# Tatool: A Java-based open-source programming framework for psychological studies

Claudia C. von Bastian · André Locher · Michael Ruflin

Published online: 22 June 2012

**Abstract** Tatool (*T*raining *a*nd *T*esting T*ool*) was developed to assist researchers with programming training software, experiments, and questionnaires. Tatool is Java-based, and thus is a platform-independent and object-oriented framework. The architecture was designed to meet the requirements of experimental designs and provides a large number of predefined functions that are useful in psychological studies. Tatool comprises features crucial for training studies (e.g., configurable training schedules, adaptive training algorithms, and individual training statistics) and allows for running studies online via Java Web Start. The accompanying "Tatool Online" platform provides the possibility to manage studies and participants' data easily with a Web-based interface. Tatool is published open source under the GNU Lesser General Public License, and is available at www.tatool.ch.

**Keywords** Cognitive training · Experimental software · Java · Open-source · Online experiments

Various software packages for conducting behavioral experiments exist—for example, E-Prime (Schneider, Eschman, & Zuccolotto, 2002), Inquisit (Draine, 1989), and SuperLab (Haxby, Parasuraman, Lalonde, & Abboud, 1993). However, the majority of this experimental software is proprietary, and thus is often expensive and not extensible to the demands encountered in the wide range of computer-based psychological research. There are also a growing number of freely available or open-source experimental software packages (e.g., Psychophysics Toolbox for MATLAB, Brainard, 1997; Pelli, 1997; PyEPL, Geller, Schleifer, Sederberg, Jacobs, & Kahana, 2007; Functional Measurement Builder Suite, Mairesse, Hofmans, & Theuns, 2008; TScope, Stevens, Lammertyn, Verbruggen, & Vandierendonck, 2006; and PsyToolkit, Stoet, 2010), which are more cost-effective and often more flexible. However, most of these software packages generate platform-dependent code that is suitable solely for specific operating systems (e.g., Microsoft Windows or Mac OS). Moreover, some software packages can only be used for offline research in the lab. Tatool is an open-source programming framework based on Java, and hence is extensible and platform-independent. Tatool was developed specifically with a focus on facilitating cognitive training research, but it can be used for setting up experiments and questionnaires as well, both online and offline. With Tatool, one compiled version of the experimental tasks can be distributed online to all participants. This is particularly useful for cognitive training studies, in which it is often required that training sessions run self-administered from participants' home computers with different operating systems. Moreover, Tatool provides multiple features that are crucial for conducting training studies (e.g., session schedulers, adaptive training algorithms, and a Web-based interface to monitor participants' commitment), which will be described in more detail below.

Tatool is published open source under the GNU Lesser General Public License. By editing the source code of Tatool, the software can be extended to virtually every need that may be encountered during the technical development

C. C. von Bastian (✉)
Department of Psychology, University of Zurich,
Binzmühlestrasse 14/22,
8050 Zurich, Switzerland
e-mail: c.vonbastian@psychologie.uzh.ch

A. Locher
Bristol, UK

M. Ruflin
Sydney, New South Wales, Australia

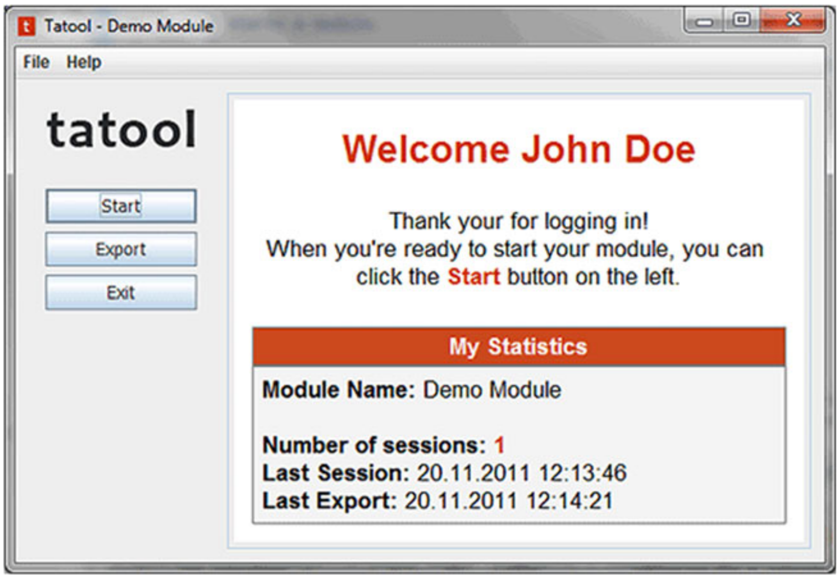
of psychological tasks or studies. Because Tatool is open source, this also implies that fellow researchers can—and are invited to—contribute their own ideas to the Tatool main release in order to optimize the framework for different fields of research. The code of training or experimental tasks can be published under the GNU General Public License, so researchers can easily exchange tasks or code snippets without compromising their authorship of a specific task.

