WRITING AND DIGITAL MEDIA

EDITED BY

LUUK VAN WAES
University of Antwerp, Belgium

MARIËLLE LEIJTEN
University of Antwerp, Belgium

CHRISTINE M. NEUWirth
Carnegie Mellon University, Pittsburgh, USA

ELSEVIER
Amsterdam • Boston • Heidelberg • London • New York • Oxford
Paris • San Diego • San Francisco • Singapore • Sydney • Tokyo
Logging Writing Processes with Inputlog

Luuk Van Waes and Mariëlle Leijten

In this subchapter we describe a new logging program, called Inputlog. Most logging tools are either developed for a specific computing environment, or not adequately adapted to the current Windows environment. As such, they cannot be used for writing studies in which 'natural' writing and computer environments employ commercial word processors. This was the main incentive to develop a new logging tool, i.e., Inputlog. This program consists of three modules: a data collection module that registers on-line writing processes on a very detailed level; a data analysis module that offers text and pause analysis; and a play module that enables researchers to review the writing session. The functional description of Inputlog is complemented by two examples of research studies in which the program was used. We wind up the chapter with a preview of the plans for further developments.

1 Introduction

The basic concept of most logging tools that have been developed is more or less comparable (Sullivan & Lindgren, in this volume; or for a review, see Sullivan & Lindgren, in preparation). All keystrokes and mouse movements during a writing process are logged and stored for later processing. This continuous data storage does not interfere with the normal usage of the computer, creating an ecologically valid research context. At a later stage, the logged data can be made available for further analysis, either within the program environment itself or as exported data in statistical programs such as SPSS or SAS. Depending on the research question, researchers can choose to analyze different aspects of the writing process and the writing behavior by combining, for instance, temporal data (e.g., time stamps, or pauses) with process data (keystrokes or mouse movements). The data collection (and processing) can be performed much faster and more accurately by means of the computer, than it could ever be done manually.

At the moment most logging tools are either developed for a specific writing environment, or not adequately adapted to the current Windows environment. As such, they cannot be used for writing studies in which 'natural' writing and computer environments employ commercial word processors (e.g., MSWord or WordPerfect). This was the main reason to develop a new logging tool, i.e., Inputlog. In this chapter we present a short description of the functional characteristics of Inputlog. Next, two research studies are presented to illustrate the use of the program. We conclude the chapter with a preview of the further developments of the program.

2 Characteristics of Inputlog

For the development of Inputlog we were able to fall back on the functionality of two existing programs: JEdit and Trace-it on the one hand (Severinson Eklundh, 1994, 1996; Severinson Eklundh & Kollberg, 1996, 2003; Kollberg, 1998), and ScriptLog on the other hand (Strömqvist & Malmsten, 1997). JEdit and Trace-it are suitable for Macintosh personal computers. JEdit only logs data in an in-house developed limited word processor. ScriptLog also mainly logs in a limited word processor that was developed for research purposes (i.e. mainly writing experiments with young children). Trace-it features an extensive revision module, while ScriptLog combines logging data with recorded eye-tracking data (Andersson et al., this volume).

As mentioned above, most logging-tools cannot be used for writing studies in which professional writing in an ethnological business environment is studied. Professional writers almost always employ commercial word processors like MSWord, mostly in a Windows environment. Existing logging tools up till now are mainly developed to operate in experimental settings, which was problematic for our writing research in which we studied professional writers in their own environment. Another impetus for the development of Inputlog has been a study on the influence of speech recognition on the writing process. For this study we observed the writing processes of 20 participants who used speech recognition software during their day-to-day work in their professional business contexts (five observations of approximately 30 min per participant; see Leijten & Van Waes, 2005b). Because it was not possible to register keyboard input in combination with speech mode data with any of the existing logging tools, the process data had to be analyzed manually.

To collect the writing process data, or should we say speech recognition data, we combined two digital observation instruments: a digital screen cam (i.e. Camtasia; see Degenhardt in this volume) and a digital sound recorder (i.e. Quickrec). The study showed that the chosen observation instruments and analyzing methods did enable us to analyze and describe the specific speech recognition writing processes, but the data analyses were very time-consuming.

Inputlog was developed to give in to these objections. It consists of three modules that should assist researchers in the collection and analyses of online writing processes in a Windows environment: a record, a generate and a play module. In short, the logging process is organized as follows. The first step is the activation of the recorder itself. To identify a particular writing session that is going to be logged, the user defines certain characteristics using a set of (predefined) variables and then starts the recorder. By default MS Word will be started to create a writing environment that is familiar to most writers. From then on, the writing process will be logged as it evolves on the screen. At the end of the registration the output data are saved in a source file, a so-called IDF-file. The data of this file can be used either to generate analyses files, or as an input for the playback module. The analyses files are the result of several conversions of the data in the source file, and enable researchers to analyze the modus data, the pause data, and the text data on a more aggregated level. Finally, the playback module enables researchers to replay the writing session exactly as it was registered, or speed up to the researcher’s preference. Furthermore, the playback module is also used as an intermediate facilitator for the revision analysis (under development).
The three main steps in the logging procedure are now described in more detail. Depending on the researchers' objectives, certain options in the modules will be more appropriate than others.

