Writing Tools: Looking Back to Look Ahead

Cerstin Mahlow
School of Applied Linguistics, Zurich University of Applied Sciences
Winterthur, Switzerland
cerstin.mahlow@zhaw.ch

ABSTRACT

Research on writing tools started with the increased availability of computers in the 1970s. After a first phase addressing the needs of programmers and data scientists, research in the late 1980s started to focus on writing-specific needs. Several projects aimed at supporting writers and letting them concentrate on the creative aspects of writing by having the writing tool take care of the mundane aspects using NLP techniques. Due to technical limitations at that time the projects failed and research in this area stopped. However, today's computing power and NLP resources make the ideas from these projects technically feasible; in fact, we see projects explicitly continuing from where abandoned projects stopped, and we see new applications integrating NLP resources without making references to those old projects. To design intelligent writing assistants with the possibilities offered by today's technology, we should reexamine the goals and lessons learned from previous projects to define the important dimensions to be considered.

CCS CONCEPTS

• **Human-centered computing** → **Text input**; **User interface design**; *Graphical user interfaces*; *Hypertext / hypermedia*; • **Applied computing** → **Document preparation**.

KEYWORDS

writing technology, natural language processing, intelligent writing tools, interactive editing

1 WHERE WE COME FROM

The term "word processing" first appeared in the 1960s, referring to a combination of hardware and software [for the history of word processing see, 18, 23]. The first writing tools for microcomputers, such as Electric Pencil or Easy Writer, were developed by hobby programmers; they were soon displaced by commercial products [4]. How people actually used these new tools was not obvious. Rosson [43] and Whiteside et al. [57] therefore asked "How do people really use text editors?" (title of [57]) in the early 1980s to gain insights for the development of future editors. But even ten years later, after the failure of a large project on writing support, Holt and Williams [29] had to admit:

> In seems clear [...] that in order to produce computer based tools to support writers and the writing process we must increase our knowledge of how writers conduct their craft. An increased understanding of writer's requirements and the task involved in writing will form the basis of the next generation of writing tools. [29, p. X]

Projects like RUSKIN [61, 62], Writer's Assistant [47, 48], Intelligent Workstation [31], and Editor's Assistant [13, 14] did not result in marketable products. They aimed to improve (post-)editing and revision based on linguistic principles using linguistic resources. The design and development, however, did not take into account the real needs of users. Natural language processing (NLP) resources were not yet mature enough to be used in real applications. The computing power of PCs at the time was insufficient for real-time analysis—Editor's Assistant relied on constant full syntactic parsing of the growing text—and generation and the resulting tools were thus too limited for practical use.

For RUSKIN, a post-editing support tool, Williams [58] notes that at the time, most writing software was rather ad hoc implementations of ideas with poor user interfaces incorporating checkers that provided inadequate or even erroneous results [58, p. 3]. The project aimed to overcome the poor quality of checkers by incorporating automatic syntax analysis—the probabilistic parser only delivered results with a correctness of around 75%, though [60, p. 117]. Furthermore, the supposed "authors' needs" were based solely on the researchers' intuitions. Only at the end of the project they realized that real authors actually would have preferred support *during* writing, not in a separate post-editing phase, as "the concept of postwriting software implies a linear model of the human writing process which is at best simplistic and at worst may be completely misleading" [62, p. 6]. The first models of writing as a complex and non-linear process had already been published at this point, e.g., by Flower and Hayes [20] and Scardamalia and Bereiter [44]. Writer's Assistant did take into account writing research, but lacked an *implementable* writing model [47, p. 26]. This is still an open issue in writing research today, but only rarely addressed [e.g., 26, 27]. The prototype of Writer's Assistant [48] focused on authors' activities and aimed to be more than just the next word processor: "The Writer's Assistant is a computer-based *cognitive support* system for people who create complex documents as part of their professional live" [47, p. 22, emphasis in original]. Writer's Assistant was supposed to be a combination of word processor, idea processor, and outliner/structure editor to provide authors with different views on different aspects on the text under development [47, p. 22]. But at that time, it could not be implemented and remained a thought model.