Tatool is based on the programming language Java, which is an object-oriented language with a common syntax. Furthermore, a Java-based programming framework entails three key benefits. First, Java is a portable language. Therefore, Tatool applications can be written on one operating system (e.g., Windows) and run on every other operating system, such as Mac OSX or Linux, without the need to change the code. Hence, this tool is truly platform independent. Second, the Java Runtime Environment (JRE) necessary for executing Java applications can be downloaded for free and is easily installed. Tatool requires Java version 1.6 or later. To date, many applications in digital everyday life use Java technology, which means that most participants will probably have the JRE already installed. Third, the Java Web Start technology included in the JRE allows for the online distribution of the Tatool application client. With a few mouse clicks, the client is installed on a local computer. After installation, the application runs on the local computer and does not require an Internet connection anymore, avoiding typical issues of Internet testing (e.g., random noise produced by the Internet connection). Hence, it allows for running experiments or training studies online without sacrificing the advantages of an offline application (see also Schmiedek, Bauer, Lövden, Brose, & Lindenberger, 2010). Java Web Start can also be used to facilitate the distribution of software updates. When starting the local application, Java Web Start can optionally search for changes to the online code and update the local version accordingly.

The Tatool framework can be used with different levels of complexity, depending on the user's programming skills. For experimenters or software developers with at least moderate prior knowledge of Java or object-oriented programming, the Tatool framework provides an extensive application programming interface (API). It comprises a large number of Java classes (in simplified terms, bundles of functions) and methods that are useful in the context of psychological studies in general, and in particular for computer-based training studies. The full API documentation, including a comprehensive list of Tatool's classes and methods, is accessible online via www.tatool.ch/javadoc. Once programmed in Java, study components (e.g., instructions, questionnaires, and tasks) are arranged in module files written in XML ("Extensible Markup Language"), which is a text format that is readable by both humans and machines. XML documents follow a tree structure, consisting of a single root element that branches into one or more child elements. The grammar of the XML document is constrained by a specific set of rules, called the *schema language*. Tatool utilizes the XML schema provided by the open-source Spring framework (www.springsource.org). Due to its easily comprehensible format, XML can be viewed as a translator between the experimenter and Tatool. In the module file, the parameters of an experiment, such as the display duration of stimuli or the number of trials, can be edited without having to change the Java code (see the How to Get Started section below). Therefore, it offers the opportunity for the properties of already-implemented tasks to be customized by researchers or students without or with only few programming skills. Module files can be opened and started with the Tatool application client from online and from local sources.

The Tatool application client comes with a ready-to-use graphical user interface and is multilingual (currently available in English and German) and capable of multiuser management (i.e., several individuals can use the same Tatool installation by creating different users). Custom information, such as user statistics, can be displayed in the main window, facilitating the individualization of the client application (Fig. 1). The display during module execution (e.g., an experimental task) is organized in three regions (north, center, and south, according to the Java BorderLayout) and provides two modular panels that can be displayed in these regions, the "status panel" (top or "north" panel in Fig. 2) and the "action panel" (bottom or "south" panel in Fig. 2). The status panel can display information on the current level of difficulty, the number of trials already completed, feedback on the correctness of the last trial, and a visual timer display. The action panel enables the user to interact with Tatool, for example through keypresses, mouse clicks on buttons, or text input fields. The panels can be set to be visible or hidden, for example to give feedback during practice trials, but not during test trials. The module execution display can either run in windowed or full-screen mode.

