Conducting Browser-Based User Studies in Software Engineering

Insights from Behavioral Science

Lavinia Dunagan *[1] and **Joshua Sunshine** †[2]

[1] *University of Washington, Seattle, WA*
[2] *Carnegie Mellon University, Pittsburgh, PA*

Abstract

User studies of software engineers are an invaluable part of the evaluation of new tools in academic software engineering (SE). Unfortunately, significant barriers to the efficient execution of these user studies still exist. Researchers in software engineering and related fields continue to struggle with recruiting, orchestration, experimental design, and IRB approval (Buse 2011). In this paper we address these challenges by comparing them to problem domains already identified and addressed in the behavioral sciences. We present (1) a survey of the current frameworks for administering online experiments in behavioral science and (2) a pair of software engineering-specific tools extending jsPsych, a front-end framework from behavioral science.

Keywords: Research methodology. Tools. Human subjects. Experiments.

1 Introduction

One of the central goals of software engineering and programming languages research is the creation of new tools for people, e.g., environments, languages, and techniques. User studies of software engineers are thus an invaluable part of the evaluation of new tools — in order to, for instance, validate that a new tool does in fact help engineers produce more robust code. Unfortunately, significant barriers to the efficient execution of these user studies still exist. As a result, user studies evaluating new programming tools produced by researchers remain uncommon.

Researchers in software engineering and related fields continue to struggle with recruiting, orchestration, experimental design, and IRB approval [1]. Software engineering is itself a set of specialized skills, so among developers there is a wide range of levels of experience with any given combination of tools; it is especially difficult to recruit people who accurately represent the future users of a tool, and in fact user studies frequently default to using undergraduate students as subjects (72% of the time, in one measure of papers from 1993 to 2002) even when the ultimate target audience is developers, who are themselves notoriously difficult to recruit [2]. Orchestration, or the actual implementation of an environment (in both the broad and development senses) and the provision of access to that environment, presents difficulties on the basis of its engineering complexity and the conflict between realism and control. Experimental design is challenging due to the relative lack of experience many software engineering researchers have with methodological questions — in contrast to behavioral scientists, who may themselves lack technical skills that are more expected of scholars in computer science. Similarly, the process of IRB approval and the primacy of other ethical considerations in human studies is seen as too great an obstacle by many researchers [2].

The first three challenges described above (recruiting, orchestration, and design) roughly map on to problem domains already identified and addressed by tools for browser-based behavioral science. We extend jsPsych, one of these tools, in the creation of our own prototypical tool; jsPsych is a front-end library, and our solution is focused on orchestration and design of a study timeline (as opposed to, for instance, server administration). Some specific practices in behavioral science are not necessarily workable in software engineering, such as recruiting from a general pool of participants like that of Mechanical Turk [2]. Nonetheless, there are sufficient similarities between the experimental philosophies of user-focused software engineering and behavioral science on the whole to allow us to effectively compare the two [3]. Both disciplines involve collecting data on human participants'

*Email: laviniad@uw.edu
†Email: sunshine@cs.cmu.edu

PLATEAU
12th Annual Workshop at the Intersection of PL and HCI

Organizers:
Sarah Chasins, Elena Glassman, and Joshua Sunshine

completion of tasks in an experimental context that balances control and relevance to practices ”in the wild” [1], [4]. In the first place, behavioral software engineering has its roots in an empiricist turn from the late 90’s and early 00’s which looked to theories of work in other fields to understand software development [3], [5]. It follows that we may use tools from behavioral science as models for the extension of methods in software engineering.

Builder for Software Engineering Experiments (BSEE), the system we describe below, uses an existing framework from psychology; it is intended to illustrate a path towards alleviating the burden of experimental design and instrumentation, as well as evince some of the challenges of constructing a tool that is both appropriately general and sufficiently structured.[1] Our secondary aim is to compare and to some extent unify the experimental paradigms in behavioral science and user-focused software engineering in order to bolster the validity of the latter. In this paper we present (1) a brief survey of the current state of online experiments in behavioral science with a focus on frameworks, where we find that current practice already includes tools that are extensible and lend themselves to various disciplines, and (2) a pair of software engineering-specific tools enabling greater ease of use and reproducibility: a code editor and testing plugin for one of these formats, jsPsych, and a command-line wizard for an associated experiment builder [7], [8].