Serious research on writing tools stopped in the late 1980s, once corporate customers—including universities—had decided what to purchase [see 10, 25, 42, 49, 55] and Microsoft Word had achieved monopoly status in the consumer market [see 5, 18, 64]. The main reason was that MS Word was bundled with many PCs. From a customer's perspective, everything was ready: one could just turn on the new computer and start writing with MS Word. Purchasing and installing another word processor such as WordPerfect would have required purchasing another license and installing another program. This ubiquity had several effects: writers became accustomed to the appearance, features, and affordances of MS Word;

the format of text files produced with MS Word became the default file format expected and demanded for submissions of academic theses and the like, for interchange between writers when writing collaboratively, and for further processing in publishing houses.

The first effect led to the general assumption that any other new writing facility, e.g. in the first learning management systems that appeared in the early 2000s, should be designed to resemble the look and feel of MS Word and include its main features to provide a familiar user experience. This also applies to the first versions of Google's web-based word processor Google Docs, which became available in a beta version in early 2006. Experimental projects such as the British Telecom-funded Editor's Assistant had no progress in sight in the 1990s to overcome technological obstacles (computing power, quality of NLP) that would justify further investment.

The integration of NLP technology into word processors beyond checkers for spelling and grammar has been a research topic since the 1980s [e.g., 31, 32], but did not result in commercial products either. To overcome the challenges for parsers arising from what Van De Vanter [51] calls "the three I's": *ill-formedness*, *incompleteness*, and *inconsistency* of sentences during writing, experimental word processors attempted to incorporate *syntax orientation* as derived from *syntax-oriented* text editors such as EMILY [24], Cornell [50], PEN [1], JANUS [8], PARSE [9], Mentor [22, 33], PAN [3, 53], or CodeProcessor [52]. These editors handled documents as tree structures and were implementations of programming principles like *stepwise refinement* and *structured programming* [17, 63]. However, similar to programmers, writers objected to always produce complete, well-formed sentences, as this was not compatible with their writing habits. It also does not reflect the writing process as has been observed in various studies: authors often start revising a sentence *before* a complete first version of this sentence is finished [see 34, 41, 56]. Dale [12] predicted in 1997:

> The major developments in the next five to ten years are likely to be of an augmentative nature, with increasingly sophisticated systems that have people and machines doing what they each do best. The key here is to add intelligence and sophistication to provide *language sensitivity*, enabling the software to see a text not just as a sequence of characters, but as words and sentences combined in particular structures for particular semantic and pragmatic effect. [12, p. 235, emphasis in original]

No such systems were available for the general public in the 2000s, though. At that time, Mahlow and Piotrowski [38] proposed language-aware functionality, but only developed a proof of concept as extension to Emacs [39]. In writing research, the influence of the writing tool and medium are only occasionally acknowledged [7, 36, 45]; the field concentrates on cognitive aspects and writing strategies.

2 WHERE WE ARE NOW

Starting in the 2010s, the emphasis on writing experience, personalization of tools, and the growing diversity of input devices (and methods) and displays prompted the development of "new writing tools." Their functionalities are often working implementations of methods and concepts originally described and developed in the 1960s and 1970s that used to be considered failures—but had actually only failed due to the limitations of computers at that time. Only now we see the inverse development, back to ideas and applications of the 1960s, when projects like NLS (oN-Line System) [19] where "pushed aside in favor of computer systems more oriented toward print practices" [54]. NLS already combined functionality to write text, messages similar to what later became known as e-mail, and "computer conferencing" for allowing collaborative simultaneous editing of documents [6]. There was no fixed final document format—e.g., a printed page—the focus was on facilitating online text production by implementing

> text editing capabilities of later word processors, including word wrap, search and replace, and scrolling, and the use of a mouse to select text to be cut and pasted between documents. Indeed Engelbart's system was much more complex than most of subsequent word processing systems [23, p. 21]

One type of functionality that Mahlow and Piotrowski [38] suggest are information functions that use NLP techniques to highlight certain aspects of the evolving text, commonly referred to as "syntax highlighting." Since 2013, iA Writer has offered such a feature in a commercial product to specifically highlight nouns, verbs, adjectives, etc., advertising it as "using parts of speech to improve your writing" and explicitly stating that writers deserve the same professional support as programmers.[1] The use of NLP has been feasible for some time now, both in terms of quality and the computing power required.