Data produced during the execution of a module are stored in a local database coming with Tatool, from which they can be exported as comma-separated value (CSV) files. These files are composed of one row per trial and several columns for the variables of interest, which can be virtually any stimulus properties or (measurable) variables. The CSV file can be opened with a spreadsheet program such as Microsoft Excel, in which data can be aggregated with a few mouse clicks via pivot tables. The CSV file can also be viewed with any text processor or imported into statistic programs such as SPSS. To facilitate data aggregation, the column headers in the CSV file are named according to



Welcome John Doe

Thank your for logging in!
When you're ready to start your module, you can click the Start button on the left.

My Statistics

**Module Name:** Demo Module

**Number of sessions:** 1
**Last Session:** 20.11.2011 12:13:46
**Last Export:** 20.11.2011 12:14:21

**Fig. 1** Main window of the Tatool application client

definitions made in the XML module file and are, therefore, meaningful to the experimenter.

Data can be exported either locally, on the client computer, or online, to an FTP server or a Web server running Tatool Online, which allows for managing studies via a Web interface. This component can be installed on any Web server that supports PHP and MySQL. An example installation can be viewed at www.tatool.ch. Experimenters can create studies, add groups, and assign participants to these groups. Different experimenter profile settings would facilitate complex study designs, such as double-blinding (i.e., when one or more experimenters must not be informed about the identity and group membership of participants). Furthermore, Tatool Online provides helpful features for studies that require participants to do the experimental tasks at home, as is the case in a growing number of training studies. The data can be uploaded automatically, reducing data loss due to possible technical problems. Importantly, data are also stored locally in an embedded database (HSQLDB). Hence, in the case of technical problems (e.g., a loss of Internet connection), data can be uploaded later or exported locally. Tatool Online processes uploaded data in real time and lists all participants and their sessions, including the date and time of the last upload, session duration, and custom information (e.g., performance and reaction time). This enables experimenters to gain a quick overview of the participants' commitment and to intervene if necessary. Moreover, Tatool Online allows for monitoring the overall progress of a study by providing basic descriptive statistics and simple graphs on the aggregated level, such as

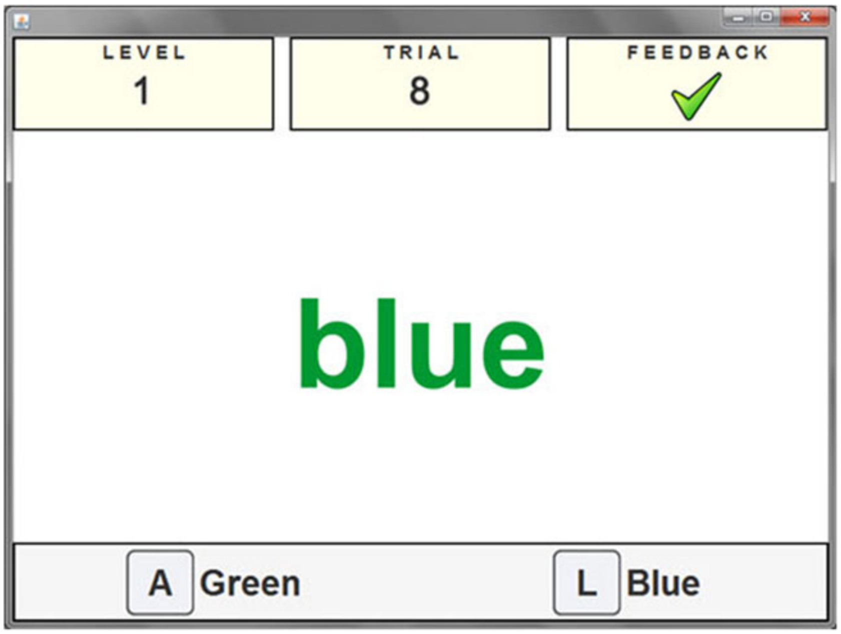


**Fig. 2** Display during the execution of a module

performance means and minimum and maximum values. The experimenter can either access the data via Tatool Online or download it from the FTP server hosting the Tatool Online installation.

