

TOWARDS EVALUATION OF GRAPHICAL USER INTERFACES BASED ON A REPRODUCIBLE QUALITY MODEL: APPLICATION TO VISUAL DESIGN AESTHETICS

Nicolas Burny

*Thesis submitted in partial fulfillment of the requirements for
the Degree of Doctor in Computer Science*

March 2022

Louvain Research Institute in Management Sciences
Louvain School of Engineering
Université catholique de Louvain
Louvain-la-Neuve
Belgium

Thesis Committee:

Prof. Sophie **Dupuy-Chessa** (Examiner)
Prof. Peter **Van Roy** (President of the Jury)
Prof. Ahmed **Seffah** (Examiner)
Prof. Jean **Vanderdonckt** (Advisor)
Dr. Mathieu **Zen** (Reader)

Université Grenoble Alpes, France
UCLouvain, ICTeam/INGI, Belgium
Concordia University, Canada
UCLouvain, LouRIM & ICTeam, Belgium
UCLouvain, LS/SGSI/SIPS & SSH/ESPO/COMU, Belgium

Towards Evaluation of Graphical User Interfaces based on a Reproducible Quality Model: Application to Visual Design Aesthetics

by Nicolas Burny

© Nicolas Burny 2021

LouRIM

Université catholique de Louvain

Place des Doyens, 1

1348 Louvain-la-Neuve

Belgium

Knowledge grows when you share it.
SOCRATE

Abstract

User interfaces (UI) are one of the main media of interaction with the devices we use on a daily basis, ranging from smartphones to smart TVs and cars. Evaluation of UIs is and will remain a topical research domain as long as existing evaluation methods are refined, improved and new methods appear to evaluate new types of UIs such as virtual reality or large user interfaces. However, not all the existing evaluation methods include explicit, formal, calculable and interpretable quality models that are a necessary condition to ensure their replicability.

In addition to the large number of evaluation methods, an important number of experimental studies investigated, evaluated, and reported the impact of various aspects of Graphical User Interfaces (GUI) on other dimensions of user experience (UX). These experimental studies make heavy use of large datasets to infer relationships between the considered elements. However, the construction of datasets requires time- and resource-consuming tasks implying human intervention, making it a hardly scalable and error-prone process.

The large variety of evaluation methods and studies coupled to other various factors led to a replicability crisis which consequences are manifold and hinder the scientific progress in the field of Human-Computer Interactions.

In order to address the aforementioned challenges and concerns, there is a need to provide the field of experimental studies on GUI visual design with methodologies and tools that make the process of dataset creation process efficient and realistic to foster the replicability of experimental studies in the research field.

The purpose of this thesis is to provide a way of evaluating the quality of GUIs in an explicit, formal, calculable and thus reproducible way. This thesis introduces, motivates, defines, implements and validates a methodology for producing explicit, formal, calculable and interpretable quality models for evaluation of GUI visual design. This methodology is based on a conceptual model and is supported by a software tool¹ implementing this model to automate to various degrees the different processes implied in the dataset construction process. To validate the methodology, this thesis proposes a novel approach based on factor analysis to establish interpretable quality models for GUI visual design. This approach is supported by a dataset collected with the help of the created tool, validating in this way the whole methodology.

¹<https://github.com/uilab-app>

Acknowledgments

I would like to express my deepest gratitude to my advisor, Prof. Jean Vanderdonckt, who trusted more than 4 years ago when I applied to the Research & Teaching assistant position at the Louvain School of Management. Throughout our discussions during these 4 years, he provided me all the guidance, advices and motivation I needed when things went well but also during hard times, such as the COVID-19 crisis. He played a important role in the success of my PhD thesis and sending him an e-mail on December 2nd, 2016 was probably one of the best decisions I ever took.

I would also like to thank the other members of my thesis committee : Prof. Ahmed Seffah, Prof. Sophie Dupuy-Chessa and Dr. Mathieu Zen, for their support and the confidence they expressed by accepting to be examiners of my PhD thesis. I also express my gratitude to Prof. Peter Van Roy who accepted to assume the Presidency of the Jury for my PhD thesis.