Williams [59] stated that professional writers, including academics and journalists, seemed to be satisfied with the tools available in the early 1990s. They had adapted to these tools and did not seem to be aware of other options. In the early 2000s, only writers who had used WordPerfect or other word processors "back in the days" sometimes complained about missing functionality in MS Word.

Today, users are willing to try out new interfaces and new writing experiences. The implementation of applications with appealing user interfaces is easier than ever: current programming languages and toolkits allow for fast development and roll-out of responsive applications. At the same time, the assumption that any writing tool must resemble MS Word is fading, which is also driven by developments in creating, sharing, and accessing documents beyond the paper-based structure [35, 37].

For some time now, new writing applications as stand-alone tools or integrated into other services—e.g., learning management systems or blogging software—are being developed. As for the first wave of writing tools, we also see the adoption of tools originally intended for writing code now for writing all kinds of texts. The shift of academic writing to include dynamic aspects of "text," e.g., code (snippets), data plots, and other visualizations clearly supports the use of these affordances.

3 WHERE WE SHOULD BE GOING

The failed projects from the late 1980s addressed issues that can be considered general considerations for the design and implementation of writing technology:

[1] https://ia.net/writer/support/writing-tips/parts-of-speech

1. user-friendly interfaces, carefully designed functionality instead of ad-hoc hacks [58];
2. support not only for writing, but also for teaching and learning how to write [62], including sophisticated feedback on various levels to stimulate reflection on the writing and decisions for revising and editing [28];
3. help for interpreting system messages and feedback [40];
4. easy extension of features based on user needs [28];
5. real interaction with the system that enables writers to stay in control of edits [11];
6. application of NLP technology users can trust [14];
7. various views on the evolving text (rhetoric, linguistic, typographical, graphemic) to stimulate creativity [31].

Most of these requirements are generic requirements for software development and emphasize the need for input from real users, both for functionality and for the user interface. Strong collaboration between designers/developers of writing tools and writing researchers modeling human writing processes at multiple levels (e.g., the cognitive or the linguistic level) should be established.

We are already seeing experimental applications that use recent technological possibilities to finally approach writing in ways that previous experiments could not realize: One such example is Tilio, which tried to implement the ideas proposed by Sharples [46] by understanding writing as design and incorporating aspects and techniques today known as "design thinking." While this endeavor was halted by the COVID-19 situation in 2020, the technical feasibility was demonstrated in an alpha version, so we may see another attempt later. Similarly, the combination of different features and services in an application like Scrivener for seamless integration of idea creation, management of sources and references, connection to data tracing, and communication channels (chat and messaging) can be seen as a functional implementation of Engelbart [19]'s ideas, even if the developers do not explicitly refer to it.

In the late 1990s, projects like Intelligent Workstation, intended as an instance of the "fifth generation of text-editing programs" [31], and Integrated Language Tools for Writing and Document Handling from KTH Stockholm suffered from insufficient NLP resources. Today's NLP tools make it worth to reconsider the underlying ideas of those projects. They are also of interest for document creation processes, as they already abstract from the print-oriented document, which is in line with current developments: the creation of texts for documents that can be rendered according to need and display device.

The Web generally allows for *dynamic* documents with respect to form and content. Linking of documents as hypertext challenges authors during writing but can be supported using recommender functionality based on artificial intelligence (AI). The understanding of "text" changed at the turn of the century to include "interactive, hypertextual documents—many of which reside on the Internet—[which] use color, sound, images, video, words, and icons to express their messages" [21, p. 282]. This clearly requires tools that allow writers to create and edit such documents; here again, writers could be supported by powerful AI-based components.

