Evaluating a Large Language Model on Searching for GUI Layouts

PAUL BRIE, teleportHQ, Romania
NICOLAS BURNY, Université catholique de Louvain, Belgium
ARTHUR SLUŸTERS, Université catholique de Louvain, Belgium
JEAN VANDERDONCKT, Université catholique de Louvain, Belgium

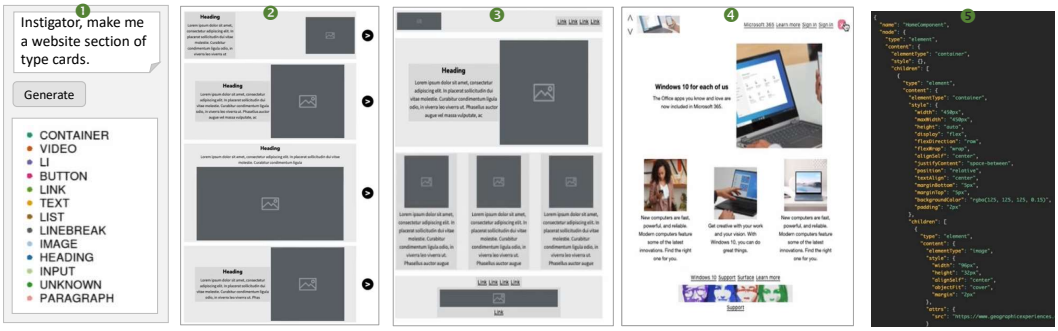

Fig. 1. Overview of our GUI design process which: ❶ prompts the user to search for a GUI layout in partial natural language, entered by speech or by typing; ❷ retrieves corresponding GUI layouts using a generative pre-training transformer and displays them in low-fidelity, in decreasing order of the computed ranking; ❸ enables the user to compose and edit selected layout elements in a wireframing environment to create a final GUI layout; ❹ renders this composition in high-fidelity; and ❺ generates code.

The field of generative artificial intelligence has seen significant advancements in recent years with the advent of large language models, which have shown impressive results in software engineering tasks but not yet in engineering user interfaces. Thus, we raise a specific research question: would an LLM-based system be able to search for relevant GUI layouts? To address this question, we conducted a controlled study evaluating how INSTIGATOR, an LLM-based system for searching GUI layouts of web pages by generative pre-trained training, would return GUI layouts that are relevant to a given instruction and what would be the user experience of (N=34) practitioners interacting with INSTIGATOR. Our results identify a very high similarity and a moderate correlation between the rankings of the GUI layouts generated by INSTIGATOR and the rankings of the practitioners with respect to their relevance to a given design instruction. We highlight the results obtained through thirteen UEQ+ scales that characterize the user experience of the practitioner with INSTIGATOR, which we use to discuss perspectives for improving such future tools.

Authors' addresses: Paul Brie, teleportHQ, Calea Moţilor, 84, Cluj-Napoca, 400370, Romania, paul.brie@teleporthq.io; Nicolas Burny, Université catholique de Louvain, Louvain Research Institute in Management and Organizations, Place des Doyens, 1, Louvain-la-Neuve, 1348, Belgium, nicolas.burny@uclouvain.be; Arthur Sluÿters, arthur.sluyters@uclouvain.be, Université catholique de Louvain, Louvain Research Institute in Management and Organizations, Place des Doyens, 1, Louvain-la-Neuve, 1348, Belgium; Jean Vanderdonckt, Université catholique de Louvain, Louvain Research Institute in Management and Organizations, Place des Doyens, 1, Louvain-la-Neuve, 1348, Belgium, jean.vanderdonckt@uclouvain.be.

2573-0142/2023/6-ART178 $15.00
https://doi.org/10.1145/3593230

CCS Concepts: • **Human-centered computing** → **Interactive systems and tools**; *Wireframes*; **Graphical user interfaces**; Web-based interaction; • **Theory of computation** → **Models of learning**; • **Computing methodologies** → **Natural language generation**; Unsupervised learning; • **Software and its engineering** → *Design languages.*

Additional Key Words and Phrases: Generative pre-training; GUI Design; GUI Layout; Large Language Model; Web pages.

ACM Reference Format:
Paul Brie, Nicolas Burny, Arthur Sluÿters, and Jean Vanderdonckt. 2023. Evaluating a Large Language Model on Searching for GUI Layouts. *Proc. ACM Hum.-Comput. Interact.* 7, EICS, Article 178 (June 2023), 37 pages. https://doi.org/10.1145/3593230

1 INTRODUCTION

Generative Artificial Intelligence (GenAI) [102] can be defined as an Artificial Intelligence (AI) system that uses existing media to create new, plausible media. Applied to the area of engineering layouts of Graphical User Interfaces (GUIs) [60], this could be a system that takes advantage of existing layouts to propose new ones that are plausible enough to be incorporated into the GUI design process. For this purpose, we are witnessing a paradigm shift moving from past expert and model-based systems [19, 24, 25, 76] exploiting knowledge to generate GUI layouts towards generative models, such as Generative Adversarial Networks (GANs), which are trained on huge datasets to produce plausible human-like results and Large Language Models (LLMs) that can generate text, source code, and images from *instructions*, or *prompts*.

Large language models (LLMs) [9] utilize huge datasets to learn the structure and relationships between words in any language, be it a programming language or a natural language, using various algorithms, such as Recurrent Neural Networks (RNNs) or Generative Pre-trained Transformers (GPT). During the training, LLMs are fed huge text datasets that have been previously labeled with parts-of-speech information and other features like grammar relations. During the exploitation, LMMs predict future data points related to unseen data points from existing ones.

LMMs, such as OpenAI's Codex [16] and GPT-3 [9] and some recent implementations such as ChatGPT, have shown great promise in the field of program synthesis and code generation. These models learn from huge code bases to automatically generate code in classical programming languages, such as Python or Javascript. With recent advances in Deep Learning (DL) and Machine Learning (ML), LLMs are being largely used in software engineering [80], including code generation from natural language [36], performance evaluation [68, 70], natural language explanation for code [61], Codex's Python coding capabilities [16], programming exercises and code explanations quality [83], code summarization [1], and code generation [97]. These recent studies demonstrate that LLMs can surpass state-of-the-art models for these tasks. Languages seem to be suitable for dealing with GUI examples, even if they have been little or not exploited in this sense up to now. While LLMs showed their potential to greatly aid developers in computer science, the lack of research and practical applications in specific areas remains a challenge, with GUI design being a particularly notable example of this issue [37]. GUI design is an essential aspect of software development and plays a crucial role in providing end users with a smooth user experience.

GUI practitioners, such as designers, developers, and stakeholders involved in participatory design, increasingly seek support [52, 54] or inspiration [44] in their design process through proven or recommended GUI layouts and examples. They can be sourced from various inspiration categories [27, 90], such as personal files and folders [44], daily repositories like blogs and social networks [93], technological resources such as galleries [52] and datasets [21, 22], other disciplines' knowledge, fieldwork [105], and design practices like case studies [7].

GUI layouts are sought for various reasons, such as starting the design process or completing a partial layout [33], introducing new ideas [29], adapting existing layouts to a new context of use [52] (*e.g.*, to align with a style guide [20]), comparing current layouts to alternatives [33, 52] or competition [56, 105], or simply improving the overall layout. Lee *et al.* [52] highlight the dominant role of design by example, stating that practitioners prefer layouts created with the help of examples, which may benefit novices more than experienced designers. Inspiration sources are a common subject in design research, yet few studies have thoroughly explored concrete inspiration search strategies. These strategies usually search GUI layouts by code, keyword, image, sketch, wireframe, or component, like text-based searches [7]. The level of constraints influences the search for GUI layouts [7]: low constraints lead to divergent and iterative searches resulting in diverse designs, intermediate constraints cause slow and deliberate iterations and homogeneous designs, and high constraints result in flexible bracketing and quick design task use, resulting in diverse designs.

Providing practitioners with an effective and efficient engine for searching GUI layouts should achieve multiple goals [7, 13]: to open the GUI design space, to facilitate its understanding, to focus on relevant GUI layouts in a more targeted way, and to ensure a smooth transition from a low level of fidelity to a high level of fidelity. To this end, LLMs are particularly suitable in that they could offer such a search engine for GUI layout. Based on the aforementioned motivations, this paper would like to raise the following research question:

RQ=*Would an LLM-based system be able to search for relevant GUI layouts?*

To address this question, we conducted a controlled study evaluating Instigator, an LLM-based system for searching GUI layouts, for two questions: how would Instigator enable a practitioner to search for a GUI layout that adequately corresponds to instruction and what is the user experience of a practitioner interacting with Instigator. The remainder of this paper is structured as follows: Section 2 reviews work related to the search for GUI layout by various methods and some recent advances in LLMs. Section 3 illustrates, via a system walkthrough, a typical use case for Instigator, our system for GUI layouts using a generative pre-trained transformer, a particular LLM. Section 4 gives an overview of the Instigator architecture and functioning. Section 5 describes the controlled study, reports the results obtained, and discusses their overall observations to end up with a detailed discussion of individual evaluation scales. Section 6 positions Instigator with respect to the state of the art, discusses some of its limitations and the main threats to validity. Section 7 concludes the paper and proposes some future avenues.

2 BACKGROUND AND RELATED WORK

This paper contributes to a recent stream of research on using Machine Learning to enable practitioners to search for GUI layouts (Section 2.1) and relies on existing streams of research on LLMs for this purpose (Section 2.2).

2.1 Methods for Searching for GUI Layouts

Such an engine represents a valuable tool for practitioners to support their quest for inspiration [7] and comparison [33, 52] using different methods. Some tools are based on a single method while others combine multiple methods. This section discusses some tools with respect to the method that is predominantly used.

By code search. Semantics-based code search [77] allows the user to specify a search in project repositories using class or method signatures, test cases, contracts, and security constraints, which are then subject to program transformations to retrieve the corresponding GUI code. For example, Java input/output examples can be used to retrieve GUI code [38]. Code-based search methods are

most useful and practicable for developers experienced in the programming language used, but not necessarily for other stakeholders.

By keyword. By using automated dynamic analysis on a large dataset of applications downloaded from Google Play, Guigle [6] extracts and indexes GUI layouts with their metadata (*e.g.*, application name, labels, element name, colors used, domain). Based on these keywords, the user searches for GUI layouts by component (*e.g.*, `Login AND color=orange` returns orange login screenshots), which are displayed accordingly. The Adaptive Ideas Web [52] extends a direct manipulation GUI editor with generated GUI layouts, from which elements can be borrowed and exported to the current design based on keywords. These layouts are generated by optimizing content-value.

By image or screenshot. Erica [22] records a user interacting with a mobile application in terms of GUI screenshots, their transitions, and user interaction data in interaction traces, aggregated into user flows. In this way, a dataset of ~3k flows was clustered into 23 categories, allowing the user to search for corresponding GUI layouts. By automatic and crowd-based manual exploration, Deka *et al.* [21] have built up Rico, a huge dataset of ~10k GUI layouts spanning 27 categories and featuring ~72k screenshots with their metadata, such as textual, visual, structural, and interactional data. These data served to classify designs by using multilayer perceptron using six layers and to identify both structural and functional roles played in these layouts [59]. Each screenshot is characterized in terms of textual *vs.* non-textual bounding boxes to support element detection. Leiva *et al.* [55] used a deep convolutional autoencoder to incorporate 20 topics (*e.g.*, bare, dialer, camera, news, profile) of Rico entries into a Enrico, a new dataset with topic modeling.

By sketch. Ge [26] transformed each Rico [21] entry into a GUI hierarchy of elements to train a deep learning model. The user draws a low-fidelity, free-hand GUI sketch that is also transformed into a GUI hierarchy by a grammar-based sketch parser. This hierarchy is then used to search for similar GUI layouts in the dataset ranked by a similarity score. From a dataset of >3,8k sketches, Swire [33] retrieves GUI layouts that are similar to a sketch drawn by the user, which is transformed into an embedding space using a deep neural network and located by a *k*-nearest neighbor search.

By wireframe. Given a small set of wireframes with transitions between them and their keywords, GUIFetch [5] searches public repositories to find applications by keyword matching, then builds models of the identified apps' GUI screens, using static and dynamic analyses, and computes a similarity metric to rank the GUI layouts. VINS [10] searches for similar mobile GUI layouts as follows: a GUI layout, either as a high-fidelity complete design or as a low-fidelity partial wireframe, is submitted to an object detection model to identify which elements are incorporated in this layout and where, and to segmentation of these elements. This layout is then used to retrieve a ranked list of similar layouts, which is achieved with an accuracy of >76%. While there are significant advances in GUI element detection in a layout, their wide diversity makes it a challenging, if not elusive, problem: Chen *et al.* [12] compared classical methods (*e.g.*, edge/contour detection and pattern matching) with deep learning methods (*e.g.*, anchor-based with one or two stages) to identify which features should be incorporated into a DL method coupled with a classical one. Instead of multi-layered approaches, Chen *et al.* [13] developed a Wireframe Image autoencoder, a CNN to classify 16 GUI elements in a dataset of ~54k layouts with their wireframes from 25 categories, to create a wireframe-based system. This method uses a *autoencoder* [28] (chap. 14), which learns a representation (*encoding*) of GUI layouts in terms of their elements in an unsupervised way to regenerate a GUI layout from the encoding. This generative model is not only useful for searching for GUI layout by wireframe but also to generate new layouts similar to the searched one [42].

By component. LayoutGAN [57], a Generative Adversarial Network (GAN), receives as input a set of randomly placed GUI elements to infer geometric constraints between them to generate the layout of a GUI layout using self-attention modules. A differentiable wireframe rendering engine then maps this layout to a wireframe, upon which a CNN-based discriminator optimizes the layout.

By exploration. Design Maps [56] enable the user to visualize the mobile GUIs of the Rico dataset [21] according to point clouds structured into four sets of five clusters obtained by ML techniques (*e.g.*, Principal Component Analysis) run on the GUI feature vectors. When starting from scratch, any selected GUI can be imported into wireframing software and ungrouped into icons representing GUI elements. When one starts from an initial wireframe, it automatically positions the user in any chosen point cloud, thus enabling visual exploration by proximity or by similarity.

2.2 Recent Advances in Large Language Models

The field of Natural Language Processing (NLP) has greatly profited from advancements in Machine Learning (ML), specifically in regard to improvements in model architecture. The implementation of encoder-decoder models, facilitated by Recurrent Neural Networks (RNNs), has been a crucial component in the development of NLP. The introduction of an attention mechanism by Bahdanau *et al.* [4] has significantly impacted NLP, as it allows for the processing of an entire input at once.

A **Transformer** is a deep learning model that embodies the self-attention mechanism, allowing for differential weighting of the significance of each component of the input data [104]. Although RNNs have proven to be effective in handling sequences, such as sentences in natural language or computer language instructions, their sequential nature results in a bottleneck in training time, as parallelization is not possible. Vaswani *et al.* [100] proposed a solution to this issue by proposing a Transformer that utilizes a stack of self-attention mechanisms and fully connected layers, allowing for a wider context to be processed in the training input. The implementation of a multi-head attention mechanism also enables the learning of diverse representations and connections between tokens, regardless of their distance. The adoption of Transformers has resulted in a reduction of computational complexity and time, due to the easily parallelizable operations, and has been proven to deliver state-of-the-art results [100, 104].

The abundance of unlabeled text-based content has sparked the development of OpenAI's Generative Pre-trained Transformer (**GPT**) [15]. Unlike the conventional approach of training models on specific, labeled datasets, GPT was trained through an initial unsupervised phase on a substantial corpus of unlabeled data, consisting of approximately 7k books. This was followed by a supervised fine-tuning process utilizing labeled datasets, utilizing a modified version of the transformer decoder architecture. The efficacy of the GPT model was demonstrated through its outstanding performance in various natural language processing tasks [104].

While GPT-like models process their data in a unidirectional manner, Google hypothesized that the deep context of language could be better understood through a bidirectional parse. This led to Bidirectional Encoder Representations from Transformers (**BERT**) [23], which incorporates the encoder part of the transformer and employs a similar unsupervised pre-training and supervised fine-tuning technique. The pre-training phase was accomplished through the use of Masked Language Modeling, where the model predicts the word missing from a sentence based on the surrounding context [78]. These techniques provide the model with information about both words and sentences in context. BERT has achieved state-of-the-art results after fine-tuning 11 domain-specific tasks, such as natural language understanding challenges from the GLUE dataset [101] and questions answering or sentence-pair completion. Rawi [46] used a BERT-based Learning-To-Rank approach that is trained on GUI layout data obtained by crowdsourcing techniques. They showed that this approach outperformed a classical approach fro searching GUI layout based on TF-IDF.

iGPT [14] proves the further application of transformers in visual processing. It performs image completion without any information about the 2D structure of the pictures and uses the same model architecture as in GPT-2. This approach, with additional fine-tuning, makes the model competitive with self-supervised benchmarks on the ImageNet dataset.