**Step 1: Record the data of a writing session in MSWord.** Before a new session is started the researcher can first specify the file information and the identification data for a specific session. This information will be included in all the analysis files that will be generated based on the session source file, facilitating the identification of each writing session. The file that will be generated in the recording session has the extension *.idf, which is added automatically. This file will be used as an input for the generate and play functions (cf. infra).

Each logging session can be identified by a maximum of 10 variables (six predefined and four user-defined variables). These variables should enable the researcher to identify a writing session in detail.

**Step 2: Generate data files for statistical, text, pause and mode analyses.** In the generate module of the program, analysis files can be generated on the basis of a source file that was recorded in a previous logging session. In other words, any IDF file can be opened at any time to generate data output files for specific analyses. Inputlog 1.4 offers four different analyses:

1. **General logging file:** a spreadsheet with a basic log file of the writing session in which every line represents an input action (letter, function, mouse click, or movement); for every input action the session information is stored together with an identification of the input, the timestamp, the pausing time that followed it, and — for a mouse operation — the xy-value of the screen position.

2. **Statistical analysis:** a spreadsheet with basic statistical information on the writing session such as the session information, some basic data about the written text (product and process), pausing behavior and the use of the different writing modes.

3. **Text analysis:** a plain linear text in HTML format with the complete linear production of the text including mouse movements and other activities; extra options allow for the production of a linear output in which the writing activities are divided into periods (time periods of x seconds) or intervals (number of intervals in which the writing process is to be divided).

4. **Pause analysis:** a spreadsheet with analyses of every non-scribal period; the threshold for the pauses can be set to 1, 2, or 5 s as a standard or to any user-defined level.

Two other analyses are under construction for the next release:

5. **Mode analysis:** a spreadsheet with information about the distribution of the writing modes (keyboard, mouse, speech technology) that were used as an input device during the writing session.

6. **Revision analysis:** a spreadsheet with a basic analysis of the number, the level and the kind of revision that has taken place during the writing session.

**Step 3: Play the recorded session at different speeds.** A recorded writing session can be replayed using Inputlog. Again, the IDF file is used as a source file for the replay. To verify the information labels of the file that is selected for a play back session, all defined variables of the session identification appear in the dialog box on the left side of the screen. The writing session can be replayed at different speeds.
3 Usage

To illustrate the possible usage of Inputlog, we briefly discuss two fragments of writing processes taken from experiments in which we have used the program as a logging device. The first experiment concerned a more technical writing experiment that focused on the working memory requirements necessary for error detection in the ‘text produced so far’ (Leijten, Ransdell, & Van Waes, 2006); the second example is taken from a case analysis of a writer producing a bad news letter.

3.1 The Text Produced So Far and the Use of Working Memory

An experimental setting was set up to assess the memory load during the ‘text produced so far’ interaction in the text production phase. The design included the most frequently occurring error types found in a case study of professional writers that were using speech recognition for the first time to write business texts (Leijten & Van Waes, 2005b). In the example at hand, we selected two text fragments in which a sentence with large speech recognition errors is corrected. The errors that occurred in the text produced so far were considered as major because the number of characters that differed from the intended text was more than two. Besides, the selected errors could only occur in speech recognition.

The task in the experiment consisted of a set of sentences that was presented to the participants to provide a context. In Figure 1, an example of a correct and an incorrect sentence is given (see sentence 2). After every sentence (1) the participants had to click the ‘ok’ button, to indicate that they had finished reading the sentence. A sub clause of the previous sentence was then presented as text produced so far (TPSF) in a subordinate causal structure (2), and the participants were prompted to complete the sentence (3) on the basis of the context provided earlier (1).

The participants could either prefer to correct the errors first and then complete the sentence, or to complete the sentence first and then correct the error. In case, for this sentence 83% of the participants preferred to complete the sentence first and correct the error afterwards. Figure 2 shows the linear output of the writing session of two participants that used another writing strategy. Pauses longer than 1 s are included in the output; texts are in Dutch.

