The Journal of Systems and Software 211 (2024) 111990

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Addressing combinatorial experiments and scarcity of subjects by provably orthogonal and crossover experimental designs[☆]

Fabio Massacci[a,b,*], Aurora Papotti[b], Ranindya Paramitha[a]

[a] University of Trento, Trento, I-38122, Italy
[b] Vrije Universiteit Amsterdam, Amsterdam, 1081 HV, The Netherlands

ARTICLE INFO

Keywords:
Design of experiments
Crossover experimental design
Full factorial design
Orthogonal design

ABSTRACT

Context: Experimentation in Software and Security Engineering is a common research practice, in particular with human subjects.
Problem: The combinatorial nature of software configurations and the difficulty of recruiting experienced subjects or running complex and expensive experiments make the use of full factorial experiments unfeasible to obtain statistically significant results.
Contribution: Provide comprehensive alternative Designs of Experiments (DoE) based on orthogonal designs or crossover designs that *provably* meet desired requirements such as balanced pair-wise configurations or balanced ordering of scenarios to mitigate bias or learning effects. We also discuss and formalize the statistical implications of these design choices, in particular for crossover designs.
Artifact: We made available the algorithmic construction of the design for $\ell = 2, 3, 4, 5$ levels for arbitrary K factors and illustrated their use with examples from security and software engineering research.

1. Introduction

Design of Experiments (DOE) is a common research practice in Science and Engineering, and laboratory randomized experiments, possibly with human subjects, are increasingly common practices in Software and Security Engineering (SSE). Designing the experiment, arranging subjects, and gathering data properly is often a challenge and an error-prone activity. Although perfection in experimentation is unattainable, varying degrees define the quality of the experiment, leading to better or worse designs (Juristo and Moreno, 2013). For example, the meta-analysis by Shepperd et al. (2014) over 42 studies to compare methods to predict fault-proneness (Shepperd et al., 2014) found that the choice of the intervention (i*n*that case a classifier) had an impact of only for 1.3% of the explanatory factor, while the research group accounted for 31%. In other words *"it matters more who does the work than what is done"*.

A key differences between biology and software engineering is the difficulty of performing realistic SSE Experiments (Sjoberg et al., 2002). SSE experiments feature a *combinatorial presence of many factors and experimental levels* due to the flexibility of software artifacts while the tested solutions have a comparative *scarcity of experimental subjects*.

While biologists can order online batches of bacteria or mice of certified genetic quality, SSE professionals are hard to recruit. MSc students may have the same proficiency as professionals when both face a task for the first time (Salman et al., 2015), which might be the case for experimenting SSE innovations, but experiments with them have ethical and time limitations (Chong et al., 2021; Naiakshina et al., 2017, 2018; Rong et al., 2012; Basak et al., 2023). Using recruiting platforms introduces significant quality concerns for SSE tasks. For example, a recent study by Tahaei and Vaniea (2022) who compared programming skills and secure development self-efficacy of subjects from a CS students mailing list and four crowdsourcing platforms (Appen, Clickworker, MTurk, and Prolific), found that 89% of CS students answered all programming skill questions correctly compared to 27% of crowdsourced subjects.

1.1. Main gap

Therefore, experiments with $N \geq 100$ or even $N \geq 50$ subjects are far from common in SSE and typically require multiple universities to join forces (for students) or deep pockets (for professionals). This makes the assignment of experimental subjects to the experimental configurations (Charness et al., 2012) a challenge for SSE Experimenters.

☆ Editor: Laurence Duchien.
* Corresponding author.Fabio Massacci at fabio.massacci@ieee.org
E-mail addresses: fabio.massacci@ieee.org (F. Massacci), a.papotti@vu.nl (A. Papotti), ranindya.paramitha@unitn.it (R. Paramitha).

https://doi.org/10.1016/j.jss.2024.111990
Received 10 August 2023; Received in revised form 22 January 2024; Accepted 4 February 2024
Available online 14 February 2024

Table 1
Subjects vs. Factors and levels in full-factorial design.

	Distributing N across K and ℓ					
	Levels (ℓ)	Factors (K)				
		1	2	3	4	5
$N = 50$	2	25	13	6	~~3~~	~~2~~
	3	17	6	~~2~~	~~1~~	~~0~~
	5	10	~~2~~	~~0~~	~~0~~	~~0~~
$N = 100$	2	50	25	13	6	~~3~~
	3	33	11	~~4~~	~~1~~	~~0~~
	5	20	~~4~~	~~1~~	~~0~~	~~0~~

This table shows how total subjects in full-factorial design with K variables (design factors), where each variable can take ℓ-levels, are distributed to each configuration, i.e. $N/(\ell^K)$. The more factors and levels one has, the lower the number of data points for each configuration drops up to a point where statistically significant results are close to impossible to get for many SSE Experimenters. A strike marks all cases when the number of subjects per configuration is below the classical rule of thumb of 10 or at the very least 5 per configuration (Agresti and Franklin, 2007).

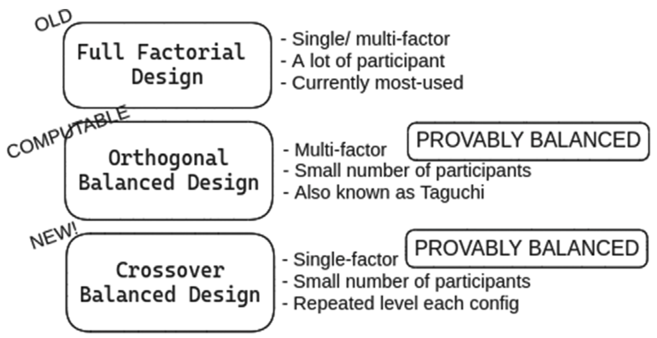

Fig. 1. Overview of experimentation designs in this work. While the full factorial design is currently the most-used design in software engineering/usable security research, this design is expensive and needs a lot of subjects. Considering the scarcity of subjects, orthogonal balanced designs (also known as Taguchi Kacker et al., 1991) can be used. To get even more data points in crossover-balanced experiments with a small number of subjects, we propose a repeated measure design.

In a nutshell, an SSE experimenter has (few) N experimental subjects and K variables that describe a configuration. If a variable can assume two or more values the number of configurations will climb to $O(2^K)$ and the number of subjects assigned to an individual configuration will drop to $O(N/2^K)$. To hope to get any statistically significant result, the SSE experimenter will forcibly choose K to be (very) small. Typical examples in reference texts (Wohlin et al., 2012; Juristo and Moreno, 2013) have one or two variables, each with less than a handful of possible values.

The numbers in Table 1 illustrate the problem in a clear way. The 101 statistics rules of thumb (Agresti and Franklin, 2007, pag. 434) of having around 10 subjects per condition, or at least 5 "successes" and 5 "failures" for the simplest χ^2-test to work, are essentially voided for most combinations. Achieving $n \gg 100$ is not easy for a SSE experiment.

Several solutions to tradeoff accuracy for feasibility exist in industrial DoE (Kacker et al., 2021) but are less well known in SSE. Further they do not provide easy-to-reference artifacts to be used by SSE researchers as designs are often presented in tables that do not allow for actual computations of the right configuration and automated assignments of subjects to configurations.

Further, given the "disposable" nature of experimental 'subjects' such as bacteria, crops, etc. those experimental designs do not consider the possibility of re-using, subjects in repeated experiments. In contrast, SSE experiments can build on the "re-usable" subjects who, once the hard job of recruiting them is done, and have analyzed a configuration, can also analyze a different configuration on a *different scenario* (e.g. code snippet, use-case scenario, source commit, etc.). This is a *crossover design*. The presence of different scenarios can make the system less vulnerable to the perils of learning effects (Vegas et al., 2015) provided the experimental configuration is appropriately designed.

1.2. Our contributions

The main contribution of this paper is to provide an alternative to traditional full-factorial designs used in SSE research that we summarize in Fig. 1

1. *Orthogonal Balanced Designs* We describe an algorithmic method to generate orthogonal designs for arbitrary K factors of $\ell = 2, 3, 4, 5, 7$ levels also known as Taguchi design (Kacker et al., 1991). While they are known in the general domain of DoE (Kacker et al., 2021) they are less frequently used in SSE and are always represented as tables. We present them with computational rules that can be automated and formally prove that our construction achieves the desired property.
2. *Crossover Balanced Designs* We propose a new methodology to compute crossover balanced design (Vegas et al., 2015) in the presence of different scenarios on which the same experiment can be repeated. Repeated designs are typically used to test an intervention before and after experimental configuration. Instead, we combine them with tests on different scenarios to test the original factors.
3. *Formal Proofs* We formally show that our design can balance the absolute number of exposures and even the order in which factors and levels are presented to subjects thus ensuring that learning effects cancel each other out. We also formally detail the impact of the designs on means and variances used for statistical tests.
4. *Artifact.* To boost plug-and-play applicability, we provide an Excel file that shows how to apply the proposed design methodology for K factors and ℓ levels so that experimenters in SSE can simply map our result to their experimental scenarios.
5. *Examples.* We illustrate our scenarios with examples from the SSE domain.

The next section (Section 2) provides the motivating examples of experiments in SE that we will use in the entire paper. We then introduce some preliminary terminologies and their corresponding notation in Section 3). We also introduce some terminology on DoE which gives an overview of the type of experimental designs that we analyzed in detail, and it introduces the experimental design contribution that we present in this work (Section 4). Section 5 discusses the experimental designs that have been used in the field of SSE and a nearby field such as Human Computer Interactions or Usable Security. Section 6 describes the full factorial experiment designs, showing their pros and cons; we do the same in Section 7 explaining the orthogonal balanced design. Then, in Section 8 we explain our experimental design for crossover balanced design. For Sections 6, 7, 8 we show how the configurations tables for the examples described in Section 2 change with the different types of experimental designs. We then elaborate on the statistical impact of the orthogonal and cross-over design consecutively in Sections 9 and 10. Before concluding our work with a summary of limitations and conclusions (Section 12), we describe our artifact (as one of our contributions) in Section 11.

2. Motivating examples

We present two different experimental goals that we use in the next sections as examples to show the difference among the different type of experimental designs, and how the number and the construction of configurations change among the different designs.

Example 1 (Finding Vulnerabilities)**. Goal**: The notion of slicing has been introduced for the first time by Weiser in 1979, and it is the process of extracting program parts based on some special criteria to improve further processing, and to facilitate debugging. We want to investigate whether slicing supports code comprehension. We would like to know if slicing helps code inspectors in identifying vulnerabilities in source code. **Intervention**: We first identify vulnerable lines in a code by means of expert judgment (outside the experiment). Then we use a slicing algorithm to identify a slice of the file in which the vulnerability is contained. Finally, we give to each subject either the original file (control) or the slice (intervention). **Measure of Success**: subjects are asked to identify the vulnerable lines of code.

Example 2 (Explainability of ML Findings)**. Goal**: we want to understand whether the ML methods used to find bugs in code are actually effective when used by developers in combination with some explanability techniques. **Intervention**: outside of the experiment we run an ML model on some code snippets. The ML model will return a binary output i.e. 1 for vulnerable code and 0 for not vulnerable code fragments. We run different explainability mechanisms on these code snippets (e.g. white-box, black-box, and control with no explanation whatsoever). These code snippets in which key tokens responsible for the decision are highlighted (or nothing is highlighted for the control condition) are given to the subjects. **Measure of Success**: subjects are asked whether they agree with the ML method if the suggested snippets are vulnerable or not.

3. Preliminary terminology and notation

We introduce first common terminologies in DOE.

- *Experimental subject*: the human participants in charge to solve a one or more tasks of an experiment. We index subjects as the nth and mth subjects, and by N we denote the total number of subjects available to the SSE experimenter.
- *Factor* and *Level*: A *factor* is a independent variable, which has several values called *levels* assigned by the experimenter (Barron, 1997). There are $K \geq 1$ factors. For simplicity, we assume that all factors have the same number of ℓ levels ranging from $0 \dots \ell - 1$. We denote by x_i or y_j the ith factor or the jth factor. The variable x_i is also called as a *full factorial variable*, i.e. a variable that the experimenter intend to explored for all possible combinations of values.

Example 1.1 (Finding Vulnerabilities)**.** A possible design of this experiment has $K = 3$ factors. The first could be the 'size of the analyzed file' with $\ell = 2$ levels: `full,slice`. Another factor could be the 'type of the vulnerability' with also two levels: `XSS, Path traversal`. The final factor could be the repository from where the codes fragments are sampled, again with two levels: `juliet`,[a] `tomcat`.

[a] https://samate.nist.gov/SARD/test-suites/111.

Remark 1. Some factors, called *design* factor, can be chosen by the experimenter. The experiment is then useful to determine the optimal configuration to be deployed on the field. Other factors, called *noise* factors, are encountered on the field. Still, to guarantee the robustness, it is important to test those factors as well, to understand what would happen if the corresponding level is actually encountered.

In the examples above, the vulnerability type is a noise factor whereas the presence of the slicing algorithm is a design choice. Indeed, the tool will not know in advance whether the vulnerability it is seeking to identify will be of a given type.

Example 2.1 (Explainability of ML Findings)**.** A possible experiment has $K = 2$ factor(s). The first factor is the 'used machine learning model' which has $\ell = 3$ Levels: `TNN, DL, GNN`. Another factor is the 'used explainability mechanism' which also has three levels: `control (no mechanism), black-box, and white-box`.

- *Experimental configuration*: An *experimental configuration* (also known as *treatment*) is the set of tasks that will be assigned to an experimental subject. We denote by $x_i[c]$ the value of the ith factor at cth experimental configuration. Depending on the type of design, a configuration might be a single task in which the variables denote the level of the corresponding factor at which the subject has to work or a set of tasks (i.e., they would be repeated measures, see Section 4) in which the variables identify the types of tasks that should be carried out.

Example 1.2 (Finding Vulnerabilities)**.** For finding vulnerabilities experiment with $K = 1$ factor and $\ell = 2$ levels, one group could have one `sliced` code with a `XSS` vulnerability while the second group could have one `full` code fragment with the same `XSS` vulnerability. The pairs of `Slice,XSS` and `Full,XSS` are two experimental configurations.

