5-11-2023

How Much Are Machine Assistants Worth? Willingness to Pay for Machine Learning-Based Software Testing

Maren F. Mehler
Technical University of Darmstadt, maren.mehler@tu-darmstadt.de

Oliver A. Vetter
Technical University of Darmstadt, oliver.vetter@tu-darmstadt.de

HOW MUCH ARE MACHINE ASSISTANTS WORTH? WILLINGNESS TO PAY FOR MACHINE LEARNING-BASED SOFTWARE TESTING

Research Paper

Maren F. Mehler, Technical University of Darmstadt, Germany, mehler@is.tu-darmstadt.de

Oliver A. Vetter, Technical University of Darmstadt, Germany, vetter@is.tu-darmstadt.de

Abstract

Machine Learning (ML) technologies have become the foundation of a plethora of products and services. While the economic potential of such ML-infused solutions has become irrefutable, there is still uncertainty on pricing. Currently, software testing is one area to benefit from ML services assisting in the creation of test cases; a task both complex and demanding human-like outputs. Yet, little is known on the willingness to pay of users, inhibiting the suppliers' incentive to develop suitable tools. To provide insights into desired features and willingness to pay for such ML-based tools, we perform a choice-based conjoint analysis with 119 participants in Germany. Our results show that a high level of accuracy is particularly important for users, followed by ease of use and integration into existing environments. Thus, we not only guide future developers on which attributes to prioritize but also which characteristics of ML-based services are relevant for future research.

Keywords: Software Testing, Machine Learning, Choice-based Conjoint Analysis, Willingness to Pay.

1 Introduction

An enormous amount of novel machine learning (ML) services has recently gained traction in the market. However, pricing these services is notoriously difficult and how much users are willing to pay is uncertain, especially since it is not clear which attributes are important to the users of ML services. Various use cases of ML services exist that support users in their day-to-day tasks and support independent decision making (Peters, Pumplun and Buxmann, 2020; Berente et al., 2021; von Wedel and Hagist, 2022). A so far rarely explored use case of a ML-infused service supporting decision making, which shows high potential, is the application of ML in the context of software testing. Software testing is an essential part of the software development process as it ensures the quality of the software (Zhao, Hu and Gong, 2021). Due to the high share of human labor in form of experience and knowledge, software testing consumes a lot of time and hence leads to high costs (Enoiu, Tukseferi and Feldt, 2020; Guveyi, Aktas and Kalipsiz, 2020). According to a survey among CIOs, 23% of the annual IT expenditure is spent yearly on software quality assurance. Although the costs decreased slightly in recent years (cf. 35% in 2015), approximately a quarter of the budget remains a significant percentage (Walgude and Natarajan, 2019). One approach for reducing time and costs while ensuring high quality is the use of technologies based on machine learning (in the following referred to ML-based software testing tools) (Hourani, Hammad and Lafi, 2019).

As software becomes more complex in general and the range of available software increases, the number of tests required also increases and needs to be more comprehensive. Costs and effort increase especially towards the end of the development process and even after the software has been deployed (Madera and Tomoń, 2017). In addition, safety-critical software and ML software make testing more complicated (Ebert, Bajaj and Weyrich, 2022). Furthermore, software companies aim to be cost-efficient and rapid in the development of software (Storey, Houck and Zimmermann, 2022). These objectives reveal great potential for providers of respective tools (Durelli et al., 2019). Although some ML-based software testing tools already exist in practice and many algorithms have been presented and tested in theory (Shafiq et al., 2021), they still do not seem to be established on the market. For example, algorithms that generate test data, select test cases, or even estimate the duration of tests can be found in theory. In addition, these algorithms have been proven to be profitable based on individual use cases (Rathore and Kumar, 2021), especially since traditional testing – i.e., without the use of ML-based tools – is prone to human error (Durelli et al., 2019). Nevertheless, this emerging technology does not yet seem to have been adopted by companies, possibly because of their heterogeneous needs and individual requirements that such a tool must fulfill.

However, companies seem to have a considerable amount of budget available, which is already being spent on testing (Ebert, Bajaj and Weyrich, 2022). Therefore, a solution for low-cost, fast, and still high-quality testing would be ML-based tools that relieve the software testers and thus their respective companies (Shen et al., 2018). The objective of this paper is to provide first insights into the willingness to pay for ML services used in their day-to-day work. This leads to the following research questions:

1. Which attributes of ML-based software testing tools determine the willingness to pay of users?
2. How much are users willing to pay for ML-based software testing tools?

Our study can thus contribute by identifying attributes for ML-based software testing tools besides those already existing in the literature and rank them according to their importance for potential users. In addition, specific prices are linked to the attributes and the resulting product combinations. Those attributes and prices can then be generalized and used for ML services.

The remainder of the paper is structured as follows: First, an overview of willingness to pay for ML is given and the fundamentals and the most researched types of ML-based software testing in the literature are explained in section two. Then, in section three, the methodology of the conjoint analysis is presented, whereby a focus is on the literature search and Delphi study necessary for the determination of the characteristics. This is followed by the analysis of the results in section four, focusing on the results of the conjoint analysis as well as providing insights on possible user segments, which are then discussed in section five. Finally, the paper ends with a conclusion in section six.

2 Theoretical Background

In the following, first, an overview of previous research on the willingness to pay in the field of ML is shown. Then, ML-based software testing is introduced by presenting existing literature and defining the use case of test case generation.

2.1 Willingness to Pay for ML

Willingness to pay (WTP) describes the maximum amount a person is willing to pay for a product or service (Kalish and Nelson, 1991). The consumer is indifferent to buying the product for this amount (Moorthy, Ratchford and Talukdar, 1997). Knowing the WTP, both the sales volume and the profit margin can be maximized, and the concept is therefore used for pricing (Le Gall-Ely, 2009). Since the needs of individuals are subjective and influence WTP, determining the WTP remains challenging. Several approaches, such as experiments, direct surveys, or indirect surveys like conjoint analyses, exist to quantify consumers' willingness to pay (Breidert, Hahsler and Reutterer, 2006).

Before several approaches are pointed out how WTP for ML has already been addressed, it is important to define and frame the term ML first. In this paper, artificial intelligence (AI) is understood as an overarching term for ML, and ML makes decisions based on experiences (Russell and Norvig, 2021). In this paper, we restrict ourselves to the term ML.

In the field of ML, only a few papers examined the willingness to pay. One cluster addresses ML-designed products: Although people were first willing to pay more for t-shirts with an ML-generated print, the willingness to pay decreases slightly when people are informed that the print was created by a

ML algorithm instead of a human (Sohn et al., 2021). Zhang, Bai and Ma (2022) obtain a similar result: Consumers are willing to pay more for t-shirts designed by a ML algorithm. Most similar to our focus are the papers regarding the willingness to pay for ML in the medical context. For example, von Wedel and Hagist (2022) examine the experience and willingness to pay for ML-based assistance systems in the medical field. Due to the absence of adequate tools, they first identify fundamental characteristics such as the provider as key attributes for the willingness to pay. Afterward, they were able to show that the willingness to pay for such tools exists. However, due to the specific context and possible regulations in the medical context, these papers provide initial insights, and we are expanding to consider a novel use case. Other papers on WTP for ML in a broader sense are Peters, Pumplun and Buxmann (2020) who can demonstrate in their study that consumers are particularly willing to pay more for transparency features in intelligent systems. In addition, Morita and Managi (2020) can show that consumers in Japan are willing to pay more for an autonomous vehicle, but the willingness remains lower than in the U.S. Finally, a research area on ML used for the calculation of the WTP exists (e.g., (Ramsey and Bergtold, 2021; Nguyen et al., 2022)).