2 Online Experiments in Behavioral Science

Disciplines in the social sciences with a focus on individual behavior (like psychology and linguistics), hereafter referred to as the behavioral sciences, have a robust ecosystem for online experiments [9]. A number of services exist for virtually every aspect of behavioral science experimentation, including both paid and free options. These tools may cover one or more of the aspects of online experimentation described below.

Prior surveys of the state of online experimentation in behavioral science have presented three infrastructure requirements, and consequently three categories of tools: “(1) a builder for a browser-based experiment front-end, (2) a server to host the experiment, and (3) a participant recruitment tool” [9]. Experiment creation tools, like jsPsych, PsychoPy/PsychoJS, and lab.js, are Javascript-based and may be modified via both code and provided GUIs (although jsPsych lacks a GUI for this purpose). Hosting tools include Pavlovia and cognition.run, which provide managed servers and integration with popular participant management tools. Recruitment generally entails the use of Amazon Mechanical Turk, Prolific, or CrowdFlower. Gorilla, Testable, and Inquisit all support integrated experiment/task creation and hosting; however, none of them are free [10]. The hosting sites mentioned, especially the Experiment Factory, have as a secondary goal reproducibility in experimentation. Consequently, several of them contain demo repositories or projects one can simply copy (or fork) to begin a new study [11]. This supports the goal in software engineering of establishing more clear standards surrounding human-centered tool evaluations [1], [2]. Extending a figure in the paper introducing Gorilla, we present a taxonomy of browser-based behavioral science frameworks [10]:

Type	Name	Free	Open-source
Hosted experiment builder	Gorilla	✗	✗
	Testable	✗	✗
	Inquisit	✗	✗
Experiment builder (GUI)	E-Prime	✗	✗
	PsychoPy Builder	✓	✓
	lab.js	✓	✓
Experiment builder (coding)	jsPsych	✓	✓
	PsychoPy/PsychoJS	✓	✓

1 We may take as a reference point the Research Through Design paradigm, which holds that the design process, especially the production of artifacts that embody prior assumptions surrounding a given field’s methodology, can enable broader exploration of issues in HCI [6].

	PsychToolBox	✓	✓
Hosting service	Pavlovia	✓	✗
	cognition.run	✓	✗
	LIONESS Lab	✓	✓
Hosting configuration	JATOS	✓	✓
	The Experiment Factory	✓	✓
Recruiting service	Prolific	✗	✗
	Amazon Mechanical Turk	✗	✗

Table 1. Tools and services for browser-based experiments in behavioral science.

None of the tools listed here explicitly enable fully integrated code editor functionality (i.e. a text box with autocompletion, syntax highlighting, etc. and the ability to connect to a backend). While most of the frameworks which expose Javascript could be coerced into operating this way, and it is even relatively simple to use `<iframe>` elements to add elements like a read-only CodePen embedding, none (to our knowledge) support this aspect of software engineering research. That being said, additions could be made for all three of these areas. As discussed in previous work, desiderata for controlled experimentation in software engineering include standardized environments, which is in some sense a hosting issue, and participants with expertise in a given field, which is a recruitment issue. One behavioral science platform, Lookit, is focused on behavioral science experiments involving children; it is arguably dealing with its own specialized recruitment problem, and its solution is the maintenance of a participant pool. We envision something similar in the future as an extension to our project.