## Architecture

Tatool's architecture matches the design of typical psychological experiments. The core concept of Tatool's architecture is the *module*, which represents the experiment or study (e.g., a cognitive training regimen, an experimental design, or a series of questionnaires). A module comprises multiple elements that can be of a number of types: list elements, compound elements, or executable elements. The list and compound elements serve to group executable elements, and can be nested within each other. List elements set the order in which elements listed in it are executed. Compound elements contain a primary and a secondary element and can be used as a basis for the implementation of dual tasks (e.g., a Brown–Peterson task [Brown, 1958] or a complex span task [Daneman & Carpenter, 1980]). First, the primary element is executed (e.g., the display of a list of words to memorize). During the following execution of the secondary element (e.g., a distraction task), the primary element is suspended until the secondary element is completed. Afterward, the primary element is executed again (e.g., the display of another list of words or of a prompt to recall the words memorized before). Only the third type of element, the executable element, can contain an executable. The executable is the implementation (i.e., the Java class) of the actual task that the participants have to do, such as the memorization and recall of a list of words, in the example above. Every executable element can only contain precisely one executable. Therefore, modules including more than one task will comprise several executable elements nested within list and/or compound elements.

All three element types can (but do not have to) contain handlers, which are reusable functions that can act during different phases of the execution process and across different executables. For example, a handler triggered after the completion of a compound element could sum up the words recalled correctly in this task and compare the sum to the performance in another task. Figure 3a illustrates the hierarchy of a module that consists of an executable element nested within a list element.

The module hierarchy is defined in an XML file. It can be opened with the Tatool application, which then saves the file's contents in a local Java database (HyperSQL) used within Tatool. With every execution of a module, Tatool creates a session and saves the data produced during this session in the same database. Each execution of an executable can be recorded and stored as a trial within the given session. Figure 3b shows the different elements of a module as they are reflected in the data view (i.e., data output).

Tatool subdivides the runtime of a module into different phases (see Fig. 4), thereby allowing elements (e.g., handlers or executables) to be triggered at given times during the execution. In simplified terms, there are three types of phases: before, during, or after the execution of the actual task. As soon as a module file is opened with the Tatool client application, the "execution_start" phase begins. After the user presses the "Start" button (cf. Fig. 1), Tatool enters the "session_start" phase. The phase immediately before the execution of an executable is the "pre_process" phase. After the actual execution of an executable ("execute_executable"), each phase in the first half of the module is ended by a counterpart phase: immediately after the execution of an executable ("post_process"), when a session ends ("session_end"), and when the module itself is finished ("execution_end"). For example, as a module file is opened ("execution_start"), the display of an individual user statistic in the main window of the application could be refreshed. When the participant starts a session ("session_start"), it might be useful to reset the stimulus set and to save the start time of the session. The "pre_process" phase can be used for cleaning up the screen before the next stimulus is shown by "execute_executable." Afterward ("post_process"), a handler could count and save the number of correct answers. At the "session_end," the session end time could be saved in order to calculate the overall duration of the session, and when the module is finished ("execution_end"), the main window could be refreshed again to display the updated user statistics.

## Using tatool

So far, Tatool has been used for programming a wide range of tasks in the form of executables—for example, dual tasks (complex span or Brown–Peterson), visual matching tasks (e.g., face matching), task switching, visual search, a trivia quiz, and several questionnaires. Moreover, we implemented several handlers for different purposes, two examples of which we will present here.

### Adaptive training algorithm

Some of the tasks listed above were used as training tasks, which means that the difficulty of the respective task was automatically increased according to the participants' individual performance, for example by increasing memory load in a dual task or reducing the display duration in switching and matching tasks. For this purpose, we applied variations of an adaptive training algorithm (Fig. 5). The open-source release includes three adaptive training algorithms to choose from. The