OpenAI has introduced a highly innovative model, known as **GPT-3** [9], that bypasses the fine-tuning aspect of the training process by utilizing a "few-shot" learning approach. Unlike fine-tuning, Few-Shot Learning (FSL) does not alter the weights of the model and instead conditions it by providing a small number of examples. The evaluation of the model is conducted through zero-shot [106] and one-shot learning [67], in which the prompt is reduced to either one or zero examples. The results obtained from GPT-3 have demonstrated a level of performance that is comparable to or exceeds the state-of-the-art fine-tuned models in several language modeling tasks, including open-close exercises, translation, common sense reasoning, and arithmetic. Additionally, GPT-3 is capable of generating synthetic news articles that are challenging for humans to accurately identify, with a success rate of 52%. When the development of Instigator was completed, only GPT-3 was available: GPT-4 and ChatGPT appeared later, which calls for a comparison between their respective results.

The recent development of iGPT [14] has further solidified the potential applications of Transformers in visual processing. This model performs image completion without any prior knowledge of the 2D structure of the pictures and employs the same model architecture as in GPT-2. With the addition of fine-tuning, this approach proves to be competitive with self-supervised benchmarks on the ImageNet dataset. The domain of reasoning, while not as popular as image processing or natural language processing, has also been tackled through the transformer architecture using **GPT-f**, a proof assistant [75]. The model performs the proof steps given a final objective and achieved state-of-the-art results on the Metamath environment, closing 56% of the proofs.

In conclusion, GTP-like approaches, especially based on pre-trained transformers [13, 15], have the potential to handle large sets of GUI layouts captured by their HTML code, interpreted as the language to process to generate layouts based on them. This processing has never been implemented and evaluated in that way. Other usages have been explored for GUI layouts, but not for generating layouts. Scones [34] uses GPT-2 to transform a user's textual scenario into a series of graphical sketches that depict the instructions contained in the scenario. Stylette [41] applies an LLM to support and improve a GUI layout: by interpreting the user's goal of re-shaping a web page in natural language, this browser plug-in extracts GUI styling suggestions from a dataset of 1.7m GUI elements and generates CSS properties with their values to be applied by the user.

3 INSTIGATOR WALKTHROUGH

Before proving an overview of how Instigator works, we present how it supports practitioners through a running example. Anna is a communication manager who is now responsible for designing a few web pages, a skill for which she does not have any particular background. She is working with web page design software and she is looking for some inspiration regarding GUI layouts that are generally accepted and widely used on the web (Figs. 2 and 3).

❶ **Searching for an overall GUI layout**. Anna types an initial instruction "Build a page with contents" since she does not know how to start designing, so she is looking for some starting point in order to work iterative and by refining layouts progressively.

❷ **Generating an initial layout**. Since the contents can be very general, Instigator generates a first series of general GUI layouts corresponding to the initial instruction entered by Anna, ranging from simple ones to more advanced ones. A subset of them is reproduced in Fig. 2-❶ sorted in decreasing order to relevance. For each GUI layout suggested, the corresponding hierarchy of GUI elements is displayed on-demand and is here reproduced to give an idea of the GUI model (not all features are represented). For example, the third entry consists of a layout starting with a paragraph and followed by a series of containers containing images.

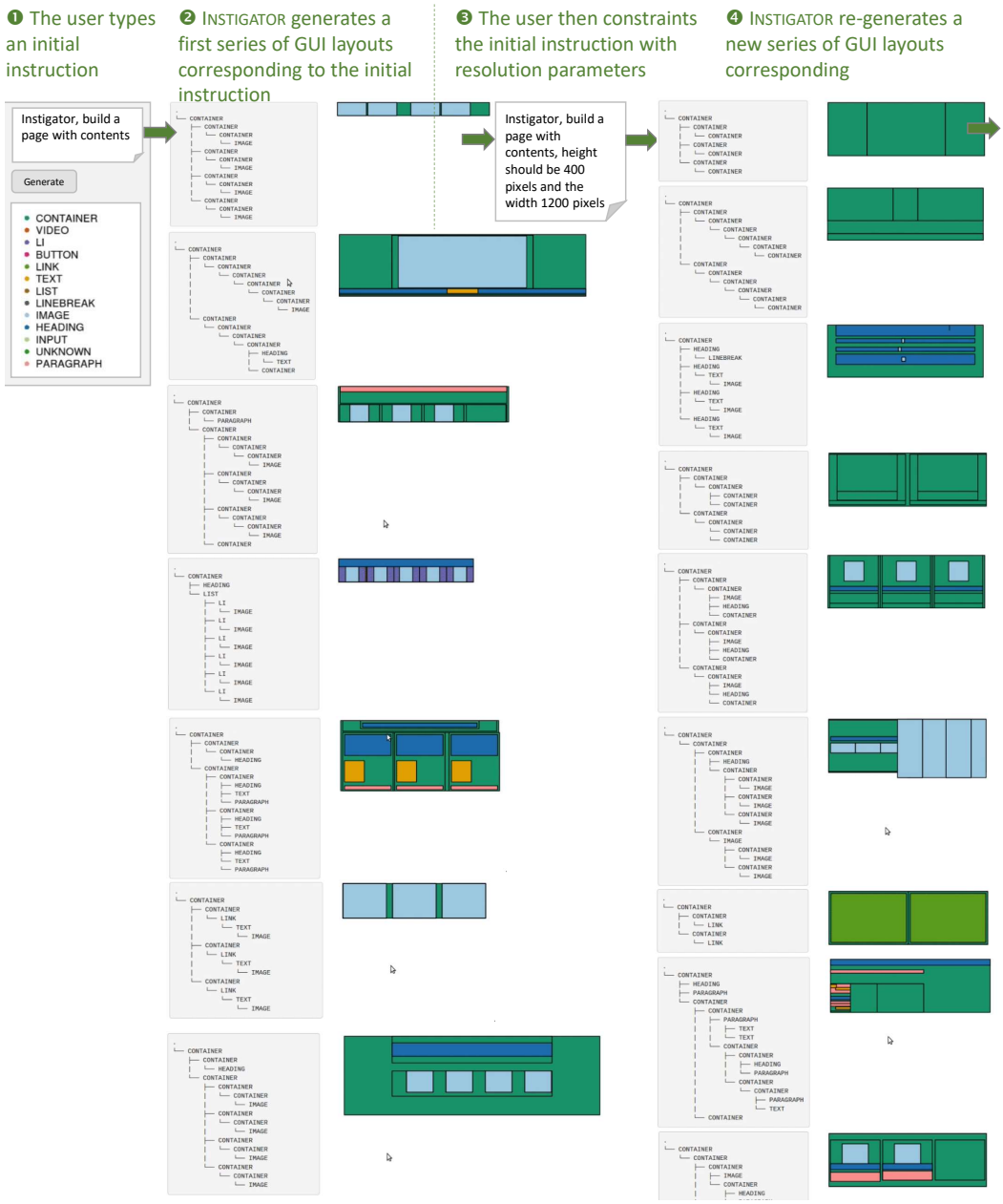

Fig. 2. Interactive session with INSTIGATOR: part 1/2.

❸ **Specifying design constraints.** Anna wants to consider a specific device having a resolution of 400 × 1200 pixels (such as the Hiro Vision Mirror). Without losing the current status, Anna edits the previously entered prompt to add "height should be 400 pixels and the width 1200 pixels".

❹ **Applying design constraints.** INSTIGATOR re-generates a new series of GUI layouts corresponding to the new query, which results in a new series of GUI layouts addressing these constraints

Fig. 3. Interactive session with INSTIGATOR: part 2/2.

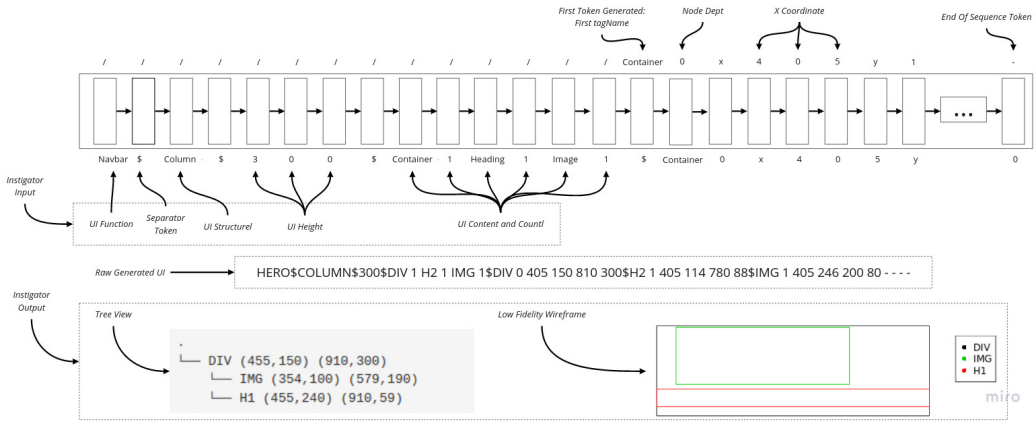

Fig. 4. Instigator Flow.

(right part of Fig. 2). For example, the fifth layout is composed of three identical containers, containing each an image, a header, and another container. The dimensions of the GUI layout respect the constraints.

❺ **Refining the contents.** Observing that the contents should be refined, Anna edits the prompt again by adding "It should contain 3 images, 3 headings, and 6 paragraphs" to the end of the string.

❻ **Populating the layout.** The new instruction results in an updated series of GUI layouts (left of Fig. 3), all sharing the same repeated container with an image, a header, and paragraphs, but with various layouts and dimensions. The fourth one introduces a variation by adding more content to the right of previously entered elements according to the right/bottom strategy [99].

❼ **Altering any parameter.** Observing that 6 paragraphs do not really make sense, Anna replaces the value of this parameter: the "6" indicating the number of paragraphs is replaced by "3" without altering the rest of the query, and a button is added (top right of Fig. 3).

❽ **Updating the layout.** Instigator updates the layout according to the last instruction.

❾ **Refining the contents.** Observing that the button interferes, Anna decides to remove the button and to specify "The orientation should be row and flex", which results in a refined series of GUI layouts (bottom right of Fig. 3).

4 OVERVIEW OF INSTIGATOR

This section provides an overview of Instigator, our LLM-based tool for searching GUI layouts in a dataset. The principle behind Instigator is to use the generative pre-trained training (GPT) architecture [9] in a zero-shot learning [106] (Figure 4). At inference time, meta-information is supplied about the desired GUI layout and its elements to generate, and Instigator will generate it token after token. Meta-information consists of a GUI function (*e.g.*, a splash screen, a "hero" page, a "call-to-action" page, a navigation bar, a login screen), a layout structure (*i.e.*,, a hierarchical decomposition of the layout into elements along with their row and column alignments inspired by FormsVBT [3]), and GUI contents (*e.g.*, elements and count of each element). Instigator will then generate the layout token after token until an End Of Sequence (EOS) token is generated. We choose a YAML format where each node appears one after another separated by an End Of Line (EOL) token and containing the node content (*e.g.*, a container, an image, a paragraph, the node depth, and the node coordinates and size). This sequence of tokens is then converted into a data frame and into a low-fidelity layout (Fig. 5).

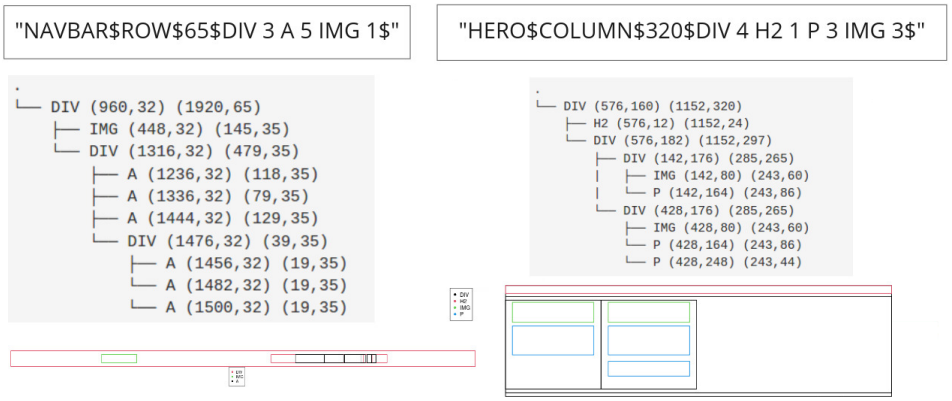

Fig. 5. Some GUI layouts returned by Instigator.

Train-test Split. For example, The Webzeitgeist dataset [48] was obtained by a crawling seeded with 20 URLs, including the Alexa Top 500 US sites, the Webby Awards gallery, and popular design blogs. To build a diverse repository, pages were crawled in breadth-first order, self-referential domain links were skipped, and only a single copy of each resolved URL was stored. For all mentioned machine learning models trained, the same strategy was used to split the dataset between training and testing sets. Because of the multiple occurrences of data points for the same domain name, a random split was not adequate, instead, the split was made based on the first letter of the domain name, *i.e.*,, domain names starting with the letter 'c', 'm' 'o', 'v', 'z' were put int he test set while the other where used for training.

Web Parser. To build a large dataset and conduct our study, we need huge amounts of data to work with, which was made possible by using a Web Parser built with Puppeteer and AWS Cloud Services. By using Puppeteer, we can access the Document Object Model (DOM) of the website, subtract the required data, and export it as a JSON file. By launching a Chromium instance, we fully control the DOM to extract information about all GUI element nodes forming the website. The process is enclosed into an AWS Lambda that is launched every time a new website is added and thus uses the concurrency feature to its full potential. This allowed us to parse up to 4,5k websites/minute, enabling a fast construction of the dataset and an easy way to add new features if required. For each website, we gather high-level information about the nodes, including, but not limited to the tag name, *z* Index, depth, the coordinates of each GUI element relative to the HTML tag or its children, to be used in low-fidelity GUI generation. The segmentation of each GUI layout in elements and their identification has been performed based on an Application Programming Interface (API) specifically trained for GUI layouts, a computer-vision software architecture based on Resnet101 for extracting features, and Faster R-CNN for bounding-box proposals [8]. Because we also consider high-fidelity GUI layouts (❺ in Fig. 1), we also extract the computed styles of the elements, inspired by Stylette [41]. This information is then exported into a JSON file stored in an Amazon S3 bucket, a public web-based cloud storage resource available in AWS Simple Storage Service (S3). All the internal links inside it are parsed, thus, exponentially increasing the number of websites. The need for more specific data brought about the extraction of the sections inside a website. A *section* is a container that satisfies certain conditions and helps us be more accurate when generating designs. Each section is exported in a JSON which contains, along with the information about the nodes, low-level information about how the content is structured inside it (*e.g.*, how the content is oriented or how it is aligned). Moreover, for each segmented section or element, we computed aesthetic metrics [71] like balance, alignment [99], or concentricity [108], which will influence how the sections are generated. We then save a clipped screenshot of each section.

Metadata Extraction. The metadata extracted from each UI could most of the time be directly and objectively extracted from the data itself, *i.e.*,, section height, and section structure via the flex information of the root, and aesthetic metrics. However, the section function is a notion too subjective to be extracted this way, and we relied on machine learning to extract it. We created a custom annotation tool to annotate and classify ~5k sections into 7 categories: hero, navigation bar, footer, testimony, cards, forms, call-to-action (CTA), and contents (text). As 5,000 annotated sections were not enough to train the GPT model, we needed a system to obtain this information on the rest of the dataset. To do so, we trained two ML models: one using Convolutional Neural Network from the images, and one using XGBoost (an open-source software library which provides a regularizing gradient boosting framework [17]) after feature extraction from the JSON, and averaged the prediction results on each of the remaining data points to obtain this final meta-information. Because many websites are not fit to be used in our project, we used the same approach to predict the probability of a website being good. We could then only keep data points above a given threshold and work with a much cleaner dataset. Elements are: container, video, li, button, link, text, list, linebreak, image, heading, input, paragraph, and unknown.