Example 2.2 (Explainability of ML Findings)**.** In this experiment with $K = 2$ factors and $\ell = 3$ levels, one group could have three tasks to perform: first assess some `control` code, i.e. unmarked, analyzed by a `TNN` algorithm, then one `black-box`-highlighted code analyzed by a `DL` algorithm and a third `blackbox`-higlighted code analyzed by `GNN`. The sequence of 3 tasks `Control,TNN; Black,DL; Black,GNN` would be one possible experimental configuration. There are 9 possible pairs (ℓ^k) that can be combined in all possible sequences of three different tasks for a total of $9 * 8 * 7 = 504$ different configurations.

- *Scenario*: In one experimental configuration, there can be several *scenarios* (also known as case studies), in which one level of one/several factors is applied. We index scenarios with the variables s and t. For SSE experiments (including the examples used in this paper) those could be code fragments with different features.

Example 1.3 (Finding Vulnerabilities)**.** Several different code snippets each containing a `XSS` vulnerability can be used.

Remark 2. To avoid learning effects the scenarios have to be sufficiently different (see Section 10).

Depending on the design choice, what is a scenario in an experiment can be a factor in another experiment and viceversa.

Example 2.3 (Explainability of ML Findings)**.** One scenario could be a code fragment containing a XSS vulnerability, and another one could be a code fragment containing a Path Traversal Vulnerability.

- *Repetition*: this happens when the experimenter decides to repeat one factor (with the same or different levels) with different scenarios in one experimental configuration. A factor (with the same or different levels) can be repeated *repeated* r times.

Example 1.4 (Finding Vulnerability)**.** For this experiment, one can design an experimental configuration with 4 scenarios, 2 are code `slices` and the other 2 are `full` codes. In this case, there is a repetition of the 'size of the analyzed code' factor with the same and different levels.

Example 2.4 (Explainability of ML Findings)**.** For this experiment, one can design an experimental configuration in which a subject has to analyze 6 code fragments as scenarios, 2 are not highlighted (`control`), 2 are highlighted with a `blackbox` mechanism, and the other 2 with a `whitebox` mechanism. In this case, there is a repetition of the 'used explainability mechanism' factor with the same and different levels.

- *Outcome* variables: while a factor is a controlled independent variable, an outcome or dependent variables are the variables that are measured to determine the success of an intervention as we change the level of one or more factors. We denote by $o_s[n]$ the value of the outcome variable measuring the performance of the nth subject on the sth scenario.

Example 1.5 (Finding Vulnerability)**.** An example of dependent variables that can be measured in this experiment is the number of false positives identified in a code fragment by an experimental subject.

Example 2.5 (Explainability of ML Findings)**.** An example of dependent variables that one can measure in this experiment is how many vulnerable lines an experimental subject correctly identifies in one code fragment.

- *Data point*: one *data point* is one dependent variable measured on one scenario done by one experimental subject. Therefore, if each subject has been assigned one experimental configuration with 6 scenarios, each subject will produce 6 data points. The total data points for the entire experiment is the number of experimental subjects times the number of scenarios, e.g. in this case $6N$.

4. Terminology on DoE

4.1. Assigning subjects to configurations

In SSE experiments where the experimental units are subjects, assigning subjects to configurations is the first design choice to make. There are three ways of assigning subjects to the different levels of the factor (experimental configurations) (Charness et al., 2012).

1. *Independent measures (either parallel or between-subjects).* Different subjects are used in each condition of the independent variable, which means that each subject is assigned to only one experimental configuration. The allocation should happen randomly, ensuring that each subject has an equal chance of being assigned to one group.
2. *Repeated measures (or within-subjects) designs.* Each subject is asked to repeat a measurement r-times in time. This is typically used to measure some metrics before and after a single intervention (i.e. $K = 1$, $\ell = r = 2$). In alternative, they participate in each condition of the independent variables, i.e. each subject is assigned to all interventions.
3. *Matched-pairs designs.* This design is used when an experiment has a factor with only two experimental configurations ($K = 1$, $\ell = 2$), and the subjects are grouped together into pairs based on some variables (e.g. age or gender), and each pair is randomly split into to a different group.

In Table 2 we summarize the pros and the cons for each different experimental design. Given the difficulty of recruiting a large number of subjects in SE, matched pairs designs have a significant risk of having an unacceptable number of datapoints. Therefore, they are not further considered in this paper.

4.2. Assembling factors and levels

The second issue is the actual procedure for designing a robust experimental design. In this case we have two types of techniques:

1. *Full or fractional factorial designs*. This design is the classical design in DoE and the most popular in SE. If we consider to have K factors, and ℓ levels, with this design we test all the possible combinations, which are in total $O(\ell^K)$ configurations in which a subject performs a single task (independent measures). Even if this design potentially reveals all the combinations, it is time-consuming, and it requires many subjects.
2. *Orthogonal design*. It is also known as Taguchi design (Kacker et al., 1991). This design allows us to get trade-off interactions between factors with a decreasing number of configurations and an increasing number of factors. If we consider to have K factors, and ℓ levels, for every pair of factors each level of one factor is paired with all levels of the other factors. The number of experiments is an order of $O(K \cdot \ell)$. It is the smallest balanced design, and the columns of the matrix describing the configuration have the property that in every pair of columns, each of the possible ordered pairs of elements appears the same number of times.
3. *Crossover design.* It is a special form of Repeated measure design where each subject participates in each condition of the independent variables in a possibly random sequence, i.e. $\ell = r$. This is quite frequent in SSE but might be vulnerable to learning effects (see here in Section 10 or Vegas et al. (2015)). To counter this problem, the same subject is asked to validate different scenarios.

Full factorial designs are used when the accuracy of the result is critical, and the cost (in terms of enough time to run all the trials) is not an issue. As the number of configurations grows exponentially by increasing the number of factors and levels and it makes it hard in some cases, if not impossible, to retrieve enough subjects to run the experiment, and it takes too much time to run all the trials.

Orthogonal design, also called Taguchi designs has a smaller number of balanced runs at the price of not measuring complex interactions. This design is ideal when the number of factors and levels is too big for a full factorial and complex multi-factor interaction are unlikely.

Crossover designs can be used with a repeated measure on different levels of the factor on a slightly different scenario (or case study) when there is some evidence that the different nature of the scenario will limit the presence of a learning effect. In this case, we typically have $\ell = r$. Our proposed design will have $r = 2\ell$. However, not all possible crossover designs are immune from learning effects just because they use different scenarios. One cannot use Taguchi arrays directly for crossover designs as one of the assumptions for using it is that the factors are independent.

Table 2
Pros and cons of different assignments.

Design	*Pros*	*Cons*
Independent measures	Prevents the presence of order effects (such as practice or fatigue) by having the subjects engage in just one condition.	High number of subjects is required, and the differences between the subjects (e.g. variations in age, gender, or social background) may affect the results.
Repeated measures	Fewer people are needed as they participate in all conditions. They are typically measured after a single intervention.	There might order effects aka learning effect (as the subjects by the time of the second condition know what to do), or the result might get worse because they are tired.
Matched pairs	There are no order effects. We control lurking variables by pairing the subjects with a specific variable eliminating the effects that the variables could have on the result.	If one subject of the pair decides to drop out, those two data points are lost at once since also the twin must be removed. It can be time-consuming to match the subjects on certain variables.

5. DoE in software and security engineering

Classical books on experimentation on SE (Wohlin et al., 2012; Juristo and Moreno, 2013) typically reports factors with two levels and not multiple levels. Yet more complex DoE are increasingly becoming the norm especially outside SE.

General DoE in computer science and industry. Taguchi (Kacker et al., 1991; Roy, 2010) provides a balanced design, but the number of experiment configurations is not small. This means we need more experimental subjects, which scarcity in the field is high and makes the experiment expensive. However, the implementation of Taguchi, cross-sectional design, or even Design of Experiments (DoE) in general in the field is still rare, as mentioned in several surveys (Antony et al., 2021; Tanco et al., 2008).

On Human-Computer Interaction, Aoyama and Yokoyama (2017) used Taguchi's method to design aesthetic product functions concerning human emotion. Other than for getting a robust design on product functions, Taguchi's method has also been used to optimize heuristic design on the evaluation of human–computer interaction (Ling and Salvendy, 2007).

Several works (Seo et al., 2021; Lee et al., 2022) have used Taguchi's design to design robots, mostly to tune the design parameter Seo et al. (2021) used Taguchi's design and orthogonal arrays to find the best combination of design parameters for the robot's screw wheels. They use 4 factors, each with 3 levels. Lee et al. (2022) also used Taguchi's orthogonal design to tune the design parameters for their Angled Spoke-based Wheels (ASW). They also have 4 factors with 3 levels each. On the other hand, Chew and Nakamura (2023) use a modified orthogonal design to teach their robot. They used the design in their experiment to define the optimal experiment condition for a robot to learn strategies for facilitating a multi-party interaction. They used 4 factors, 1 with 4 levels, and the other 3 each with 2 levels.

Challenge. One of the challenges experimenters faced in computer science/software engineering experimentations was the lack of subjects. Some studies (Naiakshina et al., 2018; Papotti et al., 2022) are using students as a proxy to practitioners, as students tend to be easier to recruit and cost less. This approach has been validated by several studies (Falessi et al., 2018; Naiakshina et al., 2020) that show that students are good proxies for real-world developers. Several studies (Serafini et al., 2023; Rainer and Wohlin, 2022) suggested some designs/frameworks to recruit subjects for software field study. Even though there has been some effort to get more subjects, in the field, the small pool of subjects or highly-cost subjects is still an inevitable challenge. Therefore, to make the best out of this small number of subjects, experimenters need to have a good design with less configuration but as many data points as possible.

Software engineering. In the context of SE, Taguchi designs are not frequently used. Vescan et al. (2021) use Taguchi design in their experiment on feature modeling using genetic programming. Gümüş et al. (2023) compared Taguchi design with the F-Race method and Steady-State Memetic Algorithm (SSMA) in parameter tuning. Still about tuning parameters, in several machine learning studies (Tyasnurita et al., 2017; Dell'Amico et al., 2018; Atta et al., 2022; Gargiulo et al., 2021), Taguchi's orthogonal design is used to tune the hyperparameters of the machine learning model. Some others (McCormick, 2022; Cicirello and Giunta, 2022) use Taguchi to design an experimental dataset to train their machine-learning models. Taguchi design has also been used to assess the performance of expensive-to-evaluate system (Solouma and El Berry, 2022; Cicirello and Giunta, 2022). Kanchana and Sarma (1999) use Taguchi to optimize the design for improving software quality.

Crossover designs are more popular and used in SSE as stated in Wohlin et al. (2012). However, their use is ad hoc and no formal guarantee is provided for the chosen configurations. Vegas et al. (2015) perform a SSE literature review of crossover experiments. The objective of the study is dual: (1) provide a picture of the practice of crossover experiments, (2) and provide mechanisms for SSE researchers to run valid and reliable crossover experiments. As result they show that crossover designs and between-subjects experiments are the most used: they appear in 47.5 and 41.5 percent of the papers, respectively, and are used in 54.8 and 30.6 percent of the experiments, respectively. Instead, repeated measures, and matched-pairs designs are less common.

Even if experiment reporting guidelines (Jedlitschka et al., 2008) requires to specify information for rating the quality of the experiment, in some cases (especially conference papers) omitted information regarding the experimental designs or analysis (Vegas et al., 2015). Moreover, some authors describe the designs without any support in tabular form, and there is also confusion with respect to what a crossover experiment is among researchers. Some papers named the experiment designs either incorrectly, or not specific enough.

Another aspect in crossover experiments that is relevant is the correlation between the outcome measures on the same subjects (Kitchenham et al., 2021) As the goal of crossover experiments is to compare software engineering techniques that must be undertaken by human subjects, and the individual skill differences are often emphasized by the software engineering theory, we would expect a relatively high value of correlation. However, most of the studies do not report the correlation value. Kitchenham et al. (2021) analyzed 35 crossover experiments, and they observed that the correlation value was small, which it means that a subject that performs well on a specific technique does not imply that performs well on another one. Low values of correlation threat the validity of the crossover designs for software engineering experiments. The cause of this result can be due training limitations, therefore it is important to invest in intensive training.

Vegas et al. (2015) conclude that despite crossover designs have been defined complex and criticized both in SE (Kitchenham et al., 2003) and other disciplines (Fleiss, 1989; Freeman, 1989; Grieve, 1985; Grizzle, 1965; Jones and Kenward, 2014), they state they can yield to valid results if the experiments are addressed and designed properly.

Usable security. In the security community, full-factorial designs are also commonly used (Cummings et al., 2021; Lee et al., 2021; Sebastio et al., 2020). For example, Cummings et al. (2021) assigned 1 scenario (out of 2) and 1 condition (out of 7, total 2 × 7 levels) to each subject. In the usable privacy and security community, a subfield with a stronger focus on human experiments, the common practice is also to use full-factorial design. Single-factored between-subject experiments divided their subjects into multiple experimental configurations e.g. two (McCall et al., 2023; Volkamer et al., 2022; Langer et al., 2022), three (Liu et al., 2023; Kühtreiber et al., 2022; Zheng and Becker, 2022; Zibaei et al., 2022), or even more like Zibaei et al. (2023) with six and Kühtreiber et al. (2022) in their second experiment with 11. Each subject only works on 1 configuration (1 factor with multiple levels). With this kind of setup, we do not have any learning effects, but the more levels we have, the smaller the subject group for each level will be. This means we will have less data for each level, while in most cases, we want the contrary: get more data.

Multi-factored between-subjects experiments (Amador et al., 2023; Rader, 2023; Huaman et al., 2022) usually need more subjects as the number of configuration grow quickly as the number of level increases. To reduce the number of needed subjects, some studies use a subset of full-factorial design, e.g. Kersten et al. (2023) which has 10 configurations with 20 levels, each level is repeated twice in the whole experiment, not on each configuration.

