To conduct a comprehensive discovery of established principles, best practices, existing research, influential tools, and well-regarded examples related to information design, visual hierarchy, and overall design language specifically for text-based output from Command-Line Interface (CLI) tools. The aim is to understand how the clarity, usability, and aesthetics of CLI output can be systematically improved, drawing parallels from established design disciplines like print, advertising, UI/UX, signage, and information dashboard design, but adapted for the constraints and capabilities of a terminal environment.

Ten studies confirm that CLI output clarity and usability can be systematically improved by adapting established design principles and specific formatting techniques from various design disciplines.

Abstract

Ten studies document that CLI output can be improved by applying clear visual design principles. Monospace typefaces, structured indentation, and judicious spacing support legibility and explicit grouping of information. Syntax and semantic highlighting—while aiding rapid parsing—carry risks of overuse that may lead to eye fatigue. Several reports endorse ASCII visualizations and diagrams for inline explanation in constrained terminal environments, with one study noting that interactive ASCII graphs preserve workflow.

Design approaches drawn from print, UI/UX, and dashboard practices are adapted in CLI contexts. Four studies use structured text visualization methods, and one study outlines guidelines for accessible, screen reader–compatible output. Among the ten studies, five focus on visualization or usability principles while three provide user-preference evidence. These findings indicate that established design disciplines, when carefully tailored, foster improvements in clarity, usability, and aesthetics for text-based CLI tools.

Paper search

Using your research question ”To conduct a comprehensive discovery of established principles, best practices, existing research, influential tools, and well-regarded examples related to information design, visual hierarchy, and overall design language specifically for text-based output from Command-Line Interface (CLI) tools. The aim is to understand how the clarity, usability, and aesthetics of CLI output can be systematically improved, drawing parallels from established design disciplines like print, advertising, UI/UX, signage, and information dashboard design, but adapted for the constraints and capabilities of a terminal environment.”, we searched across over 126 million academic papers from the Semantic Scholar corpus. We retrieved the 50 papers most relevant to the query.

Screening

We screened in papers that met these criteria:

- **CLI/Text Interface Focus**: Does the study examine text-based user interfaces, command-line interfaces, or terminal-based applications and their output formatting?
- **Text Display Analysis**: Does the study include analysis of user interaction with or readability of monospace/text displays?
- **Design Principles**: Does the research address information design principles or visual hierarchy specifically within text-based or constrained digital environments?
- **Cognitive Processing**: Does the study examine how users cognitively process and interact with textual information displays?
- **Empirical Evidence**: Does the research include empirical data, case studies, or validated examples of CLI tool implementation?
- **Typography/Layout**: Does the study address typography or layout considerations within constrained digital/text environments?
- **Interface Type**: Does the study include text-based interface components (rather than focusing exclusively on graphical interfaces)?
- **Study Scope**: Does the study focus on output design and presentation (rather than solely on input methods, command parsing, or programming language syntax)?

We considered all screening questions together and made a holistic judgement about whether to screen in each paper.

Data extraction

We asked a large language model to extract each data column below from each paper. We gave the model the extraction instructions shown below for each column.

- **Study Design Type**:

  Identify the primary type of research methodology used in the study. Options may include:

- Empirical user study
- Theoretical/conceptual design research
- Prototype development and evaluation
- Comparative analysis
- Qualitative investigation

Look in the methods section or research approach description. If multiple approaches are used, select the primary methodology that best characterizes the overall study. If unclear, note "Unclear" and provide a brief explanation.

- **Research Participants/Context**:

  Describe the participants or user groups involved in the study:

- Professional roles (e.g., system administrators, developers, power users)
- Expertise level (novice, intermediate, expert)
- Number of participants
- Selection criteria

Extract this information from the methods section, participant description, or recruitment details. If no explicit participant information is provided, note "Not specified". If participant details are partial, extract what is available and note any limitations.

- **CLI Design Approach or Intervention**:

  Describe the specific CLI design approach, intervention, or prototype:

- Key design principles or innovations
- Specific features or mechanisms introduced
- Technological or interface modifications
- Theoretical framework guiding the design

Locate this information in the methods, design description, or results sections. Be precise about the specific design elements or interventions. If multiple approaches are described, list them in order of prominence.

- **Outcome Measures and Evaluation Criteria**:

  Identify the primary outcomes or evaluation metrics used to assess the CLI design:

- Usability metrics
- Task completion time
- User satisfaction measures
- Cognitive load
- Error reduction
- Learning curve

Extract from results, discussion, or methodology sections. Prioritize quantitative metrics if available. Include specific measurement techniques or scales used. If outcomes are qualitative, describe the key themes or findings.