Model Training and Hyperparameters. The GPT model was created from scratch and trained using the minGPT library using PyTorch [73] with the following options:

- Block size=256: this hyperparameter, expressing the length of the bit string, is set to this value that is common for both the input and the output which should have the same length; the output cannot be shorter than the input [81]. It is also the minimal value for the convolutional block that uses filters of this size.
- Number of layers=8: this hyperparameter is set to include all traditional layers such as Transformer, TransformerEncoder, TransformerDecoder, TransformerEncoderLayer, TransformerDecoderLayer, MultiHeadAttention, and MultiHeadAttentionForward.
- Embedding size=512: this hyperparameter is set to the minimum value typically found in more recent NLP papers which use 512, 768, and 1024 [74], contrarily to older papers having an embedding size of 300 conventionally.
- Number of heads=16: this hyperparameter is set to the value recommended by Michel et al. [66] and by Jaime Sevilla and Villalobos [35] for Transformers.
- Batch size=32: this hyperparameter is set to the lower bound of the interval [32,...,512] recommended for optimizing the generalization. When using a larger batch size, a significant degradation could be observed in the model quality, as measured by its ability to generalize [40].

The model was trained for 50 epochs on a 2080Ti GPU. To transform the raw data into a token, we convert the nested JSON structure into a flattened table, with a depth column to keep the nesting information. Tag names were grouped into the following classes: 'CONTAINER', 'VIDEO', 'LIST ITEM', 'BUTTON', 'LINK', 'TEXT', 'LIST', 'LINEBREAK', 'IMAGE', 'HEADING', 'INPUT'. Tag names that did not belong to those classes were labeled as 'UNKNOWN'. The rest of the information in the low-fidelity table is numbers, they were tokenized by digits.

5 CONTROLLED STUDY

This section firstly introduces the apparatus to evaluate: Instigator, an LLM-based software that generates GUI layouts from a textual instruction and at composing its elements into a cohesive layout (Fig. 1). Given textual instruction for a layout, the pre-trained language model of Instigator generates a series of layouts that match the distribution of HTML elements on which it was trained. To avoid jargon and to facilitate the understanding of this process by most stakeholders, who are not necessarily experts, we used the term “searching” instead of “generating”.

However, we do not know whether the GUI layouts returned by Instigator correspond to layouts expected by practitioners (for example, 68.8% of the GUI layouts returned by Guigle were relevant [6]; 81% of the GUI layouts returned by GuiFetch were relevant and suggestions can be found for 65% of the initial GUIs [5]) and how they appreciate the composition of elements found in these suggestions. The remainder of this section will successively define the various aspects of our controlled study structured according to Ko *et al.* [43]: the participants, the tasks, and the quantitative and qualitative measures.

5.1 Participants

Participants were recruited from a national and an international mailing list of ~200 practitioners in total, both mailing lists being maintained at Anonymous. An email was sent to the mailing lists asking practitioners willing to assess Instigator. No compensation was offered to volunteers. To select participants with representative coverage and experience, each volunteer was asked to:

- Fill in a GDPR-compliant demographic questionnaire, which includes statements about the frequency of use of a computer, a smartphone, and a tablet (expressed on a 7-point rating scale, 1=minimum, 7=maximum), as well as the number of years of experience in GUI layout.
- Position herself/himself on Costabile *et al.*'s [18] continuum of development expertise, which consists of nine levels ranging from 'Internet surfing' (L1, where no genuine development actually occurs) to 'professional application development' (L9). Each level defines both the activities a practitioner is able to perform and the corresponding skills required to reach this level. The levels are defined as follows:

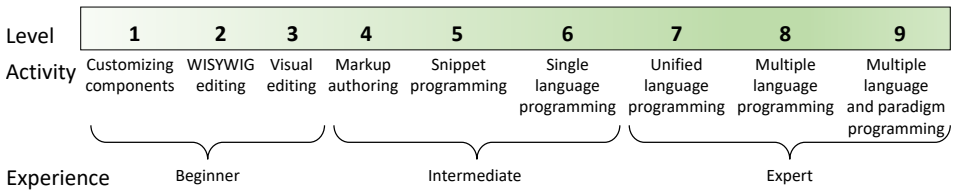

Fig. 6. The nine development expertise levels from [18].

(1) *Customizing components*: practitioners adjust existing GUI layouts by setting values to parameters, such as customizing service preferences, adaptation, and display.
(2) *WYSIWYG editing*: practitioners edit the GUI layout in a WYSIWYG editor.
(3) *Visual editing*: practitioners create GUIs visually by adding, customizing, and connecting components, such as in mashup editors and component-based editors. Visual editing is possible without having learned programming, but often requires a basic understanding of the programming paradigms (*e.g.*, imperative, declarative, functional).
(4) *Markup authoring*: practitioners design GUI layouts using plain markup languages. This is code editing with a simplified markup language, which involves some minimal understanding of declarative programming.
(5) *Snippet programming*: practitioners copy, paste, and manually edit existing source code for templates called *snippets*. This is performed within an editor that automatically generates source code to be tweaked afterward.
(6) *Single language programming*: practitioners have sufficient knowledge of a single programming language and they are able to develop a piece of functionality from scratch.
(7) *Unified language programming*: practitioners have sufficient knowledge of a single programming language, which can be used to create all necessary components of an entire interactive application.

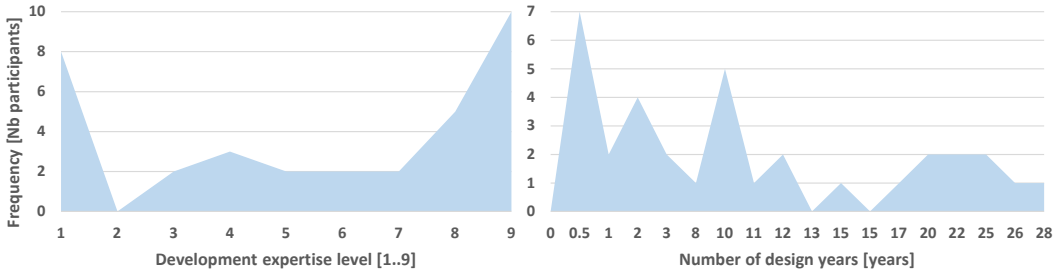

Fig. 7. Design experience of participants: expertise level [18] (left) and the number of design years (right).

(8) *Multiple language programming*: practitioners have a thorough command of several programming languages and combine them. The difference with respect to the previous level is that developers need to master and integrate several languages together to produce an interactive application.
(9) *Multiple language and paradigm programming*: practitioners are professional programmers as they combine several different technologies to entirely build an interactive application. Programming paradigms can be declarative, imperative, or multi-paradigm (*e.g.*, declarative for the GUI and imperative for the controller).

While this scale was primarily invented to express the level of development expertise of a developer, we used it as it mentions software that are typically used by practitioners at each level, thus making the test concrete even for somebody having little or no development expertise. After discarding outliers (*e.g.*, not satisfying the conditions, not fulfilling a complete evaluation), we recruited thirty-four (N=34) participants (11 female, 23 male, and 0 identified as gender variant/non-conforming) aged between 22 and 56 years old (M=36.51, SD=9.91, MD=35 years). Participants were coming from 11 countries located on four continents: North America, Latin America, Europe, and Africa. Participant occupations included various domains of human activity, such as management, social sciences, design science, education, law, and office administration. Their device usages were distributed as follows on a 7-point Likert scale (1=strongly disagree to 7=strongly agree): computer (M=6.89, SD=0.4), smartphone (M=6.49, SD=0.98) and tablet (M=3.74, SD=2.09). They reported level of expertise ranging from 1 to 9 (M=5.64, SD=3.16, Fig. 7, left) and between 0.5 (min) and 28 years (max) of GUI design experience (M=9.98, SD=9.06, Fig. 7, right). No one has ever seen or used Instigator in any circumstance.

5.2 Tasks

We devised four tasks to be carried out by each participant:

(1) A *discovery task*, in which participants attended a 2:33 min. demonstration video (see the supplementary material) and then interacted freely with the Instigator for five minutes, starting from a clean Prompt stage (❶ in Fig. 1). They did not receive any specific instruction or information.
(2) A *warm-up task*, in which participants had to design the homepage of an electronic commerce website in a free manner with Instigator. Although no time limit was imposed, a 15-minute slot was recommended. Participants were instructed to design a GUI to their best knowledge from the initial stage to the final (❺ in Fig. 1).
(3) A *GUI layout search and ranking task*, in which participants had to search for Instigator GUI layouts and to order them in decreasing order of their relevance to a given instruction (*e.g.*, "Build a section with 5 images, 5 paragraphs, and 5 buttons" - Fig. 8). For the 21 instructions tested (see first column of Table 2), we extracted three subsets of Instigator-generated

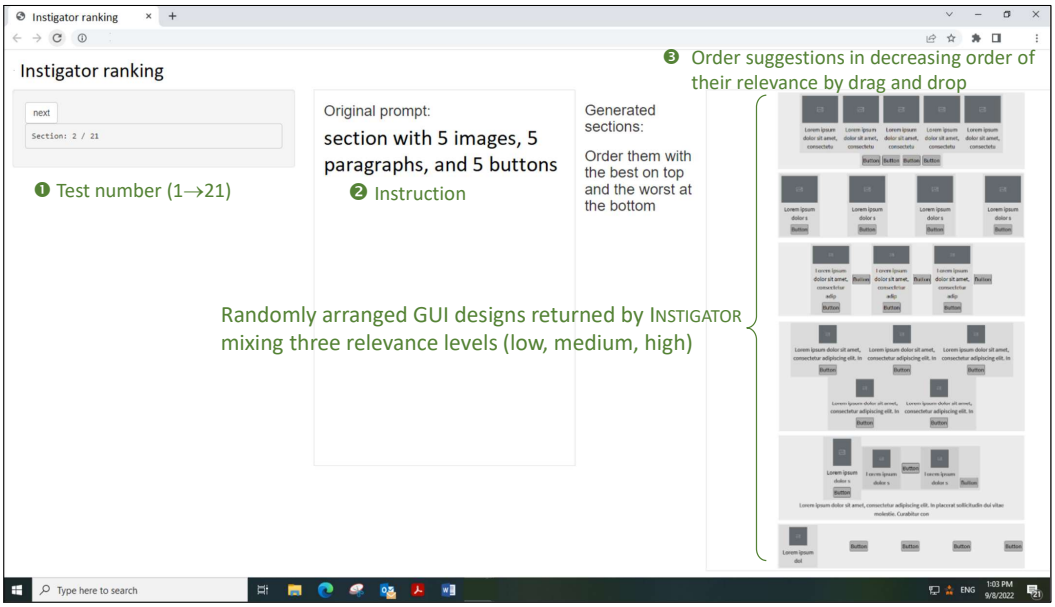

Fig. 8. Example of a testing sample.

layouts with three levels of relevance (*i.e.*,, low, medium, high - see file Instigator-rankings.pdf in the supplementary material for details). The instructions and their stimuli were randomly presented in a web application that allowed participants to arrange the stimuli by drag and drop (see Fig. 8 for a test sample). The setup was a within-subject design with 21 instructions × 7,8 stimuli = 165 trials (56 for a function + 54 for contents + 55 for a function combined with contents).

(4) An *evaluation task*, in which participants were instructed to fill in a UEQ+ questionnaire (User Experience Questionnaire) [86], a modular extension of UEQ in which we selected 13 scales covering general aspects, pragmatic and hedonic qualities according to suggestions made by Schrepp [85], *i.e.*, ATTRACTIVENESS, EFFICIENCY, PERSPICUITY, DEPENDABILITY, STIMULATION, USEFULNESS, NOVELTY, ADAPTABILITY, USEFULNESS, VALUE, VISUAL AESTHETICS, INTUITIVE USE, TRUSTWORTHINESS OF CONTENT, and QUALITY OF CONTENT to focus on evaluating the user experience of participants interacting with INSTIGATOR. Free comments could be added. We justify the use of UEQ+ for the following reasons: it explicitly covers pragmatic and hedonic dimensions of user experience (Fig. 13), not just usability, it consists of several scales to be analyzed individually and globally, there are analysis instruments and published norms for interpreting their results, which is not the case for other similar methods[1].

5.3 Quantitative and Qualitative Measures

In addition to the demographic information collected from the participants at the beginning of the session, we recorded their rankings for the search task and their answers to the scales for the evaluation task. The search task returned the participant's ranking, an integer variable defined as follows: $\forall$ instruction I_i $\forall i \in \{1, \ldots, 21\}$, $\forall$ stimulus S_{ij} $\forall j \in \{1, \ldots, 7 \text{ or } 8\}$, R_{ij} = participant's ranking of the j^{th} stimulus of the i^{th} instruction, where $R_{ij} \in \{1, \ldots, 7 \text{ or } 8\}$. Similarly, INSTIGATOR's GUI layouts were assigned to a system ranking based on their relevance (see the

[1]See https://measuringu.com/pragmatic-hedonic/

rankings in Instigator-rankings.pdf). The evaluation consists of a series of scales, where: ∀ scale $S_i, \forall i \in \{1, \ldots, n\}$ decomposed in m_i subscales $SC_{ij}, \forall j \in \{1, \ldots, m_i\}$ or items to be evaluated (*e.g.*, the scale Attractiveness is decomposed in four subscales: annoying *vs.* enjoyable, bad *vs.* good, unpleasant *vs.* pleasant, and unfriendly *vs.* friendly), each subscale SC_{ij} being a differential scale with 7 points between items of each pair (*e.g.*, annoying o o o o o o o enjoyable). We measure each subscale SC_{ij} employing a 7-point Likert scale with response categories "Strongly disagree" (=1) to "Strongly agree" (=7). For each S_i, we have in addition:

(1) Subscale score (SS): an ordinal integer variable that measures the score for each subscale SC_{ij} of a scale S_i, ranging from 1="Strongly disagree" to 7="Strongly agree".
(2) Subscale importance (SI): an ordinal integer variable that measures the weight for each subscale SC_{ij} of a scale S_i, ranging from 1="Least important" to 7="Most important".
(3) Scale mean score (SMS): a real variable that measures the average score obtained on all subscales SC_{ij} of a scale SC_i for all participants, ranging from -3="Mostly negative" to +3="Mostly positive" calculated as follows: $\mathrm{SMS}(SC_i) = \frac{\sum_{j=1}^{m} \mathrm{SS}(SC_{ij})}{n} - 4, \forall i \in \{1, \ldots, n\}$, where 4 is the scale median.
(4) Scale mean importance (SMI): a real variable that measures the average weight of importance of all subscales SC_{ij} of a scale SC_i for all participants, ranging from -3="Mostly negative" to +3="Mostly positive" computed as follows: $\mathrm{SMI}(SC_i) = \frac{\sum_{j=1}^{m} \mathrm{SS}(SI_{ij})}{n} - 4, \forall i \in \{1, \ldots, n\}$, where 4 is the scale median.

5.4 Ranking of GUI Layouts

To assess the relevance of the GUI layouts returned by Instigator with respect to the search instruction, we compared Instigator rankings and participants' rankings with four measures (Table 2): a cosine similarity to test the similarity between the rankings, a Pearson's ρ coefficient to investigate potential correlation, and a Wilcoxon test to determine whether there is any significant difference between the conditions.

We compute the **cosine similarity** [49] between vectors expressing the Instigator rankings and the participants' rankings averaged for each instruction. Cosine similarity was chosen for its simplicity of interpretation and its large adoption [96], along with other properties such as independence from vectors' length making it independent of the layout size. The measure S_C computes the cosine of the angle between the vector of Instigator rankings and the vector of participants' averaged rankings. A value of 0 means that the two vectors are 90 degrees from each other (orthogonal) and have no match. The closer the cosine value to 1, the smaller the angle and the greater the match between vector similar (positive co-linear) vectors. The first column of Table 2 shows that the individual values, ranging from 0.88 to 0.98, are all very high, suggesting significant goodness of fit between Instigator rankings and participants' rankings, expressing their perception of the adequacy of the same items to the given instructions. For example, "Build a section of type navbar" received the highest score S_C=0.98, meaning that the angles between the two vectors could be said 98% similar: when Instigator produced a layout with a low, a medium, or a high relevance to the instruction, participants mostly recognized the same relevance. This very high value is obtained because participants share a common representation of what a good layout is for this instruction. "Build a section with 2 inputs, 1 button, and 2 paragraphs" received the lowest S_C=0.88 as there are multiple ways to arrange the individual GUI elements into a cohesive layout and because this was interpreted as have the paragraphs below or after the button, which is unusual. This instruction was deliberately proposed as it is ending with a paragraph, which

is supposed to appear sooner in the instruction. Apparently, Instigator returns a worse result when the instruction mixes the structures. On the contrary, it returns a better result when the instruction contains progressively finer structuring elements: putting a paragraph at the beginning of an instruction is structuring.

