Evaluating the Usefulness and Ease of Use of an Experimentation Definition Language

Florian Auer, Michael Felderer
University of Innsbruck
Innsbruck, Austria
florian.auer@uibk.ac.at, michael.felderer@uibk.ac.at

***Abstract*—Before any online controlled experiment, a hypothesis has to be formulated. Moreover, the design, execution, and analysis have to be planned. Given that the definition of an experiment varies considerably amongst experimentation platforms, no common experiment definition exists. Furthermore, there is to the best of the authors' knowledge no platform-independent experiment definition model proposed in the literature.**

Thus, we aim to propose an experimentation definition language and evaluate its usefulness and ease of use. Therefore, we developed a domain-specific language based on the results of a previous study and conducted a technology acceptance model study with 30 participants. It revealed that the proposed experiment definition language is considered useful amongst the majority of participants. Moreover, most of the participants rated the language easy to use. Participants without prior knowledge of the domain-specific language's host language (JSON – JavaScript Object Notation) rated the language considerably less easy to use.

To conclude, the proposed experimentation definition language supports practitioners in their experimentation process by providing them a structure and pointing them out to experiment characteristics that could be considered. Furthermore, the machine-readable definition of experiments represents a first step for many research directions, like the automated verification of experiments, or the development of an experiment knowledge base.

***Index Terms*—continuous experimentation, domain-specific language, technology acceptance model, online controlled experiment definition**

I. Introduction

Online controlled experimentation of software features allows to quickly assess ideas and to make data-driven decisions about them [1]. Furthermore, the technique of deploying a change, exposing it to a subset of the users and collecting telemetry about it, has the potential to be a vehicle for software quality assurance [2] of modern technologies like machine learning or the internet of things (IoT) that are challenging for offline software quality assurance techniques [3] like traditional software testing. Given the potentially large impact of decisions that are based on experiments, the correct execution and therefore, the reliability of experiments are of great importance to organizations. Recent empirical studies [4], [5] found that the majority of practitioners use in-house built experimentation platforms. Thus, they use self-built tools to execute their experiments. Although this seems to be a reasonable choice to adapt the experimentation process to the respective needs of an organization, it complicates the knowledge and experience exchange within the community and the development of platform-independent techniques. A platform-independent experimentation definition language would represent a foundation for a more structured experiment definition, static experiment verification, and software quality assurance for modern technologies like machine learning or IoT – independent of the concrete used experimentation platform. Thus, this paper presents a tool-independent experimentation definition language (EDL) based on the findings of [5]. The language is evaluated on its usefulness and ease of use using the technology acceptance model by Davis et al. [6].

The remainder of this paper is organized as follows. Section II describes the proposed experimentation definition language. Section III discusses the applied research methods and their threats to validity. Section IV presents the findings and discusses them. Finally, Section V concludes the paper.

II. An Experimentation Definition Language (EDL)

A domain-specific language to define online controlled experiments needs to describe an experiment through all its lifecycle phases to provide a complete definition of an experiment. Fabijan et al. [4] present an experiment lifecycle with three phases (see Fig. 1).

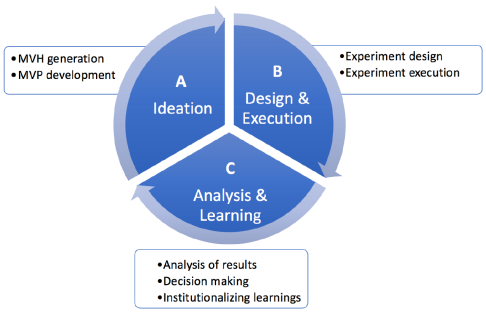

Fig. 1. Experiment lifecycle by Fabijan et al. [4].

In the first phase called the ideation phase, ideas are formalized as treatment descriptions and their success criteria (minimal viable hypotheses). Based on a hypothesis the treatment is developed as a minimal viable product. Thus, the implementation is tailored to meet the experiment's requirements.

