Translog-II: a Program for Recording User Activity Data for Empirical Reading and Writing Research

Michael Carl
Copenhagen Business School,
Dalgas Have 15, 2000 Frederiksberg, Denmark
E-mail: mc.isv@cbs.dk

Abstract

This paper presents a novel implementation of Translog-II. Translog-II is a Windows-oriented program to record and study reading and writing processes on a computer. In our research, it is an instrument to acquire objective, digital data of human translation processes. As their predecessors, Translog 2000 and Translog 2006, also Translog-II consists of two main components: *Translog-II Supervisor* and *Translog-II User*, which are used to create a project file, to run a text production experiments (a user reads, writes or translates a text) and to replay the session. Translog produces a log files which contains all user activity data of the reading, writing, or translation session, and which can be evaluated by external tools. While there is a large body of translation process research based on Translog, this paper gives an overview of the Translog-II functions and its data visualization options.

Keywords: Translation Process Research, Keyboard logging, Eyetracking

1.Introduction

Human translation process research analyses the translation behaviour of translators such as properties of reading and text production rhythms, mental memory and search strategies, types of textual units that translators focus on, etc. It investigates the temporal and contextual structure of those activities and describes inter- and intra personal variation in terms of translation competence and translation performance. In order to acquire objective data about human translation processes, the program Translog has been designed. Translog can be used to study translation processes, hence the name Translog, but it can be equally used for other kinds of computer-based reading or writing. Since it's first conception in 1995, Translog has gone through several re-implementations. Right from its beginnings, Translog had two main components, originally called *Writelog* and *Translog*, (Schou et al 2009) the former component was designed for recording writing processes in time, while the latter component served for playback. These components are now referred to as the *Translog-User* and the *Translog-Supervisor* which are two interdependent programs. A major extension was introduced in the context of the EU project Eye-to-IT in 2006 when a new version *Translog-2006* could connect to eye-tracker through the GWM module (Sparkov, 2008) so as to record both, keyboard and gaze behaviour in time. Translog-2006 was a complete re-implementation in C#, supporting Unicode and XML. However, the communication with the eye tracker through GWM[1] turned out to be too inflexible and so a further development of Translog-II now communicates directly with the eyetracker. This paper describes the purpose and usage of the Translog-II software.

Similar programs such as ScriptLog (http://www.scriptlog.net/demo.asp), and InpuLog (http://www.inputlog.net/download.html) are mainly intended for logging and analyzing writing processes, while Translog is specially designed for the acquisition of data for translation process research, and is widely used in the translation process research community. Schou et al (2009) count more than 80 publications making use of Translog, for translation process research of linguistic phenomena, (e.g. the translation of metaphors, cognates, idioms, etc.) as well as translator behaviour and cognitive processes (e.g. translator's awareness, memory constraints, (self)revision etc.), translation expertise, translation under time pressure, and machine translation post-editing. Translog is also used for translator training, teaching and learning purposes.

Translog-II records user activity data (UAD), that is, all the keystrokes and gaze movements (if an eye-tracker is connected). It classifies the keystroke data as 1) insertion, 2) deletion (delete and backspace), 3) navigation (cursor movements), 4) copy/cut-and-paste, 5) return key or 6) mouse operations. Since the keylogger runs in the background, the recording does not interfere with the writing or translation process. Translog-II logs the exact time at which each keystroke operation is made. If connected to an eye-tracker[2], Translog-II also records 7) gaze-sample points, 8) computes fixations (i.e. clusters of gaze-samples) and 9) mappings of fixations to the closest character on the screen. This latter operation performs a mapping from the spacial location of the gaze on the screen to a character offset in the text. That is, an X/Y coordinate of a fixation center is mapped onto a character position of the text that is being looked at. Since there is some noise in the recordings of gaze-sample points, the representation in the log file is such that fixations and to a certain extent also mappings can be re-computed offline. The gaze and the keystroke information can then be

1 which were Borland C++ implementations communicating with Translog through COM `