I am grateful to my colleagues of the research & teaching staff, with a special mention to Marco, Sylvain, Pierre, Ghazaleh, Mehdi and Flore. I would also thank my colleagues from the LouRIM administrative staff : Sylvie, Heike, Maud and Barbara, to name a few. I really enjoyed our discussions, all these moments spent together and by your every day work, support and commitment, you contribute to make the LouRIM institute one of the best workplaces I know.

I would like to thank my parents, Martine and Joël, and my sister Pauline for their unconditional support throughout my education and career. I am also grateful to my friends for their support, especially to my closest friends Pierre-Yves, Thibault, Arpad, Quentin and Louis, some of them being part of my life for more than 20 years.

Finally, I would like to thank the most important person to me, Elisabeth, with whom I've spent every moment for the last 11 years. I thank her for her infinite and unconditional love, kindness and support all these years and I can't wait to see what the future holds for us.

List of Figures

1.1	Senach’s model for user interface evaluation [138].	1
1.2	Whitefield’s framework of UEMs [163].	2
1.3	Rohrer’s classification of UEMs [130].	2
1.4	Experimental workflow illustrated on an experiment for as- sessing the GUI visual design of shopping web sites [27]. . . .	6
1.5	Scope of the research question.	10
2.1	Density computation of an alphanumeric UI by DAP [152]. . .	12
2.2	A boxing analysis (left) and a hot-spot analysis (right) of an alphanumeric display [145].	12
2.3	The three first categories of visual techniques: physical, com- position, and association/dissociation.	15
2.4	The two last categories of visual techniques: ordering and photographic techniques.	16
2.5	The four-phase flow diagram of our SLR examination.	23
2.6	Layout GUIs generated by DON (right) and their automatic metric computation (left) [70].	26
2.7	An optimal GUI layout computed by AIDE when only align- ment is considered (left) with the AIDE Control Panel display- ing values and interpretations for all measures [136].	26
2.8	Screenshot of the SHERLOCK interface.	27
2.9	Screenshots of the GUILayout++ interface.	28
2.10	Screenshot of the BGLayout interface.	29
2.11	QUESTIM: introductory screen	30
2.12	QUESTIM: evaluation screen.	30
2.13	LabInTheWild - Home page with hosted experiments	31
2.14	LabInTheWild - example of experiment	32
2.15	Aalto Interface Metrics (AIM): Measure selection	33
2.16	Aalto Interface Metrics (AIM): Summary of computed measures. .	34
2.17	Aalto Interface Metrics (AIM): Computation details	34
2.18	Influencing factors and sub-factors of User Experience	42
3.1	The ISO 25010 standard of software quality	43
3.2	Quality Model for Interface Trustworthiness	45
3.3	Olsina’s Ontology for Quality Views [110].	46

3.4	Example of SQuaRE-based quality model	48
3.5	Our framework for experiment replicability.	50
4.1	UILAB- Overview of architecture.	54
4.2	UILAB- Extended Entity-Relationship (EER) conceptual model.	56
4.3	UILAB- EER model of galleries and related concepts	57
4.4	UILAB- EER model of experiments and related concepts.	61
4.5	UILAB- Table for workflows and related concepts.	64
4.6	UILAB: Docker image for screenshot capture	65
4.7	UILAB: Javascript script for screenshot capture	66
4.8	UILAB: (a) Gallery creation screen, (b) Galleries overview screen.	67
4.9	UILAB: (a) Screenshots form screen, (b) Screenshots overview screen.	67
4.10	UILAB: Gallery creation and screenshot capture - sequence diagram	68
4.11	UILAB: Experiment creation process.	69
4.12	UILAB- Example of experiment flow.	71
4.13	UILAB- Experiment participation process.	72
4.14	UILAB: Measure creation process.	73
4.15	UILAB: Examples of measures: Luminance Std. dev (top-left); PNG to JPG (middle-right); PNG file size (top-right); Number of colors (bottom)	73
4.16	UILAB computations: Execution flow.	75
4.17	UILAB- Docker image for measure computation	76
4.18	UILAB- main shell script for measure computation	76
4.19	UILAB- main python script for measure computation	77
4.20	UILAB- Measures computation - sequence diagram	78
5.1	Screenshot measures - statistical distributions of variables	81
5.2	Density (top) and color hue (bottom) distributions for variables <i>color_harmony_V_alpha</i> , <i>color_harmony_Linverse_alpha</i> , <i>color_harmony_L_alpha</i> or <i>color_harmony_T_alpha</i> respectively (left to right)	82
5.3	Screenshot measures - correlation matrix.	83
5.4	Screenshot measures - Examples of skewed variable distributions due to statistical outliers	84
5.5	Scree plot of Principal Component Analysis	84
5.6	Comparison of the different factorial analysis rotations with 10 factors	88
5.7	Comparison of the different factorial analysis rotations with 7 factors	89
5.8	Distribution of factors across all categories	90