DOI reference number: 10.18293/SEKE2020-067

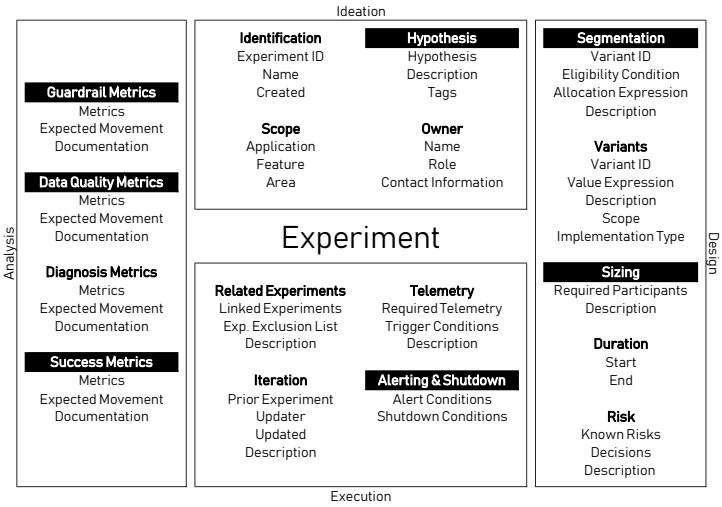

Fig. 2. Experiment definition characteristics taxonomy [5].

In the next phase, the design and execution of an experiment are addressed. The design of an experiment includes amongst others, the segmentation of the customers and the size and duration required for reliable findings. During the execution of an experiment monitoring is necessary to detect harmful experiments (e.g. through alert conditions).

Finally, in the last phase, the results are analyzed, decisions based on the findings are made and lessons from an experiment are shared. Therefore, the collected data of an experiment is analyzed and metrics are computed. After the comparison of the success criteria with the outcome of the experiment, data-driven decisions are made. Finally, the lessons learned about the experimentation process (e.g. execution) and the influence of the change on metrics are shared within an organization.

Authors in [5] assembled a taxonomy of characteristics (see Fig. 2) that is used to describe online controlled experiments in each phase of the experimentation lifecycle. It contains in comparison to other experiment models (e.g. [4]), properties that define the experiment design for each phase. The study considered the literature on online controlled experimentation, experimentation platforms, and the opinions of industrial experts. The developed taxonomy consists of 17 characteristics (e.g. experiment owner) and their properties (e.g. name, role, contact). Based on these findings and the observations made during the research, an experimentation definition language was developed. The described taxonomy of experimentation characteristics served as a domain model for the development of the domain-specific language.

The analysis of open-source as well as proprietary experimentation platforms in [5] revealed that the data exchange format JSON was the preferred data format of most platforms. Thus, it seemed reasonable to use JSON as host language given its widespread use in the community and its mature support by all common programming languages. As a result, the domain model based on the taxonomy was translated into a JSON schema that describes the experimentation definition language, its objects, and properties. This decision has the advantage that definitions written in the domain-specific language (DSL) are not only machine-readable, but they can be validated with standard JSON tools too. Listing 1 of an experiment defined in the language shows that the structure follows the taxonomy closely.

Listing 1. Structure of an experiment written in EDL.

```
{
  "Ideation":{
    "Hypothesis":...,
    "Owners":...
  },
  "Design":{
    "Variants":...,
    "Segmentation":...
  },
  "Execution":{
    "AlertingAndShutdown":...
  },
  "Analysis":{
    "SuccessMetrics":...,
    "GuardrailMetrics":...
  }
}
```

Given that the taxonomy already considers the experimentation lifecycle, the structure of the taxonomy was transferred to the language. Furthermore, the lifecycle-oriented arrangement of the characteristics gives guidelines during the planning of an experiment.

The second main technology used for the implementation of the language, is the — according to the Stack Overflow Developer Survey[1] — popular code editor Microsoft Visual Code[2]. It has a mature support for both JSON and JSON schemas, including additional JSON schema constructs to add online documentation, auto-completion of predefined partial templates and instant validation of the JSON document according to a JSON schema. Hence, in addition to the JSON schema, online documentation, and partial templates for all properties were developed. As a result, the editor allows language users to explore the language through auto-completion (e.g. what could be written at this location?) and provides rapid feedback on the syntactical validity of the defined experiment (e.g. is the experiment valid against the schema?).