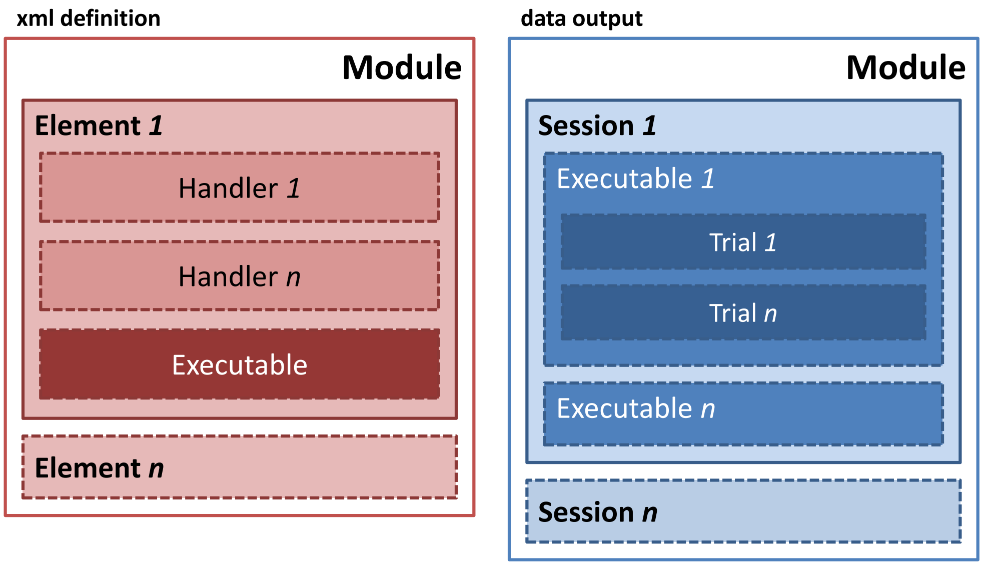


**Fig. 3** Hierarchy of Tatool's architecture. In the XML view (**a**), a module comprises multiple elements (list, compound, or executable elements), which can contain one or more handlers and one or more nested elements, such as executable elements. Only executable elements can contain executables. In the data view (**b**), each execution of a module corresponds to a session, and each execution of an executable corresponds to a trial

simplest one is the "default points and level handler." If a participant's performance is above a defined threshold (*maxThreshold) after a specified number of trials (sampleSize), task difficulty is increased. If performance is below a certain threshold (minThreshold), task difficulty is decreased. The experimenter can set the number of trials and the thresholds for changing the difficulty in the module file via XML (see the How to Get Started section).*

The adaptive training algorithm is implemented as a handler, which acts across all phases during module execution and can be used by any executable. At the beginning of a session, it loads the level achieved in the last session. Before the task is executed (i.e., in the "pre_process" phase), it displays the current level of difficulty in the status panel (in the top of Fig. 2). After execution of the executable, the algorithm calculates the participant's performance and, depending on the number of trials completed, checks whether the level of task difficulty has to be increased or decreased. At the end of the session, the algorithm saves the level of difficulty, thereby allowing participants to start the next session on the same level.

## EEG trigger handler

In another experiment, we recorded task-related brain activity with electroencephalography (EEG). To allow communication between the Tatool application that ran the cognitive test battery and the EEG recording device, we needed a parallel port interface. Therefore, we used the Java native interface,