Taking into account that communication takes place on various channels with specific and complex formats emphasizes the need for structure within texts. This allows the display of the content/text according to features of devices and tailored to the needs of readers. Writing in these scenarios used to be challenging and required knowledge of specific markup for rendering. Abandoning WYSIWYG and its focus on printed paper documents, together with the development of truly augmented and responsive writing tools based on generative AI could actually free writers to "fully embracing the new opportunities offered by digital media" [2].

Dale and Viethen [15] analyze the "automated writing assistance landscape in 2021." GPT-3 was already available at that time and was integrated into several tools aimed at supporting writers as co-authors. These applications addressed specific genres like blog posts and poetry, and specific writing tasks like expanding, rewriting, and shortening texts [15]. Some months later, they were included as writing aids into experimental editors [e.g., 16, 65]. However, they were not widely used and did not trigger the same discussions that we see now. We also see reimplementations of popular applications with integrated access to LLMs. One such example is Lex, intended as a "Google Docs style editor" [30]. It has access to GPT-3 and GPT-4 so that writers can invoke the language model to produce plausible continuations of the text, taking into account everything before the current cursor position, and to rewrite and summarize paragraphs. However, research from a writing research perspective on how humans and AI-based language models produce text through *co-creation* is still pending at this point.

In contrast to Dale's 1997 prediction of augmented language-sensitive editors within 10 years, Dale's 2021 prediction seems feasible and even close to reality, given the current pace of development in both machine learning-based NLP and writing application implementations:

> But the big shift is the transition from tools that help with editing to tools that help with authoring. It's conceivable that, in 5 years' time, no automated writing assistance tool will be considered complete without a functionality that finishes your sentences, writes you out of tight corners and fills in background paragraphs while you doze at the keyboard. And given Microsoft's exclusive licencing deal with OpenAI in regard to GPT-3, it won't be a surprise if, before too long, we see some of these capabilities as yet another item on Microsoft Word's ribbon menu. [15, p. 518]

Note that this prediction does not include the part of language-sensitive or language-aware functionality supporting authors during production and revision for semantic and pragmatic aspects.

4 CONCLUSION

To design and implement writing tools effectively and efficiently, HCI researchers must work closely with writing researchers to both foster the development of operationalizable writing models and base the implementation of writing tools on the latest insights into writing processes. Many ideas for designing writing tools that actually address the needs of writers can be gleaned from earlier projects by exploring the technical feasibility of the underlying concepts. In this way, the development of writing tools would finally respond to the demands and predictions made in the 1990s by Holt and Williams [29] and Dale [12].

REFERENCES

[1] Todd Allen, Robert Nix, and Alan Perlis. 1981. PEN: a hierarchical document editor. In *Proceedings of the ACM SIGPLAN SIGOA symposium on Text manipulation*. ACM, Portland, Oregon, United States, 74–81. DOI: 10.1145/800209.806457.

[2] Claus Atzenbeck. 2021. Interview with Beat Signer. *SIGWEB Newsletter*, Winter, Article 2, (Feb. 2021), 5 pages. DOI: 10.1145/3447879.3447881.

[3] Robert A. Ballance, Susan L. Graham, and Michael L. Van De Vanter. 1992. The Pan language-based editing system. *ACM Transactions on Software Engineering Methodology*, 1, 1, 95–127. DOI: 10.1145/125489.122804.

[4] Thomas J. Bergin. 2006. The Origins of Word Processing Software for Personal Computers: 1976–1985. *IEEE Annals of the History of Computing*, 28, 4, 32–47. DOI: 10.1109/mahc.2006.76.

[5] Thomas J. Bergin. 2006. The Proliferation and Consolidation of Word Processing Software: 1985–1995. *IEEE Annals of the History of Computing*, 28, 4, 48–63. DOI: 10.1109/mahc.2006.77.