As a result, the experiment definition language has the following key characteristics:

- *Human-readable*: Given that experimentation involves stakeholders from different functional expertise (e.g. UX design, software engineering, business), the experiment definition has to be easy to understand and readable independent of the reader's expertise.
- *Machine-readable*: The definition of an experiment is based on a common data exchange format that can be processed with standard tools.
- *Living documentation*: The definition of an experiment should be the single point of truth concerning the respective experiment. Thus, it should serve the stakeholders as a discussion base and plan of action, and it should provide

[1]https://insights.stackoverflow.com/survey/2019#development-environments-and-tools, accessed 2020-02-25.

[2]https://code.visualstudio.com/

all necessary properties for the experiment platform to execute the experiment.

- *Knowledge sharing*: Lessons learned from the design, execution, and analysis of the experiment are valuable knowledge that should be captured and shared. Therefore, the definition of an experiment includes comments, decisions, and the reasoning behind them. Moreover, properties like tags improve the structured archival of experiment definitions.

III. Research Method

To evaluate the proposed language, the technology acceptance model (TAM) by Davis et al. [6] was applied. It is a model based on the theory of reasoned action (TRA) [7] that allows to assess the user's technology acceptance behavior. The used sets of questions (see Table I) were adapted from [6], [8], and [9]. They measure the three main constructs perceived usefulness, ease of use, and self-predicted future use (see Fig. 3).

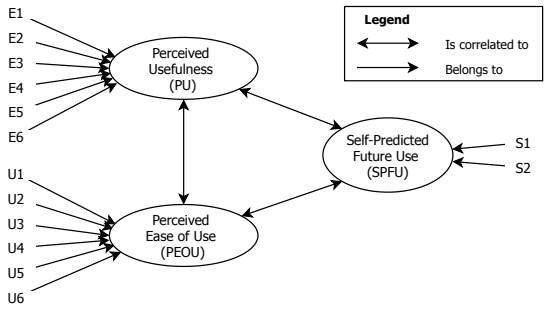

Fig. 3. Model of usefulness, ease of use, and self-predicted future usage (TAM). Abbreviations are defined in Table I.

A. Objective

The experiment was performed for two reasons. First, to evaluate the usefulness of a platform-independent and machine-readable experimentation definition language. Second, to evaluate the ease of use of the proposed experimentation definition language.

B. Variables

The following variables were considered in this study:

- *Perceived usefulness* is the degree users expect that the system will improve their job performance.
- *Perceived ease of use* is the degree users expect that the system will be free of effort.
- *Self-predicted future usage* is the degree users expect to use the system in the future.

C. Subjects

The subjects of the experiments were 30 graduate students of the University of Innsbruck (Austria) that were enrolled in a course on advanced concepts and techniques of software engineering. The course is part of the computer science as well as information systems master's (where some students did a bachelor in business administration) program. Thus, the students had a mixed background in computer science and business. Given that online controlled experimentation involves multiple stakeholders with different professional backgrounds, either more technical or more business-oriented, the setting seemed to be advantageous to represent potential users of the language. To further classify the subjects, some demographic questions about their prior knowledge on key technologies used by the experimentation definition language were asked (see Fig. 4). The results indicate that most participants (76%) mentioned that they know JSON. In contrast, only a quarter of all participants stated that they know the code editor Microsoft Visual Code.

TABLE I
Scale Items.

Perceived Usefulness	
U1	Using the software in my job would enable me to accomplish tasks more quickly.
U2	Using the software would improve my job performance.
U3	Using the software in my job would increase my productivity.
U4	Using the software would enhance my effectiveness on the job.
U5	Using the software would make it easier to do my job.
U6	I would find the software useful in my job.
Perceived Ease of Use	
E1	Learning to operate the software would be easy for me.
E2	I would find it easy to get the software to do what I want it to do.
E3	My interaction with the software would be clear and understandable.
E4	It was easy to become skillful using the software.
E5	It is easy to remember how to perform tasks using the software.
E6	I would find the software easy to use.
Self-predicted future usage	
S1	Assuming the software would be available on my job, I predict that I will use it on a regular basis in the future.
S2	I would prefer using the software to other forms for defining experiments.