We compute a **two-tail Pearson's ρ coefficient** with α=0.05. The second column of Table 2 shows a highly significant correlation for most instructions (17/21 with $p \leq 0.001^{***}$). 2/21 instructions and 1/21 instruction received $p \leq 0.01^{**}$ and $p \leq 0.05^{*}$, respectively. Only "Build a section with 1 heading, 2 paragraphs, 2 images" did not receive any significant correlation as a potential confusion between the grouping of paragraphs and images arose. Participants were wondering if these elements were related, and therefore subject to grouping, or not. Seven instructions received a high degree of correlation, five, a moderate one, and eight, a low degree. Overall, the degree for all instructions is still moderate, thereby suggesting that Instigator rankings are correlated to participants' rankings.

We finally compute a **Wilcoxon signed-rank test** (with ties and continuity corrections and 1,000 iterations) for paired samples by aligning Instigator' rankings and participants' rankings for the same instruction. No significant difference was found between the two series of rankings, both individually for each instruction and globally for all instructions (see last column of Table 2).

5.5 Scale Mean Scores and Scale Mean Importances

This section provides an overview of the results that will be further detailed, scale by scale, in Section 5.9. We compute and interpret participants' responses to the selected scales with the UEQ+ data analysis tool to report the results in Table 3 and shown in Fig. 9 and Fig. 10, respectively, for each scale on the reference interval [-3,+3] [87].

Overall, all scales were positively assessed above 0.76, thereby suggesting that the participants judged Instigator fulfilling their expectations regarding these 13 aspects. Perspicuity (M=2.06, SD=1.10) and Intuitive Use (M=1.88, SD=1.10) received the two highest scores along with high importance (M=2.0, SD=1.0 and M=2.03, SD=1.04, resp.). The importance of all scales was also uniformly positive above 1.12. Adaptability (M=0.76, SD=1.39) and Visual Aesthetics (M=0.77, SD=1.29) were judged by the participants as being the less convincing aspects of Instigator, while being of relative importance (M=1.5, SD=1.22 ; M=1.12, SD=1.28, resp.). Usefulness (M=1.67, SD=0.99) was estimated as the most important scale (M=2.24, SD=0.92), followed by Intuitive Use (M=1.88, SD=1.10) which also received a highly positive mean score (M=2.03, SD=1.04). As the scale mean scores are significantly different from each other (a Kruskall-Wallis test gives H_{stat}=155.94, H_{ties}=166.71, df=12, $p \leq 0.0001^{****}$), we compute a Nemenyi pair-wise test to find the scales that differ (the results of Table 4 are reproduced in Fig. 9). We compute Cohen's d coefficient for determining the effect size (8 small, 15 medium, and 5 large). Similarly, we compute a second Kruskall-Wallis test to determine whether any significant difference exists between scale mean importances. We find out only one difference: Usefulness is significantly larger than Visual Aesthetics (R_{crit}=120.60, SD=21.90, q_{stat}=5.50, $p \leq 0.0067^{**}$).

Cronbach's α ranges from 0.73 for Dependability to 0.95 for Visual aesthetics with nine scales interpreted as 'Excellent', three scales as 'Good' and only one as 'Acceptable' (Table 3), the mean being equal to 0.89, interpreted as 'Good', almost as 'Excellent'. Despite their varying expertise level and design experience, participants answered the scales in a consistent way. To assess the internal consistency of participants for the subscale importance, we compute three measures: Cronbach's coefficient α=0.82 (interpreted as 'Good'), Spearman's rank correlation coefficient ρ=0.93 (interpreted as 'Very strong', the highest value), and Guttman's reliability λ_2=0.95 (computed with 2,000 iterations and interpreted as 'Very high'). This internal consistency establishes a sound basis for the discussion of the results.

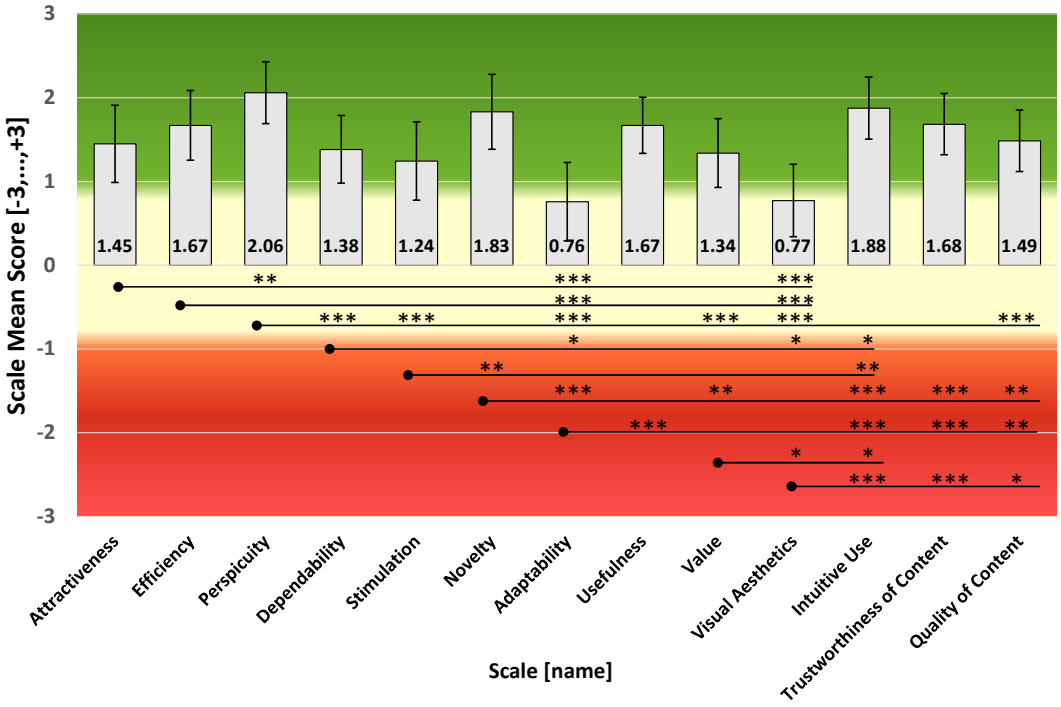

Fig. 9. Scale Mean Scores (SMS) obtained for Instigator. Error bars show a confidence interval of 95%. Significance level: $p \leq 0.05^{*}$, $p \leq 0.01^{**}$, $p \leq 0.001^{***}$, *n.s.*=not significant.

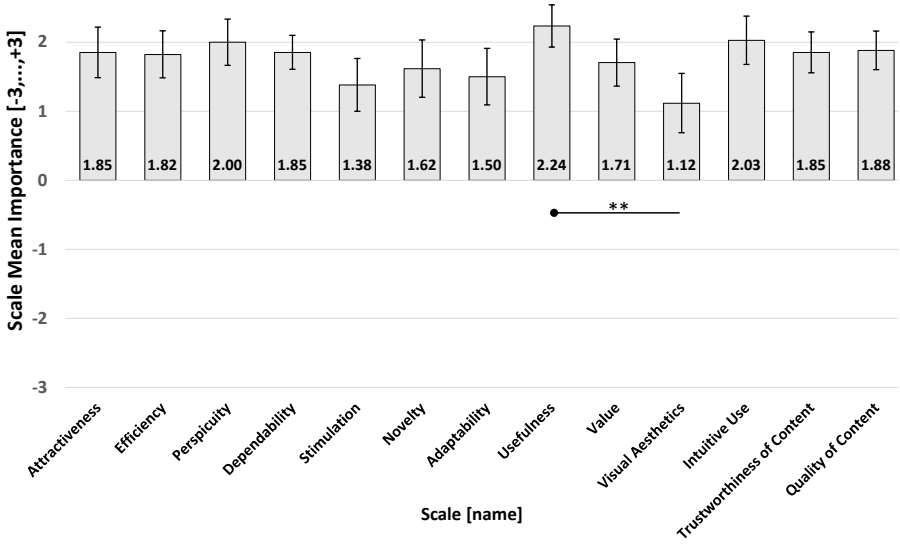

Fig. 10. Scale Mean Importances (SMI) for Instigator. Error bars show a confidence interval of 95%.

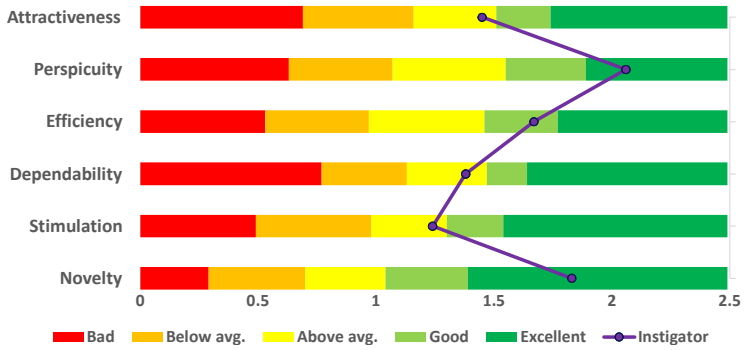

Fig. 11. Benchmarking of the six main scales.

5.6 Benchmarking of Scales

Schrepp *et al.* [86] mention intervals that are more precise than the 3 zones (bad, neutral, good in Fig. 9) initially defined to interpret some scales, based on a benchmarking performed on a set of evaluations. Fig. 11 shows the distribution of the six scales, decomposed into five intervals based on this benchmarking: bad, below average, above average, good, and excellent. Perspicuity and Novelty are rated as 'Excellent', Efficiency is rated as 'Good', and Attractiveness, Dependability, and Stimulation are rated as 'Above the average', all these being positive signs of their appreciation.

5.7 Key Performance Indicators

Another recent development of UEQ+ is the construction of the Key Performance Indicators (KPI) extension [32]. The KPI combines the subjectively perceived importance of user experience factors and the results of the UEQ+ into one figure to better understand the relationship between the scale mean scores and their importance. The mean KPI (M=1.51, SD=0.67) is above the positive threshold of 1. Although the theoretical value of the KPI is its value is again in an interval [-3, +3], Hinderks *et al.* [31] empirically determined that the practical value range is between -0.286 and 2.14. Having a mean KPI_M=1.51, we could estimate that the overall KPI estimation is quite positive for Instigator. Two individual KPIs are low, KPI_6=−0.10 and KPI_{30}=0.19, which means that these two participants estimated some scales as irrelevant or less important and therefore rated them with corresponding lower values. Six individual KPIs are more positive, but below the threshold of 1, *i.e.*,, KPI_5=0.98, KPI_{15}=0.76, KPI_{19}=0.76, KPI_{21}=0.91, KPI_{25}=0.76, and KPI_{32}=0.78, the remaining 26 participants being above the threshold and $\frac{15}{34}$=44% above the mean KPI.

5.8 Importance Performance Analysis

The Importance-Performance Analysis (IPA) [30] aims to assign every scale to four different quadrants determined by two methods: (1) a differentiation by the coordinate origin at $(0, 0)$, which is represented by a solid green line in Fig. 12; (2) a differentiation by the coordinate origin in the mean value of all scale values ($M_{\text{Performance}}$=1.48, $M_{\text{Importance}}$=1.76, two very high values), which is represented by a green dotted line in Fig. 12. With respect to the first method, all scales are located in the first quadrant "Q1=Keep Up the Good Work". With respect to the second method which is more demanding than the first one, six important scales remain in the top quadrant "Q1=Keep Up the Good Work": Usefulness, Intuitive Use, Perspicuity, Quality of Contents, Trustworthiness, and Efficiency as sorted in decreasing order of their preference. Novelty is located in "Q2=Possible Overkill", thereby suggesting that the novelty of the process supported by Instigator is certainly original, but perhaps overestimated. Four scales are located in "Q3=Low

Priority": Adaptability and Visual Aesthetics –which were already identified as such in the KPI analysis–, and Stimulation and Value. This scale, along with the two remaining scales located in "Q4=Concentrate Here", *i.e.*, Attractiveness and Dependability are very close to the mean.

5.9 Individual Scale Analysis

Having accumulated the various results obtained with the different instruments, this section now discusses each scale individually in detail by gathering all results per scale.

Attractiveness. This scale is the most important among all scales as it captures "a user's general impression" of Instigator, one of the three dimensions of user experience, and the top scale of UEQ+ structure (Fig. 13). The perceived attractiveness is considered to be the result of an averaging process of the perceived quality of the software with respect to the relevant aspects in a given usage scenario [50]. The mean score (±*SD*) of Attractiveness was 1.45 (1.37) with high mean importance (±*SD*) of 1.85 (1.09), all these figures belonging to the positive zone (Fig. 10), thus being interpreted as a good assessment. This assessment is also supported by an 'Above average' position in the benchmarking (Fig. 11) and by Q1 and Q4 positions (Table 1). All these results concur with a positive assessment of this scale, being credible by an excellent internal consistency of participants (α=0.90). This perceived attractiveness is suggested to be good enough to be incorporated into a user interface design process, especially with expected GUI layouts (Section 5.4).

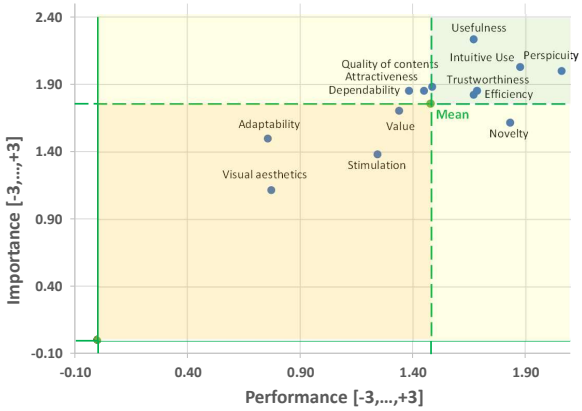

Fig. 12. Importance Performance Analysis of Instigator.

Scale	Center	
	(0,0)	Mean
Attractiveness	Q1	Q4
Efficiency	Q1	Q1
Perspicuity	Q1	Q1
Dependability	Q1	Q4
Stimulation	Q1	Q3
Novelty	Q1	Q2
Adaptability	Q1	Q3
Usefulness	Q1	Q1
Value	Q1	Q3
Visual Aesthetics	Q1	Q3
Intuitive Use	Q1	Q1
Trustworthiness of Content	Q1	Q1
Quality of Content	Q1	Q1

Table 1. Assignment Scales to IPA quadrants according to the two methods: Q1="Keep Up the Good Work", Q2="Possible Overkill", Q3="Low Priority", Q4="Concentrate Here".

Efficiency. This scale is considered to be a pragmatic quality of user experience and is "goal-directed" (its assessment is based on tasks that can be performed with Instigator) [86]. Efficiency (M=1.67, SD=1.54) belongs to the positive zone with its high importance (M=1.82, SD=1.00), is interpreted as 'good' in the benchmarking and is located in Q1 in both scenarios, with a good interrater consistency (α=0.88). This perceived efficiency suggests that participants felt quick enough to search for GUI layouts and to find them in order to incorporate them into the rest of the design process. This efficiency is appreciated in the capability to decompose a suggested layout

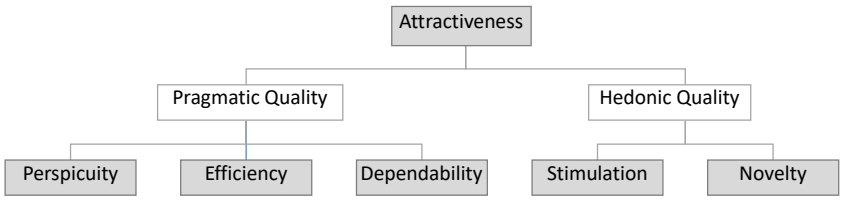

Fig. 13. Scale structure of UEQ+ [85, 87].

into elements that can be edited independently of each other, therefore not forcing the practitioner to either completely adopt a layout or reject it.