Our interest here is mostly in jsPsych, the framework that we build on as part of our prototype. It is a tool for building browser-based behavioral science experiments, although it has been extended to HCI applications (as discussed below) and other fields that do not fall into the category of conventional psychology; it effectively represents the front-end of one instantiation of an experiment, with some explicit support for connections to backend and recruiting services like Amazon Mechanical Turk. It is also supported in some capacity by virtually all of the hosting platforms which allow the use of code to define experiments. The foundation of jsPsych is the experiment timeline, which is composed of "trial" blocks and sub-timelines. A typical jsPsych experiment consists of a series of stimuli and during each a wait for a response from the user, along with one or more surveys and display text. From the experimenter's perspective, this entails the declaration of trial blocks representing each of these phases of the experiment. For example, in the jsPsych demo implementation of a simple reaction time experiment the "welcome" block is declared in the following manner:

```javascript
/* define welcome message trial */
var welcome = {
type: "html-keyboard-response",
       stimulus: "Welcome to the experiment. Press any key to begin."
};
timeline.push(welcome);
```

The philosophy of jsPsych is that it should be usable for both experienced programmers of browser-based behavioral science experiments and those who have not previously worked with programs of any complexity. It prioritizes extensibility and welcomes additions in the form of plugins. To investigate the current use of jsPsych, we conducted a brief study of projects tagged with it on a popular behavioral science hosting platform and a review of its citations.

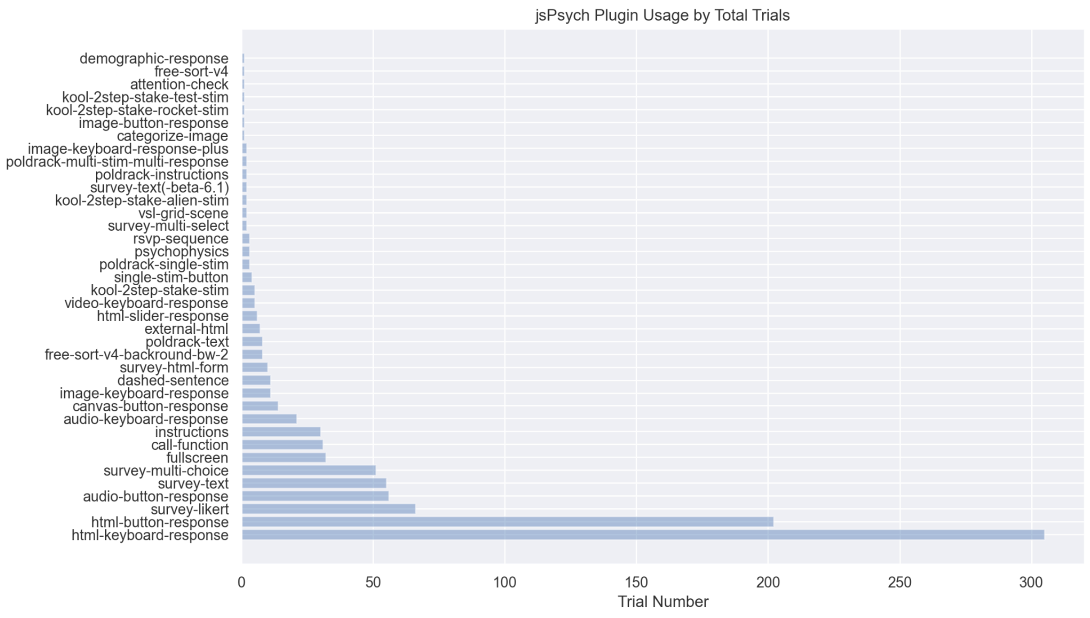
2.1 Datasets

The results here were drawn mainly from Pavlovia and Web of Science. Pavlovia is a hosting website created by the maintainers of PsychoPy and its Javascript counterpart, PsychoJS [11]. Its website lists projects according to whether they use PsychoJS, lab.js, or jsPsych; we gathered a corpus of relevant projects by manually collecting projects from the jsPsych category that were not duplicates, currently piloting, or demos. This yielded a set of approximately 50 projects. The following plugin-specific data was drawn from this corpus.

The Web of Science dataset was gathered via a "Cited By" search for the most recent papers associated with jsPsych (from 2015). Web of Science is human-curated and draws from a specific set of validated journals, as opposed to "scholarly" sources throughout the Internet, so it is generally a higher quality resource than Google Scholar given our interest in excluding research that may not have been completed and published [12].