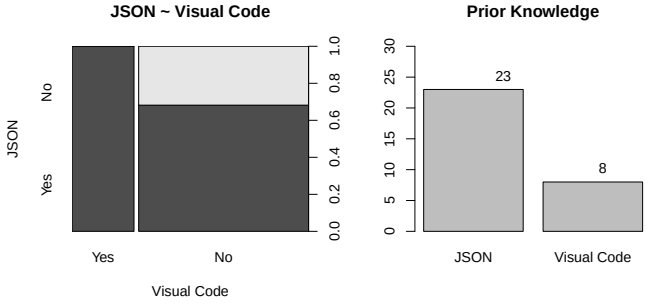

Fig. 4. Participants prior knowledge about the technologies JSON and Microsoft Visual Code.

D. Running the experiment

Given that the students already learned about experimentation in software engineering as part of the lecture, only a quick repetition of this topic was given. Thereafter, current approaches to experimentation (e.g. experimentation platforms, best practices) were discussed. After that, the development

and purpose of the experimentation definition language was presented. Finally, a short introduction of the tool (Microsoft Visual Code) was given. The language itself was introduced by some example experiment definitions. Next, all students received the same task to model an experiment in the language based on a written description of an experiment. Finally, the students submitted their written experiment definition and filled out a questionnaire containing demographic questions, the adapted TAM questions (see Table I), and a comment question.

E. Threats to validity

Potential threats of the study's validity [10] were considered and minimized whenever possible.

Construct validity was improved by considering the technology acceptance model [6] to evaluate the usefulness and ease of use of the experimentation definition language. Moreover, best practices and lessons learned from similar technology acceptance model studies (e.g. [8]) were considered.

Internal validity threats are caused by faulty conclusions that could for example happen because of mistakes in the statistical analysis. To mitigate this, the statistical analysis considers more than one measure (e.g. Pearson product correlation and Spearman rank-order correlation). Besides, the measurement instrument itself was analyzed regarding its reliability (Cronbach's alpha). Furthermore, the participants' demography was considered during the statistical analysis to draw more detailed conclusions of observations.

External validity covers to which extend the generalization of the results is justified. Given that experiments with students are usually considered to be of low external validity [11], the possibility to generalize the results of the study is limited. Nevertheless, this threat was slightly mitigated by considering graduate students from a course that addresses the subject online controlled experimentation and that have varying backgrounds (computer science, business) that reassembles teams in practice.

Reliability was improved by recording every step and decision of this study carefully and reporting the most important decisions as well as the reasoning behind them. Moreover, all questions of the study are reported.

IV. Results and discussion

A. Reliability

To test the reliability of the questionnaire, Cronbach's alpha is used commonly in empirical studies. Values above 0.8 are typically considered indicating the questionnaire as a highly reliable measurement instrument [12]. The analysis of the answers revealed a Cronbach's alpha of 0.967 for usefulness and 0.962 for ease of use. This indicates that our questionnaire is a reliable measurement instrument.

B. Factorial validity

In factor analysis it is examined whether the two factors ease of use and usefulness form distinct constructs (see Table II). The analysis clusters the variables together that tend to be correlated and assigns them a factor loading that describes the correlation of the variable to the factors. In Table II the results for all variables are given. Like correlations, the factor loading ranges from -1 (negative correlation) to +1 (positive correlation). A variable should have at least a factor loading of 0.7 to be a meaningful factor [13]. The results in Table II indicate that there are two factors on which the variables load. All variables associated with questions about usefulness tend to be loaded on the factor usefulness. The same can be observed for ease of use. The questions about ease of use tend to load on the factor ease of use. Thus, the result confirms that there are two factors ease of use and usefulness in the data.

TABLE II
Factorial analysis.

	Usefulness	**Ease of use**
Quick (U1)	0.92	0.03
Performance (U2)	0.96	-0.01
Productivity (U3)	0.92	-0.08
Effectiveness (U4)	0.91	0.03
Easy (U5)	0.91	-0.04
Useful (U6)	0.84	0.12
Easy to learn (E1)	-0.05	0.91
Easy to do (E2)	0.00	0.89
Clear (E3)	0.13	0.90
Skillful (E4)	0.24	0.81
Remember (E5)	-0.17	0.94
Easy to use (E6)	-0.03	0.95