<table>
<thead>
<tr>
<th>Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Because the height was not indicated, the pick-up truck drove by the underpass.</td>
</tr>
<tr>
<td>2. The pick-up truck drove by the underpass,</td>
</tr>
<tr>
<td>3. because the height was not indicated.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Incorrect</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Because the height was not indicated, the pick-up truck drove by the underpass.</td>
</tr>
<tr>
<td>2. The <strong>picot trug</strong> drove by the <strong>underparts</strong>.</td>
</tr>
<tr>
<td>3. because the height was not indicated.</td>
</tr>
</tbody>
</table>

Dutch “De heeft week reed onder de brief door, want de hoogte was niet aangegeven.”

Figure 1: Examples of correct and incorrect sentences in the TPSF experiment.
<table>
<thead>
<tr>
<th>Sentence completion</th>
<th>Error correction (83%)</th>
<th>Error correction</th>
<th>Sentence completion (17%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[MwC:910,701-391,348] [Mcl:Left]</td>
<td>[2.66][Mcl:Left][MwC:910,711-149,347]</td>
<td>[Mcl:Left] [MwC:149,347-375,355]</td>
<td>[MwC:910,711-149,347]</td>
</tr>
<tr>
<td>de hoogte was niet aangegeven [2.19]</td>
<td>[Mcl:Left] [MwC:149,347-375,355]</td>
<td>[BS 9]fterunk (1.6)</td>
<td>[MwC:372,357-269,346] [Mselect]</td>
</tr>
<tr>
<td>[MwC:491,352-145,546] [Mcl:Left]</td>
<td>[Mcl:Left] [MwC:149,347-375,355]</td>
<td>[MwC:270,340-342,341]</td>
<td>[Mcl:Left]</td>
</tr>
<tr>
<td>[BS 9]</td>
<td>[BS 4]</td>
<td>[BS 9]fterunk (1.6)</td>
<td>[Mcl:Left]</td>
</tr>
<tr>
<td>efruck (1.27)</td>
<td>ug &lt;1.17&gt;</td>
<td>ug &lt;1.81&gt; [RIGHT 15]</td>
<td>ug &lt;1.81&gt; [RIGHT 15]</td>
</tr>
<tr>
<td>[RIGHT 17] [DEL 4]</td>
<td>de hoogte stond niet aangegeven &lt;1.25&gt;</td>
<td>de hoogte stond niet aangegeven &lt;1.25&gt;</td>
<td>de hoogte stond niet aangegeven &lt;1.25&gt;</td>
</tr>
<tr>
<td>ug &lt;1.17&gt;</td>
<td>[MwC:165,335-333,699] [Mcl:Left]</td>
<td>[MwC:348,338-384,695] [Mcl:Left]</td>
<td>[MwC:348,338-384,695] [Mcl:Left]</td>
</tr>
</tbody>
</table>

Figure 2: Example of linear output of a text fragment generated by Inputlog. Remark: MwC, mouse movement without click; Mcl:Left, mouse click left; Mselect, selection of text by mouse; BS, backspace; right 17, arrow right 17.

In these examples of two short writing fragments, different writing strategies can be distinguished. The first writer prefers to continue completing the text first, before correcting the mistake in the TPSF. He positions his cursor after the first segment (TPSF; cf. _xy_-value of left mouse click) — without a significant previous pause — and then completes the sentence. After the sentence is completed he pauses for 2.19 s. He then positions the mouse after the incorrect word and deletes it by using the backspace key. Next, he navigates through the text by pressing the right arrows key and deletes the second error. Finally, he has a short pause before he continues to the next sentence. The logging of the writing session of the second participant reveals a different pattern. He pauses before he starts writing and then positions the cursor behind the error to correct this first. After correcting the second error he completes the sentence.

In the analysis of the research data presented here, we were mainly interested in a description of the interaction with the text produced so far. Differences in writing strategies can be related to both individual differences and to error characteristics in the TPSF. This short example illustrates how the detailed process information that is generated by Inputlog provides a basis for analyzing writing strategies that are no longer visible in the final written product.

### 3.2 Pausing and Revision Behavior in Writing Bad News Letters

In Figure 3, we show the linear text representation of a writer producing a bad news letter in which he declines an offer to deliver a keynote address for an international conference. We were especially interested in the process characteristics of the writing episode that concerned the wording of the bad news itself. As we know from the literature on this issue, the strategic considerations to make the bad news as acceptable as possible for the reader are crucial in the perception of the message (and may also determine the future interpersonal relation with the reader).

The writer needed about 10 min to finish the letter of about 130 words. The replay function of Inputlog was used as a stimulus for a retrospective thinking aloud protocol. Figure 3 shows the sequential linear output (periods of 30 s) of a fragment that illustrates the writer's strategic considerations at the beginning of the second paragraph.
Figure 3: Linear output of two fragments of a writing process in which a bad news letter was produced.