2.2 Plugin Distribution

We observe that many of the plugins found on Pavlovia are essentially one-offs (some of which are nonstandard, i.e. not part of vanilla jsPsych), while the classic use of jsPsych includes three fairly limited groups of basic formats. These groups can be described in brief as those that allow the insertion of arbitrary text and HTML, those that support surveys, and those that support the insertion of various kinds of stimuli (audio, image, etc.). For the purposes of software engineering, only the first two are directly relevant to the common study designs previously described in the literature [2]; given their popularity, we include them in the CLW described in the following section. As shown in the below measure, html-keyboard-response and its cousin, html-button-response, are the most popular plugins, while the survey-* plugins are also widely used (plugins with zero use on Pavlovia are excluded from the graph):



It is worth noting that while there are 48 official plugins, **18** of the 38 plugins above (the ones that were used on Pavlovia) were one-offs created specifically for the project they were attached to. That is, less than half of the official plugins that are part of vanilla jsPsych are used in our corpus, while almost half of the plugins that are used were created by non-contributor practitioners.

As an aside, the distribution of plugins and packages in similar software ecosystems in general is also more or less exponential. We also looked at the top 1000 packages in `npm` (as of August 2019, the last time when they were publicly pre-computed) and the modules in PsychoJS, and found that all three have module distributions that can be extremely well-modeled by an exponential of the form $y = m * e(-t * x) + b$ where m, t, and b are parameters [13].

	R^2	m	t	b
`npm`	0.869	$3.938 * 10^4$	$4.797 * 10^{-2}$	$1.118 * 10^3$
PsychoPy/PsychoJS	0.997	$2.909 * 10^5$	$7.073 * 10^{-1}$	$1.539 * 10^2$
jsPsych	0.962	$4.980 * 10^2$	$5.090 * 10^{-1}$	3.298

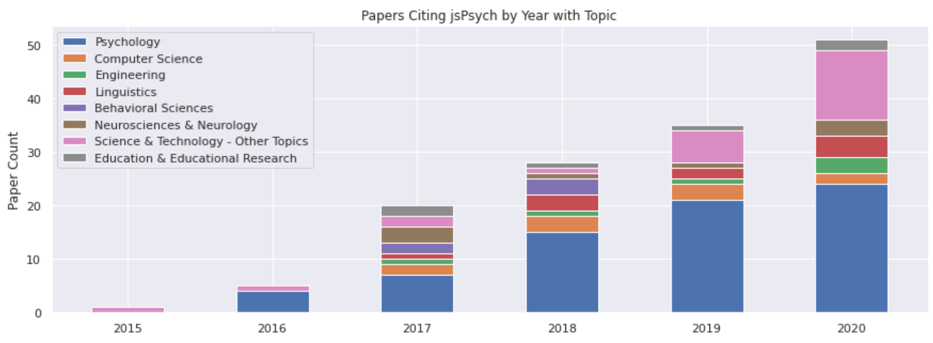
2.3 Randomness and Trial Number

The standard jsPsych paradigm of the timeline allows conditional flow and random selection of stimuli within an experiment; by nature it does not inherently support the management of participants with respect to other participants' live responses. However, its flexibility in the form of dynamic parameters means changes can be made to task details and experiment structure on the basis of the statuses of outside resources (for instance, one might decide in a behavioral science setting to set participants' stimulus values according to prior ones to achieve coverage of a stimulus space).

In our survey of jsPsych Pavlovia projects, we found that randomness is widely used — albeit frequently simply for the generation of random identification numbers. Of the 50 projects surveyed, 23 used randomness for something besides simple identification. Almost all of these 23 projects were attempting to cover large parts of some stimulus space, and instead of implementing a costly backend that did this programmatically they selected different stimuli uniformly at random.

2.4 Distribution Across Disciplines

The basic jsPsych framework is used in several disciplines. Using a "Cited By" search of Web of Science, we found that (unsurprisingly) psychology is the most popular use case of jsPsych, while other fields that in some way touch on individual behavior (linguistics, neurology, etc.) make appearances. The figure below represents the disciplinary distribution over time. Notably, papers from the "Behavioral Research Methods" journal were excluded because of how many of them corresponded to papers describing other frameworks, while disciplines where less than five papers were published using jsPsych are not represented here either. "Science & Technology — Other Topics" and "Computer Science" also generally refer to reviews of online behavioral science frameworks, i.e. they do not connote original research done using jsPsych.