Within-subject experiments (Kaushik et al., 2023; Yoshikawa et al., 2022; Mayer et al., 2022; Whalen et al., 2022; Malkin et al., 2022; Chen et al., 2022) require each subject to do all different experimental configurations (multi factors with 1 level each). This is common for surveys or interviews in which all subjects have to go through the same set of tasks/questions. Some studies also have repetitions of configurations (Chen et al., 2022), e.g. Kaushik et al. (2023) have 3 factors but each subject has to do each factor twice, while in the study by Mayer et al. (2022) only the first factor out of 5 is repeated (total tasks = 4 times the first factor + once the rest = 8 tasks per subject). While we need less number of subjects to analyze all factors the experiment is not balanced and bias or learning effects might be introduced. However, the subject's task can become too heavy and a large number of subjects still determines the total datapoints we get in the end.

6. Full factorial design

This is the traditional type of experiment used in SSE research. To define an experiment we need to define the variable of interest which we call *factors* and the corresponding values that they can be assigned which we call *levels*.

In a full factorial design with K factors all possible levels are combined with all possible factors. The number of configurations is therefore $c \in \{0, \dots, \ell^K - 1\}$. The index i runs across all possible factors ($1 \le i \le K$). Each factor x_i can then be represented by a vector with ℓ^K elements, one for each configuration cth, which stores the level of the factor for that particular configuration. We can formally capture this combination as follows:

$$x_1[c] = c \mod \ell \tag{1}$$

$$x_2[c] = \lfloor c/\ell \rfloor \mod \ell \tag{2}$$

$$\vdots$$

$$x_i[c] = \left\lfloor \frac{c}{\ell^{i-1}} \right\rfloor \mod \ell \tag{3}$$

When the accuracy of the result is critical, the full factorial design has the advantage of (potentially) revealing all interactions and test all the possibilities.

However, the cost of run this experimental design is extremely high because the number of experimental configurations increases exponentially with the number of factors and levels. If we have N *experimental subjects* this means that every possible configuration can only be tested by N/ℓ^K. If we consider the rule of thumb to have at least 16 elements for each possible configuration we have the following requirements. Table 1 shows some examples of the number of data points per level and factor we would get with 50 and 100 subjects. In SSE, it is often practically impossible to have enough subjects to populate all possible configurations.

We provide below a possible realization of Example 1 (Finding Vulnerabilities) and Example 2 (Explainability of ML Findings) with the full factorial design (Example 1.7 and 2.7 respectively).

Example 1.6. Finding Vulnerabilities - Full Factorial which is a possible realization of Example 1 with $K = 2, \ell = 2$.
Instance. The $K = 2$ factors will be the intervention and the type of vulnerability. For the intervention, we will have $\ell = 2$ levels (the original full file, $x_1 = \texttt{Full}$, or the sliced fragment of the file, $x_1 = \texttt{Slice}$). For the type of vulnerability, we will also have two $\ell = 2$ levels (Cross Site Scripting, $x_2 = \texttt{XSS}$, or Path Traversal vulnerabilities, $x_2 = \texttt{Path}$).
Example Configurations Therefore an example of a configuration is $\{\texttt{File}, \texttt{Path}\}$ when $x_1 = 0$ and $x_2 = 1$. Another example with $x_1 = 1$ and $x_2 = 0$ will make $\{\texttt{Slice}, \texttt{XSS}\}$ configuration.
Task of subjects Each subject will be exposed to a single vulnerable file.

The full factorial table would be the one below:

c	x_1	x_2
0	0	0
1	1	0
2	0	1
3	1	1

=

c	Code	Vuln
0	Full	XSS
1	Slice	XSS
2	Full	Path
3	Slice	Path

From Table 1 Experiment 1.6 looks a solid experiment. With $n = 100$ subjects we will have around 25 subjects per configuration which is comfortably higher than the rule of thumb of 10 and the danger threshold of 5.

Example 1.7 (Finding Vulnerabilities - Full Factorial). Boosted by the success we try more factors $K = 4, \ell = 2$.
Scaling up. For example the sources of the vulnerability (whether it comes from a synthetic dataset – `juliet` or from a real project – `tomcat`) or some changes we made to the presentation (e.g. we provide an IDE with highlighted the lines changed by the most recent commit – `commit`, or just the file for inspection – `as-is`) we would have a much bigger table. The last column of the Table below also reports the possible outcome of an experiment (for example the mean number of vulnerable lines across all possible subjects having that configuration).

c	Code	Vuln	Project	IDE	Outcome
0	Full	XSS	juliet	as-is	10
1	Slice	XSS	juliet	as-is	11
2	Full	Path	juliet	as-is	9
3	Slice	Path	juliet	as-is	10
4	Full	XSS	tomcat	as-is	11
5	Slice	XSS	tomcat	as-is	9
6	Full	Path	tomcat	as-is	10
7	Slice	Path	tomcat	as-is	1
8	Full	XSS	juliet	commit	11
9	Slice	XSS	juliet	commit	10
10	Full	Path	juliet	commit	10
11	Slice	Path	juliet	commit	9
12	Full	XSS	tomcat	commit	11
13	Slice	XSS	tomcat	commit	9
14	Full	Path	tomcat	commit	10
15	Slice	Path	tomcat	commit	1

A quick look at Table 2 reveals this is doomed plan for SSE researchers: with $N = 100$ subjects we will have around 6 subjects per configuration and be dangerously close to the threshold of 5. With $N = 50$ we would only have three subjects per configuration. Too few to have meaningful results. To reach close to 25 subject we would need $N > 400$. This is a very large number for any task that is not simply answering a survey. At a cost of 75Euro per hour to pay a professional developer, it would require more than 25KEuro.

Example 2.6. Explainability of ML Findings - Full Factorial which is a possible realization of Example 2) with $K = 2, \ell = 3$.
Instance The $K = 2$ factors will be the models and the type of explainability. For the models, we will have $\ell = 3$ level (a token-based Neural Network model – `TNN`, a Graph-based Neural Network – `GNN`, and a Deep Learning Model – `DL`. For the explainability method, we will have $\ell = 3$ levels which are `Control`, `Blackbox`, and `Whitebox` explainability methods.
Example Configurations Therefore an example of a configuration is {`DL`, `Black`} when $x_1 = 2$ and $x_2 = 1$. Another example with $x_1 = 0$ and $x_2 = 2$ will make {`TNN`, `White`} configuration.
Task of subjects Each subject will be exposed to a single vulnerable file.
The full factorial design would be

c	x_1	x_2
0	0	0
1	1	0
2	2	0
3	0	1
4	1	1
5	2	1
6	0	2
7	1	2
8	2	2

=

c	Model	Explain.
0	TNN	Control
1	GNN	Control
2	DL	Control
3	TNN	Black
4	GNN	Black
5	DL	Black
6	TNN	White
7	GNN	White
8	DL	White

This looks like a borderline experiment with $N = 100$ subjects we will have around 12–13 subjects per configuration which is slightly higher than the rule of thumb of 10 and the danger threshold of 5.

Example 2.7 (Explainability of ML Findings - full factorial). We explore possibility of adding more factors with $K = 3, \ell = 3$.
Scaling up. If we have 1 more factor e.g. type of vulnerabilities with $\ell = 3$ levels: `XSS`, SQL Injection (`SQL`), and `DoS` we will have a much bigger table as below. Once, again, the last column of the Table below reports the outcome of a possible experiment (for example the mean).

c	Model	Explain.	Vuln	Outcome
0	TNN	Control	XSS	7
1	GNN	Control	XSS	6
2	DL	Control	XSS	8
3	TNN	Black	XSS	10
4	GNN	Black	XSS	7
5	DL	Black	XSS	7
6	TNN	White	XSS	9
7	GNN	White	XSS	8
8	DL	White	XSS	6
9	TNN	Control	SQL	7
10	GNN	Control	SQL	6
11	DL	Control	SQL	8
12	TNN	Black	SQL	8
13	GNN	Black	SQL	7
14	DL	Black	SQL	7
15	TNN	White	SQL	9
16	GNN	White	SQL	8
17	DL	White	SQL	6
18	TNN	Control	DoS	7
19	GNN	Control	DoS	6
20	DL	Control	DoS	8
21	TNN	Black	DoS	8
22	GNN	Black	DoS	7
23	DL	Black	DoS	7
24	TNN	White	DoS	9
25	GNN	White	DoS	8
26	DL	White	DoS	6

Once again, this looks like an experiment that is close to impossible: with $N = 100$ subjects we will have only 3–4 subjects per configuration. To match the rule of thumb (at least 10 per cell) we would need more than $N = 270$ subjects which considering the scarcity of subjects the experiment would be very expensive. To reach 25 per configuration we would need more than $N > 600$, which is possible if you work for a multinational corporation and your fellow colleagues feel obliged to answer.

Summary of full factorial design. The available N subjects will be divided among the configurations of K factors and ℓ levels as follows

- *first factor*: ℓ configurations
- *second factor*: ℓ^2 configurations
- *additional $K-2$ factors*: ℓ^K configurations

7. Orthogonal balanced design

If we do not have that many subjects a full factorial design is nearly impossible.

7.1. Intuition

As a first step, we can forget all possible interactions for a lower number of configurations while keeping the comparison of pairwise comparison between factors. They are also known as Taguchi design (Kacker et al., 1991).

The key intuition is that by suitably combining factors and levels we can make sure that every possible value of a factor is combined with every possible value of another factor. So we have a *pairwise* factorial. What we cannot guarantee is that all possible combinations of the first two factors are compared with all possible levels of the third factor and so on.

So if we have two factors for which we are interested to get a full factorial combination x_1 and x_2 we can add $\ell - 1$ additional factors without changing the numbers of possible configurations. The levels of the first two factors are determined by the full factorial combination as in Eq. (3). So we have $c \in \{0, \dots, \ell^2 - 1\}$.

Once we add one more independent factor x_3 we can also add $2 \cdot (\ell - 1)$ factors without adding more experimental configurations. In this set-up we have $c \in \{0, \dots, \ell^3 - 1\}$ levels.

In the general case, each time we add a new full factorial variable x_i we can add $(i-1)(\ell-1)$ factors y_j which will not vary in all possible ways but only in some 'clever' ways to make sure that all possible combination is present. This is obtained by taking the modulo addition of all pairs of factorial variables for all possible values of the levels.

In summary is we have i full factorial variables $x_1, \dots, x_i$ then we can have $(i-1)(\ell-1)$ computed factors y_j for $j \in \{1 \dots (i-1)(\ell-1)\}$. However, the number of all possible combinations that we have to consider is only ℓ^i and not $\ell^{i+(i-1)(\ell-1)}$ which would be way beyond what can be possibly computed with limited resources.

7.2. Two full factorial variables

The simplest case is when we have only two variables for which we would like (or can afford) to explore all full factorial combinations. Since we have only two variables we have that for $c = \{0, \dots, \ell^2 - 1\}$ configurations.

$$x_1[c] = c \mod \ell \qquad \text{as in Eq. (1)}$$
$$x_2[c] = \lfloor c/\ell \rfloor \mod \ell \qquad \text{as in Eq. (2)}$$
$$y_j[c] = (x_1[c] + j \cdot x_2[c]) \mod \ell \quad \text{for } j \in \{1, \dots \ell - 1\} \tag{4}$$

In this way, we can test $K = 2 + (\ell - 1)$ factors for pairwise interaction of the levels by only testing ℓ^2 configurations rather than $\ell^{2+(\ell-1)}$ configurations as we would have needed to do with a full factorial design.

The intuition behind this construction is that for ℓ prime, e.g. $\ell = \{2, 3\}$, the modulo operation generates a Galois Field where both addition and multiplication are invertible and each combination generates the complete set of values. For $\ell = 4$ one needs to change multiplication and addition to behave as in a Galois Field. Table A.8 in the Appendix shows the correct values to use for multiplication and addition 'modulo' $\ell = 4$. We also discuss additional values in Appendix A.

7.3. Three or more full factorial variables

For three variables, the equations for the previous configurations remain unchanged and c must simply be applied with the new range.

$$x_3[c] = \left\lfloor \frac{c}{\ell^2} \right\rfloor \mod \ell \qquad \text{for } i = 3 \text{ according to (3)}$$
$$y_j[c] = (x_1[c] + j \cdot x_3[c]) \mod \ell \text{ for } j \in \{1 \dots \ell - 1\} \tag{5}$$
$$y_{j+\ell-1}[c] = (x_2[c] + j \cdot x_3[c]) \mod \ell \text{ for } j \in \{1 \dots \ell - 1\} \tag{6}$$

In this way, we can test $K = 3 + 2(\ell - 1) = 2\ell - 1$ factors for pairwise interaction of the levels by only testing ℓ^3 levels, rather than $\ell^{3+2(\ell-1)} = \ell^{2\ell+1}$ as we would have needed to do with a full factorial design.

In the general case, each time we add a new variable x_i for which we are willing to explore the full factorial combination with the previous full factorial variables we can add the combination of the previous x_j-variables for $j = 1 \dots i - 1$ for each of the ℓ levels.

$$x_i[c] = \left\lfloor \frac{c}{\ell^{i-1}} \right\rfloor \mod \ell \qquad \text{as in (3)}$$
$$y_{j' \cdot j}[c] = (x_{j'}[c] + j \cdot x_i[c]) \mathrm{mod}\ \ell \quad \text{for } j' \in \{1 \dots i - 1\} \quad \text{for } j \in \{1 \dots \ell - 1\} \tag{7}$$

Theorem 1. *In a modulo-ℓ experimental design, every level of a factor is compared to every other level of another factor.*

Proof. We prove this by contradiction for the case with $2 + (\ell - 1)$ factors, i.e. with two full factorial factors. For additional factors, the proof is essentially identical.