One important method for determining WTP is a conjoint analysis. Conjoint analysis has been successfully applied for many years and was continuously improved (Green and Srinivasan, 1978). The choice-based conjoint analysis as an extension has been applied to many contexts in the lasts years like smartphone mobile operating systems (Böhm, Adam and Farrell, 2015), platform as a service solutions (Giessmann and Stanoevska-Slabeva, 2013), content preferences of newspaper readers (Kanuri, Thorson and Mantrala, 2014), web identity management systems (Roßnagel et al., 2014), augmented reality in production environments (Schuir and Teuteberg, 2021), smart meters (Albani, Domigall and Winter, 2017), digital assistants (Ebbers et al., 2021), in-vehicle assistants (Mihale-Wilson, Zibuschka and Hinz, 2019). Similar to our use case, most of these papers first identify attributes, and then determine the WTP using choice-based conjoint analysis. While WTP is usually measured for consumer products, the considered product of our use case is intended for employees. Goebel et al. (2018), for example, examined the WTP of purchasing managers successfully, but noted that WTP methods are mostly applied to consumers. Since our aim is to determine the preferred features of the product, we perform a choice-based conjoint analysis with employees.

2.2 Machine Learning-based Software Testing

Software testing is a crucial part of the process of software development. It is a way of measuring and determining the quality of an IT system (referred to as a system under test). Depending on the project phase, single functional parts are tested as part of a unit test, or the integration of several parts, or even the entire system is tested. The primary goal is to uncover errors in order to improve and correct them. An error or bug occurs when the software does not meet the requirements or expectations of the users (Pandey, Rathee and Tripathi, 2020). It is important to emphasize that a system can never be fully tested, especially only relying on human resources (Ahmed and Zamli, 2011). Therefore, it is recommended to perform diverse and as many tests as possible (Segura and Zhou, 2018). To support the software testers in their day-to-day work, numerous ML-based software testing tools have been developed in theory (e.g., (Dejaeger, Verbraken and Baesens, 2013; Tahvili et al., 2018; Kesri, Nayak and Ponnalagu, 2021)). In the context of this paper, ML-based software testing can support manual testing, but especially improve automated testing.

Both in literature and practice, two concepts should be distinguished (Gezici and Tarhan, 2022): Software testing, where ML provides support – this is the focus of this paper – and the testing of ML algorithms. The latter may also be facilitated by ML and could utilize the same algorithms, but the use case is highly specific and could lead to confusion in the survey conducted as part of this paper. In the following, ML-based software testing is thus referred to as the testing of ordinary software with the support of ML.

There is neither a single nor a standardized solution for ML-based software testing since the variety of different tests and testing approaches is matched by the variety of algorithms and tools that exist or have been proposed in the literature. They differ, for example, in terms of the applied AI algorithms – neural networks, clustering such as kNN or decision trees are commonly utilized (Lima, da Cruz and Ribeiro, 2020). In the context of this paper, the available ML-based software testing tools are categorized according to their areas of application (Durelli et al., 2019; Hourani, Hammad and Lafi, 2019; Shafiq et al., 2021). The following exist:

- Test Case Generation – Generation of test cases from requirements or code using Natural Language Processing (NLP) (Memon, Pollack and Soffa, 2001; Dadkhah, Araban and Paydar, 2020; Kesri, Nayak and Ponnalagu, 2021).
- Test Data Generation – Generation of test data, meaning the automatic identification of input data for test cases, e.g., through specific ML algorithms like backtracking, AI planning, or swarm intelligence (Gupta et al., 2004; Xing et al., 2014; Rabbi, Mamun and Islam, 2018).
- Test Oracle Generation – Test oracles are the set of program results depending on the input data (Barr et al., 2015); these can also be extracted automatically using classification, support vector machines, or neural networks (Gholami et al., 2018; Khatibsyarbini et al., 2021).
- Test Case Selection/ Prioritization – To save time and costs, it is convenient to perform only certain tests or to begin with the most important ones. The selection or prioritization can be supported by ML algorithms in the areas of classification, clustering, or swarm intelligence (Kazmi et al., 2017; Khatibsyarbini et al., 2021; Pan et al., 2022).
- Test Estimation – ML can also be integrated into secondary tasks such as estimating the remaining duration of tests, e.g., using regression. This can support managers in their planning of the remaining software project or test case selection/prioritization (Tahvili et al., 2018).
- Software Fault Prediction/ Bug Prediction/ Defect Prediction – All terms are used synonymously and describe the prediction in which modules of the software errors are likely. Algorithms such as dimensionality reduction, Naive Bayes, or neural networks are applied here (Gondra, 2008; Catal and Diri, 2009; Dejaeger, Verbraken and Baesens, 2013; Li, Jing and Zhu, 2018).

Throughout this paper, the focus is primarily on test case generation, since this use case is a common task for software developers and software testers. Additionally, the adoption of ML for test case generation is considered to be the most suitable use case by the experts in our – later described in detail – Delphi study. Therefore, this use case will be presented in more detail.

It is extremely time-consuming to manually generate test cases from requirements, as these often do not match a given standard since they are written by several domain experts. Thus, natural language processing (NLP) algorithms can support software testers by reducing time and costs, as well as producing results without human errors (Kesri, Nayak and Ponnalagu, 2021). In addition, test cases can be derived directly from code. Since many dependencies exist between the system modules, the generation of test cases can become very complex. Therefore, a form of knowledge management is often applied during the generation of test cases (Dadkhah, Araban and Paydar, 2020; Kesri, Nayak and Ponnalagu, 2021). Furthermore, automated planning for graphical user interfaces was utilized for test case generation (Memon, Pollack and Soffa, 2001).

ML appears to have arrived in software engineering research, especially in software testing (Shafiq et al., 2021) and numerous opportunities exist to reduce software testing costs and time and minimize errors. However, although many tools assist programmers, human testers are not obsolete and will not be replaced. Many tasks, such as defining the test objectives or analyzing the test results, remain the responsibility of individuals, which means that the focus of ML-based software testing tools is on providing support (Itkin, Novikov and Yavorskiy, 2019; King et al., 2019; Marselis, 2020).

Although a substantial amount of literature on potential algorithms already exists and an increasing number of tools are being developed, the economic aspects of ML-based software testing have hardly been examined. Most papers present a new algorithm that is evaluated based on ML-specific criteria such as accuracy (e.g., (Dejaeger, Verbraken and Baesens, 2013; Tahvili et al., 2018; Kesri, Nayak and Ponnalagu, 2021)). Little literature can be found on whether ML-based software testing is profitable. Rathore and Kumar (2021) perform a cost-benefit analysis on their algorithm and can demonstrate for almost all datasets that the application of their algorithm for fault prediction can reduce software testing costs. Herbold (2021) addresses the costs and profit of software defect prediction in his paper. He presents a cost model which includes different aspects such as the costs for software defects which are discovered after the release of the software. To the best of our knowledge, no research goes beyond the focus on costs and investigates the willingness to pay for such solutions.

3 Methodology

To answer the research questions, we conducted a conjoint analysis, as such analysis is used "to measure the joint effects of a set of independent variables on the ordering of a dependent variable" (Green and Rao, 1971, p. 355). This involves determining the preferences of (future) users and generating predictions about purchasing behavior (Green and Rao, 1971). In this paper, we apply a choice-based conjoint analysis, originally by Louviere and Woodworth (1983), which differs from the traditional conjoint analysis by presenting respondents with different product combinations several times in a row and asking the respondents to decide between them (see Figure 1) and thus selecting the product with the highest personal utility (Ebbers et al., 2021). This results in more realistic data through the questioning of potential users, since the method of questioning leads to real preferences (Mihale-Wilson, Zibuschka and Hinz, 2019).