2 Currently connection to Tobii eye tracker is supported, but other interfaces are planned.

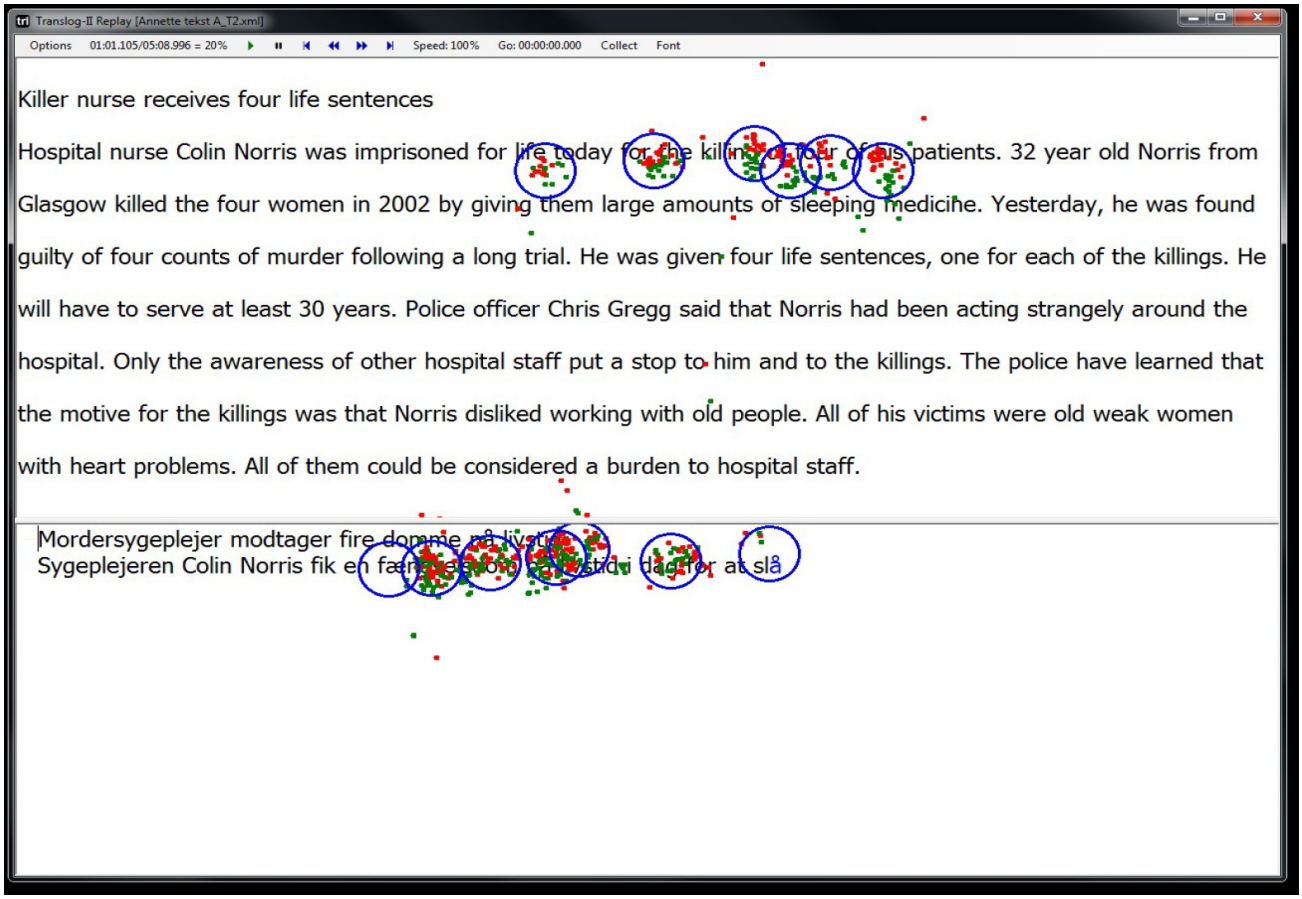

Figure 1 The screen shot of a Translog-II Supervisor replay session shows a fragment of a translation experiment with the source text (top) and the beginning of a translation (bottom) in the target text window. It also shows the gaze movement during the translation of the past 8 words. Red and green dots are gaze-sample points (sampling rate 60Hz.) for the right and left eye respectively, and the blue circles represent fixations. Much more gaze activity is takes place on the lower target window during translation.

correlated, as they both refer to textual positions. The information is stored in an XML format and can be replayed or analyzed with Translog-II or analyzed in external tool. In Carl and Müller (2011) and Carl and Jakobsen (2009) is given more information on the XML representation. Here we describe the functions of the data acquisition software *Translog-II*.