Assume that there are two factors z and z' such that some value $v \in z$ is not paired with all possible values $v' \in z$. Since we have ℓ^2 configurations corresponding to all possible pairs, this means that the same level v of z is compared at least twice with some level v^* of z', once in a configuration c and once in a configuration c^*. The proof works by cases:

1. Assume first that $z \equiv x_1$ and $z' \equiv x_2$, this is impossible because by construction this pair is organized as a full factorial design without repetitions.
2. Suppose then $z \equiv x_1$ and $z' \equiv y_j$ for some $j \in \{1, \dots, \ell - 1\}$. By replacing the values in Eq. (4) generating y_j for one of the configuration c, we have $v^* = v + j \cdot x_2[c] \mod \ell$. Similarly for c^* we would have $v^* = v + j \cdot x_2[c^*] \mod \ell$ However, operations in ℓ are designed to be a Galois field so all linear equations have a unique solution. So there cannot be two values of x_2, one for c and one for c^* generating the same value of v^* unless $j = 0$ which cannot happen by construction.
3. The reasoning for the case where $z \equiv x_2$ and $z' \equiv y_j$ is symmetrical to the case above.
4. Suppose that $z \equiv y_j$ and $z' \equiv y_{j'}$ for some j and j' different from each other. Since there is a repetition we have $y_j[c] = v$ and $y_j[c^*] = v$. By replacing the value v for the configuration c in the equation generating $y_j[c]$, we have $v = x_1[c] + j \cdot x_2[c] \mod \ell$. Since this is a Galois field we can invert the equation and obtain $x_1 = v - j \cdot x_2[c] \mod \ell$. We can then replace it into $y_{j'}[c]$ obtaining $v^* = (v - j' \cdot x_2[c] \mod \ell) + j \cdot x_2[c] \mod \ell$. This value simplifies to $v + ((j - j') \mod \ell) \cdot x_2[c] \mod \ell$. The same equation has to hold for $x_2[c^*]$. Since this is a Galois field the only way for the solution x_2 not to be unique would be $0 = (j - j') \mod \ell$ which is only possible for $j = j'$ which is against the hypothesis.

The cases for x_3 are simply a permutation of the above argument with x_1, x_2 and x_3. □

We provide below a possible realization of Example 1 (Finding Vulnerabilities) and Example 2 (Explainability of ML Findings) with the orthogonal balanced design (Example 1.8 and 2.8 respectively).

Example 1.8. Finding Vulnerabilities - Orthogonal Balanced Design is a possible realization of Example 1, which has the factors of Example 1.7 but way less configurations.
Instance. As in the full factorial design Example 1.7 we have $K = 4, \ell = 2$. The first two factors {`Full`, `Slice`} and {`XSS`, `Path`} are the same as in Example 1.6 and we start with them the full factorial combination. Besides the slicing intervention and the type of vulnerability we also consider the sources of the vulnerability (`juliet`,`tomcat`)and the changes we made to the presentation (`as-is`,`commit`).
Example Configurations Also the configurations are the same for Example 1.7 Therefore an example of a configuration is {`Full`, `Path`, `juliet`, `as-is`} when $x_1 = 0$, $x_2 = 1$, $x_3 = 0$, $y_1 = 0$. Another example with $x_0 = 1$, $x_1 = 0$, $x_3 = 1$, $y_1 = 1$ will make {`Slice`, `XSS`, `tomcat`, `commit`} configuration.
Task of subjects Each subject will be exposed to a single vulnerable file.
Scaling up In contrast to the full factorial design, we have a number of rows that is half the number of rows for the full factorial table

c	x_1	x_2	x_3	y_1	Outcome
0	0	0	0	0	10
1	1	0	0	1	10
2	0	1	0	1	10
3	1	1	0	0	10
4	0	0	1	0	11
5	1	0	1	1	9
6	0	1	1	1	10
7	1	1	1	0	1

=

c	Code	Vuln	Project	IDE	Outcome
0	Full	XSS	juliet	as-is	10

1	Slice	XSS	juliet	commit	10
2	Full	Path	juliet	commit	10
3	Slice	Path	juliet	as-is	10
4	Full	XSS	tomcat	as-is	11
5	Slice	XSS	tomcat	commit	9
6	Full	Path	tomcat	commit	10
7	Slice	Path	tomcat	as-is	1

For this particular design, the configuration $c = 0$ corresponds to the configuration $c = 0$ of the full factorial design in Example 1.7. However, this does not happen for every configuration. For example, configuration $c = 2$ from this example, corresponds to the configuration $c = 10$ from the full factorial Example 1.7. Not all the configurations from Example 1.7 have corresponding entries because the whole purpose of an orthogonal design is to test *fewer* configurations that the full factorial. For example configuration 2 and configuration 15 are missing.

This looks like a doable experiment. With $N = 100$ subjects we will have around 12 subjects per configuration which is slightly higher than the rule of thumb of 10 and the danger threshold of 5.

Example 2.8. Explainability of ML Findings - Orthogonal Balanced Design which is another possible realization of Example 2 with the same number of factors and levels $K = 3, \ell = 3$ as in Example 2.7 but less number of configurations compared to the full factorial table.
Instance. The factors would be the same Example 2.7. The first two factors $\{\texttt{TNN}, \texttt{GNN}, \texttt{DL}\}$ and $\{\texttt{Control}, \texttt{Black}, \texttt{White}\}$ are the same as in Example 2.7 and we start with them the full factorial combination. Besides the type of neural network model and the type of explainability we also consider the type of the vulnerability $\{\texttt{XSS}, \texttt{SQL}, \texttt{DoS}\}$.
Example Configurations The configurations are a subset of those of Example 2.7. Therefore an example of a configuration is $\{\texttt{TNN}, \texttt{Black}, \texttt{SQL}\}$ when $x_1 = 0$, $x_2 = 1$ and $y_1 = 1$. Another example with $x_1 = 1$, $x_2 = 2$, and $y_1 = 2$ will make $\{\texttt{GNN}, \texttt{White}, \texttt{XSS}\}$ configuration.
Task of subjects Each subject will be exposed to a single vulnerable file.

The orthogonal balanced table has only one-third of the number of rows: 9 instead of 27.

c	Model	Expl.	Vuln	Outcome
0	0	0	0	8
1	1	0	1	4
2	2	0	2	11
3	0	1	1	10
4	1	1	2	6
5	2	1	0	8
6	0	2	2	12
7	1	2	0	8
8	2	2	1	6

=

c	Model	Expl.	Vuln	Outcome
0	TNN	Control	XSS	8
1	GNN	Control	SQL	4
2	DL	Control	DoS	11
3	TNN	Black	SQL	10
4	GNN	Black	DoS	6
5	DL	Black	XSS	8
6	TNN	White	DoS	12
7	GNN	White	XSS	8
8	DL	White	SQL	6

Table 3
Subjects vs. factors and levels in orthogonal balanced designs.

	Distributing n across K and ℓ					
	Levels (ℓ)	Factors (K)				
		1	2	3	4	5
$n = 50$	2	25	13	13	6	6
	3	17	6	6	6	$\cancel{3}$
	5	10	$\cancel{2}$	$\cancel{2}$	$\cancel{2}$	$\cancel{2}$
$n = 100$	2	50	25	25	13	13
	3	33	11	11	11	6
	5	20	$\cancel{4}$	$\cancel{4}$	$\cancel{4}$	$\cancel{4}$

The number of subjects per configuration, i.e. $\langle factor, level\rangle$ pairs, in orthogonal balanced designs is larger than the number available in to full factorial designs. This happens because the number of configuration does not increase exponentially with the number of factors.

Also in this case, the configuration $c = 0$ corresponds to the configuration $c = 0$ of the full factorial design in Example 2.7. However, the configuration $c = 2$ from this example, corresponds to the configuration $c = 20$ from the full factorial Example 2.7. Again, not all the configurations from Example 2.7 have corresponding entries in this example because the purpose of an orthogonal design is to test *fewer* configurations.

This now becomes a doable experiment: with $N = 100$ subjects we will have 12–13 subjects per configuration which pass the rule of thumb of 10.

The example of the number of data points we will get with the orthogonal balanced design is shown in Table 3.

Summary of orthogonal balanced design. The following rules apply

- *first factor*: ℓ configurations;
- *two factors*: ℓ^2 configurations;
- $2 + (\ell - 1)$ *factors*: no additional configurations beyond ℓ^2;
- $i + (\ell - 1)\frac{i(i-1)}{2}$ *factors*: no additional configuration beyond the first ℓ^i configurations.

Remark 3. The notation of xs and ys is purely for presentational reasons. There is no difference between them. If one selects y_1 and y_2 and sort the table by those value would obtain an isomorphic construction of x_1 and x_2.

8. Crossover balanced design

A key feature of the SSE experiments is the presence of slightly different *scenarios* on which the same experiment can be repeated.

8.1. Intuition

Using a repeated measures design is easier as each subject performs each experimental configuration condition typically ($r = \ell$). It is easier to ask a subject to perform one more task rather than recruiting an additional subject to perform a different task. However, researchers should be aware of order effects, and counter-balance them by balancing the order of the different experimental configurations that the subjects have to experiment.

The key idea is to generalize the design structure of orthogonal designs so that instead of considering x_2 as a separate factor we just consider x_2 a repetition of the same experiment on a different scenario. This allows us to boost the number of times a factor has been analyzed.

We, therefore, have two dimensions. The same dimension of the horizontal configurations (number of rows) with repeats ($K = r.\ell$), number of columns equals repeats times the number of levels).

In this way, in each configuration, we can test each level of a factor at least $r = 2$-times. To do so without incurring too large learning effects, we need $2 \cdot \ell$ *different scenarios*. In general, to be balanced we always need r to be a multiple of two.

8.2. The two factors case

For simplicity in this case we consider the case of two factors $K = 2$ in which the first factor $i = 1$ is used as a traditional factor and the second factor $i = 2$ is used through repeats with $r = 2 \cdot \ell$ repeats.

To generate a permutation of a block of ℓ-elements we can shift a value by $r \in \{1, \dots, \ell - 1\}$ modulo ℓ and therefore we have $c \in \{0, \dots, \ell(\ell - 1)\}$.

$$x_1[c] = (c + \lfloor c/\ell \rfloor) \mod \ell \tag{8}$$
$$y_j[c] = (x_1[c] + j \cdot (1 + \lfloor c/\ell \rfloor)) \mod \ell \tag{9}$$
$$\text{for } j \in \{1, \dots \ell - 1\} \tag{10}$$
$$y_{j+\ell-1}[c] = y_{\ell-j}[c] \text{ for } j \in \{1, \dots \ell - 1\} \tag{11}$$
$$y_{2\ell}[c] = x_1[c] \tag{12}$$

In this way, we can run $\ell(\ell - 1)$ configurations but we test a factor level for 2-times per configuration and $\ell - 1$-times for each possible scenario.

The first theorem provides a guarantee that is very important in terms of work, in particular, if subjects are asked to perform the task.

Theorem 2 (*Fairness and Balanced Levels*). *Every configuration tests every factor and each level exactly twice.*

In this way, all subjects will be exposed to the same technology and everybody will benefit from being exposed to a particular approach.

Proof. Since operations on ℓ are designed to be a Galois Field each value of $0 \dots \ell - 1$ can occur only once in the sequence $x_1[c]$ and $y_j[c]$ for $j \in \{1 \dots \ell - 1\}$. Since $y_{j+\ell-1}[c]$ is specular to $y_{\ell-j}[c]$ and $y_{2\ell}[c]$ is specular to $x_1[c]$ we have that exactly two values can occur in each configuration. □

Corollary 3. *With n subjects to be distributed among all configurations each level of the factor is tested $2 \times n/\ell^2$.*

Theorem 4 (*Balanced Scenarios*). *Every scenario is tested with all levels.*

Proof. The proof is identical to the one above: each operation in a Galois field is invertible. For a scenario to be tested with less than the total number of levels it means that one level never occurs among the result of the operations (and another value occurs twice). Then the missing value would not have an inverse which is a contradiction for being a field. □

Theorem 5 (*Balanced Start*). *Each levels is presented at the beginning of the experiment in ℓ configurations.*

Proof. Since ℓ is designed to be a Galois field we have that for the first ℓ elements of $x_1[c]$ we have all possible values from 0 to $\ell - 1$ and for each of the subsequent $\ell - 1$ blocks of ℓ configurations each we have a shift of the previous block by a constant factor. Hence each level occurs once and only once in each of the ℓ blocks of ℓ configurations. □

Theorem 6 (*Compensating Learning Effects*). *The number of times a level occurs before another level in a configuration is the same across all configurations.*

Proof. Since operations on ℓ are designed to be a Galois Field for each configuration the sequence $x_1[c]$ and $y_j[c]$ for $j \in \{1 \dots \ell - 1\}$ is a permutation of the sequence $0, \dots \ell - 1$. Since the remainder of the scenario is the permutation in reverse order the number of inversion of each pair of elements in the entire sequence is the same. Building upon Theorem 5 all configurations' initial permutation of the first ℓ-elements start with an equal number of levels and therefore across all configurations the number of inversions is also the same across the entire design. □

The example of the number of data points we will have per level in an experiment with crossover balanced design is shown in Table 4.

We provide a possible realization of Example 1 (Finding Vulnerabilities) and Example 2 (Explainability of ML Findings) for the crossover design

Example 1.9. Finding Vulnerabilities With Crossover is an instance of the Experiment 1.
Instance. We only have the slicing intervention ($K = 1$, single factor) with $\ell = 2$ levels (Full, Slice) but $r = 2\ell$ repeats using the types of vulnerabilities to provide different scenarios.
Example Configurations. We provide *different source codes containing different CVEs* so that each file contains a different vulnerability. We want to assign to each subject two sliced files and two original files so that the number of times sliced files are seen before original files is the same as the number of times original files are seen before sliced files.
Task of subjects Each subject will be exposed to 4 different vulnerable code source files.

The full table will be as below.

c	Repeat $r = 2\ell$ code source files			
	XSS	User	Path	DoS
0	Full	Slice	Slice	Full
1	Slice	Full	Full	Slice

We have also marked vertical lines since each configuration that will be assigned to a subject will require testing four different vulnerabilities. In contrast to Experiment 1.7 and Experiment 1.8 the subject will only be exposed to a single vulnerability.

This looks like a doable experiment. With $N = 100$ subjects we will have around 50 subjects per configuration which is comfortably higher than the rule of thumb of 10 and the danger threshold of 5.