Before conducting a choice-based conjoint analysis, the product attributes and their characteristics (levels) must be defined. To determine the attributes and levels, a two-step approach is chosen: First, a structured literature search according to vom Brocke et al. (2009) is conducted, followed by a Delphi study according to Skinner et al. (2015) to determine additional attributes and levels and eventually select them.

3.1 Literature review

The literature search was conducted during the first quarter of 2022. In the **first phase** ("definition of review scope") we follow Cooper's taxonomy as suggested by vom Brocke et al. (2009) (see Table 1). We *focus* on the *research outcome* of the considered papers. Our *goal* is the *integration* of our results. We take a *neutral perspective*. The *coverage* is *representative* and the *organization* is *conceptual*. Our *audiences* are *specialized scholars and practitioners* (Cooper, 1986; vom Brocke et al., 2009).

Characteristic	**Categories**			
Focus	Research outcome	Research methods	Theories	Applications
Goal	Integration	Criticism	Central issues	
Organization	Historical	Conceptual	Methodological	
Perspective	Neutral Perspective	Espousal of position		
Audience	Specialized scholars	General scholars	Practitioners/ politicians	General public
Coverage	Exhaustive	Exhaustive and selective	Representative	Central/ pivotal

Table 1. Taxonomy of literature review following Cooper (1986).

For the **second phase** ("conceptualization of topic") we got an overview of the different concepts of ML-based software testing using a general Google search and Google Scholar. We started by using the terms "artificial intelligence" and "machine learning" as synonyms in combination with "software testing". We quickly realized that there is not one, but many different approaches on the one hand regarding the ML algorithms used, on the other hand regarding the different possible applications of ML in software testing. Overviews provided by Durelli et al. (2019), Hourani, Hammad and Lafi (2019) and

Shafiq (2021) were used for the background section of ML-based software testing. Our search also revealed that besides "software testing", the terms "quality assurance" and "quality management" are often used. However, the combination of "quality assurance" and "artificial intelligence" or "machine learning" also results in many articles on quality assurance in manufacturing using AI. To limit the search, we added "software" to our search term.

Finally, the search string "((“software testing” OR “quality assurance” OR “quality management”) AND (“artificial intelligence” OR “machine learning”) AND “software”)" resulted, which is used in the **third phase** ("literature search"). The search term gives 827 results in Web of Science and 158 results in Ebscohost Business Source Premier, respectively. After removing duplicates, 953 papers are left. In the next step, the titles and abstracts of those papers were analyzed. In this process, we excluded papers with 1) very specific use cases such as medicine, farming, manufacturing, etc. as those mainly utilize ML algorithms to increase the quality in general, not software quality. In addition, we excluded papers that 2) focus on only the testing of ML algorithms as those are – as already argued in the section on ML-based software testing – out of our scope. This resulted in 468 peer-reviewed papers. While reading these papers we noticed that they primarily focus on the application, implementation, or benchmarking of a specific or new algorithm (e.g., (Bouktif, Sahraoui and Ahmed, 2014; Rabbi, Mamun and Islam, 2018; Tahvili et al., 2018)) and thus 3) are not focusing on economic aspects and user demands of ML-based software testing. After excluding those papers, 18 papers are left; mainly literature reviews.

In the **fourth phase** ("literature analysis and synthesis"), all remaining papers were read and analyzed for possible attributes for the conjoint analysis. However, the papers rarely outlined attributes, which might be essential for ML-based software testing tools. The following features could be extracted:

- Accuracy, as a measure of the correctness of the tool (Briand, 2008)
- Additional training data needed to implement the tool successfully in the company (Arora, Tetarwal and Saha, 2015; Durelli et al., 2019)
- Ease of use of the tool, e.g., with a user-friendly interface (Dejaeger, Verbraken and Baesens, 2013; Arora, Tetarwal and Saha, 2015)
- Explainability, as the ability to explain the predictions (Bouktif, Sahraoui and Ahmed, 2014; Schieferdecker, 2020)
- Possibility for users to intervene during the calculations of the tool (Durelli et al., 2019)
- ML skills needed to use the tool (Poth, Beck and Riel, 2019)

The **last phase** ("research agenda") is addressed in the discussion (section 5).

3.2 Delphi Study

However, the above-mentioned factors are not sufficient to perform a conjoint analysis as levels and prices are missing. Therefore, a Delphi study based on Skinner et al. (2015) was conducted in June and July 2022. A Delphi study is useful when the opinion of a panel of independent experts is needed on a specific topic, as it is required to determine attributes, levels, and prices. The fifteen experts selected were software testers, in particular test/ quality assurance automation engineers including six experts with decision-making responsibilities. Half of the fifteen experts have between three months and five years of experience with ML-based software testing, particularly in the areas of test case generation, test data generation, and fault/bug/defect prediction. The other half were experienced software testers without specific knowledge of ML. This gives a good mix of both experts in the field of ML-based software testing and experts in general software testing, but inexperienced users regarding ML-based software testing.

In sequential rounds, the experts were asked via emails to participate in an online survey about features, their importance, possible levels, and prices. For the specific case of test case generation – as this was most familiar to the experts – attributes such as accuracy, adaptability, cost-saving, community, documentation, ease of use, explainability, flexibility, integration, reliability, security, and test case coverage were listed. The three which ranked highest (~10% of all mentioned attributes following Skinner (2015)) were accuracy, integration, and ease of use. The derived levels are shown in Table 2. Willingness to pay for one license per month indicated by the participants ranged from €0.99 to €2000. Here the median of €150 was chosen and the prices of €50, €150, and €250 were derived. In addition to the statements of the experts, pricing of existing tools was referred to for comparison. The tools available on the market are quite comprehensive and vary in terms of what they offer, so only a rough price range can be given here. Since ML-based software testing tools are still new, many companies do not state their pricing on the website, possibly because they are determining their customers' willingness to pay. Thus, for example, Github offers the open source tool Github Copilot, which is available for business users for $19 per month per user. Here, the focus is on the completion of code in particular (GitHub, 2023). Perfecto offers a license starting from $125 per month which provides in particular automatic debugging (Perfecto, 2023). Sofy offers for $549 per month, among other things, test case selection/prioritization (Sofy, 2023). Thus, the pricing of existing tools confirms the prices from the experts. In the last round of the Delphi study, the consensus of the participants was confirmed.

Attributes	**Attribute Levels**
Accuracy	Low (90%)
	Moderate (95%)
	High (99%)
Integration	The tool is **separated** from the programming environment.
	The tool is **integrated** into the programming environment.
Ease of Use	The tool **takes some time to get used to** and requires some familiarization.
	The tool is **simple to use** with a user-friendly interface.
Price per license per month	€50
	€150
	€250

Table 2. Attributes and attribute levels for the conjoint analysis.

During the determination of the attributes and levels, attention was also given to ensure that they meet certain requirements for the conjoint analysis, which are briefly explained in the following. The requirement of relevance is fulfilled by the prioritization of attributes and levels by the participants themselves; independence was ensured in the definition of attributes and levels in each round for the participants; symmetry in the number of levels was considered in the derivation from the Delphi study. Finally, the range of attributes chosen by the participants must cover the possible attributes. However, this was only possible to a limited extent, since many more attributes were identified, but cannot be all integrated into the conjoint analysis (Albani, Domigall and Winter, 2017).

3.3 Conjoint Analysis

After determining the attributes and levels through the iterative questioning of experts in the Delphi study, the choice-based conjoint analysis was designed using the software tool Conjointly (Conjointly, 2022) and integrated into an online survey. The survey is structured as follows: First, an introduction to ML-based software testing and in particular to the use case of test case generation was presented with a focus on the attributes and their levels (see Table 2). Then, the participants were randomly assigned 12 combinations according to the scheme in Figure 1 with the task of selecting the best combination for them or none at all. This "none" option is a special design choice that leads to a balanced response behavior (Gensler et al., 2012). Finally, we collected demographic data such as age, gender, and software testing experience, as well as company size.