Perspicuity. This scale expresses to what extent participants considered it easy to get familiar with Instigator and to learn how to use it. Therefore, it is also considered a pragmatic quality that is an important part of the user experience and is "goal-directed" [86]. Perspicuity (M=2.06, SD=1.10, the highest mean score of all scales) belongs to the positive zone with its very high importance (M=2.00, SD=1.00), is interpreted as 'excellent' in the benchmarking and is the rightmost scale in Q1 in both scenarios, with an excellent consistency (α=0.93). This perceived perspicuity suggests that participants felt extremely comfortable with the way Instigator is run and the way the process is structured in steps, which they considered to be the fundamental quality, far more than other factors (Perspicuity is significantly higher than 6 other scales in the bottom part of Fig. 10)

Dependability. This scale expresses to what extent participants felt in control of Instigator. Therefore, it is considered as the third pragmatic quality (Fig. 13 and is "goal-directed" [86]. Dependability (M=1.38, SD=1.20) belongs to the positive zone with high importance (M=1.85, SD=0.73), is interpreted as 'Above average' in the benchmarking and is located in Q1 and Q4 in the scenarios, with an acceptable consistency (α=0.74). This perceived perspicuity suggests that participants had a moderate illusion of being in control of Instigator, which is defined as an explicit dialog control, the best configuration. This should be mitigated by a potential loss of control when defining the instruction. How the controlled vocabulary could be used for instruction should be improved. For example, "Build a section of type form" is moderately controllable and predictable as "type" could have multiple meanings: typing a password is different from typing a city name or a specific term like a person's name to be used as input by a local search tool. The instruction "Build a section with 3 images and 3 paragraphs" generates paragraphs placed under each image, while a more flexible way would be to use additional constructs such as "Build a <content|navigation|album> section with 3 images and 3 paragraphs placed below" or using a direction indicator such as "ltr" (left to right). Other options could refine the terminology: "image" could be attached to some refined terms to specify the semantics, such as "photo", "diagram", "logo", "chart" with some modifiers like "big", "small", "medium". This would give for example instructions structured as follows: "Build an aside/main section with 2 small photos, 1 search input, 2 navigation links, and 1 download button", "Build a section with 5 images, 5 paragraphs 5 buttons by rows/columns". Then, if "columns" is chosen, Instigator could automatically generate 5 columns, each column having 1 image, 1 paragraph, 1 button.

Stimulation. This scale expresses to what extent participants felt motivated to use Instigator in the context of work. Therefore, it is considered a hedonic quality (Fig. 13 and is not "goal-directed" [86]. Stimulation (M=1.24, SD=1.39) belongs to the positive zone with high importance (M=1.38, SD=1.14), is interpreted as 'Above average' in the benchmarking and is located in Q1 and Q3 in the scenarios, with a good consistency (α=0.83). This perceived stimulation suggests that participants were motivated enough to consider Instigator in their context of work, particularly when they search for inspiration (*e.g.*, for novice practitioners) or for comparison (*e.g.*, for expert

practitioners). Although the participants recognized the advantage of being fed with GUI layouts, they were concerned about being forced to always reuse the same layouts that can be found elsewhere and not being creative enough, which limits their stimulation.

Novelty. In our study, this scale expresses to what extent participants felt that searching for GUI layouts in this way is novel and original enough to warrant their practical use. It is the second hedonic quality of UEQ+. Novelty (M=1.83, SD=1.33) belongs to the positive zone with high importance (M=1.62, SD=1.24), is interpreted as 'excellent' in the benchmarking and is located in Q1 and Q2, with an excellent consistency (α=0.93). This perceived stimulation suggests that participants felt convinced that Instigator was supporting them in a novel way, but that this could be overestimated (Q2) because they never saw any similar software. Note that this surprise effect has been identified at a time before ChatGPT was released end of November 2022.

Adaptability. This scale expresses the extent to which participants felt that Instigator could be adapted to a wide range of personal parameters, such as their own development expertise level or individual way of working. Adaptability (M=0.76, SD=1.39) belongs to the neutral zone (Fig. 10) with moderate importance (M=1.50, SD=1.22) and is located in Q1 and Q3, with an excellent consistency (α=0.93). Adaptability is also significantly lower than five other UEQ+ scales (Fig. 10), which makes it one of the lowest priority factors. This perceived adaptability suggests that participants felt that Instigator should be made more adaptable to their own parameters and context of work. Indeed, Instigator returns always the same layouts for the same instruction without taking into account previously used layouts, preferred layouts, or layouts depending on other conditions. While consistency is certainly appreciated in this process, each time a GUI layout is selected, composed or edited (Fig. 1) should update the results, *e.g.*, by reinforcement learning to adapt the process [94] and to improve the affordance [58].

Usefulness. This scale expresses how useful the participants felt the Instigator was in searching for GUI layouts. Usefulness (M=1.67, SD=0.99) belongs to the positive zone with the highest importance among all scales (M=2.24, SD=0.92, with the maximal median Mdn=6.5) and is located highest in Q1="Keep Up the Good Work" in both scenarios, with an excellent consistency (α=0.91). These results are particularly important since the main goal of Instigator lies in its usefulness first before guaranteeing its user experience. This perceived usefulness suggests that participants recognized the main advantages brought by Instigator, mainly the suggestion and the capability to edit them.

Value. This scale expresses to what extent the participants felt that Instigator adopted a professional look & feel and is of high quality enough to admit an added value in using it. Value (M=1.34, SD=1.22) belongs to the positive zone with high importance (M=1.71, SD=1.02) and is located in Q1 and Q3, with an excellent consistency (α=0.90). This perceived value suggests that participants felt that Instigator was quite valid from a professional rendering point of view, although they agreed that this was not the priority (since located in Q3), as if the current state of the Instigator Look & Feel did not call for negative comments.

Visual Aesthetics. This scale expresses to what extent the participants agree to recognize Instigator as having an aesthetically pleasing presentation and behavior. In our case, aesthetic quality can apply both to Instigator and to the layouts produced by that software. The aesthetic quality of the produced layouts could very well have been evaluated, notably by tools such as Questihm [108] or AIM [71]. The Visual Aesthetics (M=0.77, SD=1.29) belongs to the medium zone with the lowest, yet positive, importance (M=1.12, SD=1.28) and is located in Q1 and Q3, with an excellent consistency (α=0.95). Visual aesthetics as perceived by the participants suggest that

this factor is sufficient for the purpose of producing GUI layouts without touching the aesthetics of the produced layouts themselves since they are widely varying and inevitably affect the GUI usability.

Intuitive Use. This scale expresses the extent to which participants were able to interact with Instigator in the search task assigned to them with minimal use of any form of assistance, help, or guidance. Intuitive Use (M=1.88, SD=1.10) belongs to the positive zone with the second highest importance (M=2.03, SD=1.04) and is located in Q1 for both scenarios, with a good consistency (α=0.89). This perceived intuitiveness suggests that participants benefit from the discovery and the warm-up tasks (Section 5.2 to properly interact Instigator and that this familiarisation period was enough for them to manage with the software, which represents a lightweight introduction.

Trustworthiness of Content. This scale expresses the extent to which participants believed that the layouts generated by Instigator are of good quality and reliable. Trustworthiness of Content (M=1.68, SD=1.09) belongs to the positive zone with high importance (M=1.85, SD=0.88) and is located in Q1 for both scenarios, with an excellent consistency (α=0.90). This perceived trustworthiness suggests that participants believed that produced GUI layouts are of enough quality in general and that they remain in this good quality over time. They are likely to trust Instigator.

Quality of Content. This scale expresses the extent to which participants believed that the layouts generated by Instigator are actual and well-designed. Quality of Content (M=1.49, SD=1.09) belongs to the positive zone with high importance (M=1.88, SD=0.83) and is located in Q1 for both scenarios, with an excellent consistency (α=0.90). This perceived quality of content suggests that participants estimated that the layouts are representative enough of actual layouts found on up-to-date websites.

6 POSITIONING, LIMITATIONS, VALIDITY THREATS, AND PERSPECTIVES

This section first positions Instigator with respect to other tools of this state-of-the-art (Section 6.1), then discusses some limitations of this software and its evaluation as we conducted it (Section 6.2), some threats to validity (Section 6.3), and finally proposes some perspectives for improving future similar tools (Section 6.4).

6.1 Positioning Instigator with respect to Prior Works

Instigator can be compared in terms of similarities and differences to prior works for searching GUI layouts, whether they are based on machine learning or not.

Combine the computational design with a data-driven approach. Computational design [72] generates GUI layouts using an algorithm [24, 95] while data-driven approaches determine them by clustering and classifying a dataset [21, 53] and/or by visualizing it [6, 56]. Instigator combines both a GPT-generated language model and a dataset, where each GUI layout can be composed, edited, and stored in a GUI model, such as in [76]. Whereas Scones [34] and Stylette [41] generate sketches from text and restyle layouts, resp., Instigator introduces a language model of the contents, the structure, and the properties of GUI layouts.

Search by natural language. Search in datasets has been achieved primarily by code [38, 77], by keywords [79], by image, sketch [107] or wireframe [10, 13], by component [6, 22], and by visual exploration [56]. These are potentially time-consuming tasks because practitioners do not quickly find GUI layouts that are relevant to their project [13]. Most recently, Rawi [45, 46] proposed a data-driven GUI prototyping approach that retrieves GUI layouts from a large-scale semi-automatically created GUI dataset for mobile apps using natural language.

Free oneself from the constraint of having a GUI layout as input. Several tools take a GUI layout, expressed as a hand-drawn sketch [26], an image [6]), or a wireframe [5] as input for their search and retrieval. This form of input assumes that the user has already thought in terms of the physical layout before thinking in terms of content and that she has already a clear idea of the desired layout, whether it is a partial or complete layout. This assumption does not hold precisely for novice designers who have little or no experience, but also for other stakeholders who have no design background and rely solely on their user experience and wish to look for some inspiration [29]. In contrast, Instigator does not impose this constraint and offers the end user to search in terms of contents independently of any layout.

Maximize autonomy with respect to the level of fidelity. Any visual representation of a GUI layout inevitably implies some fidelity, which can span several levels [65] (low, medium, high, or mixed [19]). A sketch is typical of low to medium fidelity; a wireframe also spans from low to medium fidelity but could go up to high if GUI elements are finely represented. A screenshot is always high-fidelity (*e.g.*, screenshots in [5, 6] are tied to Android Look & Feel), but can be transformed into another level [19] or editable representation [56, 91]. Some tools are limited to only one level of fidelity, while some others benefit from several levels of fidelity (*e.g.*, VINS [10] supports both low-fidelity and high-fidelity; SketchiXML [19] forces the designer to lay out a GUI in low-fidelity and automatically switch to medium-fidelity or high-fidelity at runtime but still requires some level of fidelity). In all cases, there is a dependence on some level of fidelity that is no longer needed when layouts can be searched by natural language in terms of contents, not in terms of physical properties. Instead of imposing a certain level of fidelity, Instigator defers the decision to rely on a given level of fidelity by allowing the user to think first in terms of content and structure, and later in terms of fidelity.

Maximize autonomy with respect to the target platform. Several tools focus on GUI layouts for one target platform, device, or operating system at a time, often with the same fixed resolution. For instance, VINS [10] targets mobile designs, Ge [26] targets mobile designs for Android. On the contrary, Instigator is expected to be as independent as possible from any target platform as GUI layouts resulting from the query could be suitable for one or multiple platforms, and later refined. When it comes to tailoring a design for a particular platform, this specification could be entered in natural language to restrict the scope of admissible designs.

Search GUI layouts also by dissimilarity. Several tools retrieve a list of GUI layouts that are considered similar to the initially provided or imaginary GUI layout, whatever the search method, thus privileging similarity as the key factor. Instead, Instigator relaxes this constraint by suggesting alternate GUI layouts having layouts different from the input, provided that the GUI contents and structure are preserved to some extent, thus enabling the exploration of a larger design space than a space explored by similarity only.

Ensure continuity throughout the design process. One of the most common processes for GUI layout is to start with a low-fidelity GUI layout and then progressively refine it to reach a high-fidelity prototype [13, 19]. This is not only to support the iterative design but also to better support a participatory design in which stakeholders provide more feedback on the current design as it evolves from one level of fidelity to another [82].

6.2 Limitations

Despite Instigator's success in providing an adequate layout with respect to instructions provided by users, several improvements could be brought to the application in order to improve its efficiency or overall quality.

What Instigator can generate is what Instigator was trained for. First, tags used to train Instigator are limited to the eleven classes of GUI elements (see a reminder in the top

left of Fig. 2). Therefore, GUI layouts are limited by these elements only, thereby leaving other elements unsupported. For example, custom widgets found in APIs or libraries are left unsupported. JavaScript code for GUI widgets is also left unsupported. Second, an instruction submitted to Instigator could effectively include elements belonging to these classes, but in a structure that has not been found in the training set, thus returning weird combinations of elements or layouts missing some elements. While the instruction could, in theory, incorporate any structure of GUI elements, Instigator will in practice generate layouts that have only been trained beforehand. Instigator has been pre-trained and remains as it is, thereby meaning that there is no long-term memory that learns from each instruction. No reinforcement learning is included.

Limited instruction size. Transformer architectures accept a limited size for input, *e.g.*, 2,048 tokens for GPT-3. Consequently, the user cannot enter a too-long instruction as input for the output.

Lack of explainability. Instigator is prone to the same problems many neural networks face: their inability to explain and interpret why certain inputs result in specific outputs, *e.g.*, why a layout does not match the instruction.

Single modality for user interaction. Users can currently enter instructions to Instigator only through textual interaction in a partial language. Although textual interaction through the keyboard is the most common way of interaction today, instructions can be recognized by speech-to-text CMUSphinx, but that was rarely used. The major limitation lies in the perception of the language. While experts used the vocabulary they knew, the others did not know exactly what terms could be accepted in the language and what their syntax was. These limitations concern classical NLP challenges, such as discoverability (*e.g.*, how to discover the language, its syntax, and its relationship to the output), contextual understanding (*e.g.*, an instruction could hold different meanings depending on its structure), support for synonyms (*e.g.*, "1 button" and its synonym "one pushbutton" are both recognized but not made observable), and structure variations (*e.g.*, how the syntax could express the inclusion of GUI elements in a container was not clear enough: "Build a section with 5 headings, 5 images, and 5 paragraphs" could refer to the same structure repeated five times or to a structure with 5 headings first, then 5 images, then 5 paragraphs or a mixture of them).

Only structure- and content-oriented results. Instigator currently outputs designs that maximize the adequacy between the structure and content of the results and the instruction given by the user. While this allows users to focus on the structural part of their design, the lack of styling capabilities of Instigator makes it unsuitable for various users or workflows for which the design and styling steps are not considered sequentially.

Web-only GUI layouts. While Instigator allows users to refine a layout to multiple platforms (*e.g.*, desktop, mobile), it only allows for the design of the structure and content of web-based applications, making the design of other kinds of applications (*e.g.*, native GUIs for iOS, Android operating systems) currently beyond the reach of practitioners.

6.3 Threats to Validity

We explain the validity threats of our empirical study in terms of four categories: internal, external, construct, and conclusion threats to validity [103].

Internal validity. The learning effect, the understandability of the GUI layouts in the 21 instructions, and the instrumentation validity must be considered. The learning effect was mitigated by randomizing the order in which the 7-8 GUI layouts returned by Instigator were presented for each instruction (Fig. 8). In a validation pre-test, the understandability of the instructions was assessed and the experimental material was evaluated by an experienced researcher in HCI. We

mitigated the instrumentation validity by using a specific application for testing that was aimed to place any participant in a context comparable to web browsing.

External Validity. The representatives of the results, and the size and complexity of instructions that might affect the generalization of the results must be considered. The representatives of the results could have been affected by the number of GUI layouts ranked. We believe that the GUI layouts selected for this study can be considered as a baseline to obtain some indications of which parts perform better results on the matching between Instigator and the participants. However, the total number of GUI layouts cannot represent the whole GUI design space: layouts returned by Instigator are only based on those used for training (see Figs. 2 and 3 for some examples). We plan to conduct more studies with more and different conditions to increase the reliability of our results. Regarding the size and complexity of the task, we decided to use relatively small tasks to enable participants to complete the whole session within 30 minutes. Replications with different and more complex GUI layouts are nevertheless needed to study the effect of their variables on the observed results.