Example 2.9. Explainability of ML Findings – Crossover is an instance of Experiment 2.
Instance. In this case, we only have 1 factor: explainability mechanism ($K = 1$, single factor) with $\ell = 3$ levels (Control, Black-box, White-box) but repeated twice ($r = 2\ell$).
Example Configurations We provide $r = 2\ell$ *different code fragments* containing different vulnerabilities. We assign each subject 2 unmarked code fragments (Control), 2 highlighted with Black-box explainability mechanism, and 2 others with White-box mechanism. The permutations between Control, Black-box, and White-box are balanced, i.e. the number of Controls seen before Black-box is the same as Black-box before White-box, etc.
Task of subjects Each subject will be exposed to 6 different vulnerable code fragments.

The full table of this crossover design will be as below.

Table 4
Subjects/data points vs. levels in crossover balanced design.

N	Levels	Config.	Subject/	DP for r repeat	
	(ℓ)	(c)	Conf	1	2
50	2	2	25	50	100
	3	6	8–9	50	100
	5	20	2–3	50	100
100	2	2	50	100	200
	3	6	16–17	100	200
	5	20	5	100	200

The number of total data points (DP) per level in the crossover balanced design is $n = N \cdot 2$, and $r = 2 \cdot \ell$ is the number of repeats. As all subjects get each level r times, the number of data points is not influenced by the number of levels. The number of scenarios in each configuration $= \ell \cdot 2$.

c	Repeat $r = 2\ell$ code fragments					
	XSS	$User$	$Path$	DoS	SQL	$Info$
0	Control	Black	White	White	Black	Control
1	Black	White	Control	Control	White	Black
2	White	Control	Black	Black	Control	White
3	Black	Control	White	White	Control	Black
4	White	Black	Control	Control	Black	White
5	Control	White	Black	Black	White	Control

We have also marked vertical lines since each configuration that will be assigned to a subject will require testing 6 different vulnerabilities. While in Experiment 2.7 and Experiment 2.8 each subject will only be exposed to a single vulnerability.

This is also a doable experiment. With $n = 100$ subjects we will have around 16 subjects per configuration which is higher than the rule of thumb of 10 and the danger threshold of 5. We will also get a total of 200 data points for each level ($N \cdot r$).

Summary of crossover balanced design. The following rules apply

- *two levels*: 2 configurations.
- *three levels*: 6 configurations.
- *ℓ levels*: $\ell \cdot (\ell - 1)$ configurations.

9. Statistical impact of Orthogonal Design

For the balanced Orthogonal Design the key statistical impact is that we do not measure the full interactions, so we can only detect main effects (i.e. the difference between levels) or pairwise interactions. This is well studied in statistical textbooks, e.g. Agresti and Franklin (2007), or in texts on DOE, so we do not discuss it here in detail but just make one example of successfully captured interaction and one example of interaction that the system cannot fully catch.

Consider the outcomes of Example 2.7. By looking at the variable outcome we find an interaction between the type of model and the type of explainability. For TNN and GNN the outcome increases when moving from Control to Whitebox while for DL it decreases. Also the GNN model has always a lower outcome than the TNN model. The type of vulnerability we use does not make a difference. For example, the three configurations {`TNN, Control, XSS`}; {`TNN, Control, SQL`}; {`TNN, Control, DoS`} have the same outcome. So we do not lose information by only considering the first configuration of the three as we do in Example 2.8.

Fig. 2 illustrates the example. The x-axis captures the changes in value of the type of explainability and each different line correspond to a value of the factor capturing the type of model. We can capture this interaction in the design as shown in Fig. 2 because all pair of points have at least a row (c) in the example configuration.

We also see the outcome for type of vulnerabilities behave in a different ways but there is not a clear difference. The full dataset that would be available with the full factorial design would have shown us that for the same configurations of the neural network model and the type of explainability, all vulnerability types have the same outcome. In this case, the right side of Fig. 2 and the Table in Example 2.8 are consistent with this finding and show that there is not a clear difference among the type of vulnerabilities.

A full-fledged three-way interaction is instead presented in Example 1.8 where the combination of {`Slice, Path, Tomcat`} generate a significant drop in outcome irrespective of the IDE configuration. It is also the *only* combination that generates a drop in outcome. Plotting the results in Fig. 3 would show three possible pairwise interactions: `Slice` with `Tomcat`, `Slice` with `as-is`, as well as `Slice` with `Path` traversal. Slicing does not seem to help for commits. This strange combination of several two-ways interactions should spur additional investigations and confirming trials.

In most cases N-ways interactions are rare and therefore we may ignore them. Their presence may be suspected when several two-ways interactions are present and is typically discovered when confirmatory trials fails.

10. Statistical impact of crossover designs

For crossover designs, which are more common in SE, the situation is more nuanced than for orthogonal designs as we have a learning effect. To show the impact of such a learning effect we provide a full-fledged calculation of the resulting mean and variance for the case of a crossover design with one factor with two levels. The other cases are essentially identically in terms of qualitative findings.

10.1. Single factor factorial design

As a reference benchmark we start by calculating the mean and standard deviation of what would happen if we wanted to run a standard full factorial design with one factor and two levels (e.g. Full vs. Slice) where we want each condition to have at least n points.

With only one factor x_1 and a full factorial experimental design we would assign a subject to either a group A (where x_1 was at level 0) or to a group B (where x_1 was at level 1). W.l.o.g. suppose that the effect on the outcome from level A to B on our only scenario 1 is equal to δ_1. Since we would assign subjects to the only scenario 1, then the outcome $o_1[n]$ of subject n (respectively $o_1[m]$ of subject m) will be the random variation due to the subject n's characteristics (respectively subjects m's characteristics) assigned to the group A (respectively B) on the only scenario we had.

This scenario is summarized in the Experiment part of Table 5. For the statistical analysis we would also have the two vectors described in Table 5-Statistics, each of them with n entries. Since we wanted to have n subjects for each of the condition we would need to have at least $2n$ subjects in total.

If N is sufficiently large and the variance is sufficiently small, the random effects of the individual characteristics will cancel each other and the difference in outcome between $\mu_{A(1,N)}$ and $\mu_{B(1,N)}$would be

$$\mu_{B(1,N)} - \mu_{A(1,N)} \approx \delta_1 + error \tag{13}$$

10.2. Traditional crossover design

In the traditional crossover design with repetitions used in SE, we only have N subjects but we seek to have more points by asking them to re-do some tasks. So we will still divide the set of subjects in the two groups. Subject n will be doing A on the scenario 1 and then will be using B on a different scenario 2. Conversely, subject m will be doing B on scenario 1 and will be performing A on scenario 2. The intuition is

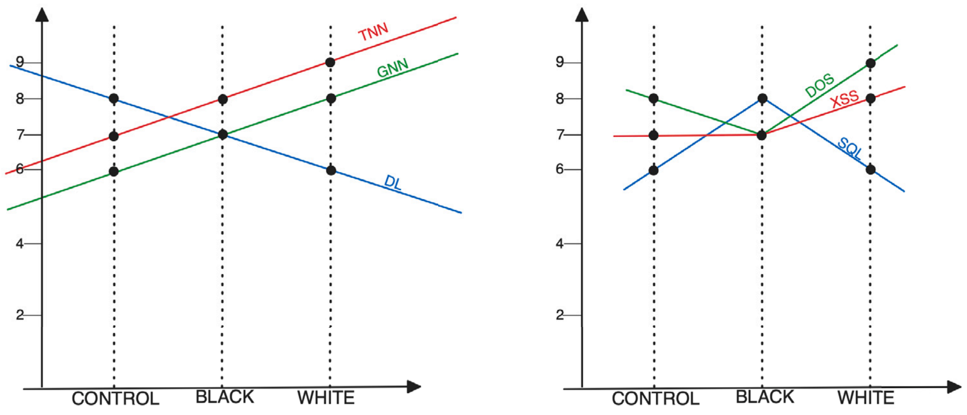

Fig. 2. 2-way factors interaction captured by explainability Example 2.8. The right part subfigure shows that TNN is always better than GNN, no matter the explainability. Also control is worse than black and the latter is worse than white. If only those two models were considered we could have concluded that there was no interaction between the ML Models and the absence or the presence of difference types of explainability mechanism. However, we see that DL behaves differently from the other two models. Therefore, we have an interaction between the type of models and the type of explainability. The left subfigure shows that there does not seems to be any relations between the type of explainability and then type of vulnerability.

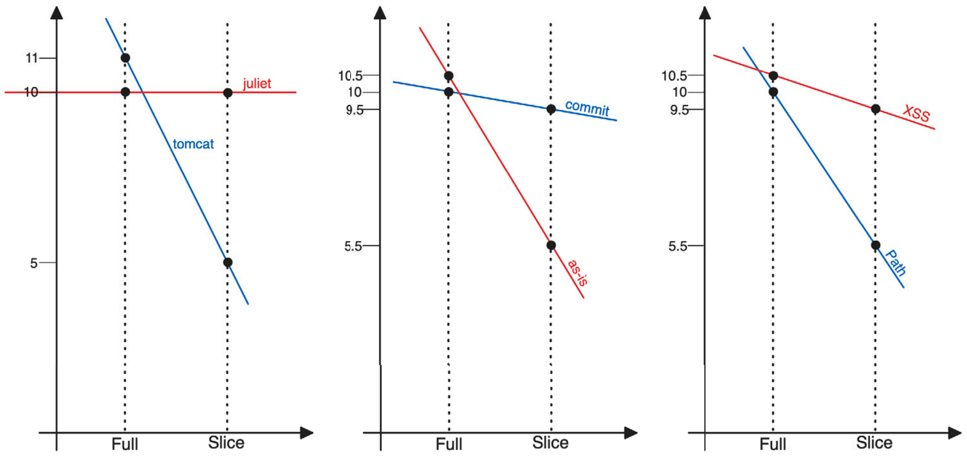

Fig. 3. Interactions on finding vulnerabilities Experiment 1.8. The right part of the figure shows the interaction of slicing with the choice of the dataset (Slice makes a difference for tomcat but not for juliet). At the center we see the interaction that only for as-is configuration slicing seems to help. Finally, the left side shows that slicing have a major impact on Path traversal. Three two-ways interactions are of course possible, but might also be the sign that a three-way interaction is present and additional confirming trials for the missing configurations might be needed.

Table 5
Simple factorial design - Measurements.

Experiment - Design with $2N$ subjects		
Subjects	Scenario	Outcome
$n = 1 \dots N$	A_1	$o_1[n]$
$m = 1 \dots N$	B_1	$o_1[m]+\ \delta_1$
Statistics - Two vectors of N elements to compare		
Points	Outcome_A	Outcome_B
1	$o_1[1]$	$o_1[1]\ +\ \delta_1$
$\vdots$	$\vdots$	$\vdots$
$n,\ m$	$o_1[n]$	$o_1[m]+\ \delta_1$
$\vdots$	$\vdots$	$\vdots$
N	$o_1[N]$	$o_1[N]+\ \delta_1$
mean	$\mu_{A(1,N)}$	$\mu_{B(1,N)}$
stdev	$\sigma_{A(1,N)}$	$\sigma_{B(1,N)}$

that we half the number of subjects but ask them to do twice the work. We show the design in Table 6.

The outcome of the experiment will be different in this case because there will be a learning effect due to the fact that the nth subject has seen A first and then has possibly transferred some of the results to the next analysis of B on the second scenario. We denote by $\lambda_{A_1<B_2}$ the learning effect of doing A first and then B and by $\lambda_{B_1<A_2}$ the effect of doing A first and then B.

Remark 4. Crossover experiment are only partly balanced because we run the same number of As and Bs cases but this is not really true as we run them on *different* subjects and scenarios. This apparent from Table 6

Since the whole point of the crossover scenario is to have a repetition of data points for the statistical analysis to increase its power one cluster together in the same set the result of the nth subject on A (on the first scenario) and the result of the mth subject on A (on the second scenario). Therefore, when we do the statistical tests we would compare the two vectors of outcomes that we show in Table 6. Statistics which now have N elements each.

Theorem 7. *The mean and variance of a crossover experiment with* $2N$ *subjects, one factor with two levels (A,B) and two scenarios (1,2) as designed in* Table 6 *can be expressed as a combination of the mean and variance of a full factorial experiment without repetitions on scenario 1 and a full factorial experiment without repetition on scenario 2 as designed in*

Table 6
Traditional crossover design in SSE - Measurements.

Experiment - Design with $2N$ subjects				
Subjects	Scenarios		.... outcomes	
	1	2	1	2
$n = 1 \dots N$	A_1	B_2	$o_1[n]$	$o_2[n]+ \delta_2 + \lambda_{A_1<B_2}$
$m = 1 \dots N$	B_1	A_2	$o_1[m]+ \delta_1$	$o_2[m]+ \lambda_{B_1<A_2}$
Statistics - Two vectors of $\mathbf{2}N$ elements to compare				
Points		Outcome$_A$		Outcome$_B$
$\vdots$		$\vdots$		$\vdots$
$n = 1 \dots N$		$o_1[n]$		$o_2[n]+ \delta_2 + \lambda_{A_1<B_2}$
$\vdots$		$\vdots$		$\vdots$
$m = 1 \dots N$		$o_2[m]+ \lambda_{B_1<A_2}$		$o_1[m]+ \delta_1$
$\vdots$		$\vdots$		$\vdots$
mean		$\mu_{A(co,N)}$		$\mu_{B(co,N)}$
stdev		$\sigma_{A(co,N)}$		$\sigma_{B(co,N)}$

Table 5 and the learning effects $\lambda_{A_1<B_2}$ and $\lambda_{B_1<A_2}$ as follows

$$\mu_{A(co,2N)} = \frac{1}{2}\left(\mu_{A(1,N)} + \mu_{A(2,N)}\right) + \frac{1}{2}\lambda_{B_1<A_2} \tag{14}$$

$$\sigma^2_{A(co,2N)} = \frac{1}{2}\left(\sigma^2_{A(1,N)} + \sigma^2_{A(2,N)}\right) \tag{15}$$

$$+ \frac{1}{4}\left(\mu_{A(2,N)} - \mu_{A(1,N)} + \lambda_{B_1<A_2}\right)^2 \tag{16}$$

The case for B is analogous by swapping A with B.