2. Functions of Translog-II

Translog-II has three main functions:

1. create a project file:
   - determine the size and orientation of a source and a target window on the screen for reading and writing permission respectively.
   - produce texts for the source and/or the target window, their layout, text font, size, color, line spacing etc.
   - determine which data are to be logged, keyboard and eye-tracking

2. run and record a Translog-II session:
   - load a project file
   - calibrate eye-tracker (if connected)
   - record and log UAD

3. replay and analyze a recorded log file:
   - statistics: figures about text production/ elimination/ navigation events
   - user view: replays the translation session in time (Figure 1)
   - linear view: plots a textual representation of the UAD (Figure 2)
   - pause plot: shows a 2D representation how the text emerge in time (Figure 3)

The Translog-II Supervisor program implements the functions 1. (create a project file) and 3. (replay a log file), Translog-II User is only used to record a Translog session and to store the UAD in a log file. A Translog-II project file can be configured for a reading experiment, where only the "source window" will be visible during the recording session, it can be configured for a writing experiment, where only the "target window" is visible in which a text can be typed, or for a translation experiment, in which both windows are visible (as in figure 1). In fact Translog-II also allows for post-editing texts, if a pre-defined text is entered in the target window. Translog-II allows the source and the target windows to be horizontally or vertically oriented and the source or target windows to be left or right, or bottom or top.

As in previous Translog versions, texts can be displayed in smaller portions, e.g. one sentence at a time. Each portion can be displayed for a certain number of pre-defined

Figure 2 Two linear view screen shots of the same text with different temporal resolution. Top: each dot represents 1 second pause. Bottom: a dot represents 0.1 seconds between successive keyboard activities.

seconds, or the writer may decide to go on to the next portion of source text when ready to do so.

3. Translog User

The Translog-II User program is an interface for displaying and typing text and for logging UAD. To start a translation session, a project file must be loaded. According to the settings in a project file, the eye-tracker needs be calibrated, then Translog-II User opens a source and/or a target text window, plots the pre-defined texts in the source window, and waits for the translator to type a translation into the target window. As the size, orientation and rendering of the windows and the font is defined in the project file, it is not possible to re-size the windows in Translog-II User, to change the font. It is possible to use Translog-II User as a post-editing, by providing (machine) translation in the target window and to record text modifications during post-editing.

4. Translog-II replay mode

The most interesting feature in Translog is the replay mode. Translog-II Supervisor computes some statistical figures on the number of keystrokes, but more interesting are certainly the possibilities to replay the log file. As mentioned above, there are three different ways to visualize the UAD, the user view, the linear view, and the pause plot which are respectively presented in figures 1, 2 and 3. The user view (a screen shot is shown in Figure 1) replays the typing process in real-time, and radio buttons can be used to accelerated or decelerated, to pause the replay, rewind or forward it etc. In addition to the keystrokes, Translog-II also plots the gaze-sample points, fixations, and fixated words. In Figure 1, gaze sample points and fixations were collected over a period of approx. 30 seconds illustrating the gaze path and the coordination of reading and writing activities. It is possible to select or un-select whether gaze and fixation information should be plotted.