Construct Validity. The 13 UEQ+ scales and their associated measures used to evaluate the UX of Instigator by the participants and the participants' apprehension about being evaluated must be considered. We mitigated the threat regarding the scales used by relying on the official SAP' UEQ+ modular version of UEQ [32, 85, 86], which calculations are based on relevant prior research in UX evaluation that has been empirically validated on many UIs [30, 31, 87] Regarding the participants' possible apprehension about being evaluated, we believe that the Hawthorne effect [64] was mitigated by two means: (1) instead of having an experimenter present the stimuli in person in front of the participants, the participants ranked the stimuli directly alone in an independent testing application (Fig. 8); (2) the participants were not evaluated on their own performance but have been told to evaluate the GUI layouts returned by a software. Bias introduced into the study by expectations on the part of the experimenter was mitigated by the warm-up task that the participants had used and also by interacting with the participants before each session. However, the scales' answers can be influenced by how participants remember their previous experience [51] and not by the real experience itself [47] or by the Hawthorne effect [64].

Conclusion Validity. With regard to conclusion validity, the data collection and the validity of the statistical tests applied must be considered. With the purpose of decreasing the data collection threat, the same data-extraction procedure was consistently applied in each session with each participant thanks to the dedicated testing application. Regarding the statistical tests, proper tests were performed to test our hypotheses [63]. To select the statistical tests, we considered the design of the experiment and the nature of the variables and their assumptions according to Maxwell [62]. However, Laugwitz et al. [50] stated that the UEQ+ scales are in theory independent of each other, which was not observed by Schankin et al. [84] who report some correlation between scales.

6.4 Perspectives

This section proposes some implications for improving the development of more supportive design search engines based on the aforementioned limitations of Instigator and the results of its evaluation. These perspectives can therefore feed a set of generative AI design practices [102].

Justify suggested GUI layouts on demand. Some search engines do not always return relevant designs. For example, GuiFetch [5] found a suggestion in 65% of the cases. Image-, sketch- and screenshot-based tools return only designs already existing in the dataset. For example, a GUI layout from a dataset [26] is suggested only if it corresponds to the hand-sketched design. Instigator always suggests some GUI layouts since they are built at run-time based on the prompt. To better understand, and therefore accept or reject a design, users requested that more explanations [89] should be provided on-demand justifying why a design is suggested, *e.g.*, in terms of similarity, of

matching the contents, the structure, and the properties when used. These parameters could be used to control the design suggestion, such as Scout [92] in its feedback panel. If a design suggested by Instigator corresponds to a previously crawled design, its source should be indicated. If a design has been created from multiple sources, they should be mentioned.

Better support the comparison among GUI layouts. Wu *et al.* [105] pointed out the importance of comparing GUI layouts, namely by juxtaposing suggestions to designs coming from case studies or from concurrence. Everything should be done to estimate the relative deviation between suggested designs. While Instigator displays the designs in decreasing order of their matching to the initial prompt, the deviation between two subsequent designs can be slight or important. Therefore, two or more suggested designs should be compared against selected criteria, perhaps by a Design Map [56]. For example, positioning each design in terms of aesthetic metrics could be reinforced by displaying it on a color gradient indicating the positive and negative zones, which is achieved by GuiComp [53] and Questim [108].

Introduce context-dependent layouts. If the context of use is defined as a triple (user, platform/device, environment) [11], some tools are considered context-dependent (*e.g.*, [26] is platform-dependent as it returns Android mobile designs) while some others are considered context-independent as they are agnostic of the target context (*e.g.*, [13] manipulates platform-independent wireframes). Similarly, Instigator manipulates GUI layouts that are agnostic of constraints imposed by the target context of use -which is an advantage - but should progressively incorporate context-dependent constraints when needed. These constraints could restrict the scope of suggested GUI layouts or adapt them accordingly. For example, accessibility constraints for diverse users, screen constraints for diverse platforms or devices (*e.g.*, for cross-device interaction), and physical constraints for diverse environments.

Support transition from novice to expert. As for any language, whether programming or natural, it is necessary to know its syntax to best exploit it. Although this syntax is based on GUI elements with their names and properties, it may be familiar to an expert, but not to a novice. To bring a novice to an expert level, the system could objectify the syntax with progressive examples and a history of prompts and related GUI layouts, possibly shared by a local community or a global crowd, to show the possibilities of the language.

Customize the suggestion of GUI layouts. Right now, Instigator remains consistent for all users: the same GUI layouts are suggested for the same prompts. However, some stakeholders want to personalize, to customize Instigator to their specific needs that are domain- or context-dependent. Reinforcement learning [2] or multi-objective deep reinforcement learning to support multiple criteria simultaneously [88] could be used to learn when and why a user has accepted (positive reinforcement), rejected (negative reinforcement), compared, and preferred (preference analysis) a suggested GUI layout to some others and tailor the system accordingly. Furthermore, the user may want to incorporate final GUI layouts in the process to facilitate its reuse, and to privilege its own designs, especially for compliance with style guide [20]. Since Instigator is context-independent, it does not consider prompts related to a particular domain: "Build a form with five sections composed of a label and an edit field" is admissible, but "Build a bank transfer form" is not. Incorporating domain-specific aspects is feasible (*e.g.*, topic modelling [55]) and should enable the user to customize the language with these terms and to re-train Instigator accordingly. Model re-training while taking into account user feedback could improve the robustness over time and in user-dependent contexts.

Introduce diversification metrics. Another issue that may hinder the capabilities of Instigator to provide adequate GUI suggestions is the lack of diversity in outputs. This lack of diversity may vary depending on the instruction entered on the prompt and may be mitigated for generic

instructions. We advocate for the implementation of diversification mechanisms based on diversification metrics [39, 98], which could potentially improve the adequacy of Instigator outputs with respect to some instructions, especially in the context of comparison. The experiment conducted in Section 5 deliberately included designs that are estimated to have high, medium, and low rankings, but this is not sufficient to support diversification.

7 CONCLUSION

To address the research question of whether a large language model-based system for searching GUI layouts would contribute needed serendipity to the design process and would not introduce irrelevant layouts, this paper reports the results of a controlled study (N=34) evaluating how Instigator, an LLM-based system for searching GUI layouts of web pages by generative pre-trained training, would return GUI layouts that are relevant and what would be the user experience of practitioners interacting with it. Instigator is obtained by parsing the code of >100k web sites, extracting their structure, and classifying their GUI elements to create a dataset of >500k individual GUI elements, such as widgets, images, banners, icons, as well as how they are structured in forms, menus, navigation bars, headers, footers. The transformer enables us to introduce a nearly natural language that practitioners can use to search for GUI layouts by content, structure, and properties.

First, we found out that the rankings of GUI layouts produced by Instigator based on a given instruction are highly correlated to the rankings expressed by the participants for the same layouts: good layouts were perceived as such, as well as bad layouts. This contributes to answering the sub-question of Instigator not producing any irrelevant layouts.

Second, we evaluated the user experience of Instigator with the N=34 practitioners, who assessed thirteen scales of the UEQ+ evaluation method and their importance. After performing a detailed inference analysis, a benchmarking of these scales, an analysis of key performance indicators, and an importance-performance analysis, we concluded with an in-depth analysis of each individual scale to pinpoint a list of scales sorted in decreasing order of their perceived quality: perspicuity, intuitive use, usefulness, trustworthiness, and efficiency were the most appreciated scale; novelty was also appreciated but perhaps overestimated due to a surprise effect; quality of contents, attractiveness, and dependability were seen as encouraging to be improved; and value, stimulation, adaptability, and visual aesthetics were the least appreciated scales, mainly due to their low importance. After positioning Instigator with respect to prior works and based on our results, we devise six perspectives for improving future tools that are similar to Instigator in principle.

Third, we demonstrated how any GUI layout returned by Instigator, in parts or whole, can be edited and composed with other layouts and elements to form a final GUI that can be rendered in high fidelity and transformed into code, which seems a seamless integration into the user interface development life cycle. While we observed some limitations that are intrinsic to natural language understanding, it still offers wide-ranging benefits to any practitioner. Through the development of Instigator, we hope to leverage the capabilities of any stakeholder to effectively and efficiently come up with quality GUI layouts that can be incorporated into the whole development life cycle.

ACKNOWLEDGMENTS

The authors of this paper are very grateful to the anonymous EICS reviewers and the associate chair for their insightful and constructive comments on earlier versions of this manuscript. We warmly thank Dimitri Fichou for inventing and developing Instigator. Nicolas Burny and Jean Vanderdonckt are supported by the EU EIC Pathfinder-Awareness Inside challenge "Symbiotik" project (1 Oct. 2022-31 Dec. 2026) under Grant no. 101071147. Arthur Sluÿters is funded by the "Fonds de la Recherche Scientifique - FNRS" under Grant n°40001931 and n°40011629.

REFERENCES

[1] Toufique Ahmed and Premkumar Devanbu. 2023. Few-Shot Training LLMs for Project-Specific Code-Summarization. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering* (Rochester, MI, USA) *(ASE '22)*. Association for Computing Machinery, New York, NY, USA, Article 177, 5 pages. https://doi.org/10.1145/3551349.3559555

[2] Kei Aoki, Hajime Kimura, and Shigenobu Kobayashi. 2004. Distributed Reinforcement Learning using Bi-directional Decision Making for Multicriteria Control of Multi-Stage Flow Systems. In *Proceedings of IAS '04*. http://sysplan.nams.kyushu-u.ac.jp/gen/papers/klab2004/IAS8AokiBDM.pdf

[3] Gideon Avrahami, Kenneth P. Brooks, and Marc H. Brown. 1989. A Two-View Approach to Constructing User Interfaces. In *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '89)*. Association for Computing Machinery, New York, NY, USA, 137–146. https://doi.org/10.1145/74333.74347

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proc. of 3rd International Conference on Learning Representations* (San Diego, CA, USA) *(ICLR '15)*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1409.0473

[5] F. Behrang, S. P. Reiss, and A. Orso. 2018. GUIFetch: Supporting App Design and Development through GUI Search. In *2018 IEEE/ACM 5th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*. IEEE Computer Society, Los Alamitos, CA, USA, 236–246. https://doi.ieeecomputersociety.org/

[6] C. Bernal-Cardenas, K. Moran, M. Tufano, Z. Liu, L. Nan, Z. Shi, and D. Poshyvanyk. 2019. Guigle: A GUI Search Engine for Android Apps. In *Proceedings of IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion '19)*. IEEE Computer Society, Los Alamitos, CA, USA, 71–74. https://doi.org/10.1109/ICSE-Companion.2019.00041

[7] Michael Mose Biskjaer, Bo T. Christensen, Morten Friis-Olivarius, Sille J.J. Abildgaard, Caroline Lundqvist, and Kim Halskov. 2020. How task constraints affect inspiration search strategies. *International Journal of Technology and Design Education* 30 (2020), 101–125. https://doi.org/10.1007/s10798-019-09496-7

[8] Paul Brie, Nicolas Burny, and Jean Vanderdonckt. [n.d.]. VisionAPI: An API for Offline and Online Segmentation and Identification of Hand-Sketched Graphical User Interfaces. In *Companion Proceedings of the 15th ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (Swansea, Wales, UK) *(EICS 2023)*.

[9] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. (2020). arXiv:2005.14165 http://arxiv.org/abs/2005.14165

[10] Sara Bunian, Kai Li, Chaima Jemmali, Casper Harteveld, Yun Fu, and Magy Seif Seif El-Nasr. 2021. VINS: Visual Search for Mobile User Interface Design. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 423, 14 pages. https://doi.org/10.1145/3411764.3445762

[11] Gaëlle Calvary, Joëlle Coutaz, David Thevenin, Quentin Limbourg, Laurent Bouillon, and Jean Vanderdonckt. 2003. A Unifying Reference Framework for multi-target user interfaces. *Interact. Comput.* 15, 3 (2003), 289–308. https://doi.org/10.1016/S0953-5438(03)00010-9

[12] Jieshan Chen, Chunyang Chen, Zhenchang Xing, Xin Xia, Liming Zhu, John Grundy, and Jinshui Wang. 2020. Object Detection for Graphical User Interface: Old Fashioned or Deep Learning or a Combination?. In *Proc. of ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Virtual Event, USA) *(ESEC/FSE '20)*. Association for Computing Machinery, New York, NY, USA. https://doi.org//10.1145/3368089.3409691

[13] Jieshan Chen, Chunyang Chen, Zhenchang Xing, Xin Xia, Liming Zhu, John Grundy, and Jinshui Wang. 2020. Wireframe-Based UI Design Search through Image Autoencoder. *ACM Trans. Softw. Eng. Methodol.* 29, 3, Article 19 (June 2020), 31 pages. https://doi.org/10.1145/3391613

[14] Mark Chen, Alec Radford, Rewon Child, Jeff Wu, Heewoo Jun, Prafulla Dhariwal, David Luan, and Ilya Sutskever. 2020. Generative Pretraining from Pixels. In *Proceedings of the 37th International Conference on Machine Learning*.

[15] Mark Chen, Alec Radford, Rewon Child, Jeff Wu, Heewoo Jun, David Luan, and Ilya Sutskever. 2020. Generative Pretraining from Pixels (v1). *Icml* (2020).

[16] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak,

Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating Large Language Models Trained on Code. https://doi.org/10.48550/ARXIV.2107.03374

[17] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

[18] Maria Francesca Costabile, Piero Mussio, Loredana Parasiliti Provenza, and Antonio Piccinno. 2008. End Users as Unwitting Software Developers. In *Proceedings of the 4th International Workshop on End-User Software Engineering* (Leipzig, Germany) *(WEUSE '08)*. Association for Computing Machinery, New York, NY, USA, 6–10. https://doi.org/10.1145/1370847.1370849

[19] Adrien Coyette, Suzanne Kieffer, and Jean Vanderdonckt. 2007. Multi-fidelity Prototyping of User Interfaces. In *Human-Computer Interaction – INTERACT 2007*, Cécilia Baranauskas, Philippe Palanque, Julio Abascal, and Simone Diniz Junqueira Barbosa (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 150–164.

[20] Niraj Ramesh Dayama, Simo Santala, Lukas Brückner, Kashyap Todi, Jingzhou Du, and Antti Oulasvirta. 2021. Interactive Layout Transfer. In *26th International Conference on Intelligent User Interfaces* (College Station, TX, USA) *(IUI '21)*. Association for Computing Machinery, New York, NY, USA, 70–80. https://doi.org/10.1145/3397481.3450652

[21] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. 2017. Rico: A Mobile App Dataset for Building Data-Driven Design Applications. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 845–854. https://doi.org/10.1145/3126594.3126651

[22] Biplab Deka, Zifeng Huang, and Ranjitha Kumar. 2016. ERICA: Interaction Mining Mobile Apps. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16)*. Association for Computing Machinery, New York, NY, USA, 767–776. https://doi.org/10.1145/2984511.2984581

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[24] Jacob Eisenstein, Jean Vanderdonckt, and Angel Puerta. 2001. Applying Model-Based Techniques to the Development of UIs for Mobile Computers. In *Proceedings of the 6th International Conference on Intelligent User Interfaces* (Santa Fe, New Mexico, USA) *(IUI '01)*. Association for Computing Machinery, New York, NY, USA, 69–76. https://doi.org/10.1145/359784.360122

[25] Krzysztof Gajos and Daniel S. Weld. 2004. SUPPLE: Automatically Generating User Interfaces. In *Proceedings of the 9th International Conference on Intelligent User Interfaces* (Funchal, Madeira, Portugal) *(IUI '04)*. Association for Computing Machinery, New York, NY, USA, 93–100. https://doi.org/10.1145/964442.964461

[26] X. Ge. 2019. Android GUI Search Using Hand-Drawn Sketches. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion '19)*. IEEE Computer Society, Los Alamitos, CA, USA, 141–143. https://doi.org/10.1109/ICSE-Companion.2019.00060

[27] Milene Gonçalves, Carlos Cardoso, and Petra Badke-Schaub. 2014. What inspires designers? Preferences on inspirational approaches during idea generation. *Design Studies* 35, 1 (2014), 29–53. https://doi.org/10.1016/j.destud.2013.09.001

[28] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[29] Scarlett R. Herring, Chia-Chen Chang, Jesse Krantzler, and Brian P. Bailey. 2009. Getting Inspired! Understanding How and Why Examples Are Used in Creative Design Practice. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Boston, MA, USA) *(CHI '09)*. Association for Computing Machinery, New York, NY, USA, 87–96. https://doi.org/10.1145/1518701.1518717

[30] Andreas Hinderks, Francisco José Domínguez Mayo, Anna-Lena Meiners, and Jörg Thomaschewski. 2020. Applying Importance-Performance Analysis (IPA) to Interpret the Results of the User Experience Questionnaire (UEQ). *J. Web Eng.* 19, 2 (2020), 243–266. https://doi.org/10.13052/jwe1540-9589.1926

[31] Andreas Hinderks, Martin Schrepp, Francisco José Domínguez Mayo, María José Escalona, and Jörg Thomaschewski. 2019. Developing a UX KPI based on the user experience questionnaire. *Comput. Stand. Interfaces* 65 (2019), 38–44. https://doi.org/10.1016/j.csi.2019.01.007

[32] Andreas Hinderks, Dominique Winter, Martin Schrepp, and Jörg Thomaschewski. 2019. Applicability of User Experience and Usability Questionnaires. *J. Univers. Comput. Sci.* 25, 13 (2019), 1717–1735. http://www.jucs.org/jucs_25_13/applicability_of_user_experience

[33] Forrest Huang, John F. Canny, and Jeffrey Nichols. 2019. Swire: Sketch-Based User Interface Retrieval. In *Proceedings of the ACM Int. Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/3290605.3300334

[34] Forrest Huang, Eldon Schoop, David Ha, and John Canny. 2020. Scones: Towards Conversational Authoring of Sketches. In *Proceedings of the 25th International Conference on Intelligent User Interfaces* (Cagliari, Italy) *(IUI '20)*. Association for Computing Machinery, New York, NY, USA, 313–323. https://doi.org/10.1145/3377325.3377485

[35] Marius Hobbhahn Tamay Besiroglu Anson Ho Jaime Sevilla, Lennart Heim and Pablo Villalobos. 2022. Estimating Training Compute of Deep Learning Models. https://epochai.org/blog/estimating-training-compute Accessed: 2023-04-01.