Proof Sketch (*Full Details in the* Appendix)**.** To prove the result we substitute the value of the outcome for the statistics part of the Table 6 in the formal definition of mean and standard deviation. We start the derivation by explicitly computing the mean

$$\mu_{A(co,2N)} = \frac{1}{2N}\left(\overbrace{\sum_n o_1[n]}^{\text{scenario 1}} + \overbrace{\sum_m \left(o_2[m] + \lambda_{B_1<A_2}\right)}^{\text{scenario 2}}\right) \tag{17}$$

we re-arrange the terms by separating the n summands steps whose purpose it to re-assemble the terms so that the one can identify the components of the statistics part of Table 5. Then we replace the explicitly computed mean into the formal definition of the variance and again re-arrange terms so that they can be mapped into the components of the statistics part of Table 5 or can be canceled out. □

Therefore the difference in outcome between $\mu_{A(co,2N)}$ and $\mu_{B(co,2N)}$ would be

$$\mu_{B(co,2N)} - \mu_{A(co,2N)} = \frac{1}{2}\left(\mu_{B(1,N)} - \mu_{A(1,N)}\right) \tag{18}$$

$$+\frac{1}{2}\left(\mu_{B(2,N)} - \mu_{A(2,N)}\right) \tag{19}$$

$$\frac{1}{2}\left(\lambda_{A_1<B_2} - \lambda_{B_1<A_2}\right) \tag{20}$$

This theorem provides us the basis for some useful observations. The first observation is that the effect size of the cross-over design depends on the average of the effect of the changing level on the two scenarios (21). This smooths effects due to small variations of performance on the scenarios thus providing more robust results.

If the scenario are sufficiently different then $\lambda_{A1<B2} \approx \lambda_{B1<A2} \approx 0$ and the impact of the learning effect on both mean and variance is negligible. However, if the two scenarios are too different, such difference may present itself in different ways. For example, the mean outcome may be different between the two scenarios (e.g. scenario 1 is generally easier than scenario 2) then the difference in variance would still be present as the terms $\mu_{A(2,N)} - \mu_{A(1,N)}$ and $\mu_{B(2,N)} - \mu_{B(1,N)}$ would not be zero. If we assume that N is sufficiently large the differences in the mean will cancel and we will have

$$\mu_{B(co,2N)} - \mu_{A(co,2N)} \approx \frac{1}{2}\left(\delta_1 + \delta_2\right) + error \tag{21}$$

$$+\frac{1}{2}\left(\lambda_{A_1<B_2} - \lambda_{B_1<A_2}\right) \tag{22}$$

This equation shows that the overall effect depends on the average effect on the two scenarios. Also in this case, if the effect of one scenario is way smaller (or way larger) than the effect of the other scenario it might mask (or boost) the overall effect. If this happens, then scenarios are not just scenarios, they should be considered as first class citizens as uncontrollable, noise factors.

For example, in the Finding Vulnerabilities Example, a file from Juliet would only span less than a hundred lines of code, while a method from Apache Tomcat might span thousands lines of code. They would hardly classify as just scenarios.

A third observation is that variance may increase due to the learning effect or due to the difference in outcome between the two scenarios (16). An increased variance may make it more difficult to have a significant result as the outcomes of the level A may overlap with the outcomes of the level B.

If the learning effect is just related to the scenarios and not to the experimental configuration by itself $\lambda_{A1<B2} \approx \lambda_{B1<A2} \approx \lambda_{1<2}$ then the impact on the mean is actually zero. However, the impact on the variance is still there, so the outcomes of the subject might be more widely scattered. Depending on the effect size, an effect might turn out not to be statistically significant. For example, the t-test divides the mean by the variance and would thus yield a lower value of the t-test and possibly a not significant result. So it is important that the scenarios are actually different.

Remark 5. In summary, the designer is faced with an essential trade-off: the scenarios must be sufficiently different so that the learning effect is negligible but they must also be sufficiently similar that the difference in effect is negligible.

10.3. Balanced crossover design

Table 7 shows the different outcomes for the case of the full balanced crossover that we have introduced in Section 8.

For simplicity we only report the mean of the crossover experiment. The variance can be calculated with the same derivations used for Theorem 7.

Theorem 8. *The mean crossover experiment with $2N$ subjects, one factor with two levels (A,B) and four scenarios (1, 2, 3, 4) as designed in Table 7 can be expressed as a combination of the mean of four full factorial experiments without repetitions of N subjects as designed in Table 5 and the learning effects as follows*

$$\mu_{A(co,2N)} = \frac{1}{4}\Big(\mu_{A(1,N)} + \mu_{A(2,N)} + \mu_{A(3,N)} + \mu_{A(4,N)} \tag{23}$$

$$+\lambda_{B_1<A_2} + \lambda_{B_1<A_3} + \lambda_{B_2<A_4} + \lambda_{B_3<A_4} \tag{24}$$

$$+\lambda_{A_1<A_4} + \lambda_{A_2<A_3}\Big) \tag{25}$$

The case for B is analogous by swapping A with B.

Table 7
Balanced crossover design - Measurements.

Experiment - Design with $2N$ subjects

Subjects	………Scenarios………				………………Outcomes………………			
	1	2	3	4	1	2	3	4
$n = 1 \dots N$	A_1	B_2	B_3	A_4	$o_1[n]$	$o_2[n]+ \delta_2 + \lambda_{A_1<B_2}$	$o_3[n]+ \delta_3 + \lambda_{A_1<B_3} + \lambda_{B_2<B_3}$	$o_4[n]+ \lambda_{A_1<A_4} + \lambda_{B_2<A_4} + \lambda_{B_3<A_4}$
$m = 1 \dots N$	B_1	A_2	A_3	B_4	$o_1[m]+ \delta_1$	$o_2[m]+ \lambda_{B_1<A_2}$	$o_3[m]+ \lambda_{B_1<A_2} + \lambda_{A_2<A_3}$	$o_4[m]+ \lambda_{B_1<B_4} + \lambda_{A_2<B_4} + \lambda_{A_3<B_4}$

Statistics - Two vectors of **4**N elements to compare

Points	Outcome_A	Outcome_B
⋮	⋮	⋮
$n = 1 \dots N$	$o_1[n]$	$o_2[n]+ \delta_2 + \lambda_{A_1<B_2}$
$n = 1 \dots N$	$o_4[n]+ \lambda_{A_1<A_4} + \lambda_{B_2<A_4} + \lambda_{B_3<A_4}$	$o_3[n]+ \delta_3 + \lambda_{A_1<B_3} + \lambda_{B_2<B_3}$
⋮	⋮	⋮
$m = 1 \dots N$	$o_2[m]+ \lambda_{B_1<A_2}$	$o_1[m]+ \delta_1$
$m = 1 \dots N$	$o_3[m]+ \lambda_{B_1<A_3} + \lambda_{A_2<A_3}$	$o_4[m]+ \delta_4 + \lambda_{B_1<B_4} + \lambda_{A_2<B_4} + \lambda_{A_3<B_4}$
⋮	⋮	⋮
mean	$\mu_{A(co,2N)}$	$\mu_{B(co,2N)}$
stdev	$\sigma_{A(co,2N)}$	$\sigma_{B(co,2N)}$

Proof Sketch. The proof is essentially identical to the one used for Theorem 7 (see also Appendix B). If we apply the formal definition for the calculation of the mean of the crossover design for the level A we have the following equation

$$\mu_{A(co,2N)} = \frac{1}{4N}\Bigg(\overbrace{\Sigma_n o_1[n]}^{\text{scenario } 1} + \tag{26}$$

$$\overbrace{\Sigma_m \left(o_2[m] + \lambda_{B_1<A_2} \right)}^{\text{scenario } 2} \tag{27}$$

$$\overbrace{\Sigma_m \left(o_3[m] + \lambda_{B_1<A_3} + \lambda_{A_2<A_3} \right)}^{\text{scenario } 3} \tag{28}$$

$$\overbrace{\Sigma_n \left(o_4[n] + \lambda_{A_1<A_4} + \lambda_{B_2<A_4} + \lambda_{B_3<A_4} \right)}^{\text{scenario } 4} \tag{29}$$

We use $4N$ as a divisor as we have $4N$ data points. We can then re-aggregate the elements of the definition above to obtain the desired results. To show that the equation holds by swapping A with B we simply use Theorem 4 as a Lemma. □

We have a similar structure of the crossover case without a balance. In contrast with the traditional crossover design we have now performed an experiment with 4 (four) different scenarios and have therefore a more robust understanding of the performance of the levels of a factor in a variety of scenarios.

This can be more easily visible by using the same approximation we have used for the unbalanced cross-over. If we assume that when n is sufficiently large, the differences in the means will cancel and we can approximate the difference between the means of different levels by the effects of the levels:

$$\mu_{B(co,2N)} - \mu_{A(co,2N)} \approx \frac{1}{4}\Big(\delta_1 + \delta_2 + \delta_3 + \delta_4 + err \tag{30}$$

$$+\lambda_{B_1<B_4} - \lambda_{A_1<A_4} \tag{31}$$

$$+\lambda_{B_2<B_3} - \lambda_{A_2<A_3} \tag{32}$$

$$+\lambda_{A_1<B_2} - \lambda_{B_1<A_2} \tag{33}$$

$$+\lambda_{A_1<B_3} - \lambda_{B_1<A_3} \tag{34}$$

$$+ \lambda_{A_2<B_4} - \lambda_{B_2<A_4} \tag{35}$$

$$+\lambda_{A_3<B_4} - \lambda_{B_3<A_4}\Big) \tag{36}$$

Also in this case, if a scenario is significantly different than the other three it might be more appropriate to consider a replication of the experiments in which we treat the type of scenarios as (uncontrollable) noise factors for a full factorial design or at least an orthogonal design.

In terms of learning effect, having a balance design has a number of advantages. Once again, if the learning effect depends on the scenario only and therefore for every pair of scenarios s and t it is $\lambda_{A_s<B_t} \approx \lambda_{B_s<A_t} \approx \lambda_{s<t}$ then all terms (31)–(36) cancel out and the impact would be close to zero. If the learning effects is not transferable from one techniques to the other, i.e. $\lambda_{A_s<B_t} \approx 0$ then all terms (33)–(36) will be zero. At this point only the self learning effect will be present from (31)–(32). If the learning effect is the same across the two levels $\lambda_{B_s<B_s} \approx \lambda_{A_s<A_s}$ also the first two term will cancel out.

This analysis shows us that the cases in which a learning effect may have an impact on the validity of the balanced cross-over experiment is when the learning effects varies significantly across the levels. For example, if by repeatedly using A the subjects improve significantly more their performance than by using B then (31)–(32) will have a negative contribution to the difference in means and therefore lower the effect of B to generate a difference due to (30).

By itself this will be an interesting finding which can be actually measured by testing pairwise the outcome the repeated scenario-level pairs without the need to run a different experience but simply adjusting with a Bonferroni–Holms correction. In presence of a significant learning effect in one level of the factor but not the other level. The comparison of B and A individually could be misleading. Replication experiments could then focus on a string of As vs. a string of Bs.

11. Description of the artifacts

We provide an Excel file that provides an example of how to implement the design. The Excel file contains 5 sheets elaborated as follows:

- *Orthogonal*: contains examples of the orthogonal balanced design for 2 levels and 3 levels.
- *Crossover*: contains examples of the crossover balanced design with 2, 3, and 5 levels.
- *Galois*: contains the Galois table to create an orthogonal balanced design with 4 levels.
- *MakeYourOwn*: contains three editable tables containing the equations for designing the three different type of experiments. It is possible to use this sheet for designing your own experiment.
- *Example1*: contains the implementation of Example 1 with 2 levels, including 2 full-factorial designs (2 and 4 factors), 2 orthogonal balanced designs (2 and 4 factors), and a crossover balanced design.
- *Example2*: contains the implementation of Example 2 with 3 levels, including 2 full-factorial designs (2 and 3 factors), 2 orthogonal balanced design, and a crossover balanced design.

Moreover, we provide a Google colaboratory file to generate the full factorial and the orthogonal design. Also, it replicates the examples for the crossover design.

The Excel file and the Google colaboratory are available on Anonymous Github (Massacci et al., 2024) for reviewers during the review process and will be open to public upon acceptance.

12. Limitations and conclusion

In this paper, we presented the challenges and issues of designing balanced experiments for experiments in SE. In particular, we described three of main designs that are used: full factorial design, orthogonal balanced design, and crossover balanced design. We also formally prove that the computational rules that we provide for each design meet the desired requirements. The orthogonal and crossover designs can then be combined to obtain richer and complex designs.

The first type of design allows us to test and possibly retrieve all the possible interactions. However, as the number of runs exponentially grows when increasing the number of factors and levels, it makes this design close to impossible to perform in SSE as it requires many subjects, and it is very expensive in terms of time and resources. The orthogonal design, also known as the Taguchi design, allows to have a balanced design with the smallest number of possible runs. However, this experiment in some cases may still have a large number of configurations, and it can be applied in the case the factors are independent. Therefore, we finally introduced a crossover design that can be used in SSE experiments when there is the presence of a slightly different scenario in which the same experiment can be repeated.

Our design is balanced as every configuration tests every factor and each level twice. Therefore, the subjects will be exposed to the same technology which provides an important fairness requirement when used on students during class experiments. Obviously, a crossover design may have order effects. However, we counter-balance it by formally assuring that the number of times a level occurs before other levels in a configuration is the same across all the configurations. Thus we provably mitigate learning effects to the extent that is possible without using a full factorial design.

Our contribution is suitable only for certain cases, as our approach is applicable only when the number of levels is a prime number. This is normally the case for most common and practically used levels $\ell = 2, 3, 5, 7$. In the Appendix, we provide an example of the calculation necessary in case the number of levels is a power of a prime number (e.g $\ell = 4, 8, 9$). Such procedure involves the use of the Galois table to create modulo-ℓ design. Unfortunately no such a system can be designed for $\ell = 6$ (this is unfortunately a theoretical limitation). Designs with larger numbers of levels would still be out of reach for SSE experiments due to the difficulty or recruiting subjects. In fact, our design allows to use a limited number of participants, however issues such as less variability and more risk of errors may impact on the results.

Despite these limitations our design offers a good trade-off of having the balanced smallest number of combinations without using a full factorial design, and we believe that such design can be useful for boosting the use of experimentation in SSE without limiting ourselves to two factors and two levels as in classical SSE textbooks.