After developing the questionnaire, we conducted a pretest among IS researchers independent of this project for clarity, framing, and the time needed to participate. Subsequently, small details were added for better understanding, such as a visual representation of the levels. Finally, the participants were approached by a research institute, as the target group is very specific. It was ensured that the participants had experience in software testing.

Integration	The tool is **integrated** into the programming environment	The tool is **integrated** into the programming environment	The tool is **separated** from the programming environment	None of them
Accuracy	Low (90%)	High (99%)	Moderate (95%)	
Ease of use	The tool is **simple to use** with a user-friendly interface	The tool **takes some time to get used to** and requires some familiarization	The tool is **simple to use** with a user-friendly interface	
Price per license per month	€150	€250	€50	

Figure 1. Sample choices for conjoint analysis.

A total of 119 software testers participated in the survey conducted in September 2022. Participations that seemed unrealistic due to a short response time and duplicates were eliminated directly by the survey tool. On average, the duration of the survey took a little over 7 minutes. By recruiting the participants through a market research institute, it was ensured that all the participants work in the quality assurance unit of an IT department, i.e., software testing, in Germany, which is important for the price range.

		#	%
Gender	Male	66	55.5
	Female	53	44.5
	Divers	0	0.0
Age	18-25	10	8.4
	26-35	50	42.0
	36-45	32	26.9
	46-55	19	16.0
	56-65	6	5.0
	Over 66	2	1.7
Experience Software Testing in years			
	Less than 1	18	15.1
	1-2	31	26.1
	3-5	38	32.0
	More than 6	32	26.9

		#	%
Number of Employees	Less than 100	29	24.4
	100-499	36	30.3
	500-999	20	16.8
	1000-2499	21	17.6
	2500-9999	7	5.9
	Over 10000	6	5.0
Experience Test Case Generation in years	Less than 1	29	24.4
	1-2	45	37.8
	3-5	22	18.5
	More than 6	23	19.3
Years employed in current company	Less than 1	9	7.6
	1-2	34	28.6
	3-5	36	30.3
	More than 6	40	33.6

Table 3. Demographic data and control variables.

The gender distribution is balanced with 55.5% of participants identifying as male, and the remaining as female. Almost half (42%) of the participants are between 26 and 35 years old; a quarter (26.9%) are in the range of 36-46 years and the remaining participants are split among the other age groups (see Table 3). A third (32%) of the participants have three to five years of experience in the field of software testing; only 15% have less than one year of experience. As expected, experience in the area of test case generation is slightly lower with about 40% having one to two years of experience and about a quarter

(24.4%) having less than one year of experience. The respondents have also been employed in their current company for quite a long time (63.9% at least three years). Regarding the size of the company, expressed as the number of employees, no clear trend can be observed; only large (> 2,500 employees) and very large companies (> 10,000 employees) are hardly represented.

4 Results

Figure 2 shows the relative importance of the attributes. Besides the most important attribute, price per license per month, participants tend to have a special interest in the accuracy of the tool. Integration and ease of use seem to be similarly relevant to the participants.

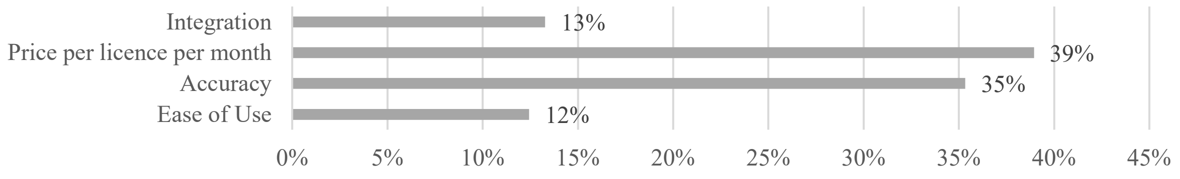

Figure 2. Relative importance of the attributes.

The respective level part worth (often referred to as level values, conjoint analysis utilities, or attribute importance scores) are shown in Figure 3, which reflect the preferences of an average consumer. More preferred levels receive a higher rating than less preferred levels. The individual levels per attribute add up to zero. The most preferred level is an accuracy of 99%; the second most is a price of €50 per license per month. The least preferred level is an accuracy of 90% and the second least preferred is a price of €250.

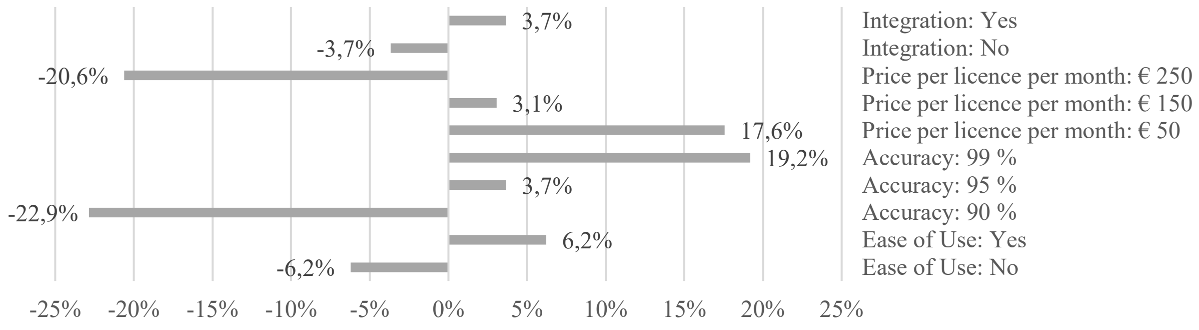

Figure 3. Level part worth.

Next, Figure 4 gives the distribution of preferences for the respective levels. For both integration and ease of use, respondents are nearly indifferent overall with about half selecting “yes” and half “no”. In addition, half of the participants chose €50 per license per month as the preferred level, and also half 99% as the most preferred accuracy.

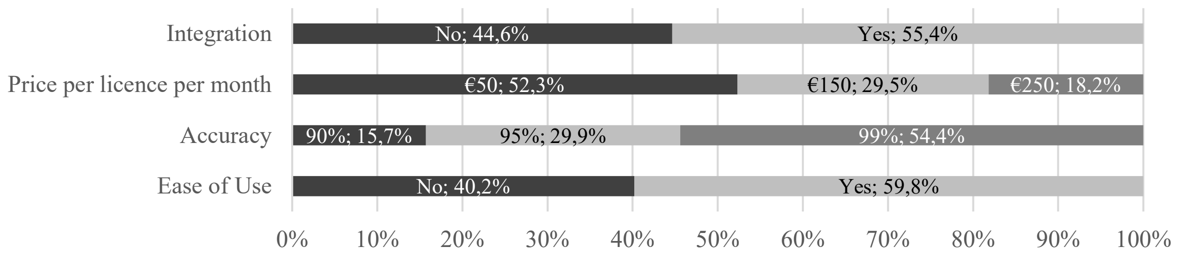

Figure 4. Distribution of preferences for the respective levels.

Combining the attributes and respective levels into products, it can be observed that the most preferred product has the following attributes: A simple-to-use tool with a user-friendly interface, an accuracy of 99%, and possible integration into the respective programming environment for €50 per license per month. This combination could generally be described as the "best" product, since all attributes achieve the highest level, while the price is the lowest.

Based on this result, the marginal willingness to pay is determined. It describes the amount that participants are willing to pay for a certain level of an attribute, i.e., the additional amount they are willing to pay to switch from level A to a superior level B (see Figure 5).