C. Usefulness

In Fig. 5 the results for usefulness are shown. It presents a box plot for each usefulness variable by the Likert scale value that ranges from 1 (extremely unlikely) to 7 (extremely likely). Note that according to the suggestion of Laitenberger et al. [8] the middle option 4 (neither) was omitted because this answer option would not give any information about the direction a participant leans to. The summative results range between 21 and 42 (ignoring the two outliers with the sums 6 and 7) with a mean and median of about 33. Given that the maximum possible rating is 42, the results suggest that the participants consider the definition language useful. Figure 5 gives a more detailed picture of the ratings. It shows that across all six variables (U1 - U6) the median rating is between 5 and 6. There are also some outliers visible that show ratings below the Likert score value of 3 (slightly unlikely). A more detailed analysis of these answers revealed that two participants rated all variables with 1 (extremely unlikely) or 2 (quite unlikely). All other participants' ratings were 3 (slightly unlikely) or above. In the final comments question one of the two participants mentioned problems with understanding the task description. The other participants provide unfortunately no final comments. Given that their responses differ considerably from the majority of participants, the problem with the task description is not considered a possible threat to all answers.

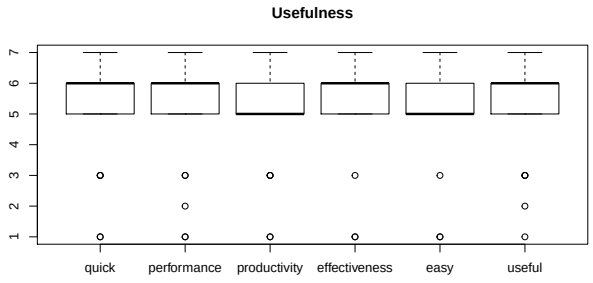

Fig. 5. Usefulness results.

D. Ease of use

The results for ease of use are summarized in Fig. 6. The summative results (excluding two outliers with a sum of 6 and 9) range between 15 and 42 (maximum score) with a median of 34 and a mean of 31. These findings suggest that the participants consider the presented solution easy to use. However, the widespread of the summative results indicate that the participants' opinion on this varies.

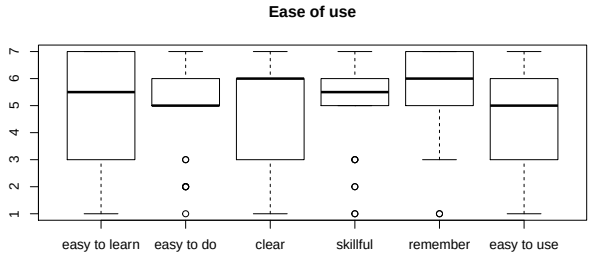

Fig. 6. Ease of use results.

The lowest rating were given easy to learn (E1), clear (E3), and easy to use (E6). Reasons for this could be the short introduction of the language by examples before to the experiment.

Nevertheless, the majority of participants (see Fig. 6) gave high ratings for the language to be easy to express experiments (E1), easy to become skillful in it (E4) and easy to remember how to perform tasks (E5). This seems to suggest that the domain model fits well with the domain concepts.

A more detailed analysis of the answers revealed that the ease of use ranking correlates with the participants' prior knowledge on the data interchange format JSON. Calculating the correlation coefficients (see Table III) revealed that there is a strong positive correlation between the participant's prior knowledge of JSON and the ease of use. Figure 7 shows the influence of the participants' prior knowledge of JSON on the ease of use rating. Participants with prior JSON knowledge rated every variable with a median of 6, whereas participants without prior knowledge of JSON rated with a median of 3. Also, the large spread of the ratings for participants without JSON knowledge could indicate that there is some unknown variable (maybe demographic) that could explain the data

TABLE III
PEARSON PRODUCT / SPEARMAN RANK ORDER CORRELATION COEFFICIENTS OF THE RESPECTIVE SUMMATIVE RESULTS.