The linear view represents the UAD in a textual (linear) manner. Each key and mouse activity[3] has a representation in the linear view, and pauses are either indicated as

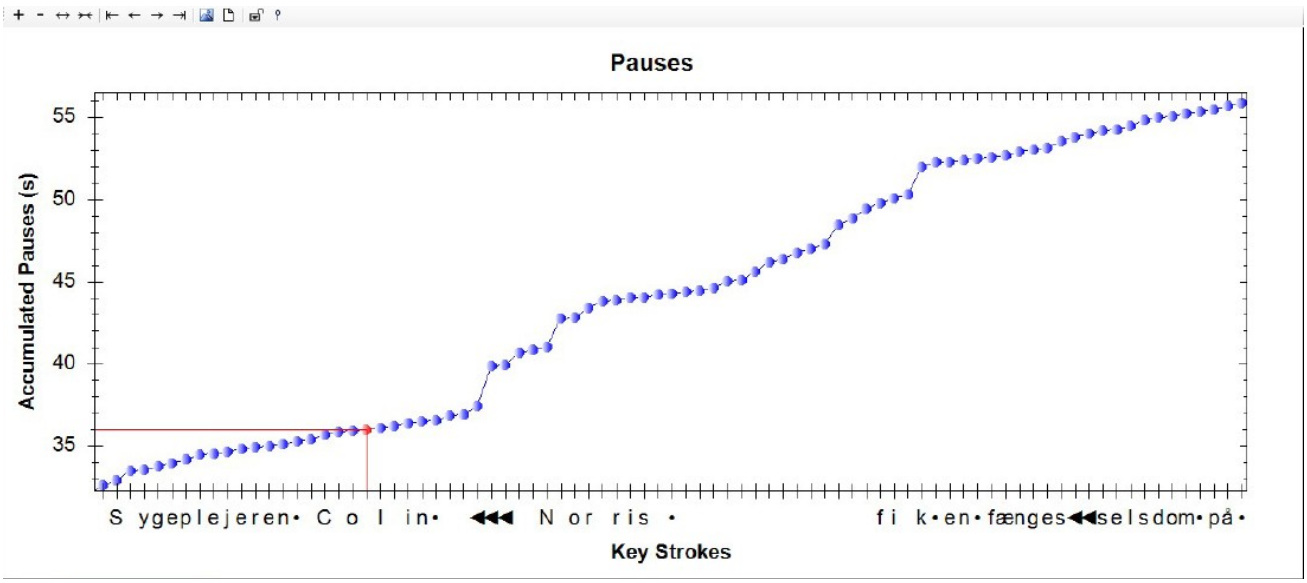

Figure 3. Screen shot of the pause plot: Blue dots indicate the accumulation of pauses during a translation session (in seconds).

asterisks, and/or numeric value indicating the duration between successive activities. The granularity of the pause display can be selected starting from 1ms up to any amount of time. This gives the possibility to get an overview over the coarse temporal structure of a translation session, reducing the temporal information to a minimum (Figure 2, top), or to zoom into a sequence to study pausing behaviour as small as a few hundreds of seconds (Figure 2, bottom).

The third Translog-II replay mode is the pause plot. A pause plot represents essentially the same information as the linear view does, this time in as 2D graph. Keyboard activities are indicated on the horizontal X-axis, while the vertical Y-axis shows the accumulation of time (pauses). Figure 3 shows a segment of a translation session.

It is possible to scroll through the pause plot, to zoom in or out. Translog-II also allows to synchronize all three visualization methods. That is, all three windows (user and linear view as well as pause plot) can be opened at the same time, and by clicking the synchronization item the cursor in all three windows will be positioned at the same time. The option in the user view would then trigger a synchronous replay in the three windows.

life today for the killing". Each keystroke is mapped onto the source text segment of the translation to which it contributes. Thus, line 11 shows all gaze and keyboard activities that relate to the production of the translation for English "was imprisoned", line 13 for that of "for", line 14 of "life" etc. The graph also shows gaze activities in relation to source segments. The blue dots are fixations on the words in the source text, while the green dots represent fixations in the target window, most of the time on the word(s) that are currently being typed.

Translation progression graphs require additional alignment knowledge of the source and the target texts, and are therefore not supported inside Translog-II. A set of additional tools are used:

1. to align the translation,
2. to compute the keystroke-to-source text mapping
3. to visualize the graph.