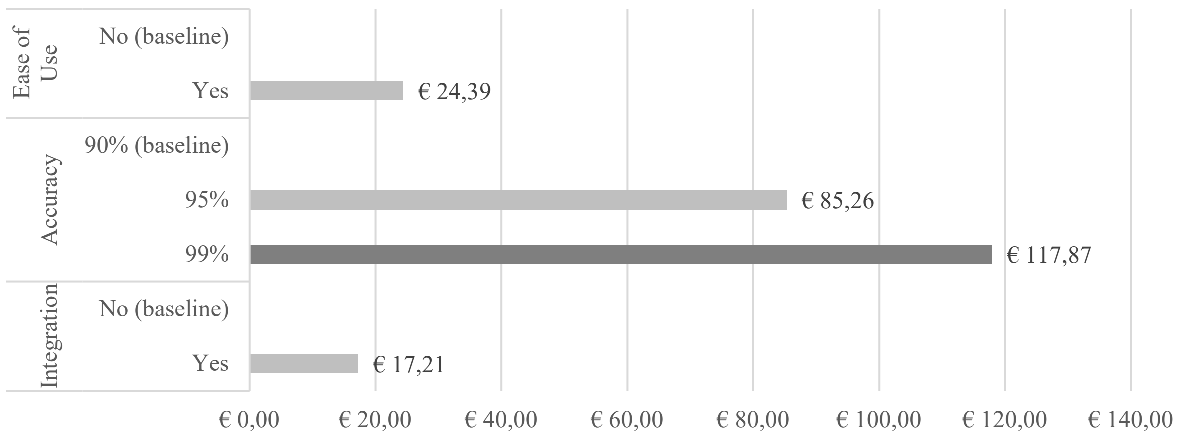

Figure 5. Marginal willingness to pay.

To the median user, an easy-to-use tool is worth as much as a price reduction of €24.39. To a median person, an increase in accuracy from 90% to 95% or 99% is worth as much as a decrease in price from €85.26 and €117.87, respectively. And to the median person, the possible integration of the tool into a programming environment is worth as much as a reduction of the price by €17.21.

Finally, the marginal willingness to pay will be examined by dividing the participants into customer segments according to the demographic data gender, age, experience in software testing and the control variable size of the company as these provide the most interesting results. However, the findings must be interpreted with caution, as they are derived from a smaller number of data points. Figure 6 shows the marginal willingness to pay according to gender. The baseline willingness to pay is omitted, as it is always zero. Interestingly, women are willing to pay more for ease of use, but less for integration. They are also willing to pay almost twice as much as men for 99% accuracy.

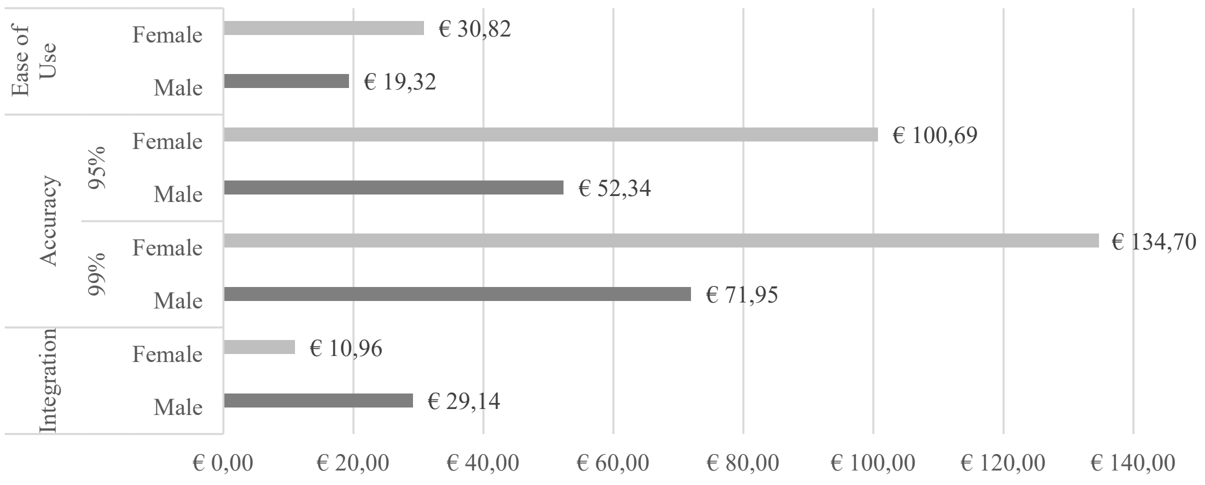

Figure 6. Marginal willingness to pay according to gender.

Likewise, the segmentation by age (see Figure 7) shows large differences. The age group between 36 and 45 is particularly willing to pay a great amount for high accuracy. When it comes to integration, it can be seen that people over 56 – participants between 56 and 65 and over 65 were grouped as the small number is not representative – have a negative willingness to pay.

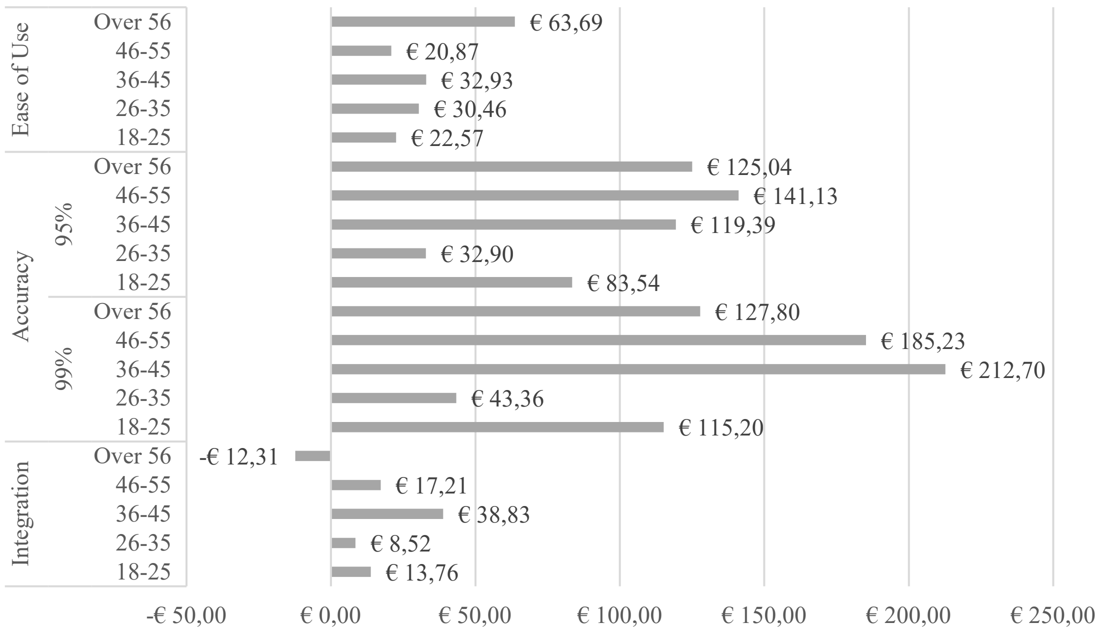

Figure 7. Marginal willingness to pay according to age.

Lastly, the segmentation according to the years of experience and to the number of employees cannot be included as a figure due to page limitations, but show interesting results. First, regarding the experience: It shows that experienced employees in particular are willing to pay more for integration (€73.51 compared to around €10). With more experience, the WTP for accuracy increases and the WTP for ease of use is balanced throughout the years of experience. Second, regarding the size of the company: Software testers from smaller companies are willing to pay less for the attributes, except for companies with 100 or less employees, here the willingness increases. The WTP increases with the size of the company with two exceptions: Employees of companies with between 2500 and 9999 employees are willing to pay the most for 99% and 95% accuracy with €386.25 and €345.92, respectively. Small companies with 100 or fewer employees would pay least for integration with only €8.74, and the WTP for integration also drops to €17.21 for companies with 1000-2499 employees.