CRediT authorship contribution statement

Fabio Massacci: Conceptualization, Methodology, Software, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Aurora Papotti:** Conceptualization, Software, Validation, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Ranindya Paramitha:** Conceptualization, Software, Validation, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization.

Table A.8
Addition and multiplication tables for $\ell = 4$.

+	0	1	2	3	*	0	1	2	3
0	0	1	2	3	0	0	0	0	0
1	1	0	3	2	1	0	1	2	3
2	2	3	0	1	2	0	2	3	1
3	3	2	1	0	3	0	3	1	2

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Fabio Massacci reports to have a financial support provided by EU under GA 952647 (H2020-AssureMoss) and 101120393 (Sec4AI4Sec), also by the Dutch Research Council (NWO) Grant NWA.1215.18.006 (HEWSTI). Aurora Papotti reports to have a financial support provided by the Dutch Sectorplan I. Ranindya Paramitha reports to have a financial support provided by EU under GA 952647 (H2020-AssureMoss) and 101120393 (Sec4AI4Sec).

Data availability

The artifact of this paper can be accessed in Zenodo through a link in the paper: Massacci et al. (2024) .

Acknowledgments

This work was partly supported by the European Union through grant number 952647 (AssureMOSS), and grant number 101120393 (Sec4AI4Sec), the Dutch Research Council (NWO) grant n. NWA.1215.18.006 (THESEUS), n. KICH1.VE01.20.004 (HEWSTI), and the Dutch Sectorplan I.

Appendix A. Galois tables for $\ell \geq 4$

Table A.8 shows the correct values to use for $\ell = 4$ for the construction with Eqs. (5)–(6).

There cannot be a design for $\ell = 6$ (because 6 is not a prime power) while there can be a design for $\ell = 5, 7$ using directly the arithmetic modulo operation, and a slightly more complicated table for $\ell = 8, 9$ using a variant of Table A.8.

The addition and multiplication tables for $\ell = 8 = 2^3$ can be generated by using as elements the triples $x_1x_2x_3$ which are the coefficients of the polynomial in z, $x_1 + x_2 \cdot z + x_3 \cdot z^2$ for all possible values of x_1, x_2, x_3 modulo 2 with a simplification constraint such that $z^3 = z + 1$.

For example, the triple 100 is the element which is the product of the element 010 and the element 101. This element can be constructed as the triple of the coefficients modulo 2 of the product of the polynomials $(0 + 1 \cdot z + 0 \cdot z^2) \cdot (1 + 0 \cdot z + 1 \cdot z^2)$ i.e. $z \cdot (1 + z^2) = z + z^3 = z + (z + 1) = 2 \cdot z + 1 = 0 \cdot z + 1 = 1$. Therefore we have 1 as the final result which is more precisely the polynomial $1 + 0 \cdot z + 0 \cdot z^2$ and therefore $x_1 = 1, x_2 = 0, x_3 = 0$.

The table for $\ell = 9 = 3^2$ can be generated by using as elements the pair x_1x_2 which are the coefficients of the polynomial in z $x_1 + z \cdot x_2$ for all possible values of x_1 and x_2 modulo 3.

Appendix B. Proof of Theorem 7

Proof. The mean of the outcome for subjects who received the experimental configuration A with a cross-over design $\mu_{A(co,2N)}$ can expressed as a function of the learning effect $\lambda_{B_1<A_2}$ and the mean that they would have had if the just performed the analysis in an ideal factorial

experiment without repetitions μ_A. By re-arranging terms in the right part of Eq. (29) we obtain

$$\begin{aligned}\mu_{A(co,2N)} &= \frac{1}{2N}\left(\Sigma_n o_1[n] + \Sigma_m o_2[m]\right) + \frac{1}{2N}\Sigma_m \lambda_{B_1<A_2}\\ &= \frac{1}{2}\left(\frac{1}{N}\Sigma_n o_1[n] + \frac{1}{N}\Sigma_m o_2[m]\right) + \frac{1}{2N} N\lambda_{B_1<A_2}\\ &= \frac{1}{2}\left(\mu_{1(A,N)} + \mu_{2(A,N)}\right) + \frac{1}{2}\lambda_{B_1<A_2} \end{aligned} \tag{B.1}$$

For the last step of the derivation above observe that $\frac{1}{N}\Sigma_n o_1[n]$ is the mean of the normal factorial experiment as defined from Table 5. The case for $\frac{1}{N}\Sigma_n o_2[n]$ would be the same as if we had run the factorial experiment on scenario 2 instead of scenario 1.

For the variance, the derivation is more involved. We start by plugging the computed mean into the definition of variance and perform some algebraic simplifications.

$$\begin{aligned}\sigma^2_{A(co,2N)} = \frac{1}{2N}\Bigg[&\overbrace{\Sigma_n\left(o_1[n] - \mu_{A(co,N)}\right)^2}^{\text{scenario 1}}\\ &+ \overbrace{\Sigma_m\left(o_2[m] + \lambda_{B_1<A_2} - \mu_{A(co,N)}\right)^2}^{\text{scenario 2}}\Bigg]\end{aligned} \tag{B.2}$$

By expanding the definition of the mean from (14) that we have just proved, we get the following term for each summand corresponding to the first scenario

$$o_1[n] - \frac{1}{2}\left(\mu_{A(1,N)} + \mu_{A(2,N)}\right) - \frac{\lambda_{B_1<A_2}}{2}$$

By adding and subtracting $\frac{1}{2}\mu_{1(A,N)}$ we obtain

$$\begin{aligned}&o_1[n] - \frac{1}{2}\left(\mu_{A(1,N)} + \mu_{A(2,N)}\right) - \frac{\lambda_{B_1<A_2}}{2}\\ &\quad+\frac{1}{2}\left(\mu_{A(1,N)} - \mu_{A(1,N)}\right)\\ &= \underbrace{o_1[n] - \mu_{1(A,N)}}_{\text{scenario 1 only}} + \frac{1}{2}\left(\mu_{A(1,N)} - \mu_{A(2,N)} - \lambda_{B_1<A_2}\right)\end{aligned}$$

We can then plug the new summand back in the definition of the variance (B.2).

$$\Sigma_n\left(o_1[n] - \mu_{A(1,N)} + \frac{1}{2}\left(\mu_{A(1,N)} - \mu_{A(2,N)} - \lambda_{B_1<A_2}\right)\right)^2 \tag{B.3}$$

Then we can expand the square power by keeping as a single term the two components $o_1[n] - \mu_{A(1,N)}$.

$$\begin{aligned}&\Sigma_n\Big(\left(o_1[n] - \mu_{A(1,N)}\right)^2\\ &\quad+\left(o_1[n] - \mu_{A(1,N)}\right)\left(\mu_{A(1,N)} - \mu_{A(2,N)} - \lambda_{B_1<A_2}\right)\\ &\quad+\frac{1}{4}\left(\mu_{A(1,N)} - \mu_{A(2,N)} - \lambda_{B_1<A_2}\right)^2\Big)\end{aligned}$$

At this point, we observe that $\Sigma_n\left(o_1[n] - \mu_{A(1,N)}\right)^2$ corresponds to the variance of the experiment without crossover from Table 5, i.e. $N\sigma^2_{A(1,N)}$ and that the term $\Sigma_n\left(o_1[n] - \mu_{A(1,N)}\right)c$ for any constant c can be simplified to $c\cdot\left(\left(\Sigma_n o_1[n]\right) - \mu_{A(1,N)}\right)$ and the latter is identically zero since $\Sigma_n o_1[n] = \mu_{A(1,N)}$ by definition from Table 5.

Hence the first summation in the variance Eq. (B.2) can be characterized as follows:

$$\begin{aligned}\sigma^2_{A(co,2N)} = \frac{1}{2n}\Big[&N\sigma^2_{A(1,N)}\\ &+\frac{N}{4}\left(\mu_{A(1,N)} - \mu_{A(2,N)} - \lambda_{B_1<A_2}\right)^2\\ &+\cdots\end{aligned} \tag{B.4}$$

For the summand corresponding to the variance of the second scenario we also replace the mean $\mu_{A(co,n)}$ with its definition and obtain the following term:

$$\left(o_2[m] + \lambda_{B_1<A_2} - \frac{1}{2}\left(\mu_{A(1,N)} + \mu_{A(2,N)}\right) - \frac{1}{2}\lambda_{B_1<A_2}\right)^2$$

Then we perform the same operation of adding and subtracting $\frac{1}{2}\mu_{A(2,N)}$ and the algebraic simplification of the terms $\lambda_{B_1<A_2}$.

$$\begin{aligned}&o_2[m] + \lambda_{B_1<A_2} - \frac{1}{2}\left(\mu_{A(1,N)} + \mu_{A(2,N)}\right) - \frac{1}{2}\lambda_{B_1<A_2}\\ &\quad+\frac{1}{2}\left(\mu_{A(2,N)} - \mu_{A(2,N)}\right)\\ &= \underbrace{o_2[m] - \mu_{A(2,N)}}_{\text{scenario 2 only}} + \frac{1}{2}\left(-\mu_{A(1,N)} + \mu_{A(2,N)} + \lambda_{B_1<A_2}\right)\end{aligned}$$

We repeat the process of expanding the squared power by keeping $o_2[m] - \mu_{A(2,N)}$ as a single term.

$$\begin{aligned}&+\Sigma_m\Big(\left(o_2[m] - \mu_{A(2,N)}\right)^2\\ &+\left(o_2[m] - \mu_{A(2,N)}\right)\left(-\mu_{A(1,N)} + \mu_{A(2,N)} + \lambda_{B_1<A_2}\right)\Big)\\ &+\frac{1}{4}\left(-\mu_{A(1,N)} + \mu_{A(2,N)} + \lambda_{B_1<A_2}\right)^2\end{aligned}$$

The same considerations that we have done for the first scenario applies in this case. Hence the second sum related to scenario 2 in the variance Eq. (B.2) can be characterized as follows:

$$\begin{aligned}\sigma^2_{A(co,2N)} = +\frac{1}{2N}\Big[&\ldots + N\sigma^2_{A(2,N)}\\ &+\frac{N}{4}\left(-\mu_{A(1,N)} + \mu_{A(2,N)} + \lambda_{B_1<A_2}\right)^2\Big]\end{aligned} \tag{B.5}$$

At this point we can assemble the contribution of the first scenario (B.4) and the second scenario (B.5) into the definition of the variance to obtain the final results:

$$\begin{aligned}\sigma^2_{A(co,2N)} &= \frac{1}{2}\left(\sigma^2_{A(1,N)} + \sigma^2_{A(2,N)}\right)\\ &\quad+\frac{1}{4}\left(\mu_{A(2,N)} + \lambda_{B_1<A_2} - \mu_{A(1,N)}\right)^2\end{aligned}$$

The case for B is analogous by swapping A and B in the formula. □

References

Agresti, A., Franklin, C., 2007. Statistics: The art and science of learning from data.

Amador, J., Ma, Y., Hasama, S., Lumba, E., Lee, G., Birrell, E., 2023. Prospects for improving password selection. In: Proceedings of the 19th USENIX Symposium on Usable Privacy and Security. SOUPS'23, pp. 263–282.

Antony, J., Viles, E., Torres, A.F., Fernandes, M.M., Cudney, E.A., 2021. Design of experiments in the service industry: results from a global survey and directions for further research. TQM J. 33 (5), 987–1000.

Aoyama, H., Yokoyama, H., 2017. Study on digital style design-robust design system for kansei using multivariate analysis and Taguchi method. In: Proceedings of the 11th International Conference on Interfaces and Human Computer Interaction. IHCI'23, IADIS, pp. 10–18.

Atta, M., Megahed, M., Saber, D., 2022. Using ANN and OA techniques to determine the specific wear rate effectors of A356 Al-Si/Al2o3 MMC. Neural Comput. Appl. 34 (17), 14373–14386.

Barron, A., 1997. Experimentation. URL: http://www.stat.yale.edu/Courses/1997-98/101/expdes.htm.

Basak, S.K., Neil, L., Reaves, B., Williams, L., 2023. What challenges do developers face about checked-in secrets in software artifacts? arXiv preprint arXiv:2301.12377.

Charness, G., Gneezy, U., Kuhn, M.A., 2012. Experimental methods: Between-subject and within-subject design. J. Econ. Behav. Organ. 81 (1), 1–8.

Chen, J., Hengartner, U., Khan, H., 2022. Sharing without scaring: enabling smartphones to become aware of temporary sharing. In: Proceedings of the 18th USENIX Symposium on Usable Privacy and Security. SOUPS'22, pp. 671–685.

Chew, J.Y., Nakamura, K., 2023. Who to teach a robot to facilitate multi-party social interactions? In: Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction. HRI'23, pp. 127–131.

Chong, C.Y., Thongtanunam, P., Tantithamthavorn, C., 2021. Assessing the students' understanding and their mistakes in code review checklists: an experience report of 1,791 code review checklist questions from 394 students. In: ICSE-SEET-21. IEEE, pp. 20–29.

Cicirello, A., Giunta, F., 2022. Machine learning based optimization for interval uncertainty propagation. Mech. Syst. Signal Process. 170, 108619.

Cummings, R., Kaptchuk, G., Redmiles, E.M., 2021. "I need a better description": An investigation into user expectations for differential privacy. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 3037–3052.

Dell'Amico, M., Hadjidimitriou, N.S., Koch, T., Petkovic, M., 2018. Forecasting natural gas flows in large networks. In: Machine Learning, Optimization, and Big Data: Third International Conference. MOD'17, Springer, pp. 158–171.

Falessi, D., Juristo, N., Wohlin, C., Turhan, B., Münch, J., Jedlitschka, A., Oivo, M., 2018. Empirical software engineering experts on the use of students and professionals in experiments. Empir. Softw. Eng. 23, 452–489.

Fleiss, J.L., 1989. A critique of recent research on the two-treatment crossover design. Control. Clin. Trials 10 (3), 237–243.

Freeman, P., 1989. The performance of the two-stage analysis of two-treatment, two-period crossover trials. Statist. Med. 8 (12), 1421–1432.