	Usefulness	**Ease of use**	**Self-pred.**	**JSON**
Usefulness	1.00 / 1.00	0.23 / 0.30	0.60 / 0.56	0.43 / 0.30
Ease of use	0.23 / 0.30	1.00 / 1.00	0.62 / 0.68	0.56 / 0.57
Self-pred.	0.60 / 0.56	0.62 / 0.68	1.00 / 1.00	0.75 / 0.67
JSON	0.43 / 0.30	0.56 / 0.57	0.75 / 0.67	1.00 / 1.00

more. Thus, it seems that the selection of the host language (in our case JSON) has a strong influence on the user's perceived ease of use of the developed DSL. In the case of JSON the researchers could observe during the experiment that the concept of the brackets was difficult to some participants, despite the provided auto-completion.

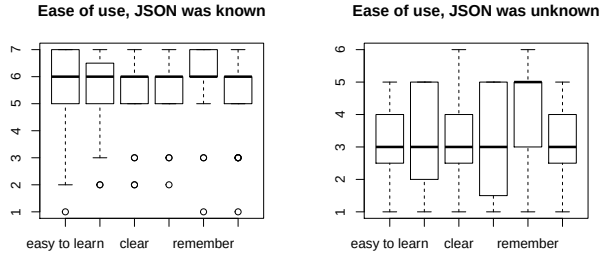

Fig. 7. Ease of use results by the participant's knowledge about JSON. The variables are from left to right: easy to learn (E1), easy to do (E2), clear (E3), skillful (E4), remember (E5) and easy to use (E6).

E. Correlations

The analysis of usefulness (see Section IV-C) and ease of use (see Section IV-D) showed that the participants seem to consider the language useful and most of them consider it easy to use too. Next we want to investigate the users' acceptance of the language by considering the correlations between the factors' usefulness, ease of use, and self-predicted future usage (see Fig. 3). In addition, we considered the influence of the participants' prior knowledge of JSON on those factors. Therefore, we want to investigate the correlation between the summative results of usefulness, ease of use, self-predicted future usage, and JSON knowledge. Table III shows the results of this analysis. It contains the Pearson product correlation coefficients together with the Spearman rank-order correlation coefficients between the summative results.

Usefulness is relatively low positively correlated with ease of use, which could mean that users that find the language useful not necessarily find it ease to use. This interpretation is supported by the observations made in Section IV-C and Section IV-D that although almost all find the language useful, not all participants find it easy to use.

Both factors usefulness and ease of use are similarly strong positively correlated with self-predicted future usage. One interpretation could be that both factors, the usefulness of the software and its ease of use are strong drivers for the participants to adapt the language. It seems that participants

TABLE IV
SIGNIFICANCE OF THE CORRELATIONS REPORTED BY THE *P*-VALUES (PEARSON PRODUCT MOMENT CORRELATIONS / SPEARMAN RANK ORDER CORELLATION COEFFICIENT). COMMON STAR NOTATION [15].

	U(sefulness)	**E(ase of use)**	**S(elf-pred.)**	**J(SON)**
U	-	ns/ns	***/**	*/ns
E	.202 / .102	-	***/***	**/***
S	<.001 / .001	<.001 / <.001	-	***/***
J	.015 / .105	.001 / <.001	<.001 / <.001	-

can find the language useful, but not easy to use (and vice versa) and still give a high rating for self-predicted future usage. Thus, usefulness could compensate faults in ease of use and vice versa. This would also explain the still high rating of self-predicted future usage, although some participants did not find the language easy to use.

An interesting fourth factor to include in the correlation analysis is the participants' prior knowledge of JSON. The analysis showed that there is a positive correlation between JSON knowledge, usefulness and ease of use. One explanation for this result could be that the usefulness of a language is more independent of the participants' prior knowledge of JSON than the ease of us. However, it also indicates the strong influence of a host language for an internal DSL project. The strong correlation between JSON and self-predicted future usage further supports this interpretation.

In addition to the calculation of the correlation coefficients, the significance of correlation among the four factors were analyzed (see Table IV). They indicate that the correlation between usefulness and ease of use is not significant, which is consistent with the results of King et al. [14]. Furthermore, all correlations to self-predicted future usage were statistically significant. Thus, it seems that these factors are indicators of the users' acceptance of the language. Moreover, it supports further the impact of the host language on the users' acceptance of an internal DSL.

As a result the analysis of the answers suggests that the participants consider the language useful and most of them easy to use too. Besides, most of the participants would use the language in the future and prefer it over other solutions.