5.9	Distribution of factors for the category “Business and consumer services”	91
A.1	The four-phase flow PRISMA diagram of our SLR examination [161]	110
B.1	Example of axes of symmetry [162]	117
B.2	Comparison between a good and a bad sequence [104]	121
B.3	The coordinate system for the HSV color space [73]	134
C.1	Distribution of factors for the category “Business and consumer services”	143
C.2	Distribution of factors for the category “Arts & entertainment”	144
C.3	Distribution of factors for the category “Community & Society”	145
C.4	Distribution of factors for the category “Computers Electronics & Technology”	146
C.5	Distribution of factors for the category “E-commerce & Shopping”	147
C.6	Distribution of factors for the category “News & Media”	148
C.7	Distribution of factors for the category “Finance”	149
C.8	Distribution of factors for the category “Food & Drinks”	150
C.9	Distribution of factors for the category “Lifestyle”	151
C.10	Distribution of factors for the category “Games”	152
C.11	Distribution of factors for the category “Health”	153
C.12	Distribution of factors for the category “Science & Education”	154
C.13	Distribution of factors for the category “Heavy industry & Engineering”	155
C.14	Distribution of factors for the category “Hobbies & Leisures”	156
C.15	Distribution of factors for the category “Home & Garden”	157
C.16	Distribution of factors for the category “Jobs & Career”	158
C.17	Distribution of factors for the category “Law & Government”	159
C.18	Distribution of factors for the category “Pets & Animals”	160
C.19	Distribution of factors for the category “Reference Materials”	161
C.20	Distribution of factors for the category “Sports”	162
C.21	Distribution of factors for the category “Travel & Tourism”	163
C.22	Distribution of factors for the category “Vehicles”	164

List of Tables

2.1	GUI Visual Design: Comparison of Experimental Studies . . .	22
2.2	Summary table for Quality Measures.	25
2.3	Level of support of visual variables expressed according to Harvey's Balls.	37
2.4	Implementation of measures by tools supporting GUI visual design evaluation	41
5.1	Factor loadings for factor analysis with oblimin rotation - 7 extracted factors	87
5.2	Predictive performance (R^2) for quality models	97
5.3	Reproduced experimental studies including uILAB	97
A.1	Syntax and advanced search option used for each selected dig- ital library (adapted from [112])	108
A.2	Summary table of the number of references analyzed at the different steps of the SLR (adapted from [161])	112

Contents

Abstract	i
Acknowledgments	iii
Table of Contents	ix
1 Introduction	1
1.1 Context of the Research Question	1
1.2 Challenges and Concerns in GUI Visual Design Experimentation	5
1.3 Research Question and Working Hypotheses	8
1.4 Reading Map	9
2 State of the Art	11
2.1 GUI Visual Design Experiments	11
2.1.1 GUI Aesthetic Measures	11
2.1.2 Derivative Experiments	19
2.2 Quality Measures in GUI Visual Design	21
2.3 Software for Experimental Study and Evaluation in GUI Vi- sual Design	24
2.3.1 DON	24
2.3.2 AIDE	25
2.3.3 SHERLOCK	27
2.3.4 SEGUIA	27
2.3.5 GUILayout++	27
2.3.6 BaLOReS	28
2.3.7 QUESTIM	28
2.3.8 LabInTheWild	29
2.3.9 GUIEvaluator	31
2.3.10 GUIExaminer	31
2.3.11 UI-CAT	32
2.3.12 PLAIN	32
2.3.13 Aalto Interface Metrics (AIM)	33
2.3.14 Web UI Visual Analyzer (WUI)	35
2.3.15 Comparative Analysis of Software	35
2.4 Reproducibility in (Computer) Science, HCI, and GUI Evaluation	38