- **Primary Research Findings**:

  Summarize the key findings related to CLI design, user experience, or interface improvements:

- Most significant discoveries
- Unexpected insights
- Design recommendations
- Limitations identified

Extract from results and discussion sections. Focus on findings directly relevant to improving text-based CLI output, visual hierarchy, or information design. Aim for a concise but comprehensive summary that captures the core research contributions.

Results

Characteristics of Included Studies

Study	Study Focus	Design Principles Examined	Environment Type	Methodology	Full text retrieved
Isaacs and Gamblin, 2019	Preservation of Command-Line Interface (CLI) workflow via American Standard Code for Information Interchange (ASCII) visualization for package dependency graphs	Interactive ASCII visualization, user-centered design, visual task alignment	CLI (Spack, remote/terminal)	Prototype development and evaluation	No
Jacques and Kristensson, 2015	Impact of code presentation on developer efficiency and accuracy	Layout, indentation, typefaces, anti-aliasing, syntax and semantic highlighting, secondary notation	Code editors, CLI code output	Theoretical/conceptual design research	Yes
Sampath et al., 2021	Accessibility of CLIs for screen reader users	Structured vs. unstructured text, accessibility heuristics	General CLI (screen reader context)	Empirical user study	No
Baecker and Marcus, 1986	Enhanced visualization of program source text	Five design principles for visualization, comprehensive framework	Source code, CLI/code editors	Theoretical/conceptual design research	No
Hayatpur et al., 2024	Role and design of ASCII diagrams in codebases	Visualization within code, design space for ASCII diagrams	Codebases, CLI/editor environments	Qualitative investigation	No

Study	Study Focus	Design Principles Examined	Environment Type	Methodology	Full text retrieved
Verma, 2013	Hybrid graphical command line interface (Gracoli)	Integration of CLI and Graphical User Interface (GUI), user experience, interactive output	Hybrid CLI/GUI	Theoretical/conceptual design research	No
Cabot, "Re-imagining the Command Line"	CLI user experience and problem-solving	Usability-enhancing features, command suggestion mechanisms, action specification model	CLI (general, expert users)	Prototype development and evaluation	No
Crichton, 2020	Auto-generated documentation as information visualization	Information-dense visualization, usability, searching and scanning	Documentation tools (CLI, Application Programming Interface (API) docs)	Theoretical/conceptual design research	No
Baecker and Marcus, 1983	Human factors in program documentation	Typeset representations, framework for interface enhancement	Source code, CLI/editor	Theoretical/conceptual design research	No
Murillo et al., 2015	Usability factors in legacy CLI for network security	Usability heuristics, cognitive science concepts, legacy CLI attributes	Legacy CLI, graphical interfaces	Theoretical/conceptual design research	No

Design Principles Examined:

- Visualization-related principles:Examined in 5 studies (including ASCII visualization, information-dense visualization, diagrams, typeset representations).
- Usability and user experience principles:Examined in 5 studies (including usability heuristics, user-centered design, usability-enhancing features).

- Layout, notation, and representation:Examined in 2 studies (layout, indentation, typefaces, anti-aliasing, syntax/semantic highlighting, secondary notation).
- Accessibility principles:Examined in 1 study.
- Integration or hybridization of CLI and GUI:Examined in 1 study.
- Command suggestion mechanisms and action specification models:Examined in 1 study.
- Cognitive science concepts:Examined in 1 study.
- Frameworks or comprehensive models for visualization/interface enhancement:Examined in 2 studies.
- Searching and scanning principles:Examined in 1 study.
- Legacy CLI attributes:Examined in 1 study.

Environment Type:

- CLI environments:Examined in 8 studies (including general CLI, remote/terminal, screen reader, and legacy CLI).
- Code editors or editor environments:Examined in 4 studies.
- Source code or codebases:Examined in 3 studies.
- Hybrid CLI/GUI environment:Examined in 1 study.
- Documentation tools or API docs:Examined in 1 study.
- Graphical interfaces:Examined in 1 study.
- Legacy CLI specifically:Examined in 1 study.

Methodology:

- Theoretical or conceptual design research:Used in 6 studies.
- Prototype development and evaluation:Used in 2 studies.
- Empirical user study:Used in 1 study.
- Qualitative investigation:Used in 1 study.

Based on available abstracts and full texts, most studies relied on theoretical or conceptual approaches, with only a small number employing empirical user studies or qualitative investigations. The majority of studies focused on Command-Line Interface environments, with some considering code editors, source code, or hybrid/documentation environments.