F. Limitations

Even though possible threats to validity were considered during the design and execution of the study, the findings of this experiment have to be interpreted within their limitations. The main limitation of the study is the selection of students instead of practitioners as participants. This is in general considered to make results less generalizable. However, in the case of this experiment the particular students have considerable similarities with relevant industrial stakeholders, given their mixed background of computer science and business as well as their prior knowledge about experimentation in general and the domain of experimentation in particular.

V. CONCLUSIONS

For organizations that practice experimentation, the reliability of its experiments are of great importance, given the large potential impact their results can have. Therefore, the definition of an experiment as documentation and plan of action is an important artifact for reliable experimentation. Therefore, we proposed an experiment definition language based on recent research about the definition of experiments. Moreover, we conducted a technology acceptance study to validate the usefulness and ease of use of it. The results show that the majority of participants found the language useful and most found it easy to use too. Interesting future research directions are the development of a tool-chain for this language. These tools could apply static analysis on an experiment definition (e.g. answering whether for every variant a segmentation is properly defined), synchronize the definition with an experimentation platform, and even conduct some dynamic analysis like interference with other experiments on the experimentation platform.

REFERENCES

[1] R. Kohavi and R. Longbotham, "Online controlled experiments and a/b testing." *Encyclopedia of machine learning and data mining*, vol. 7, no. 8, pp. 922–929, 2017.

[2] F. Auer and M. Felderer, "Shifting quality assurance of machine learning algorithms to live systems," *Software Engineering und Software Management 2018*, 2018.

[3] M. Felderer, B. Russo, and F. Auer, "On testing data-intensive software systems," in *Security and Quality in Cyber-Physical Systems Engineering, With Forewords by Robert M. Lee and Tom Gilb*, S. Biffl, M. Eckhart, A. Lüder, and E. R. Weippl, Eds. Springer, 2019, pp. 129–148. [Online]. Available: https://doi.org/10.1007/978-3-030-25312-7_6

[4] A. Fabijan, P. Dmitriev, H. H. Olsson, and J. Bosch, "The online controlled experiment lifecycle," *IEEE Software*, 2018.

[5] F. Auer, C. S. Lee, and M. Felderer, "Characteristics of continuous experimentdefinitions: Results from a systematic literaturereview, a tool review and an expert survey," submitted to 46th EUROMICRO SEAA Conference, unpublished.

[6] F. D. Davis, R. P. Bagozzi, and P. R. Warshaw, "User acceptance of computer technology: a comparison of two theoretical models," *Management science*, vol. 35, no. 8, pp. 982–1003, 1989.

[7] M. Fishbein and I. Ajzen, "Belief, attitude, intention, and behavior: An introduction to theory and research," 1977.

[8] O. Laitenberger and H. M. Dreyer, "Evaluating the usefulness and the ease of use of a web-based inspection data collection tool," in *Proceedings Fifth International Software Metrics Symposium. Metrics (Cat. No. 98TB100262)*. IEEE, 1998, pp. 122–132.

[9] I. Steinmacher, T. Conte, C. Treude, and M. A. Gerosa, "Overcoming open source project entry barriers with a portal for newcomers," 05 2016, pp. 273–284.

[10] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, "Selecting empirical methods for software engineering research," in *Guide to Advanced Empirical Software Engineering*. Springer London, 2008, pp. 285–311.

[11] D. Falessi, N. Juristo, C. Wohlin, B. Turhan, J. Münch, A. Jedlitschka, and M. Oivo, "Empirical software engineering experts on the use of students and professionals in experiments," *Empirical Software Engineering*, vol. 23, no. 1, pp. 452–489, 2018.

[12] E. G. Carmines and R. A. Zeller, *Reliability and validity assessment*. Sage publications, 1979, vol. 17.

[13] J.-O. Kim, O. Ahtola, P. E. Spector, C. W. Mueller *et al.*, *Introduction to factor analysis: What it is and how to do it*. Sage, 1978, no. 13.

[14] W. R. King and J. He, "A meta-analysis of the technology acceptance model," *Information & management*, vol. 43, no. 6, pp. 740–755, 2006.

[15] J. O. Berger and J. Mortera, "Interpreting the stars in precise hypothesis testing," *International Statistical Review/Revue Internationale de Statistique*, pp. 337–353, 1991.