5 Discussion

Although software testing is an essential part of software development and a large budget is spent on it, many organizations seem hesitant to develop suitable ML-based software testing solutions. The adoption of such a ML-based tool would not only reduce the cost but also decrease the time required and prevent human errors. Therefore, this paper examined the ideal design of ML-based software testing tools by evaluating the willingness to pay of potential users using a choice-based conjoint analysis. A two-step process was adopted to determine the required attributes, levels, and prices: First, a structured literature review and subsequently, a Delphi study with experts in the field of ML-based software testing.

This approach provides several **theoretical contributions**. First, many concepts and algorithms utilizing ML exist in the literature to support software testers and developers in their daily work. However, hardly any economic aspects or characteristics that are essential when selling these tools to users are considered (Barney et al., 2012). Some attributes such as accuracy (Briand, 2008), ease of use (Dejaeger, Verbraken and Baesens, 2013; Arora, Tetarwal and Saha, 2015), or explainability (Bouktif, Sahraoui and Ahmed, 2014; Schieferdecker, 2020) were gathered as part of the literature review. In addition, we determined new attributes, as well as their levels and prices as part of our Delphi study. In particular, accuracy was identified as an important attribute by the experts. It is not only used by most of the papers from the literature review to evaluate the quality of their algorithm or tool (e.g., (Dejaeger, Verbraken and Baesens, 2013; Tahvili et al., 2018; Kesri, Nayak and Ponnalagu, 2021)), but is also identified by the experts as a key attribute. Similarly, ease of use of the tool was rated as essential by the experts, thus this attribute from the literature is confirmed in practice. The additional attributes identified in the literature, such as ML skills or the extension of the tool with own test data, however, were rated as having less influence on the purchasing behavior of a user in the Delphi study. Instead new attributes such as flexibility, integration, and reliability were identified. Peters, Pumplun and Buxmann (2020), for example, showed that the attribute transparency is important for intelligent systems, which was also identified in the Delphi study. Thus, these attributes are not limited to just software testing tools but can also be applied to other ML-based tools, especially accuracy, which is a typical measure for ML. In addition, the attribute ease of use is generalizable for tools, since they should especially be easy to use in order to be adopted. Furthermore, the results of the conjoint analysis demonstrate the users' view on ML-based software testing tools which results in price as the most important attribute for the participants. Besides price, accuracy is considered to be the most essential attribute for potential users. The participants are willing to pay up to €120 per license per month more for an increase in accuracy from 90% to 99%. Integration and ease of use are also perceived by the participants as being relevant as the second and third most important attributes and they both receive a marginal willingness to pay of around €20. This implies that future research should focus on explainability. Although the high willingness to pay indicates the users' preference for exceptionally high accuracy, the increase from 95% to 99% is very costly in the development. This leads to the question of whether other attributes should be emphasized instead of such a high increase in accuracy during the development. From this some **practical contributions** follow. First, as only a limited number of tools exist on the market, but a lot of research is already conducted on how to develop ML-based software testing tools, we encourage the implementation of these tools from theory into practice, especially as many companies have a relative high budget available. Due to the focus of the participants on price, a specific price strategy by gaining market shares via initially low prices and bounding users by lock-in effects can be recommended. The price can then be increased over time and, above all, the willingness to pay of different user types can be skimmed off by offering different product combinations. In addition, it is possible to integrate additional use cases to test case generation into one tool or platform. Second, users can be segmented especially according to number of employees as licenses for ML-based software tools are mainly sold to companies. The analysis of user segments revealed a negative willingness to pay for the attribute integration among the oldest participants. Thus, the integration should be optional, i.e., users should be able to choose whether they want to integrate the ML-based tool into their existing programming environment.

However, our contributions are also subject to **limitations**, which must be considered when identifying and adopting the results or for the design of similar studies. First, the willingness to pay determined in the conjoint analysis depends on the attributes and levels selected and presented. Although these have been developed and tested by experts, an additional attribute or level can alter the analysis. In addition, the marginal willingness to pay is only within the range of the predetermined prices. However, different prices would only change the level of the willingness to pay and should not lead to a different result. Therefore, the conclusions drawn only apply to the selected combination. Second, since there are currently not many ML-based software testing tools on the market, it can also be assumed that most participants in the conjoint analysis have little or no experience so far. Hence, it can only be derived that the attributes in the order accuracy, integration, and ease of use are important to our participants. This might also change over time as users become familiar with ML-based tools. Finally, the focus on price must be interpreted with caution, since the respondents usually get the tools paid by their employers. Thus, if the prices of the tools are eventually within the budget, the exact price is presumably not important to a single user, but to the company.

Therefore, we can derive **further research** from the contributions and limitations. We want to focus on three: First, examining the WTP of other ML-based services, and thus other attributes, could lead to additional insights. Especially comparing the WTP of traditional services and ML-based services could lead to fascinating results. Second, it would be interesting to offer users a real tool to determine their willingness to pay. Third, to the best of our knowledge, we are one of the first to conduct choice-based conjoint analysis in a corporate context. This could lead to develop new methods for future research.

6 Conclusion

In summary, it was shown that machine learning has entered the software testing literature and offers great potential in practice. Not only can time and costs be saved through the automatic development of test cases, but human errors can also be reduced. In particular, high accuracy, an easy-to-use tool, and integration into the programming environment are important to users and lead to an increase in willingness to pay. The most preferred product combination for the users is 99% accuracy, high usability for example through a user-friendly interface, and the ability to integrate the tool into an existing programming environment for a price of €50 per month for a license. Other interesting attributes such as transparency or availability of documentation were identified which might be relevant for other ML-based services. However, as such tools and especially ML-based software testing tools are hardly available in practice, it will be worth developing them with a focus on accuracy, ease of use, and integration.

Acknowledgment

This research and development project is/was funded by the German Federal Ministry of Education and Research (BMBF) within the “Innovations for Tomorrow’s Production, Services, and Work” Program (funding number 02L19C150) and implemented by the Project Management Agency Karlsruhe (PTKA). The authors are responsible for the content of this publication.

References

Ahmed, B.S. and Zamli, K.Z. (2011). ‘A Variable Strength Interaction Test Suites Generation Strategy Using Particle Swarm Optimization’, *Journal of Systems and Software* 84 (12), pp. 2171–2185.

Albani, A., Domigall, Y. and Winter, R. (2017). ‘Implications of Customer Value Perceptions for the Design of Electricity Efficiency Services in Times of Smart Metering’, *Information Systems and e-Business Management* 15 (4), pp. 825–844.

Arora, I., Tetarwal, V. and Saha, A. (2015). ‘Open Issues in Software Defect Prediction’, *Procedia Computer Science* 46, pp. 906–912.

Barney, S., Petersen, K., Svahnberg, M., Aurum, A. and Barney, H. (2012). ‘Software Quality Trade-Offs: A Systematic Map Sebastian’, *Information and Software Technology* 54 (7), pp. 651–662.

Barr, E.T., Harman, M., McMinn, P., Shahbaz, M. and Yoo, S. (2015). ‘The Oracle Problem in Software Testing: A Survey’, *IEEE Transactions on Software Engineering* 41 (5), pp. 507–525.

Berente, N., Gu, B., Recker, J. and Santhanam, R. (2021). ‘Managing Artificial Intelligence’, *MIS Quarterly* 45 (3), pp. 1433–1450.