[6] E. David Callender. 1982. An evaluation of the AUGMENT system. In *SIGDOC '82: Proceedings of the 1st annual international conference on Systems documentation*. ACM Press, New York, NY, USA, 29–35. DOI: 10.1145/800065.801306.

[7] David S. Calonne. 2006. Creative Writers and Revision. In *Revision: History, Theory, and Practice*. Reference Guides to Rhetoric and Composition. Alice Horning and Anne Becker, (Eds.) Parlor Press, West Lafayette, IN, USA. Chap. 9, 142–176.

[8] Donald D. Chamberlin, James C. King, Donald R. Slutz, Stephen J. P. Todd, and Bradford W. Wade. 1981. JANUS: an interactive system for document composition. In *Proceedings of the ACM SIGPLAN SIGOA symposium on Text manipulation*. ACM, Portland, Oregon, United States, 82–91. DOI: 10.1145/800209.806458.

[9] Takeshi Chusho, Tan Watanabe, and Toshihiro Hayashi. 1983. A Language-Adaptive Programming Environment Based on a Program Analyzer and a Structure Editor. In *Information Processing 83, Proceedings of the IFIP 9th World Computer Congress, Paris, France, September 19–23, 1983*. R. E. A. Mason, (Ed.), 621–626. http://dblp.uni-trier.de/rec/bibtex/conf/ifip/ChushoWH83.

[10] Colette A. Daiute. 1983. The Computer as Stylus and Audience. *College Composition and Communication*, 34, 2, 134–145. DOI: 10.2307/357400.

[11] Robert Dale. 1990. A Rule-Based Approach to Computer-Assisted Copy-Editing. *Computer Assisted Language Learning*, 2, 1, 59–67.

[12] Robert Dale. 1997. Computer assistance in text creation and editing. In *Survey of the State of the Art in Human Language Technology*. Studies in Natural Language Processing. Giovanni B. Varile, Antonio Zamponelli, Ronald Cole, Joseph Mariani, Hans Uszkoreit, Annie Zaenen, and Victor Zue, (Eds.) Cambridge University Press, Cambridge, New York, Melbourne, 235–237. DOI: 10.5555/278696.278806.

[13] Robert Dale. 1989. Computer-based Editorial Aids. In *Recent Developments and Applications of Natural Language Processing*. Jeremy Peckham, (Ed.) Kogan Page, London. Chap. 2, 8–22.

[14] Robert Dale and Shona Douglas. 1996. Two Investigations into Intelligent Text Processing. In *The New Writing Environment: Writers at Work in a World of Technology*. Mike Sharples and Thea van der Geest, (Eds.) Springer, Berlin, Heidelberg, New York. Chap. 8, 123–145.

[15] Robert Dale and Jette Viethen. 2021. The automated writing assistance landscape in 2021. *Natural Language Engineering*, 27, 4, 511–518. DOI: 10.1017/S1351324921000164.

[16] Hai Dang, Karim Benharrak, Florian Lehmann, and Daniel Buschek. 2022. Beyond text generation: supporting writers with continuous automatic text summaries. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology* (UIST '22) Article 98. Association for Computing Machinery, Bend, OR, USA, 13 pages. DOI: 10.1145/3526113.3545672.

[17] Edsger W. Dijkstra. 1972. Notes on structured programming. In *Structured Programming*. Academic Press, London. Chap. 1, 1–82. http://portal.acm.org/citation.cfm?id=1243380.1243381.

[18] Daniel Eisenberg. 1992. History of Word Processing. *Encyclopedia of Library and Information Science*, 49, 268–278.

[19] Douglas C. Engelbart. 1962. Augmenting Human Intellect: A Conceptual Framework. Tech. rep. Stanford Research Institute, (Oct. 1962). http://sloan.stanford.edu/mousesite/EngelbartPapers/B5_F18_ConceptFrameworkInd.html.

[20] Linda S. Flower and John R. Hayes. 1981. A Cognitive Process Theory of Writing. *College Composition and Communication*, 32, 4, 365–387. DOI: 10.2307/356600.