[36] Naman Jain, Skanda Vaidyanath, Arun Iyer, Nagarajan Natarajan, Suresh Parthasarathy, Sriram Rajamani, and Rahul Sharma. 2022. Jigsaw: Large Language Models Meet Program Synthesis. In *Proceedings of the 44th International Conference on Software Engineering* (Pittsburgh, Pennsylvania) *(ICSE '22)*. Association for Computing Machinery, New York, NY, USA, 1219–1231. https://doi.org/10.1145/3510003.3510203

[37] Ellen Jiang, Kristen Olson, Edwin Toh, Alejandra Molina, Aaron Donsbach, Michael Terry, and Carrie J Cai. 2022. PromptMaker: Prompt-Based Prototyping with Largenbsp;Languagenbsp;Models. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI EA '22)*. Association for Computing Machinery, New York, NY, USA, Article 35, 8 pages. https://doi.org/10.1145/3491101.3503564

[38] Renhe Jiang, Zhengzhao Chen, Zejun Zhang, Yu Pei, Minxue Pan, and Tian Zhang. 2018. Semantics-Based Code Search Using Input/Output Examples. In *Proc. of IEEE 18th International Working Conference on Source Code Analysis and Manipulation (SCAM '18)*. 92–102. https://doi.org/10.1109/SCAM.2018.00018

[39] Marius Kaminskas and Derek Bridge. 2016. Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-Accuracy Objectives in Recommender Systems. *ACM Trans. Interact. Intell. Syst.* 7, 1, Article 2 (dec 2016), 42 pages. https://doi.org/10.1145/2926720

[40] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. *CoRR* abs/1609.04836 (2016). arXiv:1609.04836 http://arxiv.org/abs/1609.04836

[41] Tae Soo Kim, DaEun Choi, Yoonseo Choi, and Juho Kim. 2022. Stylette: Styling the Web with Natural Language. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 5, 17 pages. https://doi.org/10.1145/3491102.3501931

[42] Diederik P. Kingma and Max Welling. 2019. An Introduction to Variational Autoencoders. *Foundations and Trends® in Machine Learning* 12, 4 (2019), 307–392. https://doi.org/10.1561/2200000056

[43] Andrew J. Ko, Thomas D. LaToza, and Margaret M. Burnett. 2015. A practical guide to controlled experiments of software engineering tools with human participants. *Empirical Software Engineering* 20, 1 (01 Feb 2015), 110–141. https://doi.org/10.1007/s10664-013-9279-3

[44] Janin Koch, Magda László, Andrés Lucero, and Antti Oulasvirta. 2018. Surfing for Inspiration: digital inspirational material in design practice. In *Proceedings of the DRS International Conference DRS '18- Design as a catalyst for change* (Limerick, Ireland) *(DRS International Conference Series, Volume 3)*. Design Research Society, London, UK, 1247–1260. https://doi.org/10.21606/drs.2018.3520

[45] Kristian Kolthoff, Christian Bartelt, and Simone Paolo Ponzetto. 2020. GUI2WiRe: Rapid Wireframing with a Mined and Large-Scale GUI Repository using Natural Language Requirements. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering* (Melbourne, Australia) *(ASE 2020)*. IEEE, 1297–1301. https://doi.org/10.1145/3324884.3415289

[46] Kristian Kolthoff, Christian Bartelt, and Simone Paolo Ponzetto. 2023. Data-driven prototyping via natural-language-based GUI retrieval. *Autom. Softw. Eng.* 30, 1 (2023), 13. https://doi.org/10.1007/s10515-023-00377-x

[47] Sari Kujala, Virpi Roto, Kaisa Väänänen-Vainio-Mattila, Evangelos Karapanos, and Arto Sinnelä. 2011. UX Curve: A method for evaluating long-term user experience. *Interacting with computers* 23, 5 (2011), 473–483. https://doi.org/10.1016/j.intcom.2011.06.005

[48] Ranjitha Kumar, Arvind Satyanarayan, Cesar Torres, Maxine Lim, Salman Ahmad, Scott R. Klemmer, and Jerry O. Talton. 2013. Webzeitgeist: Design Mining the Web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 3083–3092. https://doi.org/10.1145/2470654.2466420

[49] Alfirna Rizqi Lahitani, Adhistya Erna Permanasari, and Noor Akhmad Setiawan. 2016. Cosine similarity to determine similarity measure: Study case in online essay assessment. In *2016 4th International Conference on Cyber and IT Service Management*. IEEE, 1–6.

[50] Bettina Laugwitz, Theo Held, and Martin Schrepp. 2008. Construction and Evaluation of a User Experience Questionnaire. In *Proc. of the 4th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the*

Austrian Computer Society, HCI and Usability for Education and Work, USAB 2008 (Graz, Austria) *(Lecture Notes in Computer Science, Vol. 5298)*, Andreas Holzinger (Ed.). Springer, 63–76. https://doi.org/10.1007/978-3-540-89350-9_6

[51] Effie Lai-Chong Law, Paul van Schaik, and Virpi Roto. 2014. Attitudes towards user experience (UX) measurement. *International Journal of Human-Computer Studies* 72, 6 (2014), 526–541. https://doi.org/10.1016/j.ijhcs.2013.09.006

[52] Brian Lee, Savil Srivastava, Ranjitha Kumar, Ronen Brafman, and Scott R. Klemmer. 2010. Designing with Interactive Example Galleries. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Atlanta, Georgia, USA) *(CHI '10)*. Association for Computing Machinery, New York, NY, USA, 2257–2266. https://doi.org/10.1145/1753326.1753667

[53] Chunggi Lee, Sanghoon Kim, Dongyun Han, Hongjun Yang, Young-Woo Park, Bum Chul Kwon, and Sungahn Ko. 2020. GUIComp: A GUI Design Assistant with Real-Time, Multi-Faceted Feedback. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376327

[54] Hsin-Ying Lee, Weilong Yang, Lu Jiang, Madison Le, Irfan Essa, Haifeng Gong, and Ming-Hsuan Yang. 2019. Neural Design Network: Graphic Layout Generation with Constraints. (2019). arXiv:1912.09421 http://arxiv.org/abs/1912.09421

[55] Luis A. Leiva, Asutosh Hota, and Antti Oulasvirta. 2020. Enrico: A Dataset for Topic Modeling of Mobile UI Designs. In *22nd International Conference on Human-Computer Interaction with Mobile Devices and Services* (Oldenburg, Germany) *(MobileHCI '20)*. Association for Computing Machinery, New York, NY, USA, Article 9, 4 pages. https://doi.org/10.1145/3406324.3410710

[56] Luis A Leiva, Asutosh Hota, and Antti Oulasvirta. 2021. Interactive Exploration of Large-Scale UI Datasets with Design Maps. *Interacting with Computers* 32, 5-6 (03 2021), 490–509. https://doi.org/10.1093/iwcomp/iwab006 arXiv:https://academic.oup.com/iwc/article-pdf/32/5-6/490/38856879/iwab006.pdf

[57] Jianan Li, Jimei Yang, Aaron Hertzmann, Jianming Zhang, and Tingfa Xu. 2019. LayoutGAN: Generating Graphic Layouts with Wireframe Discriminators. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=HJxB5sRcFQ

[58] Yi-Chi Liao, Kashyap Todi, Aditya Acharya, Antti Keurulainen, Andrew Howes, and Antti Oulasvirta. 2022. Rediscovering Affordance: A Reinforcement Learning Perspective. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 362, 15 pages. https://doi.org/10.1145/3491102.3501992

[59] Thomas F. Liu, Mark Craft, Jason Situ, Ersin Yumer, Radomir Mech, and Ranjitha Kumar. 2018. Learning Design Semantics for Mobile Apps. In *The 31st Annual ACM Symposium on User Interface Software and Technology* (Berlin, Germany) *(UIST '18)*. ACM, New York, NY, USA, 569–579. https://doi.org/10.1145/3242587.3242650

[60] Simon Lok, Steven Feiner, and Gary Ngai. 2004. Evaluation of Visual Balance for Automated Layout. In *Proceedings of the 9th International Conference on Intelligent User Interfaces* (Funchal, Madeira, Portugal) *(IUI '04)*. Association for Computing Machinery, New York, NY, USA, 101–108. https://doi.org/10.1145/964442.964462

[61] Stephen MacNeil, Andrew Tran, Dan Mogil, Seth Bernstein, Erin Ross, and Ziheng Huang. 2022. Generating Diverse Code Explanations Using the GPT-3 Large Language Model. In *Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 2* (Lugano and Virtual Event, Switzerland) *(ICER '22)*. Association for Computing Machinery, New York, NY, USA, 37–39. https://doi.org/10.1145/3501709.3544280

[62] Katrina Maxwell. 2006. What You Need To Know About Statistics. In *Web Engineering*, Emilia Mendes and Nile Mosley (Eds.). Springer, 365–408. https://doi.org/10.1007/3-540-28218-1_12

[63] Katrina D. Maxwell. 2002. *Applied Statistics for Software Managers*. Prentice Hall, Upper Saddle River, NJ, USA. https://cir.nii.ac.jp/crid/1573668924056253312

[64] Rob McCarney, James Warner, Steve Iliffe, Robbert van Haselen, Mark Griffin, and Peter Fisher. 2007. The Hawthorne Effect: a randomised, controlled trial. *BMC Med Res Methodol* 7 (07 2007), 119–142. Issue 30. https://bmcmedresmethodol.biomedcentral.com/articles/10.1186/1471-2288-7-30

[65] Michael McCurdy, Christopher Connors, Guy Pyrzak, Bob Kanefsky, and Alonso Vera. 2006. Breaking the Fidelity Barrier: An Examination of Our Current Characterization of Prototypes and an Example of a Mixed-Fidelity Success. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (Montréal, Québec, Canada) *(CHI '06)*. Association for Computing Machinery, New York, NY, USA, 1233–1242. https://doi.org/10.1145/1124772.1124959

[66] Paul Michel, Omer Levy, and Graham Neubig. 2019. Are Sixteen Heads Really Better than One? *CoRR* abs/1905.10650 (2019). arXiv:1905.10650 http://arxiv.org/abs/1905.10650

[67] E.G. Miller, N.E. Matsakis, and P.A. Viola. 2000. Learning from one example through shared densities on transforms. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, Vol. 1. 464–471 vol.1. https://doi.org/10.1109/CVPR.2000.855856

[68] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2022. CodeGen: An Open Large Language Model for Code with Multi-Turn Program Synthesis. https://doi.org/10.

48550/ARXIV.2203.13474
[69] Jum C Nunnally. 1975. Psychometric theory—25 years ago and now. *Educational Researcher* 4, 10 (1975), 7–21.
[70] Augustus Odena, Charles Sutton, David Martin Dohan, Ellen Jiang, Henryk Michalewski, Jacob Austin, Maarten Paul Bosma, Maxwell Nye, Michael Terry, and Quoc V. Le. 2021. Program Synthesis with Large Language Models. https://research.google/pubs/pub50670/
[71] Antti Oulasvirta, Samuli De Pascale, Janin Koch, Thomas Langerak, Jussi Jokinen, Kashyap Todi, Markku Laine, Manoj Kristhombuge, Yuxi Zhu, Aliaksei Miniukovich, Gregorio Palmas, and Tino Weinkauf. 2018. Aalto Interface Metrics (AIM): A Service and Codebase for Computational GUI Evaluation. In *The 31st Annual ACM Symposium on User Interface Software and Technology Adjunct Proceedings* (Berlin, Germany) *(UIST '18 Adjunct)*. Association for Computing Machinery, New York, NY, USA, 16–19. https://doi.org/10.1145/3266037.3266087
[72] Antti Oulasvirta, Per Ola Kristensson, Xiaojun Bi, and Andrew Howes. 2018. *Computational Interaction*. Oxford University Press, Oxford, UK. https://doi.org/10.1093/oso/9780198799603.001.0001
[73] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
[74] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR* abs/1802.05365 (2018). arXiv:1802.05365 http://arxiv.org/abs/1802.05365
[75] Stanislas Polu and Ilya Sutskever. 2020. Generative Language Modeling for Automated Theorem Proving. *arXiv preprint arXiv:2009.03393* (2020).
[76] Angel Puerta and Martin Hu. 2009. UI Fin: A Process-Oriented Interface Design Tool. In *Proceedings of the 14th International Conference on Intelligent User Interfaces* (Sanibel Island, Florida, USA) *(IUI '09)*. Association for Computing Machinery, New York, NY, USA, 345–354. https://doi.org/10.1145/1502650.1502698
[77] Steven P. Reiss. 2009. Semantics-based code search. In *Proc. of IEEE 31st International Conference on Software Engineering (ICSE '09)*. 243–253. https://doi.org/10.1109/ICSE.2009.5070525
[78] Nils Rethmeier and Isabelle Augenstein. 2022. A Primer on Contrastive Pretraining in Language Processing: Methods, Lessons Learned amp; Perspectives. *ACM Comput. Surv.* (aug 2022). https://doi.org/10.1145/3561970 Just Accepted.
[79] Daniel Ritchie, Ankita Arvind Kejriwal, and Scott R. Klemmer. 2011. D.Tour: Style-Based Exploration of Design Example Galleries. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA) *(UIST '11)*. Association for Computing Machinery, New York, NY, USA, 165–174. https://doi.org/10.1145/2047196.2047216
[80] Steven I. Ross, Fernando Martinez, Stephanie Houde, Michael Muller, and Justin D. Weisz. 2023. The Programmer's Assistant: Conversational Interaction with a Large Language Model for Software Development. In *Proceedings of the 28th International Conference on Intelligent User Interfaces* (Sydney, NSW, Australia) *(IUI '23)*. Association for Computing Machinery, New York, NY, USA, 491–514. https://doi.org/10.1145/3581641.3584037
[81] Olivier Rukundo. 2023. Effects of Image Size on Deep Learning. *Electronics* 12, 4 (2023). https://doi.org/10.3390/electronics12040985
[82] Ugo Braga Sangiorgi, François Beuvens, and Jean Vanderdonckt. 2012. User Interface Design by Collaborative Sketching. In *Proceedings of the Designing Interactive Systems Conference* (Newcastle Upon Tyne, United Kingdom) *(DIS '12)*. Association for Computing Machinery, New York, NY, USA, 378–387. https://doi.org/10.1145/2317956.2318013
[83] Sami Sarsa, Paul Denny, Arto Hellas, and Juho Leinonen. 2022. Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models. In *Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 1* (Lugano and Virtual Event, Switzerland) *(ICER '22)*. Association for Computing Machinery, New York, NY, USA, 27–43. https://doi.org/10.1145/3501385.3543957
[84] Andrea Schankin, Matthias Budde, Till Riedel, and Michael Beigl. 2022. Psychometric Properties of the User Experience Questionnaire (UEQ). In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 466, 11 pages. https://doi.org/10.1145/3491102.3502098
[85] Martin Schrepp. 2018. *User Experience mit Fragebögen messen*. Amazon CreateSpace. https://www.amazon.de/User-Experience-mit-Fragebögen-messen/dp/1986843769.
[86] Martin Schrepp, Andreas Hinderks, and Jörg Thomaschewski. 2017. Construction of a Benchmark for the User Experience Questionnaire (UEQ). *Int. J. Interact. Multim. Artif. Intell.* 4, 4 (2017), 40–44. https://doi.org/10.9781/ijimai.2017.445