2.5	Summary	40
3	Quality Models for GUIs	43
3.1	Related Work	43
3.1.1	The ISO 25010 Standard	43
3.1.2	Quality Models and Ontologies	45
3.2	Explicit, Formal, and Calculable Definition of a Quality Model	47
3.2.1	Application to SUS Evaluation Method	47
3.2.2	Application to SQuaRE-based models	48
3.2.3	Application to machine-learning based quality model .	49
3.3	Our Framework for Experiment Replicability	49
4	Software Support for Explicit Quality Models	53
4.1	Introduction	53
4.2	High-level Overview	53
4.3	Conceptual Model	55
4.4	System Walkthrough	64
4.4.1	Construction of Galleries: Automation of the Screen- shots Capture Process	64
4.4.2	Design and Deployment of Experiments	65
4.4.3	Participation to Deployed Experiments	68
4.4.4	Incremental Development of New Measures	70
4.4.5	Computation of Workflows on Galleries	74
5	Experimental Studies	79
5.1	Methodology	79
5.1.1	Experimental Setup	80
5.1.2	Data Analysis	80
5.2	Results	85
5.2.1	Identification of latent factors	85
5.2.2	Visual profiles	87
5.2.3	Quality models of aesthetics and complexity	96
5.2.4	Reproducibility of experimental studies	97
6	Conclusion	99
6.1	Overview of Contributions	99
6.2	Strengths and Limitations	100
6.3	Future Work	102
6.3.1	Relaxing the GUI representation (H_1)	102
6.3.2	Relaxing the represented and real users (H_2)	103
6.3.3	Relaxing the visual design (H_3)	104
6.3.4	Relaxing the preference option (H_4)	104

A	Systematic Literature Review on Visual Design Measures	107
A.1	Identification Phase	107
A.2	Screening Phase	109
A.3	Eligibility Phase	109
A.4	Backward Snowballing	109
A.5	Inclusion Phase	111
A.6	Summary of the Systematic Literature Review	111
B	Visual Design Metrics	113
B.1	Balance	113
B.2	Equilibrium	116
B.3	Symmetry	117
B.4	Sequence	121
B.5	Cohesion	123
B.6	Unity	124
B.7	Proportion	125
B.8	Simplicity	126
B.9	Density	127
B.10	Regularity	128
B.11	Economy	131
B.12	Homogeneity	131
B.13	Rhythm	132
B.14	White-space fraction	132
B.15	Composition	133
B.16	Color contrast	134
B.17	Color complexity	135
B.18	Typography complexity	136
B.19	Essential efficiency	136
B.20	Size complexity	137
B.21	Visual coherence	138
B.22	Element smallness	138
B.23	Edge congestion	139
B.24	Concentricity	139
B.25	Order and complexity	139
B.26	Inconsistency	141
C	Visual profiles distributions	143

Introduction

1

1.1 Context of the Research Question

The evaluation of User Interfaces (UI) will remain a topical subject in research and development as long as existing methods are refined, adjusted and new methods appear for new UI types, such as virtual or augmented reality. The UI evaluation constitutes a problem all the more crucial since it is confronted with an abundance of Evaluation Methods (EM), more specifically Usability Evaluation Methods (UEM) and User Experience Evaluation Methods (UXEM) [160]. Indeed, the experimental conditions and protocols of these EM vary widely such as mockup-based evaluation [86], model-based evaluation [92, 21], measure-based evaluation [170], and formal analysis [109]. Senach [138] characterizes the purpose of a UEM by comparing the UI under evaluation with a reference model, which can be implicit (*e.g.*, when the respect of usability guidelines assumes that usability is covered) or explicit (*e.g.*, when a user model is involved).