[21] Cheryl Geisler et al. 2001. IText: Future Directions for Research on the Relationship between Information Technology and Writing. *Journal of Business and Technical Communication*, 15, 3, (July 2001), 269–308. http://jbt.sagepub.com/cgi/content/abstract/15/3/269.

[22] Véronique D. Gouge, Gilles Kahn, Bernard Lang, Bertrand Melese, and Elham Morcos. 1983. Outline of a Tool for Document Manipulation. In *Information Processing 83, Proceedings of the IFIP 9th World Computer Congress, Paris, France, September 19–23, 1983*. R. E. A. Mason, (Ed.), 615–620. http://dblp.uni-trier.de/rec/bibtex/conf/ifip/Donzeau-GougeKLMM83.

[23] Thomas Haigh. 2006. Remembering the Office of the Future: The Origins of Word Processing and Office Automation. *IEEE Annals of the History of Computing*, 28, 4, (Oct. 2006), 6–31. DOI: 10.1109/mahc.2006.70.

[24] Wilfred J. Hansen. 1971. *Creation of hierarchic text with a computer display*. Ph.D. Dissertation. Stanford, CA, USA. http://portal.acm.org/citation.cfm?id=906308.

[25] Gail E. Hawisher. 1988. Research update: Writing and word processing. *Computers and Composition*, 5, 2, (Apr. 1988), 7–27. DOI: 10.1016/8755-4615(88)80002-1.

[26] John R. Hayes. 2001. Commentary on the book: Through the models of writing. In *Through the Models of Writing*. Studies in Writing. Vol. 9. Denis Alamargot and Lucile Chanquoy, (Eds.) Kluwer, Boston, Dordrecht, London, 229–236.

[27] John R. Hayes. 2012. Modeling and Remodeling Writing. *Written Communication*, 29, 3, (July 2012), 369–388. DOI: 10.1177/0741088312451260.

[28] Patrik O'Brian Holt, Dag C. Hegg, and Terje Johnsen. 1990. Engineering Written Style. *Computer Assisted Language Learning*, 2, 1, 27–35.

[29] Patrik O'Brian Holt and Noel Williams, (Eds.) 1992. *Computers and Writing: State of the Art*. Springer Netherlands, Dordrecht.

[30] Justin. 2023. Lex.page – eliminate writers block using an ai writing assistant. (Jan. 18, 2023). https://goldpenguin.org/blog/introducing-lex-page-ai/.

[31] Gerard Kempen, Gert Anbeek, Peter Desain, Leo Konst, and Koenraad De Smedt. 1986. Author environments: Fifth generation text processors. In *ESPRIT'86 Results and Achievements*. The Commission of the European Communities: Directorate General XIII: Telecommunications, Information, Industries & Innovation, (Ed.) North-Holland, Amsterdam, 365–372.

[32] Gerard Kempen and Theo Vosse. 1992. A language-sensitive text editor for Dutch. In *Computers and Writing: State of the Art*. Patrik O'Brian Holt and Noel Williams, (Eds.) Kluwer, Boston, Dordrecht, London, 68–77.

[33] Bernard Lang. 1986. On the usefulness of syntax directed editors. In *Advanced Programming Environments* (Lecture Notes in Computer Science). Vol. 244. Springer, Berlin, Heidelberg, New York, 47–51. DOI: 10.1007/3-540-17189-4_87.

[34] Mariëlle Leijten, Daniel Janssen, and Luuk Van Waes. 2010. Error correction strategies of professional speech recognition users: Three profiles. *Computers in Human Behavior*, 26, 5, (Sept. 2010), 964–975. DOI: 10.1016/j.chb.2010.02.010.

[35] Cerstin Mahlow. 2022. Structure! You get more than you see. In *XML Prague 2022 – Conference Proceedings*. XML Prague 2022 (Prague, June 9–11, 2022). Jirka Kosek, (Ed.), 125–140. https://archive.xmlprague.cz/2022/files/xmlprague-2022-proceedings.pdf#page=135.