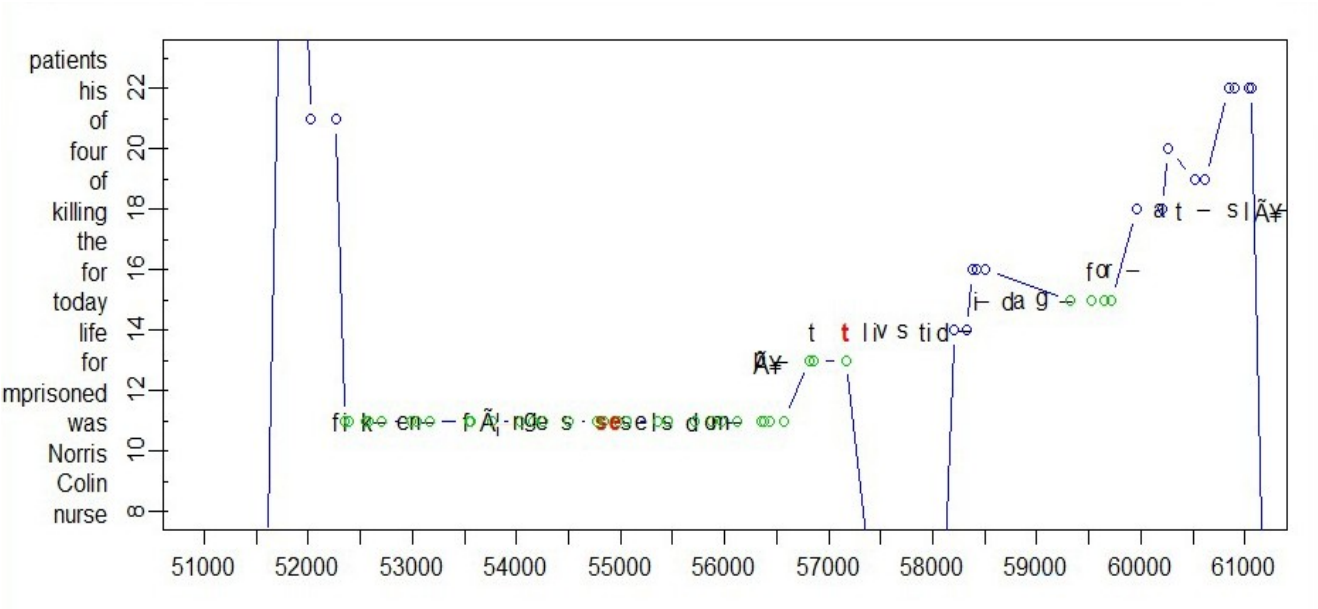

Figure 4. The graph visualizes the translation progression of the 8 English words: "was imprisoned for life today for the killings" into Danish: "fik en fængselsdom på livstid i dag for at slå". This translation segment corresponds to the accumulated gaze movements in Figure 1, which lasted approximately 10 seconds. Blue dots represent fixations on the source text, green dots fixations on the target text, the black characters are insertions and the red characters are text deletions.

5. Translation Progression Graphs and Product Data Alignment

While the visualization options in Translog-II (Figures 1-3) trace how the *target text* emerges in time, we have also developed more powerful visualization possibilities that show how the *translation* (ie. the relation between the source and the target text) evolves. Figure 4 plots the relation between the word positions in the source text (vertical axis) and the translation activity in time on the horizontal axis. The figure presents a time segment of ca. 10 seconds (from secs. 51 to 61) in the string "fik en fængselsdom på livstid i dag for at slå" is produced, which is (part of) the Danish translation of "was imprisoned for

Carl and Jakobsen (2009) describe a general method and rule-based formalism to map keystrokes on source text words. In their method, they iteratively retrieve all keystrokes which contribute to the creation of a target text segment Ti. Given alignment information for the target text segment Ti to a source segment Sj, the retrieved keystrokes can be mapped via Ti to source segment Sj. With an exhaustive fragmentation of the target text into *n* non-overlapping text segments T*1...n*, and a complete alignment of the source and the target texts, every keystroke can be associated with a source segment Sj. As illustrated in Figure 5, we have re-implemented this algorithm in a more efficient way: From the alignment of the translation product (top in Figure 5) we know which source words are linked to which cursor positions in the target text. The text processing operations can then be mapped on the target text positions and from there further on the source word(s). Figure 5 illustrates how the keystrokes which produced the correction of "trial