Each EM manipulates its own experimental conditions and produces its own results which are of different kinds: qualitative and/or quantitative, attitudinal and/or behavioral, etc. For instance, Whitefield *et al.* [163] provided a framework for classifying EMs according to two axes: UI (real vs. represented) and end-users (real vs. represented). An ideal EM would be able to evaluate a real UI with real end-users, but this ideal situation is not always possible nor is it always cost effective. Therefore we suggest replacing the real UI by a representation (*e.g.*, a prototype instead of the real UI) and the real end-users by some representation (*e.g.*, a model or substitutes of end-users).

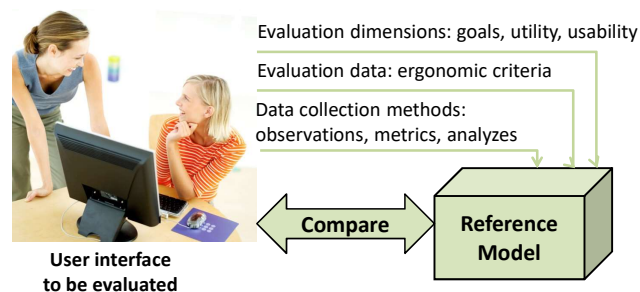


Figure 1.1: Senach's model for user interface evaluation [138].

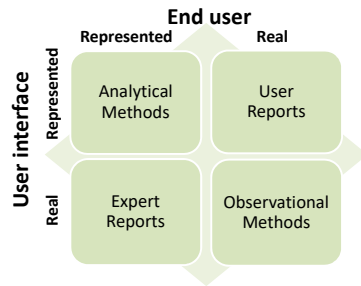


Figure 1.2: Whitefield's framework of UEMs [163].

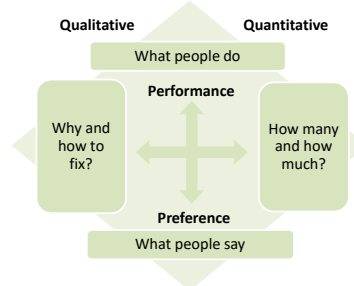


Figure 1.3: Rohrer's classification of UEMs [130].

In this way, Whitefield *et al.* break down UEMs into four categories (Fig. 1.2): (1) *analytical methods*: represent both the end-user and the UI to estimate or predict how the end-user will behave with the UI, usually through models and model-based techniques. (2) *expert reports*: represent the end-user by an expert, *e.g.*, another stakeholder, a designer, an experimenter to evaluate the real UI, usually with some reference such as a usability guide, a standard, or a style guide. (3) *user reports*: represent the UI by a substitute, such as a prototype, a mock-up, or a sketch to be evaluated by real end users, usually through questionnaires. (4) *observational methods*: evaluate how real end users interact with the real UI, usually through user testing, experiments, and records of real usage (*e.g.*, log files).

Rohrer's [130] UEMs classification (Fig. 1.3) distinguishes between UEMs producing qualitative measures capturing attitudinal characteristics of the UI and UEMs producing quantitative measures expressing behavioral characteristics of the UI. This distinction is important since attitudinal measures report end users' preferences when interacting with the UI while behavioral measures report what end users really do when interacting. End users may report a preference that is radically different from what they actually do: this expresses the eternal divergence between the end users' preferences (what they say) and their performance (what they actually do) [107].

While they were developed primarily for UEMS, these classifications still apply to UXEMs as the relationship between usability and UX is intertwined. The ISO-9241-210 standard [114] defines the usability as *the extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use*. User Experience is defined by the same standard as *the user's perceptions and responses that result from the use and/or anticipated use of a system, product or service* and is generally understood as inherently dynamic, given the ever-changing internal and emotional state of a person and differences in the circumstances during and after an interaction with a product [160].