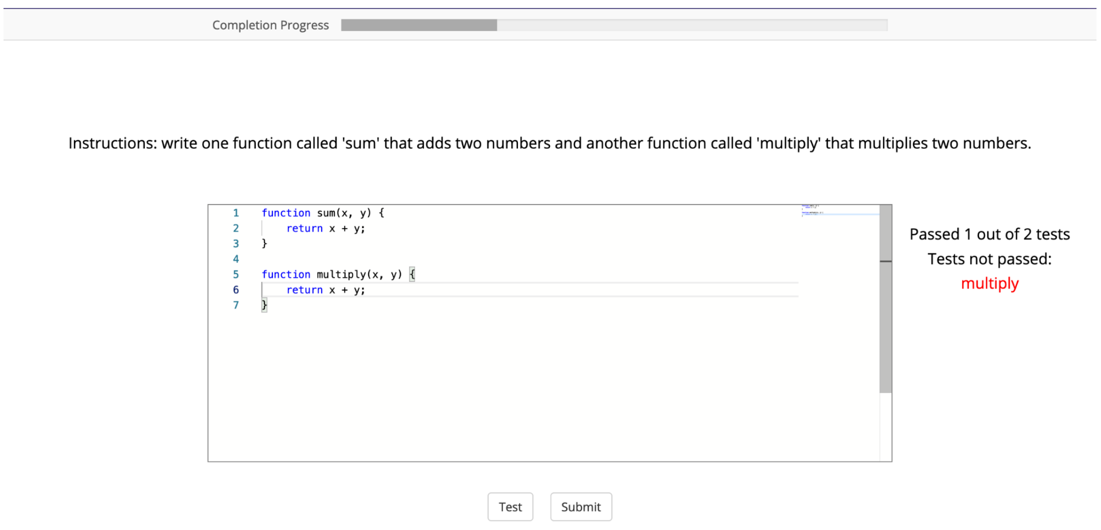
Only one study using jsPsych, to our knowledge, has been published in a computer science subdiscipline: Garaialde et al. (2021), a study on gamification and reward placement in apps published at CHI [14]. This paper recruited 70 subjects through Amazon Mechanical Turk and was hosted on a university server. The "trials" displayed consisted of data-logging tasks, interspersed with monetary rewards at different times depending on the condition. The main result was that "[u]sers selected applications more frequently when rewards were placed closer to the start of the interaction with the application." While this is not a tool evaluation, per se, it does evince the preexisting similarities between behavioral science proper and behavioral software engineering, as well as the advantages of a tool like jsPsych. The authors emphasize that previous studies in this vein had relied on relatively simple sequences of tasks and rewards, while this study was interested in more complex (and more realistic) sequences of decision-making [14].

3 BSEE: Plugin and CLW

Our contribution to the landscape of online experimentation is a plugin for jsPsych and an accompanying experiment builder. This experiment builder is divided into a task builder and a group assignment tool. The plugin for jsPsych, `jspsych-code-editor`, allows one to define a single coding task (or "trial", in the jsPsych paradigm). The experiment builder allows one to describe a jsPsych-based software engineering experiment in a small command-line wizard, thus both reducing the knowledge and effort required to design the experiment and introducing a standard format for jsPsych experiments created this way.

3.1 jspsych-code-editor Plugin

The code editor plugin consists of a code editor, a test results panel, and the data collection and experiment-wide displays, typically a progress bar, inherited by all jsPsych plugins. Secure testing native to the application is currently only possible for Javascript tasks due to the constraints of a browser-only tool. One can supply a URI to a Jest test file and/or a simple input/output specification like one shown in the demo; these tests can optionally be run multiple times before submission or only once, and the results can be either displayed to the participant or hidden. As shown in the below figure, one can optionally display both the "numeric" test results (the number passed out of the total) or "verbose" results (the names of the specific tests that were failed). The code editor is



Instructions: write one function called 'sum' that adds two numbers and another function called 'multiply' that multiplies two numbers.

```
function sum(x, y) {
    return x + y;
}

function multiply(x, y) {
    return x + y;
}
```