[36] Cerstin Mahlow and Robert Dale. 2014. Production Media: Writing as Using Tools in Media Convergent Environments. In *Handbook of Writing and Text Production*. Handbooks of Applied Linguistics. Vol. 10. Eva-Maria Jakobs and Daniel Perrin, (Eds.) De Gruyter Mouton, Berlin, Germany, (Feb. 2014), 209–230.

[37] Cerstin Mahlow and Michael Piotrowski. 2022. Academic writing and publishing beyond documents. In *Proceedings of the 22nd ACM Symposium on Document Engineering* (DocEng '22) Article 12. Association for Computing Machinery, San Jose, California, 4 pages. DOI: 10.1145/3558100.3563840.

[38] Cerstin Mahlow and Michael Piotrowski. 2008. Linguistic Support for Revising and Editing. In *Computational Linguistics and Intelligent Text Processing: 9th International Conference, CICLing 2008, Haifa, Israel, February 17–23, 2008. Proceedings* (Lecture Notes in Computer Science). Alexander Gelbukh, (Ed.) Vol. 4919. Springer, Berlin, Heidelberg, New York, 631–642. DOI: 10.1007/978-3-540-78135-6_54.

[39] Cerstin Mahlow and Michael Piotrowski. 2009. LingURed: Language-Aware Editing Functions Based on NLP Resources. In *Proceedings of the International Multiconference on Computer Science and Information Technology*. Vol. 4. Polish Information Processing Society. Mragowo, Poland, (Oct. 2009), 243–250. http://www.proceedings2009.imcsit.org/pliks/101.pdf.

[40] Steve McGowan. 1992. Ruskin to McRuskin – Degrees of interaction. In *Computers and Writing: State of the Art*. Patrik O'Brian Holt and Noel Williams, (Eds.) Kluwer, Boston, Dordrecht, London, 297–318.

[41] Thierry Olive, Rui A. Alves, and São L. Castro. 2009. Cognitive processes in writing during pause and execution periods. *European Journal of Cognitive Psychology*, 21, 5, 758–785. DOI: 10.1080/09541440802079850.

[42] Donald Ross. 1991. Prospects for Writer's Workstations in the Coming Decade. In *Evolving Perspectives on Computers and Composition Studies*. Gail E. Hawisher and Cynthia L. Selfe, (Eds.) National Council of Teachers of English, Urbana, IL, USA, 84–110.

[43] Mary B. Rosson. 1983. Patterns of experience in text editing. In *CHI '83: Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, Boston, Massachusetts, United States, 171–175. DOI: 10.1145/800045.801604.

[44] Marlene Scardamalia and Carl Bereiter. 1983. The development of evaluative, diagnostic and remedial capabilities in children's composing. In *The psychology of written language. Developmental and educational perspectives*. Margaret Martlew, (Ed.) Wiley and Sons, New York, NY, USA, 67–95.

[45] Mike Sharples. 1996. An Account of Writing as Creative Design. In *The Science of Writing. Theories, Methods, Individual Differences, and Applications*. C. Michael Levy and Sarah Ransdell, (Eds.) Lawrence Erlbaum, Hillsdale, NJ, USA, 127–148.

[46] Mike Sharples. 1999. *How We Write: Writing As Creative Design*. Routledge, New York, NY, USA.

[47] Mike Sharples, James Goodlet, and Lyn Pemberton. 1989. Developing a Writer's Assistant. In *Computers and Writing: Models and tools*. Noel Williams and Patrik O'Brian Holt, (Eds.) Intellect, Oxford. Chap. 3, 22–37.

[48] Mike Sharples and Lyn Pemberton. 1990. Starting from the Writer: Guidelines for the Design of User-Centred Document Processors. *Computer Assisted Language Learning*, 2, 1, 37–57.

[49] Lee R. Taylor. 1987. Software Views: A Fistful of Word-Processing Programs. *Computers and Composition*, 5, 1, 79–90. http://computersandcomposition.osu.edu/archives/v5/5_1_html/5_1_8_Taylor.html.