Whether an EM evaluates an end-user's preferences and/or performance with/out users, with/out UIs, its results are often assessed in terms of validity. Each experiment is subject to validity threats, which could be structured according to Wohlin's classification [166] into four types:

- *Conclusion validity*: this type of threat deals with the ability to draw the correct conclusion about relations between a treatment and an outcome. An experiment may suffer from *fishing*, which means that the experimenters are looking for a specific result, from *reliability of measures*, which means that the measures used in the statistical analysis might include mistakes, or *reliability of treatment implementation*, meaning that the treatment might be operationalized wrongly. Another threat is *random heterogeneity of participants*, which means that heterogeneity in a study group may cause bigger variations due to individual differences of participants than due to the treatment.
- *Internal validity*: this type of threat deals with influences that can affect the factor concerning causality. The experiment may suffer from *experience of the participants*, which means that the experience of the participants is not developed enough or too developed to conduct the experiment, from *history*, which means that differences may arise when treatments are applied at different times, *instrumentation*, which means that the artifacts used in the experiment could affect the results. Other threats include *interactions with selection*, which means that when there are different groups of participants, different behaviors with the UI might appear, *resentful demoralization*, which means that some treatments can be more motivating than others. *Ecological validity* [68] means that the context of use [28] in which the experiment was conducted may affect the results.
- *Construct validity*: this type of threat concerns generalizing the result of an experiment to the concept or theory behind the experiment. The experiment might suffer the following threats of this type: *evaluation apprehension*, which means that some people are afraid of being evaluated, *hypothesis guessing*, which means that when people take part in an experiment, they might try to figure out what the purpose and intended result of the experiment is.
- *External validity*: this type of threat concerns conditions that limit our ability to generalize the results of our experiments to industrial practice. The experiment may suffer the following threats of this type: *Interaction of selection and treatment*, which means the effect of having a participant population not representative of the population we want to generalize. In our experiment, we cannot ensure from our sample that

results can be generalized to participants with different profiles. Another threat that appears is *Interaction of settings and treatment*, i.e., the effect of not having the experimental setting or material representative of industrial practice.

Some EMs are proved to be externally valid based on empirical validation [64] while some others are not. For example, SUS [25], a 10-item questionnaire with answers expressed as 5-point Likert [83] scales, has been empirically validated [81] and its results are made explicit [13, 133]: the SUS score falls into five categories, i.e., A, B, C, D, and E, depending on its value (e.g., a SUS score of 77.9 results in a rating of B+=“good”). Similarly, the IBM Computer Satisfaction Usability Questionnaire (CSUQ) [78] and its successor, the IBM Post-Study System Usability Questionnaire (PSSUQ) [79], have been empirically validated on a large number of UIs, which gives them a correlation of $\rho \geq 0.89$ with usability of the UI under evaluation [80].

In order to distinguish EMs, we introduce the first criterion: **explicitness**, representing the degree to which the various processes for data collection and analysis underlying an EM are documented. Explicitness does not only refer to the textual documentation of such processes but also to the tools and various means used to facilitate the documentation and sharing of data.

In the same vein, we define the **formalness** of an UEM as the degree to which it is expressed through mathematical formulation or, more generally, in accordance with predefined rules and policies, leaving no room for ambiguity.

A third criterion to distinguish EMs is **calculableness**. An EM is said to be **calculable**, resp. **not calculable**, when its results can be measured or assessed according to a formula, a rule, or an algorithm, resp. when it is not possible. Contrary to the other properties of quality models listed here, **calculableness** is not represented on a continuum of values but rather as a binary property. Indeed, a function expressed in a formal fashion is either calculable or not and there is no degree of calculableness in-between. Also, not everything can be made calculable due to Rice’s theorem, which states that all nontrivial, semantic properties of algorithms are undecidable [60].

Finally, an important criterion to differentiate among EMs is **interpretability**, which refers to the degree to which results of an EM can be interpreted in the light of a clearly defined set of rules and policies. According to this definition, SUS, CSUQ, and PSSUQ are moderately interpretable as no clear way exists to interpret results apart from best practices coming from a large adoption in the literature. However, evaluation based on usability guidelines, like in SIERRA [155], hinders the interpretation of results and is left to the experimenter, assuming that respecting usability guidelines makes the UI usable. The absence of any means of interpreting the results leads to a risky conclusion that an interface is evaluated positively, negatively, or neutrally.