Böhm, S., Adam, F. and Farrell, W.C. (2015). ‘Impact of the Mobile Operating System on Smartphone Buying Decisions: A Conjoint-Based Empirical Analysis’, in: *International conference on mobile web and information systems*. Springer, Cham pp. 198–210.

Bouktif, S., Sahraoui, H. and Ahmed, F. (2014). ‘Predicting Stability of Open-Source Software Systems Using Combination of Bayesian Classifier’, ACM *Transactions on Management Information Systems (TMIS)* 5 (1), pp. 1–26.

Breidert, C., Hahsler, M. and Reutterer, T. (2006). ‘A Review of Methods for Measuring Willingness-To-Pay’, *Innovative Marketing* 2 (4), pp. 8–32.

Briand, L.C. (2008). ‘Novel Applications of Machine Learning in Software Testing’, in: *The Eighth International Conference on Quality Software*. IEEE pp. 3–10.

vom Brocke, J., Simons, A., Niehaves, B., Reimer, K., Plattfaut, R. and Cleven, A. (2009). ‘Reconstructing the Giant: On the Importance of Rigour in Documenting the Literature’, in: *ECIS 2009 Proceedings*.

Catal, C. and Diri, B. (2009). ‘Investigating the Effect of Dataset Size, Metrics Sets, and Feature Selection Techniques on Software Fault Prediction Problem’, *Information Sciences* 179 (8), pp. 1040–1058.

Conjointly (2022). *Survey Platform With Easy-To-Use Advanced Tools and Expert Support*. URL: https://conjointly.com/ (visited on November 10, 2022).

Cooper, H.M. (1986). ‘Organizing Knowledge Syntheses: A Taxonomy of Literature Reviews’, *Knowledge in society* 1 (1), pp. 104–126.

Dadkhah, M., Araban, S. and Paydar, S. (2020). ‘A Systematic Literature Review on Semantic Web Enabled Software Testing’, *Journal of Systems and Software* 162, p. 110485.

Dejaeger, K., Verbraken, T. and Baesens, B. (2013). ‘Toward Comprehensible Software Fault Prediction Models Using Bayesian Network Classifiers’, *IEEE Transactions on Software Engineering* 39 (2), pp. 237–257.

Durelli, V.H.S., Durelli, R.S., Borges, S.S., Endo, A.T., Eler, M.M., Dias, D.R.C. and Guimarães, M.P. (2019). ‘Machine Learning Applied to Software Testing: A Systematic Mapping Study’, *IEEE Transactions on Reliability* 68 (3), pp. 1189–1212.

Ebbers, F., Zibuschka, J., Zimmermann, C. and Hinz, O. (2021). ‘User Preferences for Privacy Features in Digital Assistants’, *Electronic Markets* 31 (2), pp. 411–426.

Ebert, C., Bajaj, D. and Weyrich, M. (2022). ‘Testing Software Systems’, *IEEE Software* 39 (4), pp. 8–17.

Enoiu, E., Tukseferi, G. and Feldt, R. (2020). ‘Towards a Model of Testers’ Cognitive Processes: Software Testing as a Problem Solving Approach’, in: *2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)* pp. 272–279.

Le Gall-Ely, M. (2009). ‘Definition, Measurement and Determinants of the Consumer’s Willingness to Pay: A Critical Synthesis and Avenues for Further Research Marine’, *Recherche et Applications en Marketing (English Edition)* 24 (2), pp. 91–112.

Gensler, S., Hinz, O., Skiera, B. and Theysohn, S. (2012). ‘Willingness-To-Pay Estimation With Choice-Based Conjoint Analysis: Addressing Extreme Response Behavior With Individually Adapted Designs’, *European Journal of Operational Research* 219 (2), pp. 368–378.

Gezici, B. and Tarhan, A.K. (2022). 'Systematic Literature Review on Software Quality for AI-Based Software', *Empirical Software Engineering* 27 (3), pp. 1–65.

Gholami, F., Attar, N., Haghighi, H., Asl, M.V., Valueian, M. and Mohamadyari, S. (2018). 'A Classifier-Based Test Oracle for Embedded Software', in: *2018 Real-Time and Embedded Systems and Technologies (RTEST)*. IEEE pp. 104–111.

Giessmann, A. and Stanoevska-Slabeva, K. (2013). 'What Are Developers' Preferences on Platform as a Service? An Empirical Investigation', in: *2013 46th Hawaii International Conference on System Sciences*. IEEE pp. 1035–1044.

GitHub (2023). *Your AI pair programmer*. URL: https://github.com/features/copilot (visited on March 21, 2023).

Goebel, P., Reuter, C., Pibernik, R., Sichtmann, C. and Bals, L. (2018). 'Purchasing Managers Willingness to Pay for Attributes That Constitute Sustainability', *Journal of Operations Management* 62, pp. 44–58.

Gondra, I. (2008). 'Applying Machine Learning to Software Fault-Proneness Prediction', *The Journal of Systems and Software* 81 (2), pp. 186–195.

Green, P.E. and Rao, V.R. (1971). 'Conjoint Measurement for Quantifying Judgmental Data', *Journal of Marketing Research* 8 (3), pp. 355–363.

Green, P.E. and Srinivasan, V. (1978). 'Conjoint Analysis in Consumer Research Issues and Outlook', *Journal of Consumer Research* 5 (2), pp. 103–123.

Gupta, M., Bastani, F., Khan, L. and Yen, I.L. (2004). 'Automated Test Data Generation Using MEA-Graph Planning', in: 1*6th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004)* pp. 174–182.

Guveyi, E., Aktas, M.S. and Kalipsiz, O. (2020). 'Human Factor on Software Quality: A Systematic Literature Review', in: *International Conference on Computational Science and Its Applications* pp. 918–930.

Herbold, S. (2021). 'On the Costs and Profit of Software Defect Prediction', *IEEE Transactions on Software Engineering* 47 (11), pp. 2617–2631.

Hourani, H., Hammad, A. and Lafi, M. (2019). 'The Impact of Artificial Intelligence on Software Testing', in: *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*. IEEE pp. 565–570.

Itkin, I., Novikov, A. and Yavorskiy, R. (2019). 'Development of Intelligent Virtual Assistant for Software Testing Team', in: *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. IEEE pp. 126–129.

Kalish, S. and Nelson, P. (1991). 'A Comparison of Ranking, Rating and Reservation Price Measurement in Conjoint Analysis', *Marketing Letters* 2 (4), pp. 327–335.

Kanuri, V.K., Thorson, E. and Mantrala, M.K. (2014). 'Using Reader Preferences to Optimize News Content: A Method and a Case Study', *International Journal on Media Management* 16 (2), pp. 55–75.

Kazmi, R., Jawawi, D.N.A., Mohamad, R. and Ghani, I. (2017). 'Effective Regression Test Case Selection: A Systematic Literature Review', *ACM Computing Surveys* 50 (2), pp. 1–32.

Kesri, V., Nayak, A. and Ponnalagu, K. (2021). 'AutoKG - An Automotive Domain Knowledge Graph for Software Testing: A Position Paper', in: *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)* pp. 234–238.

Khatibsyarbini, M., Isa, M.A., Jawawi, D.N.A., Shafie, M.L.M., Wan-Kadir, W.M.N., Hamed, H.N.A. and Suffian, M.D.M. (2021). 'Trend Application of Machine Learning in Test Case Prioritization: A Review on Techniques', *IEEE Access* pp. 166262–166282.

King, T.M., Arbon, J., Santiago, D., Adamo, D., Chin, W. and Shanmugam, R. (2019). 'AI for Testing Today and Tomorrow: Industry Perspectives', in: *2019 IEEE International Conference on Artificial Intelligence Testing (AITest)*. IEEE pp. 81–88.