[50] Tim Teitelbaum and Thomas Reps. 1981. The Cornell Program Synthesizer: A syntax-directed programming environment. *Communications of the ACM*, 24, 9, (Sept. 1981), 563–573. DOI: 10.1145/358746.358755.

[51] Michael L. Van De Vanter. 1995. Practical language-based editing for software engineers. In *Software Engineering and Human-Computer Interaction*. Lecture Notes in Computer Science. Vol. 896. Springer, Berlin, Heidelberg, New York, 251–267. DOI: 10.1007/bfb0035821.

[52] Michael L. Van De Vanter and Marat Boshernitsan. 2000. Displaying and Editing Source Code in Software Engineering Environments. In *Second International Symposium on Constructing Software Engineering Tools (CoSET'2000)*. Limerick, Ireland, (June 2000).

[53] Michael L. Van De Vanter, Susan L. Graham, and Robert A. Ballance. 1992. Coherent user interfaces for language-based editing systems. *International Journal of Man-Machine Studies*, 37, 4, (Oct. 1992), 431–466. DOI: 10.1016/0020-7373(92)90004-5.

[54] Derek Van Ittersum. 2008. Computing Attachments: Engelbart's Controversial Writing Technology. *Computers and Composition*, 25, 2, 143–164. DOI: 10.1016/j.compcom.2007.12.001.

[55] Alex Vernon. 2000. Computerized grammar checkers 2000: Capabilities, limitations, and pedagogical possibilities. *Computers and Composition*, 17, 3, (Dec. 2000), 329–349. DOI: 10.1016/s8755-4615(00)00038-4.

[56] Åsa Wengelin, Mark Torrance, Kenneth Holmqvist, Sol Simpson, David Galbraith, Victoria Johansson, and Roger Johansson. 2009. Combined eyetracking and keystroke-logging methods for studying cognitive processes in text production. *Behavior Research Methods*, 41, 2, (May 2009), 337–351. DOI: 10.3758/brm.41.2.337.

[57] John Whiteside, Norman Archer, Dennis Wixon, and Michael Good. 1982. How do people really use text editors? *ACM SIGOA Newsletter*, 3, 1-2, 29–40. DOI: 10.1145/966873.806474.

[58] Noel Williams. 1990. Editorial. *Computer Assisted Language Learning*, 2, 1, 1–4.

[59] Noel Williams. 1992. New technologies. New writing. New problems? In *Computers and Writing: State of the Art*. Patrik O'Brian Holt and Noel Williams, (Eds.) Kluwer, Boston, Dordrecht, London. Chap. 1, 1–19.

[60] Noel Williams. 1993. Postwriting software and the classroom. In *Computers and Language*. Moira Monteith, (Ed.) Intellect, Oxford, 114–123.

[61] Noel Williams. 1990. Writers' Problems and Computer Solutions. *Computer Assisted Language Learning*, 2, 1, 5–25.

[62] Noel Williams and Patrik O'Brian Holt, (Eds.) 1989. *Computers And Writing: Models And Tools*. Intellect, Oxford, (Jan. 1989). http://www.worldcat.org/isbn/187151603X.

[63] Niklaus Wirth. 1971. Program development by stepwise refinement. *Communications of the ACM*, 14, 4, (Apr. 1971), 221–227. DOI: 10.1145/362575.362577.

[64] Amy D. Wohl. 2006. How we process words: the marketing of WP software. *IEEE Annals of the History of Computing*, 28, 4, (Oct. 2006), 88–91. DOI: 10.1109/mahc.2006.66.

[65] Ann Yuan, Andy Coenen, Emily Reif, and Daphne Ippolito. 2022. Wordcraft: story writing with large language models. In *27th International Conference on Intelligent User Interfaces* (IUI '22). Association for Computing Machinery, Helsinki, Finland, 841–852. DOI: 10.1145/3490099.3511105.