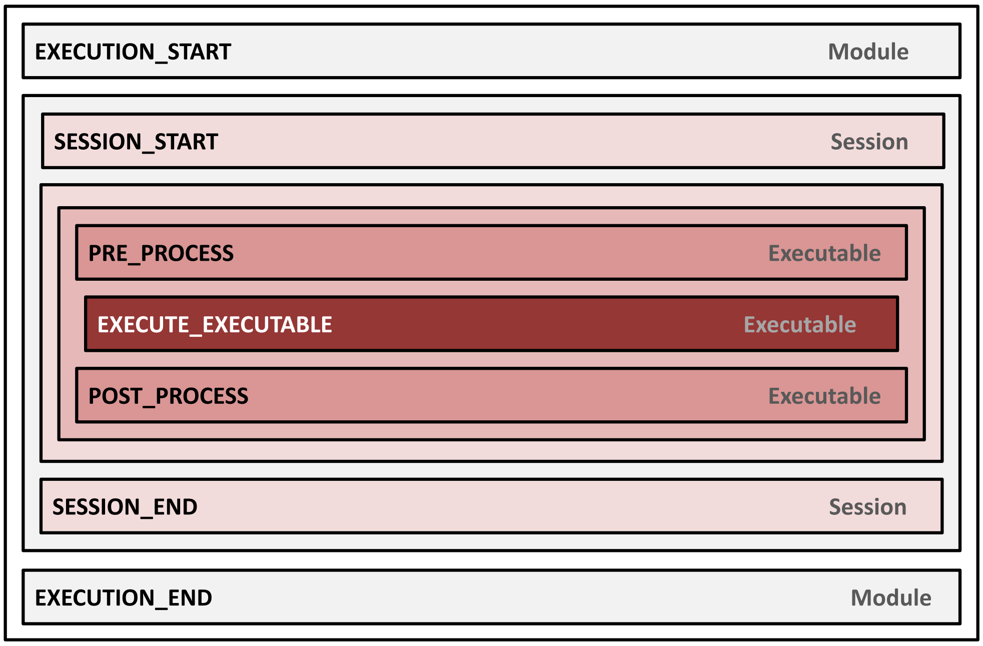


**Fig. 4** Phases during the run time of a module

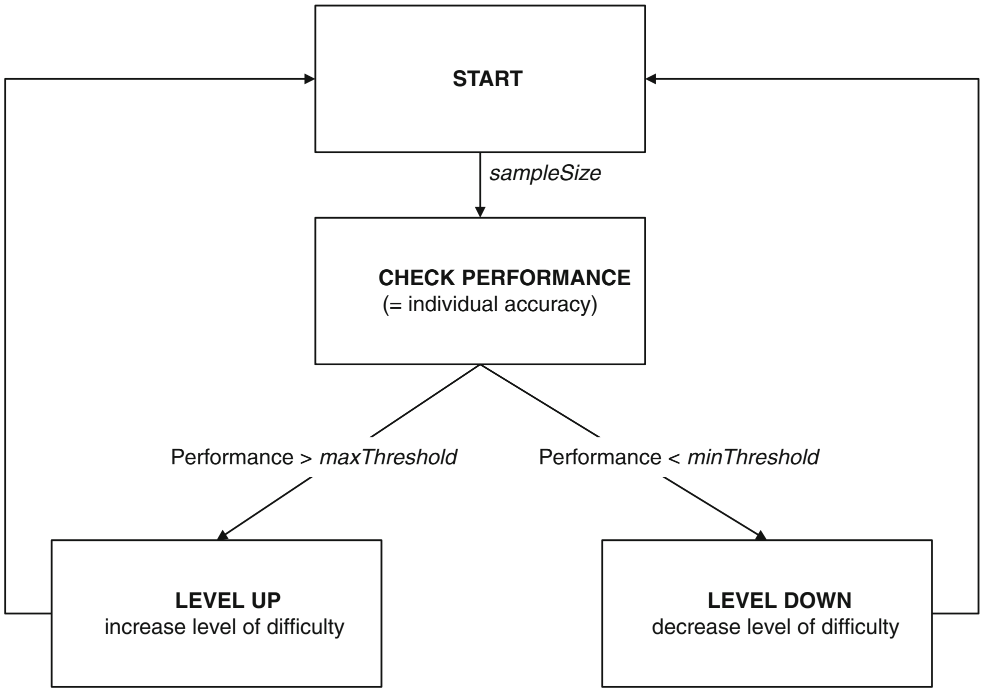

START
sampleSize
CHECK PERFORMANCE
(= individual accuracy)
Performance > maxThreshold
Performance < minThreshold
LEVEL UP
increase level of difficulty
LEVEL DOWN
decrease level of difficulty

**Fig. 5** Algorithm handler that adjust the level of task difficulty to individual changes in performance. Terms in *italics* are parameters that can be set in the module file

which utilizes C code to interact with the parallel port and any hardware attached to it. In our case, Tatool sent signals to the pins of the parallel port that were, in turn, read by the EEG device that set the markers needed for the analysis of the EEG recordings. To be able to set markers at stimulus onset and when the participants responded to this stimulus, the handler acted during the execution of the executable (i.e., in the "execute_executable" phase).

## How to get started

Tatool can be used with different levels of complexity that come with different levels of flexibility. At the top level of complexity and flexibility, users can extend Tatool to meet virtually every possible requirement and can contribute these extensions to the Tatool main release. This requires a deep knowledge of Java, the use of subversion, and XML. Most prospective users will probably use Tatool to program specific experimental or training tasks and to set up studies, which requires basic knowledge of Java and XML. In this case, the level of complexity is moderate, but the level of flexibility is still very high, because the Tatool framework comprises predefined functions that meet many of the needs of computer-based experimental psychology. To program tasks with the Tatool framework, any Java integrated development environment (IDE) can be used—for example, the open-source Eclipse IDE (www.eclipse.org). In the Tatool documentation available at www.tatool.ch, we provide instructions on how to set up the development environment and step-by-step tutorials on how to program tasks within the Tatool framework and how to set up the XML module files to run a study. Tatool can be run with or without Tatool Online. Little knowledge of MySQL database queries is required to install Tatool Online on a Web server.