Passed 1 out of 2 tests
Tests not passed:
multiply

Figure 1. Possible interface for a participant in an experiment based on jspsych and our code editor plugin.

based on Monaco, the code editor underlying VSCode. Consequently, it is highly customizable and (most importantly for PL researchers) possible to arbitrarily change the syntax highlighting seen by the user. While these options are not exposed as part of the CLW, since it is unlikely they will be broadly applicable for the typical experimenter, they can be modified as part of the trial declaration in the jsPsych file. The primary built-in options are language and starting text, which will likely be sufficient for most coding tasks writ large.

The output of the plugin is part of the jsPsych data object obtained from the experiment. This output includes the time taken for each submission, the statuses of the specified tests, and the text of the editor itself. These are the types of data most commonly used already within software engineering studies, and with the data provided as part of jsPsych's broader experiment context it is possible to associate them with other trials' results from the same experiment (e.g. to determine whether a participant solved both a first and a second debugging task faster than another) [1], [2]. The plugin collects this information for each code submission, instead of at the end of the trial; it thus introduces a notion of data collection and corresponding feedback that does not require moving on to a completely separate task (in comparison with most of the vanilla jsPsych plugins).

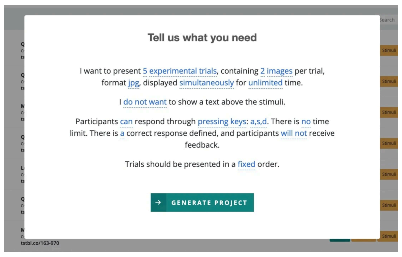
3.2 BSEE Experiment CLW

The second component of our toolbox, the command-line wizard, is itself divided into two tools: a group assignment interface and a timeline builder. The assignment interface was designed for software engineering researchers who may be previously unacquainted with different random assignment techniques. The intent here is to automate common group assignment patterns. Our wizard takes in a JSON specification of a set of participants in addition to assignment parameters (the number of groups, the demographic to balance by) and outputs group assignments for each. One can do completely random assignment, balanced random assignment, or random assignment balanced by a demographic feature; the demographic feature can be categorical or numeric.

The timeline builder includes both the actual specification of the tasks through the selection of options in the wizard and the assignment of those tasks to groups. The selection of options entails first picking a plugin category to use, then picking plugin-specific features — e.g. a URL for an `external-html` task, where an external webpage is being displayed. The four plugin options offered through the wizard are HTML, survey, code editor, or external webpage. These were included on two bases: one, that HTML elements and surveys are two of the three most popular jsPsych plugin types in a behavioral science setting (the other being stimuli), and two, survey-style participant data collection and programming tasks are critical elements of many software engineering user studies. Indeed, the three most popular methodologies are interviews and questionnaires, controlled experiments, and web surveys [1].

As shown below, the model our builder uses is one of a pool of tasks and a pool of groups, where different sequences of tasks are assigned to different groups. This framework is both flexible and easily transferable. It avoids some of the complexities of a more explicit experiment/control interface requiring the modulation of specific tasks, while nonetheless still being able to express popular study designs (including those where the variable is the presence or absence of a task). For example, one might have four tasks: a pre-survey, a debugging task, a distinct coding task, and a post-survey, and three groups, Group 1, Group 2, and Group 3. One could optionally assign the sequence `[pre-survey, debugging, coding, post-survey]` to Group 1, the sequence `[pre-survey, coding, debugging, post-survey]` to Group 2, and the sequence `[pre-survey, coding, post-survey]` to Group 3.