Gargiulo, F., Duellmann, D., Arpaia, P., Schiano Lo Moriello, R., 2021. Predicting hard disk failure by means of automatized labeling and machine learning approach. Appl. Sci. 11 (18), 8293.

Grieve, A., 1985. A Bayesian analysis of the two-period crossover design for clinical trials. Biometrics 979–990.

Grizzle, J.E., 1965. The two-period change-over design and its use in clinical trials. Biometrics 467–480.

Gümüş, D.B., Özcan, E., Atkin, J., Drake, J.H., 2023. An investigation of F-race training strategies for cross domain optimisation with memetic algorithms. Inform. Sci. 619, 153–171.

Huaman, N., Krause, A., Wermke, D., Klemmer, J.H., Stransky, C., Acar, Y., Fahl, S., 2022. If you {can't} get them to the lab: Evaluating a virtual study environment with security information workers. In: Proceedings of the 18th USENIX Symposium on Usable Privacy and Security. SOUPS'22, pp. 313–330.

Jedlitschka, A., Ciolkowski, M., Pfahl, D., 2008. Reporting experiments in software engineering. In: Guide to Advanced Empirical Software Engineering. Springer, pp. 201–228.

Jones, B., Kenward, M.G., 2014. Design and Analysis of Cross-over Trials. CRC Press.

Juristo, N., Moreno, A.M., 2013. Basics of Software Engineering Experimentation. Springer Science & Business Media.

Kacker, R.N., Kuhn, D.R., Lei, Y., Simos, D.E., 2021. Factorials experiments, covering arrays, and combinatorial testing. Math. Comput. Sci. 15, 715–739.

Kacker, R.N., Lagergren, E.S., Filliben, J.J., 1991. Taguchi's orthogonal arrays are classical designs of experiments. J. Res. Natl. Inst. Stand. Technol. 96 (5), 577.

Kanchana, B., Sarma, V., 1999. Software quality enhancement through software process optimization using Taguchi methods. In: Proceedings of the IEEE Conference and Workshop on Engineering of Computer-Based Systems. ECBS'99, IEEE, pp. 188–193.

Kaushik, S., Barbosa, N.M., Yu, Y., Sharma, T., Kilhoffer, Z., Seo, J., Das, S., Wang, Y., 2023. {GuardLens}: Supporting safer online browsing for people with visual impairments. In: Proceedings of the 19th USENIX Symposium on Usable Privacy and Security. SOUPS'23, USENIX Association, pp. 361–380.

Kersten, L., Mulders, T., Zambon, E., Snijders, C., Allodi, L., 2023. 'Give me structure': Synthesis and evaluation of a (network) threat analysis process supporting tier 1 investigations in a security operation center. In: Proceedings of the 19th USENIX Symposium on Usable Privacy and Security. SOUPS'23, pp. 97–111.

Kitchenham, B., Fry, J., Linkman, S., 2003. The case against cross-over designs in software engineering. In: Proceedings of the 11th IEEE International Workshop on Software Technology and Engineering Practice. STEP'03, IEEE, pp. 65–67.

Kitchenham, B., Madeyski, L., Scanniello, G., Gravino, C., 2021. The importance of the correlation in crossover experiments. IEEE Trans. Softw. Eng. 48 (8), 2802–2813.

Kühtreiber, P., Pak, V., Reinhardt, D., 2022. Replication: the effect of differential privacy communication on german users' comprehension and data sharing attitudes. In: Proceedings of the 18th USENIX Symposium on Usable Privacy and Security. SOUPS'22, pp. 117–134.

Langer, M., Siegel, R., Schilling, M., Hunsicker, T., König, C.J., 2022. An open door may tempt a saint: Examining situational and individual determinants of privacy-invading behavior. In: Proceedings of the 18th USENIX Symposium on Usable Privacy and Security. SOUPS'22, pp. 407–426.

Lee, S., Choi, W., Lee, D.H., 2021. Usable user authentication on a smartwatch using vibration. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 304–319.

Lee, K., Ryu, S., Kim, C., Seo, T., 2022. A compact and agile angled-spoke wheel-based mobile robot for uneven and granular terrains. IEEE Robot. Autom. Lett. 7 (2), 1620–1626.

Ling, C., Salvendy, G., 2007. Optimizing heuristic evaluation process in E-commerce: Use of the Taguchi method. Int. J. Hum.-Comput. Interact. 22 (3), 271–287.

Liu, E., Sun, L., Bellon, A., Ho, G., Voelker, G.M., Savage, S., Munyaka, I.N., 2023. Understanding the viability of gmail's origin indicator for identifying the sender. In: Proceedings of the 19th USENIX Symposium on Usable Privacy and Security. SOUPS'23, USENIX Association, pp. 77–95.

Malkin, N., Wagner, D., Egelman, S., 2022. Runtime permissions for privacy in proactive intelligent assistants. In: Proceedings of the 18th USENIX Symposium on Usable Privacy and Security. SOUPS'22, pp. 633–651.

Massacci, F., Papotti, A., Paramitha, R., 2024. Artefact for addressing combinatorial experiments and scarcity of subjects by balanced experimental designs. The clickable Anonymous Github link can be used to access the artefact anonymously. URL: https://zenodo.org/doi/10.5281/zenodo.10679479.

Mayer, P., Poddebniak, D., Fischer, K., Brinkmann, M., Somorovsky, J., Sasse, A., Schinzel, S., Volkamer, M., 2022. "I {don't} know why I check this...''-investigating expert users' strategies to detect email signature spoofing attacks. In: Proceedings of the 18th USENIX Symposium on Usable Privacy and Security. SOUPS'22, pp. 77–96.

McCall, M., Zeng, E., Shezan, F.H., Yang, M., Bauer, L., Bichhawat, A., Cobb, C., Jia, L., Tian, Y., 2023. Towards usable security analysis tools for {trigger-action} programming. In: Proceedings of the 19th USENIX Symposium on Usable Privacy and Security. SOUPS'23, pp. 301–320.

McCormick, M., 2022. An artificial neural network for simulation of an upflow anaerobic filter wastewater treatment process. Sustainability 14 (13), 7959.

Naiakshina, A., Danilova, A., Gerlitz, E., Smith, M., 2020. On conducting security developer studies with cs students: Examining a password-storage study with cs students, freelancers, and company developers. In: Proceedings of the 40nd HI Conference on Human Factors in Computing Systems. CHI'20, pp. 1–13.

Naiakshina, A., Danilova, A., Tiefenau, C., Herzog, M., Dechand, S., Smith, M., 2017. Why do developers get password storage wrong? A qualitative usability study. In: Proceedings of the 24th ACM SIGSAC Conference on Computer and Communications Security. CCS'17, pp. 311–328.

Naiakshina, A., Danilova, A., Tiefenau, C., Smith, M., 2018. Deception task design in developer password studies: Exploring a student sample. In: Proceedings of the 14th USENIX Symposium on Usable Privacy and Security. SOUPS'18, pp. 297–313.

Papotti, A., Paramitha, R., Massacci, F., 2022. On the acceptance by code reviewers of candidate security patches suggested by automated program repair tools. arXiv preprint arXiv:2209.07211.

Rader, E., 2023. Data privacy and pluralistic ignorance. In: Proceedings of the 19th USENIX Symposium on Usable Privacy and Security. SOUPS'23, USENIX Association, Anaheim, CA, pp. 457–471, URL: https://www.usenix.org/conference/soups2023/presentation/rader.

Rainer, A., Wohlin, C., 2022. Recruiting credible participants for field studies in software engineering research. Inf. Softw. Technol. 151, 107002.

Rong, G., Li, J., Xie, M., Zheng, T., 2012. The effect of checklist in code review for inexperienced students: An empirical study. In: Proceedings of the 25th IEEE/ACM International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET'12). IEEE, pp. 120–124.

Roy, R.K., 2010. A Primer on the Taguchi Method. Society of Manufacturing Engineers.

Salman, I., Misirli, A.T., Juristo, N., 2015. Are students representatives of professionals in software engineering experiments? In: Proceedings of the 37th IEEE/ACM International Conference on Software Engineering. ICSE'15, Vol. 1, IEEE, pp. 666–676.

Sebastio, S., Baranov, E., Biondi, F., Decourbe, O., Given-Wilson, T., Legay, A., Puodzius, C., Quilbeuf, J., 2020. Optimizing symbolic execution for malware behavior classification. Comput. Secur. 93, 101775.

Seo, C., Lee, K., Son, D., Seo, T., 2021. Robust design of a screw-based crawling robot on a granular surface. IEEE Access 9, 103988–103995.

Serafini, R., Gutfleisch, M., Horstmann, S.A., Naiakshina, A., 2023. On the recruitment of company developers for security studies: Results from a qualitative interview study. In: Proceedings of the 19th USENIX Symposium on Usable Privacy and Security. SOUPS'23, pp. 321–340.

Shepperd, M., Bowes, D., Hall, T., 2014. Researcher bias: The use of machine learning in software defect prediction. IEEE Trans. Softw. Eng. 40 (6), 603–616. http://dx.doi.org/10.1109/TSE.2014.2322358.

Sjoberg, D.I., Anda, B., Arisholm, E., Dyba, T., Jorgensen, M., Karahasanovic, A., Koren, E.F., Vokác, M., 2002. Conducting realistic experiments in software engineering. In: Proceedings of International Symposium on Empirical Software Engineering. ESEM'02, IEEE, pp. 17–26.

Solouma, N., El Berry, A., 2022. A predictive reliability model to assess the performance of photovoltaic systems. Appl. Sci. 12 (6), 2885.

Tahaei, M., Vaniea, K., 2022. Recruiting participants with programming skills: A comparison of four crowdsourcing platforms and a CS student mailing list. In: Proceedings of the 42nd HI Conference on Human Factors in Computing Systems. CHI'22.

Tanco, M., Viles, E., Ilzarbe, L., Alvarez, M.J., 2008. Is design of experiments really used? A survey of basque industries. J. Eng. Des. 19 (5), 447–460.

Tyasnurita, R., Özcan, E., John, R., 2017. Learning heuristic selection using a time delay neural network for open vehicle routing. In: Proceeding of the IEEE Congress on Evolutionary Computation. CEC'17, IEEE, pp. 1474–1481.

Vegas, S., Apa, C., Juristo, N., 2015. Crossover designs in software engineering experiments: Benefits and perils. IEEE Trans. Softw. Eng. 42 (2), 120–135.

Vescan, A., Pintea, A., Linsbauer, L., Egyed, A., 2021. Genetic programming for feature model synthesis: a replication study. Empir. Softw. Eng. 26, 1–29.

Volkamer, M., Kulyk, O., Ludwig, J., Fuhrberg, N., 2022. Increasing security without decreasing usability: A comparison of various verifiable voting systems. In: Proceedings of the 18th USENIX Symposium on Usable Privacy and Security. SOUPS'22, pp. 233–252.

Whalen, T., Meunier, T., Kodali, M., Davidson, A., Fayed, M., Faz-Hernández, A., Ladd, W., Maram, D., Sullivan, N., Wolters, B.C., et al., 2022. Let the right one in: Attestation as a usable {captcHA} alternative. In: Proceedings of the 18th USENIX Symposium on Usable Privacy and Security. SOUPS'22, pp. 599–612.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A., 2012. Experimentation in Software Engineering. Springer Science & Business Media.

Yoshikawa, R., Ochiai, H., Yatani, K., 2022. {DualCheck}: Exploiting human verification tasks for opportunistic online safety microlearning. In: Proceedings of the 18th USENIX Symposium on Usable Privacy and Security. SOUPS'22, pp. 19–37.

Zheng, S., Becker, I., 2022. Presenting suspicious details in {user-facing} E-mail headers does not improve phishing detection. In: Proceedings of the 18th USENIX Symposium on Usable Privacy and Security. SOUPS'22, pp. 253–271.

Zibaei, S., Malapaya, D.R., Mercier, B., Salehi-Abari, A., Thorpe, J., 2022. Do password managers nudge secure (random) passwords? In: Proceedings of the 18th USENIX Symposium on Usable Privacy and Security. SOUPS'22, pp. 581–597.

Zibaei, S., Salehi-Abari, A., Thorpe, J., 2023. Dissecting nudges in password managers: Simple defaults are powerful. In: Proceedings of the 19th USENIX Symposium on Usable Privacy and Security. SOUPS'23, pp. 211–225.

Fabio Massacci is a professor at the University of Trento, Trento, 38123, Italy, and Vrije Universiteit, Amsterdam, 1081 HV, The Netherlands. Massacci received a Ph.D. in computing from the University of Rome "La Sapienza". He received the IEEE Requirements Engineering Conference Ten Year Most Influential Paper Award on security in sociotechnical systems. He participates in the FIRST special interest group on the Common Vulnerability Scoring System and the European pilot CyberSec4Europe on the governance of cybersecurity. He coordinates the European AssureMOSS and Sec4AI4Sec projects. He is a Member of IEEE, the Association for Computing Machinery, and the Society for Risk Analysis. Contact him at fabio.massacci@ieee.org.

Aurora Papotti is a Ph.D. student at Vrije Universiteit, Amsterdam, 1081 HV, The Netherlands. She attended the EIT Digital Master School in Cyber Security. She received her master's degree in informatics with distinction from the University of Turku, Turku, 2014, Finland, and the University of Trento, Trento, 38123, Italy. Her main research interest is in software security, focusing on the empirical assessment of tools that automatically detect and fix vulnerabilities, and how developers cope with these tools while using them to code review. She started to actively serve the research community, such as by being a student volunteer (ICSE'22). Contact her at a.papotti@vu.nl.

Ranindya Paramitha is a Ph.D. student at the University of Trento, Trento, 38123, Italy. She received her master's degree in informatics with distinction from Institut Teknologi Bandung, Bandung, 40132, Indonesia. Her main research interest is in software security, focusing on empirical analysis of secure software ecosystems, mining software repositories, and how developers can apply security. She is involved in European Projects named AssureMOSS and Sec4AI4Sec. She has also started to actively serve the research community in several IEEE/ACM International Conferences/Workshops, such as by being a student volunteer (ICSE'22) and program committee (ICSE SVM'23). Contact her at ranindya.paramitha@unitn.it.