Thematic Analysis

Visual Hierarchy in Text-Based Environments

- Typography and Spacing:
  - Several included studies report the importance of typography, specifically the use of legible, monospace typefaces and structured indentation, to clarify code structure and output hierarchies.
  - Indentation is described as a means to make scope and grouping explicit, while spacing and layout facilitate the association or separation of information blocks.
  - Anti-aliasing, or the smoothing of font edges, is mentioned as a method to enhance readability, particularly at larger font sizes.
- Color and Emphasis:

    - Syntax highlighting (coloring language tokens) and semantic highlighting (assigning unique colors to variables or elements) are reported as key mechanisms for improving the visual parsing of CLI output and code.
    - Syntax highlighting aids quick identification of language constructs, while semantic highlighting can support understanding of data flow.
    - Some studies note, however, that excessive use of high-contrast color schemes can cause eye fatigue, and overuse of semantic coloring may overwhelm users.

- Information Grouping:
    - Grouping and ordering of statements, as well as the use of secondary notation (visual cues that do not alter program semantics, such as brackets or whitespace), are recommended to reduce cognitive load and aid comprehension.
    - ASCII diagrams, as described by Hayatpur et al. (2024), serve as a form of in-line grouping and visual explanation within codebases, supporting knowledge transfer and documentation.

Cross-Disciplinary Design Adaptation

- Print Design Principles:
    - Several studies advocate for adapting print design principles—such as clear visual hierarchy, effective whitespace, and typographic discipline—to CLI output.
    - Typeset representations and frameworks for code presentation are proposed to enhance readability and memorability.
- User Interface and User Experience Patterns:
    - The adaptation of user interface and user experience best practices, such as interactive elements (command suggestion, graphical overlays), is explored as a way to augment traditional CLI environments.
    - Hybrid approaches combining CLI and GUI elements are proposed to merge the efficiency of command lines with the intuitiveness of graphical interfaces.
- Information Dashboard Concepts:
    - Information-dense visualizations, inspired by dashboard design, are recommended for documentation and CLI output to support efficient searching and scanning.
    - Presenting method signatures and structural information in a compact, visually navigable format is emphasized.

CLI-Specific Design Language

- Terminal Constraints and Capabilities:
    - Multiple studies highlight the unique constraints of terminal environments: limited color palettes, fixed-width fonts, and lack of graphical widgets.
    - ASCII visualizations and diagrams are described as practical responses to these constraints, offering lightweight and portable solutions.
- Text-Based Layout Strategies:

  - Techniques such as ASCII diagrams, structured indentation, and visual task alignment are used to create visual hierarchy and guide user attention.
  - The design space for ASCII diagrams, as mapped by Hayatpur et al. (2024), illustrates the diversity and adaptability of these strategies.
- Interactive Elements:
  - The inclusion of interactive features, such as command suggestion mechanisms and interactive ASCII graphs, is reported as a way to enhance usability and efficiency in CLI environments.
  - Some users, however, perceive increased automation as less engaging.

Implementation Patterns

Study	Design Pattern	Implementation Technique	Use Cases	Effectiveness
Isaacs and Gamblin, 2019	Interactive ASCII visualization	ASCII Directed Acyclic Graphs (DAGs) in terminal, user-centered design	Package dependency analysis (Spack)	Preferred by users for workflow preservation despite ASCII limitations
Jacques and Kristensson, 2015	Indentation, layout, highlighting	Structured code output, typeface selection, syntax/semantic coloring	Code editing, CLI output	Theoretically improves comprehension and reduces cognitive load; lacks formal validation
Sampath et al., 2021	Structured output for accessibility	Recommendations for structuring CLI text, screen reader compatibility	CLI use by visually impaired users	Identified major accessibility barriers; recommendations provided but not empirically validated
Baecker and Marcus, 1986	Visualization framework	Five design principles, ten structural areas, applied to C programming language	Codebase readability and maintainability	Conceptually enhances readability; empirical validation not provided
Hayatpur et al., 2024	ASCII diagrams for visualization	Corpus analysis, design space mapping, interviews	Documentation, code understanding, knowledge transfer	Widely used and diverse; foundational for future tool development