The fragment shows a very fragmented and staccato writing process that starts off with a long pause of more than 8 sec in the beginning of the second paragraph (announcement of the refusal). In the production of about 30 words, there were 28 pauses longer than 1 sec, which is substantially more than in the previous period when the introductory context of the letter was written. About half of the time in this fragment is used for pausing, showing the time attributed to careful and strategic formulation. An interesting example of such a strategic consideration is the revision that takes place after 4 min (240 sec segment). The participant starts the sentence with ‘However’, but two words later he rereads the beginning of the sentence and realizes that this contrastive connective announces the bad news in a too early stage of the paragraph. Therefore, he decides to delete the connective to neutralize the context of this argumentative sentence.

Again, this example shows that process logging enables researchers to analyze writing processes from different perspectives enriching possible interpretation based on text analysis. Other observation methods, like recording (retrospective) thinking aloud protocols, might complement the acquired data.

4 Further Development

To facilitate a broad usage of Inputlog, the program is put at the disposal of the research community for free, provided that reference is made to Leijten and Van Waes (2005b). The users' feedback is very important for the evaluation and further development of Inputlog.

We have identified four important niches that may increase the applicability of Inputlog, especially in the domain of writing process research. In the near future we would like to further develop the following (in order of priority):

1. **A module for logging speech recognition events.** To facilitate the integration of speech recognition input, Scansoft added a new API to their latest professional version of Dragon Naturally Speaking 8.1, which enables us to integrate the dictated text with the data logged by Inputlog (for more information, see Leijten & Van Waes, in press). The implementation of speech recognition will stimulate research on the effect of this new technology on the writing process (of both professional writers and writers with learning disabilities).
(2) A module for revision analysis. In the revision analysis we would like to produce an output analysis in which different characteristics of in-process revisions are described, e.g., the number of revisions, type of revisions, level of revisions, number of words and characters involved in the revision operation, as well as the location of the revisions in relation to the point of utterance. To define revisions we have developed an algorithm and a set of rules. At the moment we have predefined about 50 sets of rules to test the algorithm for deletions and substitutions. However, after the testing phase, the rules will have to be extended and further tested to cover a more complete range of revisions.

(3) A module for progression analyses (basic and extended). So far, the development of the text in Inputlog is represented in the linear text analysis. To visualize this textual development we would like to extend Inputlog with two graphical representations of the text progression, a basic and a more extended one. In the basic progression analysis we would like to visually represent the number of characters that are produced at each moment during the writing process taking into account the characters that are deleted at that stage. This basic progression analysis is based on writing strategies researched by Perrin (2003; this volume); the extended representation is inspired by Lindgren, Sullivan, Lindgren, and Spelman Miller (2006). They use a Geographical Information System (GIS) to visualize and summarize the writing process. GIS enables researchers to analyze different subprocesses of the writing process by selecting representative variables. The graphical representations are not static, but they allow a researcher to interact with the data at different levels and to move back and forward between the data and their representation.

(4) A module for integration with Morae. Morae® is a macro-oriented observation tool developed by Techsmith (www.techsmith.com). It is our intention to complement the data of Inputlog with logging data of Morae, which is mostly used for usability research. This program captures, for instance, changes between programs on a higher level and registers, for instance, the URL-addresses of websites that are accessed during a writing session. Just like Inputlog it also logs very detailed timestamps, which should enable us to integrate the additional data registered by Morae into the output of Inputlog. For the observation of writing processes during which the participants combine MSWord with other programs especially, this integration opens new perspectives for further analyses.

In addition to these developments, we will pay special attention to the further development and optimization of the existing modules. Furthermore, the compatibility with different versions of the Windows operating systems and the Office environment will require constant attention. Finally, the integration of our logging data with data from eye-tracking observations is also on the agenda (see also Andersson, this volume).

5 Conclusion

In this paper we have briefly described the main characteristics and the functionality of Inputlog. The program differs from other keystroke logging programs in that it is not limited in its usage to a self-designed word processor. It is primarily developed to log writing processes in MSWord (Windows environment).
Inputlog offers three main functions — record, generate, and play — enabling the researchers to collect very detailed data about a writing process and to prepare some basic analyses for further study. The replay function allows for a review of the writing process and it can also be used as a stimulus for a retrospective thinking aloud protocol. For more detailed information about the use of the program we refer to the help file and the detailed description on the program’s website www.inputlog.net.

To increase the applicability of Inputlog we would like to further develop the program by adding new modules. Four new components are planned: a module to log speech recognition events, a module for revision analysis, a module for progression analyses, and a module to integrate macro-level data recorded by Morae. We hope to report about these further developments soon.

Acknowledgment

We would like to thank Wesley Cabus, Ahmed Essahi, and Bart Van de Velde for their excellent work in programming Inputlog.