participants noted significant latency issues with the recentering of the graph. The duration from the closing of the panel to the adjustment of the graph that ensued was described by several participants as "jarring" or "disorienting." One participant explicitly commented that "the latency in the moving is jarring. It is nice in the way that it moves but it takes too long," which indicates that the animation design was appreciated while execution needed to be improved.

The coherence of the user interface overall reflected inconsistencies between interaction patterns and ease of discovering functions. Toggling the description panel was problematic in that search times oscillated between 5.48 seconds and more than 50 seconds. Such wide variability indicates a salience flaw that interrupted otherwise smooth interaction with the system. Users gave feedback on particular component usability of the interface and commented that the description panel "doesn't give that much meaning/real estate" and that it "doesn't hold line breaks and things so it is smooshed in one thing." Such comments reflect the importance of more advanced text format capabilities to allow more complete task descriptions.

5.3 Key Challenges and Limitations

The research conducted revealed some serious user experience challenges. Foremost among the challenges was discoverability of the node edit option accessible by right-click, which acted as a barrier to task management effectiveness. Placing of the edit controls evenly in the context menu and in the description panel caused confusion and increased the learning curve. In addition to that, the toggle option of the description panel acted as a significant usability challenge since many of the participants had problems finding it and questioned its purpose. One participant stated that they "would never be turning off the description from up there, would be clicking outside the thing instead," which shows that the toggle feature reflected current interaction patterns in a less intuitive way. Another challenge that arose was in interpreting the structure of the graph itself. Participants tended to infer a sequential relationship between the main task and its subtasks from the visualization itself even though the visualization was intended to show hierarchical relationships and not to infer a sequence. This mismatch between the intended meaning and the perceived interpretation can cause confusion during the planning and execution of tasks.

The technical implementation uncovered many limitations that substantially affected the general user experience. On some occasions, the force-directed algorithms produced less than ideal node arrangements, particularly where many subtasks originated from a common parent task. This generated visual clutter and difficulties in comprehending the task relationships. Moreover, long transition times between panels adversely affected what otherwise should have been a smooth experience with users commenting on perceived delays between their actions and feedback from the system. One user stated that "the graph should do it as I close it," highlighting the need for instant visual feedback that was unmet. Users concluded that the text processing capabilities of the field of description were inadequate since it "doesn't save line breaks and things so it is smooshed in one thing." This limitation weakened rich task descriptions and probably discouraged users from providing additional information.

5.1 Future Development Opportunities

Future versions of TaskGraph can benefit greatly from refinements in visualization methods. A more sophisticated force simulation can improve the spacing of nodes, avoid overlaps, and clarify relationships. As one participant put it, "the nodes need to be more spaced out to make relationships clear." The status visualization mechanism can be extended to use a wider range of visual cues other than color, adding such elements as icons, text labels, or strikethroughs for completed tasks. Not only would this remedy accessibility problems, but it would also allow users to reserve color for other organizational structures. A stronger visual hierarchy would help users distinguish between primary tasks and their respective subtasks, and therefore remedy the confusion experienced by several users. This could include size variations, border types, or other visual discriminators that emphasize the parent-child relationships.

The model of interaction has to be scalable to allow multiple routes to similar functions in order to cater to multiple user expectations. For example, editability may be supported through double-click and right-click in a way that increases the system's ease of use to a large user population. In addition to that, edit controls may be standardized to avoid confusion so that all task management functions may be accessed from one context. Such a change will remove the current dichotomy between edits at the node level and modifications to the descriptions and create a unified user experience. Finally, there should be added loading indicators and transition animations to offer feedback about working procedures to the user. Having "3 dots" or similar loading indicators during processing is a good way to meet user feedback about lack of information during waiting times.

This study focused more on the visualization interface than on artificial intelligence integration, but participants indicated that they would prefer a greater variety of functions related to task decomposition. Future developments may address more adaptive decomposition methods that can modulate based on both task complexity and user preferences. In addition, the chat interface can be developed into a more conversational style, along with more advanced visual task hierarchies, which will more closely align with user expectations while still maintaining the strengths of graph-based visualizations. In-system customization features will enable the system to learn from user interactions and conform to personal working styles. Since participants had varying preferences for task structure and breakdown granularity, an adaptable program is likely to offer increasingly personalized experiences over time.

5.5 Broader Implications

TaskGraph demonstrates a next-generation task management methodology that aims to overcome the weakness in traditional linear lists and hierarchical systems. Portraying tasks as nodes in an interactive network has the prospective benefits of enabling users to better understand complex task relationships and dependencies. Outcomes from this study translate to more than one specific application. They show that visualizations involving graphs can improve users' capacity to form mental images of complex tasks and potentially ease mental load and task accomplishment. One participant commented that the visualization "allowed me to break it down into parts, made it less daunting." However, the challenges discovered—most importantly with regards to interaction patterns, feature discoverability, and visual clarity—identified that careful design is required to embrace innovative visualization methods. Users come with expectations shaped by traditional systems and deviations from these ingrained patterns require explicit guidance and feedback to deliver a productive user experience. For users experiencing executive function challenges, such as individuals with a diagnosis of ADHD, TaskGraph methodology shows great value in yielding visual assistance to undertake complex tasks. It remains to be studied specifically to evaluate its effectiveness in reducing "functional freeze" and improving task initiation and accomplishment in such demographics. While task management moves towards more visual and increasing use of AI-infused methodologies, TaskGraph demonstrates serious insight into the prospective benefits and pitfalls in applying graph-based visualization. By refining its interaction design, visual feedback, and use of AI, it may be an important step in making complex task management more intuitive and accessible.

ACKNOWLEDGEMENTS

I would like to thank Yanson Khuu for developing the AI integration portion of the application and the feedback throughout the initial design process. Further implementation of TaskGraph will be incorporated using this study and his expertise.

REFERENCES

[1] R. A. Barkley, *Attention-Deficit Hyperactivity Disorder: A Handbook for Diagnosis and Treatment*, 4th ed. New York: Guilford Press, 2015.

[2] R. Vasturiano, "react-force-graph," GitHub repository. [Online]. Available: https://github.com/vasturiano/react-force-graph.

[3] Atlassian. 2023. Jira Software: The #1 software development tool used by agile teams. Atlassian. https://www.atlassian.com/software/jira

[4] Mona Haraty, Joanna McGrenere, and Charlotte Tang. 2016. How personal task management differs across individuals. International Journal of Human-Computer Studies 88, 1 (2016), 13-37.

[5] MeisterLabs. 2023. MindMeister: Online Mind Mapping and Brainstorming. https://www.mindmeister.com/

[6] XMind Ltd. 2023. XMind - Mind Mapping Software. https://www.xmind.net/

[7] Miro. 2023. The Visual Workspace for Innovation. https://miro.com/

[8] Sönke Ahrens. 2017. How to Take Smart Notes: One Simple Technique to Boost Writing, Learning and Thinking. CreateSpace Independent Publishing Platform.
[9] Roam Research. 2023. A note-taking tool for networked thought. https://roamresearch.com/
[10] Obsidian. 2023. A second brain, for you, forever. https://obsidian.md/
[11] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. 2018. Graph Drawing: Algorithms for the Visualization of Graphs. Prentice Hall.
[12] Colin Ware. 2021. Information Visualization: Perception for Design (4th Edition). Morgan Kaufmann.
[13] Amy X. Zhang, Michael Muller, and Dakuo Wang. 2022. How AI-based systems can improve task breakdowns: A study of granularity in task management. In Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22).
[14] Goblin Tools. 2023. Goblin Tools: AI-assisted tools for everyday tasks. https://goblin.tools/