[87] Martin Schrepp and Jörg Thomaschewski. 2019. Design and Validation of a Framework for the Creation of User Experience Questionnaires. *Int. J. Interact. Multim. Artif. Intell.* 5, 7 (2019), 88–95. https://doi.org/10.9781/ijimai.2019.06.006

[88] Dusan Stamenkovic, Alexandros Karatzoglou, Ioannis Arapakis, Xin Xin, and Kleomenis Katevas. 2022. Choosing the Best of Both Worlds: Diverse and Novel Recommendations through Multi-Objective Reinforcement Learning. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining* (Tempe, AZ, USA) *(WSDM '22)*, K. Selcuk Candan, Huan Liu, Leman Akoglu, Xin Luna Dong, and Jiliang Tang (Eds.). ACM, 957–965. https://doi.org/10.1145/3488560.3498471

[89] Jiao Sun, Q. Vera Liao, Michael Muller, Mayank Agarwal, Stephanie Houde, Kartik Talamadupula, and Justin D. Weisz. 2022. Investigating Explainability of Generative AI for Code through Scenario-Based Design. In *27th International Conference on Intelligent User Interfaces* (Helsinki, Finland) *(IUI '22)*. Association for Computing Machinery, New York, NY, USA, 212–228. https://doi.org/10.1145/3490099.3511119

[90] Ying Sun. 2021. *Exploring Inspirational Sources of Selection and Transformation - Industrial Designers' Self-Perception of Idea Generation.* Ph.D. Dissertation. Technische Universität Dresden, Dresden. https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa2-741274

[91] Amanda Swearngin, Mira Dontcheva, Wilmot Li, Joel Brandt, Morgan Dixon, and Andrew J. Ko. 2018. *Rewire: Interface Design Assistance from Examples.* Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3174078

[92] Amanda Swearngin, Chenglong Wang, Alannah Oleson, James Fogarty, and Amy J. Ko. 2020. Scout: Rapid Exploration of Interface Layout Alternatives through High-Level Design Constraints. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376593

[93] Yun Yi Tan and Allan H. K. Yuen. 2015. "Destuckification": Use of Social Media for Enhancing Design Practices. In *New Media, Knowledge Practices and Multiliteracies*, Will W.K. Ma, Allan H.K. Yuen, Jae Park, Wilfred W.F. Lau, and Liping Deng (Eds.). Springer Singapore, Singapore, 67–75.

[94] Kashyap Todi, Gilles Bailly, Luis Leiva, and Antti Oulasvirta. 2021. Adapting User Interfaces with Model-Based Reinforcement Learning. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 573, 13 pages. https://doi.org/10.1145/3411764.3445497

[95] Kashyap Todi, Daryl Weir, and Antti Oulasvirta. 2016. Sketchplore: Sketch and Explore with a Layout Optimiser. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems* (Brisbane, QLD, Australia) *(DIS '16)*. Association for Computing Machinery, New York, NY, USA, 543–555. https://doi.org/10.1145/2901790.2901817

[96] National Institute of Standards U.S. Commerce Department and Statistical Engineering Division Technology. 2017. *Cosine Distance, Cosine Similarity, Angular Cosine Distance, Angular Cosine Similarity.* https://www.itl.nist.gov/div898/software/dataplot/refman2/auxillar/cosdist.htmm

[97] Priyan Vaithilingam, Tianyi Zhang, and Elena L. Glassman. 2022. Expectation vs.nbsp;Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI EA '22)*. Association for Computing Machinery, New York, NY, USA, Article 332, 7 pages. https://doi.org/10.1145/3491101.3519665

[98] Flore Vancompernolle Vromman and François Fouss. 2021. Filter Bubbles Created by Collaborative Filtering Algorithms Themselves, Fact or Fiction? An Experimental Comparison. In *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology* (Melbourne, VIC, Australia) *(WI-IAT '21)*. Association for Computing Machinery, New York, NY, USA, 153–159. https://doi.org/10.1145/3498851.3498945

[99] Jean Vanderdonckt, Missiri Ouedraogo, and Banta Ygueitengar. 1994. A Comparison of Placement Strategies for Effective Visual Design. In *Proceedings of BCS Int. Conf. on Human-Computer Interaction, People and Computers IX* (Glasgow, UK) *(HCI '94)*, Gilbert Cockton, Stephen W. Draper, and George R. S. Weir (Eds.). Cambridge University Press, 125–143.

[100] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Proceeding of Annual Conference on Neural Information Processing Systems - Advances in Neural Information Processing Systems* (Long Beach, CA, USA), Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008. http://papers.nips.cc/paper/7181-attention-is-all-you-need

[101] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP.* Association for Computational Linguistics, Brussels, Belgium, 353–355. https://doi.org/10.18653/v1/W18-5446

[102] Justin D. Weisz, Michael Muller, Jessica He, and Stephanie Houde. 2023. Toward General Design Principles for Generative AI Applications. https://doi.org/10.48550/ARXIV.2301.05578 Submitted to the 4th Int. Workshop on Human-AI Co-Creation with Generative Models HAI-GEN '23.

[103] Claes Wohlin, Martin Höst, and Kennet Henningsson. 2003. Empirical Research Methods in Software Engineering. In *Empirical Methods and Studies in Software Engineering*. Springer Berlin Heidelberg, 7–23. https://doi.org/10.1007/978-3-540-45143-3_2

[104] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online, 38–45. https://doi.org/10.18653/v1/2020.emnlp-demos.6

[105] Ziming Wu, Qianyao Xu, Yiding Liu, Zhenhui Peng, Yingqing Xu, and Xiaojuan Ma. 2021. Exploring Designers' Practice of Online Example Management for Supporting Mobile UI Design. In *Proceedings of the 23rd International Conference on Mobile Human-Computer Interaction* (Toulouse amp; Virtual, France) *(MobileHCI '21)*. Association for Computing Machinery, New York, NY, USA, Article 44, 12 pages. https://doi.org/10.1145/3447526.3472048

[106] Yongqin Xian, Christoph H. Lampert, Bernt Schiele, and Zeynep Akata. 2019. Zero-Shot Learning - A Comprehensive Evaluation of the Good, the Bad and the Ugly. *IEEE Trans. Pattern Anal. Mach. Intell.* 41, 9 (2019), 2251–2265. https://doi.org/10.1109/TPAMI.2018.2857768

[107] Tom Yeh, Tsung-Hsiang Chang, and Robert C. Miller. 2009. Sikuli: Using GUI Screenshots for Search and Automation. In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology* (Victoria, BC, Canada) *(UIST '09)*. Association for Computing Machinery, New York, NY, USA, 183–192. https://doi.org/10.1145/1622176.1622213

[108] Mathieu Zen and Jean Vanderdonckt. 2014. Towards an evaluation of graphical user interfaces aesthetics based on metrics. In *Proc. of IEEE 8th International Conference on Research Challenges in Information Science* (Marrakech, Morocco) *(RCIS 2014)*, Marko Bajec, Martine Collard, and Rébecca Deneckère (Eds.). IEEE, 1–12. https://doi.org/10.1109/RCIS.2014.6861050

A RESULTS OF THE STATISTICAL TESTS

Instruction: Build a ...	Cosine similarity		Pearson correlation			Wilcoxon test	
	S_C	Similarity	ρ	p-value	Degree	z-score	p-value
page with contents	0.98	Very high	0.61	$\leq.001^{***}$	High	0.22	0.82 (*n.s.*)
section of type hero	0.93	Very high	0.42	$\leq.001^{***}$	Moderate	0.092	0.92 (*n.s.*)
section of type navbar	0.98	Very high	0.50	$\leq.001^{***}$	High	0.24	0.80 (*n.s.*)
section of type footer	0.92	Very high	0.19	0.0014^{**}	Low	0.56	0.57 (*n.s.*)
section of type call-to-action	0.96	Very high	0.41	$\leq.001^{***}$	Moderate	0.050	0.99 (*n.s.*)
section of type form	0.93	Very high	0.33	$\leq.001^{***}$	Moderate	0.21	0.83 (*n.s.*)
section of type cards	0.98	Very high	0.64	$\leq.001^{***}$	High	0.29	0.76 (*n.s.*)
section with 3 images and 3 paragraphs	0.91	Very high	0.23	$\leq.001^{***}$	Low	0.13	0.89 (*n.s.*)
section with 2 images, 1 input, 4 links, 4 buttons	0.90	Very high	0.15	0.0013^{**}	Low	0.091	0.93 (*n.s.*)
section with 4 links, 1 image, 1 input, 1 button	0.93	Very high	0.27	$\leq.001^{***}$	Low	0.021	0.98 (*n.s.*)
section with 1 heading, 1 paragraph, 1 video	0.96	Very high	0.51	$\leq.001^{***}$	High	1.44	0.15 (*n.s.*)
section with 1 heading, 2 paragraphs, 2 images	0.89	Very high	0.14	0.023^{*}	Low	0.45	0.65 (*n.s.*)
section with 2 inputs, 1 button, 2 paragraphs	0.88	Very high	0.063	0.29 (*n.s.*)		0.022	0.98 (*n.s.*)
section with 5 images, 5 paragraphs, 5 buttons	0.97	Very high	0.68	$\leq.001^{***}$	High	0.21	0.84 (*n.s.*)
contents with 2 paragraphs and 1 heading	0.92	Very high	0.28	$\leq.001^{***}$	Low	0.037	0.97 (*n.s.*)
section hero with a heading, 2 images, a paragraph, and 3 links	0.93	Very high	0.30	$\leq.001^{***}$	Moderate	0.11	0.91 (*n.s.*)
navbar with an image, 1 button, 4 links	0.98	Very high	0.60	$\leq.001^{***}$	High	0.25	0.80 (*n.s.*)
section of type cta with 1 image, 1 paragraph, 1 heading, 1 link	0.93	Very high	0.22	$\leq.001^{***}$	Low	0.046	0.96 (*n.s.*)
form with 2 inputs, 2 buttons	0.98	Very high	0.55	$\leq.001^{***}$	High	0.21	0.84 (*n.s.*)
cards with 3 images, 3 paragraphs, 3 buttons	0.91	Very high	0.30	$\leq.001^{***}$	Moderate	0.038	0.97 (*n.s.*)
footer with 1 image, 8 links, 2 images	0.92	Very high	0.29	$\leq.001^{***}$	Low	0.081	0.94 (*n.s.*)
All instructions	0.60	High	0.36	$\leq.001^{***}$	Moderate	4.64	0.94 (*n.s.*)

Table 2. Correlation between participants' rankings and INSTIGATOR rankings (α=0.05 for all tests).

Scale	Scale Mean Score				Scale Mean Importance		
	M (Int., B.)	*SD*	*Mdn*	α (Int.)	*M* (Int.)	*SD*	*Mdn*
Attractiveness	1.45 (P, E)	1.37	2	0.90 (E)	1.85	1.09	6
Efficiency	1.67 (P, E)	1.54	2	0.88 (G)	1.82	1.00	6
Perspicuity	2.06 (P, E)	1.10	2	0.93 (E)	2.00	1.00	6
Dependability	1.38 (P, A)	1.20	2	0.74 (A)	1.85	0.73	6
Stimulation	1.24 (P, A)	1.39	1	0.83 (G)	1.38	1.14	6
Novelty	1.83 (P, G)	1.33	2	0.93 (E)	1.62	1.24	6
Adaptability	0.76 (M)	1.39	1	0.93 (E)	1.50	1.22	6
Usefulness	1.67 (P)	0.99	2	0.91 (E)	2.24	0.92	6.5
Value	1.34 (P)	1.22	1	0.90 (E)	1.71	1.02	6
Visual Aesthetics	0.77 (M)	1.29	1	0.95 (E)	1.12	1.28	5
Intuitive Use	1.88 (P)	1.10	2	0.89 (G)	2.03	1.04	6
Trustworthiness of Content	1.68 (P)	1.09	2	0.90 (E)	1.85	0.88	6
Quality of Content	1.49 (P)	1.09	2	0.90 (E)	1.88	0.83	6

Table 3. Scale Mean Scores and Scale Mean Importances evaluated on Instigator. Legend: *M* (Int., B.)= mean with its interpretation based on [32, 87] (>1=Positive, $\in [-1, ..., +1]$=Moderate, < -1= Negative) and its benchmarking based on [86] (Excellent=among the best 10% of all cases), Good=10% of the cases are better than Instigator, Above average=25% of the cases are better than Instigator), Below average=50% of the cases are better than Instigator, and Bad=Instigator is among the worst 25% of cases). *SD*=standard deviation. *Mdn*=median. α (Int.): Cronbach's coefficient α with its interpretation based on [69] ($\geq$0.9=Excellent, 0.9>$\alpha\geq$0.8=Good, 0.8>$\alpha\geq$0.7=Acceptable, 0.7>$\alpha \geq$ 0.6=Questionable, 0.6>$\alpha \geq$ 0.5=Poor, 0.5>α=Unacceptable).

Received October 2022; revised February 2023; accepted April 2023

Group 1	Group 2	R Mean	q-stat	p-value	Cohen's d	Interp.
Attractiveness	Perspicuity	256.75	5.86	0.0024^{*}	0.49	Small
"	Adaptability	268.85	6.14	0.0010^{**}	0.50	Medium
"	Visual Aesthetics	288.73	6.59	0.00024^{***}	0.51	Medium
Efficiency	Adaptability	345.22	7.88	$\leq 0.001^{***}$	0.69	Medium
"	Visual Aesthetics	365.10	8.34	$\leq 0.001^{***}$	0.71	Medium
Perspicuity	Dependability	311.77	7.12	$\leq 0.001^{***}$	0.59	Medium
"	Stimulation	344.17	7.86	$\leq 0.001^{***}$	0.65	Medium
"	Adaptability	525.61	12.01	$\leq 0.001^{***}$	1.03	Large
"	Value	332.50	7.59	$\leq 0.001^{***}$	0.62	Medium
"	Visual Aesthetics	545.49	12.46	$\leq 0.001^{***}$	1.07	Large
"	Quality of Content	289.83	6.62	0.00022^{***}	0.52	Medium
Dependability	Adaptability	213.83	4.88	0.032^{*}	0.47	Small
"	Visual Aesthetics	233.71	5.33	0.0104^{*}	0.49	Small
"	Intuitive Use	208.66	4.76	0.042^{*}	0.43	Small
Stimulation	Novelty	246.56	5.63	0.0047^{**}	0.43	Small
"	Intuitive Use	241.06	5.50	0.0066^{**}	0.51	Medium
Novelty	Adaptability	428.00	9.77	$\leq 0.001^{***}$	0.78	Medium
"	Value	234.88	5.36	0.0097^{**}	0.38	Small
"	Visual Aesthetics	447.88	10.23	$\leq 0.001^{***}$	0.81	Large
Adaptability	Usefulness	309.05	7.05	$\leq 0.001^{***}$	0.75	Medium
"	Intuitive Use	422.49	9.65	$\leq 0.001^{***}$	0.89	Large
"	Trustworthiness of Content	329.64	7.53	$\leq 0.001^{***}$	0.73	Medium
"	Quality of Content	235.77	5.38	0.0092^{**}	0.58	Medium
Value	Visual Aesthetics	212.99	4.86	0.033^{*}	0.34	Small
"	Intuitive Use	229.38	5.23	0.013^{*}	0.46	Small
Visual Aesthetics	Intuitive Use	442.38	10.10	$\leq 0.001^{***}$	0.92	Large
"	Trustworthiness of Content	349.52	7.98	$\leq 0.001^{***}$	0.76	Medium
"	Quality of Content	255.66	5.84	0.0025^{**}	0.60	Medium

Table 4. Nemenyi pair-wise test: significant differences between scale mean scores (R_{crit}=205.095, SD=43.77).