The task builder on a whole was heavily influenced by the paradigm used by Testable. The experiment design interface for that platform is contained in a natural-language form. This amount of tooling is ideal given that many software engineering researchers do not generally have significant training in the design of studies with human participants [2]. A comparison is shown below:



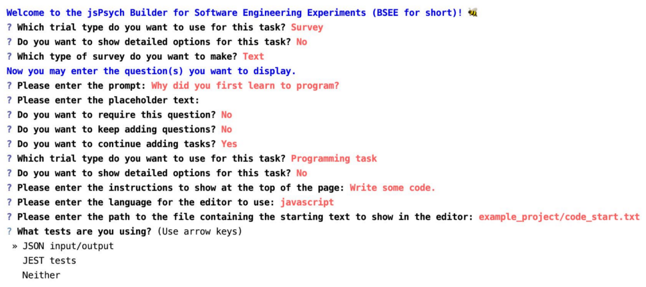
Tell us what you need

I want to present 5 experimental trials, containing 2 images per trial, format jpg, displayed simultaneously for unlimited time.

I do not want to show a text above the stimuli.

Participants can respond through pressing keys: a,s,d. There is no time limit. There is a correct response defined, and participants will not receive feedback.

Trials should be presented in a fixed order.



```
Welcome to the jsPsych Builder for Software Engineering Experiments (BSEE for short)! 🐝
? Which trial type do you want to use for this task? Survey
? Do you want to show detailed options for this task? No
? Which type of survey do you want to make? Text
Now you may enter the question(s) you want to display.
? Please enter the prompt: Why did you first learn to program?
? Please enter the placeholder text:
? Do you want to require this question? No
? Do you want to keep adding questions? No
? Do you want to continue adding tasks? Yes
? Which trial type do you want to use for this task? Programming task
? Do you want to show detailed options for this task? No
? Please enter the instructions to show at the top of the page: Write some code.
? Please enter the language for the editor to use: javascript
? Please enter the path to the file containing the starting text to show in the editor: example_project/code_start.txt
? What tests are you using? (Use arrow keys)
 » JSON input/output
   JEST tests
   Neither
```

Figure 2. Comparison between Testable's behavioral science experiment builder (left) and our prototypical command-line experiment design interface (right).

A potential user of the tool is led through a series of questions which lead to an exact specification of the experiment that is nonetheless relatively easy to understand. Anecdotally, the CLW method takes less time to specify a small programming task than raw programming for jsPsych, even for those who have some experience in jsPsych. We envision this effect being more pronounced for those who may not have used an experiment design framework or designed an experiment at all.

4 Conclusion

In this paper we presented a survey of the state of browser-based experimentation tools in the behavioral sciences and a prototype motivated by insights from that survey specifically for software engineering user studies. The tools available to behavioral scientists are diverse and numerous; they have only increased in diversity and number since the beginning of the COVID-19 pandemic [15]. In presenting another field's solutions to several salient challenges in online experimentation, we hope to encourage more frequent use of user study evaluations in software engineering, especially given their decline in the past decade [2]. To that end, our jsPsych plugin and command-line wizard prototypes are built as evidence of the feasibility of online experimentation for software engineering. Systematic and large-scale empirical studies of human participants in software engineering are both desirable and possible with limited additions to existing infrastructure.

5 Acknowledgments

We would like to express our appreciation for everyone involved in the REUSE program at Carnegie Mellon, especially Hita Kambhamettu, who worked with us on the plugin. We thank Thomas LaToza and Xiaoyin Wang for their collaboration and comments on earlier drafts. We also thank Josh de Leeuw for his creation and maintenance of jsPsych and his willingness to discuss this project with us.

References

[1] R. P. Buse, C. Sadowski, and W. Weimer, "Benefits and barriers of user evaluation in software engineering research," in *Proceedings of the 2011 ACM International Conference on Object Oriented Programming Systems Languages and Applications*, ser. OOPSLA '11, Portland, Oregon, USA: Association for Computing Machinery, 2011, pp. 643–656, ISBN: 9781450309400. DOI: 10.1145/2048066.2048117. [Online]. Available: https://doi.org/10.1145/2048066.2048117.

[2] A. J. Ko, T. D. LaToza, and M. M. Burnett, "A practical guide to controlled experiments of software engineering tools with human participants," en, *Empirical Software Engineering*, vol. 20, no. 1, pp. 110–141, Feb. 2015, ISSN: 1382-3256, 1573-7616. DOI: 10.1007/s10664-013-9279-3. [Online]. Available: http://link.springer.com/10.1007/s10664-013-9279-3 (visited on 07/16/2021).