Study	Design Pattern	Implementation Technique	Use Cases	Effectiveness
Verma, 2013	Hybrid CLI-GUI interface	Gracoli system, graphical overlays on CLI	Complex tasks, improved output interpretation	Proposed to improve experience; lacks empirical evaluation
Cabot, ”Re-imagining the Command Line”	Command suggestion, usability features	Suggestion mechanisms, theoretical action model	Problem-solving, exploratory CLI use	Increased efficiency but reduced engagement; integration challenges
Crichton, 2020	Information-dense visualization	Enhanced auto documentation layouts for APIs	API documentation lookup	Design principles proposed; effectiveness not empirically measured
Baecker and Marcus, 1983	Typeset source code interfaces	Automated typesetting, framework development	Program documentation, code review	Conceptual benefits for human factors; empirical data not reported
Murillo et al., 2015	Usability heuristics for legacy CLI	Statistical analysis, cognitive science strategies	Network security admin tools, legacy CLI	Identified key usability factors; translation to GUI remains challenging

Design Patterns:

- ASCII or structured text visualization approaches:Used in 4 studies.
- Visualization frameworks or design principles:Proposed in 3 studies.
- Usability or accessibility features for the CLI:Focused on in 2 studies.
- Hybrid CLI-GUI interface:Described in 1 study.

Effectiveness:

- Empirical user preference or usage evidence:Found in 3 studies.
- Conceptual or theoretical benefits without empirical validation:Reported in 3 studies.
- Recommendations or design principles proposed, but not empirically validated:Found in 2 studies.
- Identification of barriers or challenges, without empirical validation:Found in 2 studies.
- No empirical evaluation:Noted in 1 study.

Based on available abstracts and full texts, empirical effectiveness data is limited; only three studies provided user preference or usage evidence. Most studies described conceptual benefits, recommendations, or identified barriers without formal validation.

: Isaacs and Gamblin, 2019 : Jacques and Kristensson, 2015 : Sampath et al., 2021 : Baecker and Marcus, 1986 : Hayatpur et al., 2024 : Verma, 2013 : Cabot, ”Re-imagining the Command Line” : Crichton, 2020 : Baecker and Marcus, 1983 : Murillo et al., 2015 : Jacques and Kristensson, 2015, on typography and spacing : Baecker and Marcus, 1986, on visualization principles : Baecker and Marcus, 1983, on typeset representations : Jacques and Kristensson, 2015, on color and overuse : Hayatpur et al., 2024, on ASCII diagrams : Isaacs and Gamblin, 2019, on CLI visualization : Verma, 2013, on hybrid CLI-GUI : Cabot, ”Re-imagining the Command Line” : Crichton, 2020, on information-dense visualization : Cabot, ”Re-imagining the Command Line”, on engagement : Isaacs and Gamblin, 2019, on user preference : Jacques and Kristensson, 2015, on theoretical benefit : Sampath et al., 2021, on accessibility barriers : Baecker and Marcus, 1986, on conceptual enhancement : Hayatpur et al., 2024, on ASCII diagrams in practice : Verma, 2013, on proposed improvement : Crichton, 2020, on design principles : Baecker and Marcus, 1983, on human factors : Murillo et al., 2015, on usability factors

References

Devamardeep Hayatpur, Brian Hempel, Kathy Chen, William Duan, Philip J. Guo, and Haijun Xia. “Taking ASCII Drawings Seriously: How Programmers Diagram Code.” *International Conference on Human Factors in Computing Systems*, 2024.

Harini Sampath, Alice Merrick, and A. Macvean. “Accessibility of Command Line Interfaces.” *International Conference on Human Factors in Computing Systems*, 2021.

Jason T. Jacques, and P. Kristensson. “Understanding the Effects of Code Presentation.” *PLATEAU@SPLASH*, 2015.

Katherine E. Isaacs, and T. Gamblin. “Preserving Command Line Workflow for a Package Management System Using ASCII DAG Visualization.” *IEEE Transactions on Visualization and Computer Graphics*, 2019.

Pramod Verma. “Gracoli: A Graphical Command Line User Interface.” *Conference on Computer Supported Cooperative Work*, 2013.

R. Baecker, and A. Marcus. “Design Principles for the Enhanced Presentation of Computer Program Source Text.” *International Conference on Human Factors in Computing Systems*, 1986.

———. “On Enhancing the Interface to the Source Code of Computer Programs.” *International Conference on Human Factors in Computing Systems*, 1983.

Rachel B. Cabot. “Re-Imagining the Command Line User Experience for Problem Solving,” 2017.

Sandra R. Murillo, Enrique Sánchez-Lara, and Enrique Sánchez. “Enhancing Interfaces for Network Security Administrators with Legacy Attributes.” *Latin American Conference on Human Computer Interaction*, 2015.

Will Crichton. “Documentation Generation as Information Visualization.” *arXiv.org*, 2020.