At its lowest level of complexity and flexibility, users can easily modify the parameters of already-programmed experiments or training implementations (e.g., number of trials, display durations of the stimuli, or performance thresholds for adaptive training algorithms) within the module file, written in XML. In the following example, we show how to modify single parameters in a working experimental task that is included in the Tatool demo client available at www.tatool.ch/demo.htm. In this Stroop-like task (Stroop, 1935), the words GREEN and BLUE are randomly presented on the screen in the color green or blue, and the correct color has to be identified as quickly and accurately as possible by a keypress. To modify the experimental parameters of this task, the module file for the demo client has to be downloaded from www.tatool.ch/download.htm. The text file, with a .xml extension, can be opened with any text editor. Next, the Tatool demo client (www.tatool.ch/demo.htm) has to be downloaded, following the instructions. Instead of (or in addition to) adding the module file from Tatool Online, the option "Module from local file" should be selected, in order to open the previously downloaded module file from its location on the hard drive.

Table 1 gives an overview of the experimental parameters that can be modified in the example module file and their possible ranges. The parameters can easily be found within the module file by using the built-in search function of the text editor used (e.g., in Windows programs, pressing the keys CTRL+f). After changing the value of a parameter, the file has to be saved. Importantly, after saving the module file, it has to be reloaded in Tatool via the option "Module from local file," so that the modifications will be applied.

## Limitations and future directions

Tatool offers a multitude of features and high flexibility. Importantly, Tatool is a programming framework, and not a graphical experiment builder. Therefore, Tatool's high flexibility entails that developing tasks with Tatool is code-heavy, and prospective users thus will need basic knowledge of Java and XML. Several well-written tutorials on Java are available in the form of books or on the Web for free. On our webpage, we also offer step-by-step tutorials for novices using the Tatool framework. An accompanying discussion board on our webpage allows users to exchange code snippets and experimental tasks and to get help from the community concerning programming issues. Once an application is programmed in Java, experimental parameters can easily be modified via module files written in XML. To further facilitate the modification of experimental parameters, in the near future we plan to implement a graphical user interface for setting up module files more quickly.

The Tatool programming framework was developed mainly to facilitate cognitive training research, but it can be used for any computer-based experiment. For specific paradigms, however, specialized software packages may be more suitable. For example, the Functional Measurement Experiment Builder Suite (Mairesse et al., 2008), for conducting experiments using the functional measurement paradigm, or WebExp (Keller, Subahshini, Mayo, & Corley, 2009), for conducting simple Web experiments, are both Java-based applications, and therefore are also platform independent, but they require a lot less prior programming knowledge.

Another possible limitation of Tatool concerns timing. Generally, the timing implementation in Java does not offer the same accuracy offered by programming languages that generate native machine code or external timing hardware. The accuracy of timing in Tatool relies heavily on the timer precision provided by the underlying operating system. The nanoTime method, available since Java 1.5, uses the highest resolution clock available on the platform, and while its return value is in nanoseconds, the update resolution is typically only in microseconds. On modern hardware and operating systems, the Java methods can deliver accuracy and precision in the microsecond range. Java's timing performance can be improved by using the Java native interface, which calls functions provided by a low-level programming language such as C (cf. the implementation of the EEG trigger handler above).

To date, Tatool is available in German and in English, but translations into other languages might be required. Tatool is internationalized following the coding practice of the i18n approach. This means that strings (e.g., names of buttons) are stored externally in separate resource files that can be translated with minor effort.

## Conclusion

Tatool is a Java-based open-source programming framework that comprises a large library of functions useful for running psychological experiments. It is particularly useful for cognitive training studies and boasts multiple features, such as configurable training schedules, adaptive training algorithms, and an optional Web-based interface that supports monitoring of participants' commitment. It has high portability, can be flexibly adapted to virtually every requirement, and is published open source, and hence is available for free. Future plans include extending the documentation and tutorials already available and developing a graphical interface to facilitate the creation and modification of module files.