[3] D. Graziotin, P. Lenberg, R. Feldt, and S. Wagner, "Psychometrics in Behavioral Software Engineering: A Methodological Introduction with Guidelines," en, *arXiv:2005.09959 [cs]*, Jun. 2021, arXiv: 2005.09959. [Online]. Available: http://arxiv.org/abs/2005.09959 (visited on 06/14/2021).

[4] W. Labov, "The Social Stratification of (r) in New York City Department Stores," in *Sociolinguistics: A Reader*, N. Coupland and A. Jaworski, Eds., London: Macmillan Education UK, 1997, pp. 168–178, ISBN: 978-1-349-25582-5. DOI: 10.1007/978-1-349-25582-5_14. [Online]. Available: https://doi.org/10.1007/978-1-349-25582-5_14.

[5] K. Arnold, "Programmers are people, too: Programming language and api designers can learn a lot from the field of human-factors design.," *Queue*, vol. 3, no. 5, pp. 54–59, Jun. 2005, ISSN: 1542-7730. DOI: 10.1145/1071713.1071731. [Online]. Available: https://doi.org/10.1145/1071713.1071731.

[6] P. Stappers and E. Giaccardi, "Research through design," English, in *The Encyclopedia of Human-Computer Interaction*, M. Soegaard and R. Friis-Dam, Eds., 2nd. The Interaction Design Foundation, 2017, pp. 1–94.

[7] L. Dunagan, *Bsee*, https://github.com/se-user-studies/experiment-builder-prototype, 2021.

[8] L. Dunagan and H. Kambhamettu, *Jspsych-code-editor*, https://github.com/se-user-studies/jspsych-se-plugin/blob/dc13a8377a0c49f782b7f0fcb6dcde6b2387b0a5/plugins/jspsych-code-editor.js, 2021.

[9] T. Grootswagers, "A primer on running human behavioural experiments online," en, *Behavior Research Methods*, vol. 52, no. 6, pp. 2283–2286, Dec. 2020, ISSN: 1554-3528. DOI: 10.3758/s13428-020-01395-3. [Online]. Available: http://link.springer.com/10.3758/s13428-020-01395-3 (visited on 07/08/2021).

[10] A. L. Anwyl-Irvine, J. Massonnié, A. Flitton, N. Kirkham, and J. K. Evershed, "Gorilla in our midst: An online behavioral experiment builder," en, *Behavior Research Methods*, vol. 52, no. 1, pp. 388–407, Feb. 2020, ISSN: 1554-3528. DOI: 10.3758/s13428-019-01237-x. [Online]. Available: https://doi.org/10.3758/s13428-019-01237-x (visited on 06/11/2021).

[11] D. Bridges, A. Pitiot, M. R. MacAskill, and J. W. Peirce, *The timing mega-study: Comparing a range of experiment generators, both lab-based and online*, Jan. 2020. DOI: 10.7717/peerj.9414. [Online]. Available: psyarxiv.com/d6nu5.

[12] A. Martín-Martín, E. Orduna-Malea, M. Thelwall, and E. D. López-Cózar, "Google Scholar, Web of Science, and Scopus: A systematic comparison of citations in 252 subject categories," *Journal of Informetrics*, vol. 12, no. 4, pp. 1160–1177, 2018, ISSN: 1751-1577. DOI: https://doi.org/10.1016/j.joi.2018.09.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1751157718303249.

[13] anvaka, *Npm rank*, en. [Online]. Available: https://gist.github.com/anvaka/8e8fa57c7ee1350e3491 (visited on 08/23/2021).

[14] D. Garaialde, A. L. Cox, and B. R. Cowan, "Designing gamified rewards to encourage repeated app selection: Effect of reward placement," *International Journal of Human-Computer Studies*, vol. 153, p. 102 661, 2021, ISSN: 1071-5819. DOI: https://doi.org/10.1016/j.ijhcs.2021.102661. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1071581921000793.

[15] Sydney Wood, *Collecting behavioral data online*, en. [Online]. Available: https://www.apa.org/science/leadership/students/collecting-behavioral-data (visited on 07/21/2021).