Li, Z., Jing, X.Y. and Zhu, X. (2018). 'Progress on Approaches to Software Defect Prediction', *IET Software* 12 (3), pp. 161–175.

Lima, R., da Cruz, A.M.R. and Ribeiro, J. (2020). 'Artificial Intelligence Applied to Software Testing: A Literature Review', in: *2020 5th Iberian Conference on Information Systems and Technologies (CISTI)*. IEEE.

Louviere, J.J. and Woodworth, G. (1983). 'Design and Analysis of Simulated Consumer Choice or Allocation Experiments: An Approach Based on Aggregate Data', *Journal of Marketing Research* 20 (4), pp. 350–367.

Madera, M. and Tomoń, R. (2017). 'A Case Study on Machine Learning Model for Code Review Expert System in Software Engineering', in: *Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE pp. 1357–1363.

Marselis, R. (2020). 'Testing in the Digital Age', in: Goericke, S. (ed.) *The Future of Software Quality Assurance*. Springer, Cham pp. 77–92.

Memon, A.M., Pollack, M.E. and Soffa, M. Lou (2001). 'Hierarchical Gui Test Case Generation Using Automated Planning', *IEEE Transactions on Software Engineering* 27 (2), pp. 144–155.

Mihale-Wilson, A.C., Zibuschka, J. and Hinz, O. (2019). 'User Preferences and Willingness to Pay For In-Vehicle Assistance', *Electronic Markets* 29 (1), pp. 37–53.

Moorthy, S., Ratchford, B.T. and Talukdar, D. (1997). 'Consumer Information Search Revisited: Theory and Empirical Analysis', *Journal of Consumer Research* 23 (4), pp. 263–277.

Nguyen, K.A.T., Nguyen, T.A.T., Nguelifack, B.M. and Jolly, C.M. (2022). 'Machine Learning Approaches for Predicting Willingness to Pay For Shrimp Insurance in Vietnam', Mar*ine Resources Economics* 37 (2), pp. 155–182.

Pan, R., Bagherzadeh, M., Ghaleb, T.A. and Briand, L. (2022). 'Test Case Selection and Prioritization Using Machine Learning: A Systematic Literature Review', *Empirical Software Engineering* 27 (2), pp. 1–43.

Pandey, S.K., Rathee, D. and Tripathi, A.K. (2020). 'Software Defect Prediction Using K-PCA and Various Kernel-Based Extreme Learning Machine: An Empirical Study', *IET Software* 14 (7), pp. 768–782.

Perfecto (2023). *Perfecto Home*. URL: https://www.perfecto.io/ (visited on March 21, 2023).

Peters, F., Pumplun, L. and Buxmann, P. (2020). 'Opening the Black Box: Consumer's Willingness to Pay for Transparency of Intelligent Systems', in: Pr*oceedings of the 28th European Conference on Information Systems*.

Poth, A., Beck, Q. and Riel, A. (2019). 'Artificial Intelligence Helps Making Quality Assurance Processes Leaner', in: *European Conference on Software Process Improvement*. Springer, Cham pp. 722–730.

Rabbi, K., Mamun, Q. and Islam, M.R. (2018). 'An Efficient Particle Swarm Intelligence Based Strategy to Generate Optimum Test Data in T-Way Testing', in: *2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA)*. IEEE pp. 123–128.

Ramsey, S.M. and Bergtold, J.S. (2021). 'Examining Inferences From Neural Network Estimators of Binary Choice Processes: Marginal Effects, and Willingness-To-Pay', *Computational Economics* 58 (4), pp. 1137–1165.

Rathore, S.S. and Kumar, S. (2021). 'Software Fault Prediction Based on the Dynamic Selection of Learning Technique: Findings From the Eclipse Project Study', *Applied Intelligence* 51 (12), pp. 8945–8960.

Roßnagel, H., Zibuschka, J., Hinz, O. and Muntermann, J. (2014). 'Users' Willingness to Pay for Web Identity Management Systems', *European Journal of Information Systems* 23 (1), pp. 36–50.

Russell, S. and Norvig, P. (2021). *Artificial Intelligence, Global Edition A Modern Approach*. Pearson.

Schieferdecker, I. (2020). 'Responsible Software Engineering', in: *The Future of Software Quality Assurance*. Springer, Cham pp. 137–146.

Schuir, J. and Teuteberg, F. (2021). 'Understanding Augmented Reality Adoption Trade-Offs in Production Environments From the Perspective of Future Employees: A Choice-Based Conjoint Study', *Information Systems and e-Business Management* 19 (3), pp. 1039–1085.

Segura, S. and Zhou, Z.Q. (2018). 'Metamorphic Testing 20 Years Later: A Hands-on Introduction', in: *2018 ACM/IEEE 40th International Conference on Software Engineering: Companion Proceeedings* pp. 538–539.

Shafiq, S., Mashkoor, A., Mayr-Dorn, C. and Egyed, A. (2021). 'A Literature Review of Using Machine Learning in Software Development Life Cycle Stages', *IEEE Access* 9, pp. 140896–140920.

Shen, P., Ding, X., Ren, W. and Yang, C. (2018). 'Research on Software Quality Assurance Based on Software Quality Standards and Technology Management', in: *2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. IEEE pp. 385–390.

Skinner, R., Nelson, R.R., Chin, W.W., and Land, L. (2015). 'The Delphi Method Research Strategy in Studies of Information Systems', *Communications of the Association for Information Systems* 37 (2), pp. 31–63.

Sofy (2023). *Pricing*. URL: https://sofy.ai/pricing/ (visited on March 27, 2023).

Sohn, K., Sung, C.E., Koo, G. and Kwon, O. (2021). 'Artificial Intelligence in the Fashion Industry: Consumer Responses to Generative Adversarial Network (GAN) Technology', *International Journal of Retail and Distribution Management* 49 (1), pp. 61–80.

Storey, M.-A., Houck, B. and Zimmermann, T. (2022). 'How Developers and Managers Define and Trade Productivity for Quality', in: *15th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE'22)*, Association for Computing Machinery pp. 26–35.

Tahvili, S., Afzal, W., Saadatmand, M., Bohlin, M. and Ameerjan, S.H. (2018). 'ESPRET: Espret: A Tool for Execution Time Estimation of Manual Test Cases', *The Journal of Systems & Software* 146, pp. 26–41.

Walgude, A. and Natarajan, S. (2019). *World Quality Report*, Capgemini. URL: https://www.capgemini.com/es-es/wp-content/uploads/sites/16/2019/10/World-Quality-Report-2019-20.pdf (visited on October 30, 2022).

von Wedel, P. and Hagist, C. (2022). 'Physicians' Preferences and Willingness to Pay For Artificial Intelligence-Based Assistance Tools: A Discrete Choice Experiment Among German Radiologists', *BMC Health Services Research* 22 (1), pp. 1–14.

Xing, Y., Gong, Y.Z., Wang, Y.W. and Zhang, X.Z. (2014). 'Path-Wise Test Data Generation Based on Heuristic Look-Ahead Methods', *Mathematical Problems in Engineering* pp. 1–19.

Zhang, H., Bai, X., and Ma, Z. (2022). 'Consumer Reactions to AI Design: Exploring Consumer Willingness to Pay For AI-Designed Products', *Psychology and Marketing* 39 (11), pp. 2171–2183.

Zhao, Y., Hu, Y. and Gong, J. (2021). 'Research on International Standardization of Software Quality and Software Testing', in: *2021 IEEE/ACIS 20th International Fall Conference on Computer and Information Science (ICIS Fall)*. IEEE pp. 56–62.