Product Data

Source sentence word number	1	2	3	4
Source sentence	Das	ist	ein	Testsatz
Alignment	|	|		/ \
Translation	This	is	a	test sentence
Target text cursor position	1–4	6–7	9	11–23

Process Data

time	operation	char	pos	len	Text at time Ti			
T_{f-0}	translation			23	This	is	a	test sentence
T_{f-1}	insert	t	14	22	This	is	a	tes sentence
T_{f-2}	insert	s	13	21	This	is	a	te sentence
T_{f-3}	insert	e	12	20	This	is	a	t sentence
T_{f-4}	delete	r	12	21	This	is	a	tr sentence
T_{f-5}	delete	i	13	22	This	is	a	tri sentence
T_{f-6}	delete	a	14	23	This	is	a	tria sentence
T_{f-7}	delete	l	15	24	This	is	a	trial sentence

Figure 5. The mapping algorithm traverses the process data against the time line, from the final translation product towards the start of the translation session. The arrow on the left indicates the time flow in the process data. While traversing the process data, all operations are associated with a target text position, and hence a source sentence number.

sentence" into "test sentence" in the target text can be mapped on the source text word 5 "Testsatz".

From the alignment of the product data, we know that "test sentence" is the translation of "Testsatz", and that "test sentence" occupies cursor positions 11 to 23 in the final translation. The idea is to look backwards into the process data and collect all keyboard activities between positions 11 and 23, which are then identified to contribute to the production of the translation for "Testsatz".
Figure 5 shows that the four last letters "rial" of the word "trial" were deleted and then substituted by "est". As all deletion and insertion activities take place between cursor positions 11 and 23, these keystrokes are part of the operations that contribute to the translation of "Testsatz". To compute these mappings, the algorithm looks backwards into the activity data and decides for each operation which word it produces. The last operation in Figure 5 is the insertion of the letter "t" at time Tf-1 which took place at cursor position 14. Accordingly, the text length was one character shorter before that insertion took place and the translation of "Testsatz" consisted only of the characters 11 to 22. Insertion operations lead to a shortening of the text while deletions extend the text, as is the case for the operations Tf-4 to Tf-7. The algorithm keeps track of the length and position of each word during all times to correctly map the keyboard operations on the words. The collected keystrokes can then be linked to the source text words and plottet, together with the gaze data, as shown in figure 5.

A free version of Translog-II and auxiliary tools can be downloaded for academic use upon request to the author of this paper. A base containing approximately 200 sessions of translation process data is being constructed and released soon.

6. Acknowledgements

The implementation of Translog-II would not have been possible without previous work on the various Translog versions in the development of which were involved Lasse Schou, Arnt Lykke Jakobsen, Morten Lemvigh and Jakob Elming.

7. References

Michael Carl and Arnt Lykke Jakobsen, 2009. Towards Statistical Modeling of Translators' Activity Data. In *International Journal of Speech Technology,Volume 12, Number 4, 125-138*

Michael Carl and Henrik Høeg Müller. 2011. CRITT NLP Resources for Translation Representation of User Activity Data in Translog-II, in Proceedings of LTC, Poznan

Lykke Jakobsen, A. 1999. Logging target text production with Translog. In Hansen, G. (ed.), *Probing the process in translation: methods and results*, *Copenhagen Studies in Language*, volume 24. Copenhagen: Samfundslitteratur. Pages 9–20

Lasse Schou, Barbara Dragsted & Michael Carl (2009), Ten years of Translog. Copenhagen Studies in Language (37), pages 37-51

Špakov, O. (2007). GWM – the Gaze-to-Word Mapping Tool, available online at: http://www.cs.uta.fi/~oleg/gwm.html (abgerufen am 21. Juni 2011).