**Table 1** Modifiable parameters of the example module file

| Experimental Parameter | Description | Possible Values |
|---|---|---|
| defaultInterElementPauseDuration | Duration in milliseconds of a pause (i.e., a blank screen) between the study components | Any integer[a] |
| numIterations | Defines the number of iterations of elements within a list (i.e., number of trials) | Any integer |
| sampleSize | Number of trials that are used to calculate the user's accuracy for the adaptive training algorithm | Any integer |
| maxThreshold | If the accuracy in a specified number of trials (sampleSize) is above this value, task difficulty will be increased. | Integer between 0 and 100 (inclusive) |
| minThreshold | If the accuracy in a specified number of trials (sampleSize) is below this value, task difficulty will be decreased. | Integer between 0 and 100 (inclusive) |

[a] Whole numbers

**Author note** This work was supported by a grant to the first author from the Forschungskredit of the University of Zurich. We thank Nicole Cruz, Simone Eberhart, Julia Herkert, and the experimenters of past studies for testing earlier versions of Tatool and for their valuable feedback. We also thank Klaus Oberauer for his helpful comments on an earlier version of the manuscript.

## References

Brainard, D. H. (1997). The Psychophysics Toolbox. *Spatial Vision, 10,* 433–436. doi:10.1163/156856897X00357

Brown, J. (1958). Some tests of the decay theory of immediate memory. *The Quarterly Journal of Experimental Psychology, 10,* 12–21. doi:10.1080/17470215808416249

Daneman, M., & Carpenter, P. A. (1980). Individual differences in working memory and reading. *Journal of Verbal Learning and Verbal Behavior, 19,* 450–466. doi:10.1016/S0022-5371(80)90312-6

Draine, S. (1989). *Inquisit [Computer software]*. Seattle, WA: Millisecond Software.

Geller, A. S., Schleifer, I. K., Sederberg, P. B., Jacobs, J., & Kahana, M. J. (2007). PyEPL: A cross-platform experiment-programming library. *Behavior Research Methods, 39,* 950–058. doi:10.3758/BF03192990

Haxby, J. V., Parasuraman, R., Lalonde, F., & Abboud, H. (1993). SuperLab: General-purpose Macintosh software for human experimental psychology and psychological testing. *Behavior Research Methods, 25,* 400–405. doi:10.3758/BF03204531

Keller, F., Subahshini, G., Mayo, N., & Corley, M. (2009). Timing accuracy of web experiments: A case study using the WebExp software package. *Behavior Research Methods, 41,* 1–12. doi:10.3758/BRM.41.1.12

Mairesse, O., Hofmans, J., & Theuns, P. (2008). The Functional Measurement Experiment Builder Suite: Two Java-based programs to generate and run functional measurement experiments. *Behavior Research Methods, 40,* 408–412. doi:10.3758/BRM.40.2.408

Pelli, D. G. (1997). The VideoToolbox software for visual psychophysics: Transforming numbers into movies. *Spatial Vision, 10,* 437–442. doi:10.1163/156856897X00366

Schmiedek, F., Bauer, C., Lövden, M., Brose, A., & Lindenberger, U. (2010). Cognitive enrichment in old age. *GeroPsych, 23,* 59–67. doi:10.1024/1662-9647/a000013

Schneider, W., Eschman, A., & Zuccolotto, A. (2002). *E-Prime [Computer software]*. Pittsburgh, PA: Psychology Software Tools.

Stevens, M., Lammertyn, J., Verbruggen, F., & Vandierendonck, A. (2006). Tscope: A C library for programming cognitive experiments on the MS Windows platform. *Behavior Research Methods, 38,* 280–286. doi:10.3758/BF03192779

Stoet, G. (2010). PsyToolkit: A software package for programming psychological experiments using Linux. *Behavior Research Methods, 42,* 1096–1104. doi:10.3758/BRM.42.4.1096

Stroop, J. R. (1935). Studies of interference in serial verbal reactions. *Journal of Experimental Psychology, 18,* 643–662. doi:10.1037/0096-3445.121.1.15