In principle, any explicit, formal, calculable and interpretable EM should subsume a quality model [1, 54]. Without these properties, it is almost impossible to reproduce any UI evaluation based on this EM and it is even more impossible to compare its results with any other EM.

For example, Gray & Salzman [53] compared the design of five usability experiments involving different EMs to observe that their produced results are neither comparable nor consistent with each other. In the area of accessibility evaluation, different EMs and tools could produce inconsistent results on the same web site [23, 22].

In addition to the large number of EMs [160], a substantial body of literature investigated, evaluated, and reported various aspects of UIs as well as their impact on usability and user experience (UX). In particular, the quality of GUI visual design [29] influences the overall software quality [113] by contributing to its usability [1, 137], its aesthetics [96], and other dimensions such as credibility [172], usefulness [7, 148] or performance [140]. Visual design [29, 99] typically manipulates the GUI visual components (*e.g.*, widgets, menus, contents, pictures, videos, banners), their properties (*e.g.*, size, color, typography), and their layout by relying on techniques borrowed from visual priming [41], such as visual techniques [106, 157], symbolic qualities [58], qualitative and quantitative properties [137], and aesthetic properties [171].

1.2 Challenges and Concerns in GUI Visual Design Experimentation

A particular research direction that has grown considerably in the last few years is the experimental study of GUI visual design through software measurement [58, 44]. The typical process consists of the following steps: defining and building a GUI dataset for different configurations (*e.g.*, capturing screenshots in different resolutions on different devices, wireframes, or mock-ups), computing visual design measures for the various configurations, and analyzing their results internally and/or externally by comparing them to qualitative measures provided by participants (see figure 1.4 for an example).

However, this reliance of experimental studies on large datasets reveals problems that hinders the progress in this research field. The construction of datasets requires time- and resource-consuming processes implying human intervention. The manual nature of these processes such as interface segmentation induces a high variability in the results. Not only do these issues make the construction of datasets in the field of GUI visual design a hardly scalable and error-prone process, but they also limit the reproducibility and verifiability of experiments [122], which is a major requirement in many research settings in order to assess the value of scientific claims.

During the past decade, concerns about the replicability of scientific re-

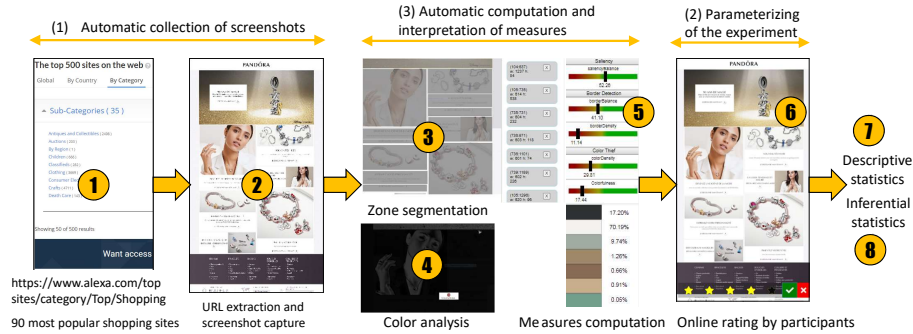


Figure 1.4: Experimental workflow illustrated on an experiment for assessing the GUI visual design of shopping web sites [27].

search have been steadily rising, with articles and papers reporting that results of experiments in numerous domains of science could not be replicated, leading to a replication crisis [10, 18, 118, 134]. The consequences of the *replicability crisis* [52] are manifold and vary depending on the research field, ranging from a lowered efficiency of scientific output to the waste of billions of dollars [47]. Replicability failures cannot be traced to a single cause, but several categories of shortcomings can explain many of the cases where research cannot be reproduced. Here are some of the most significant categories¹:

- *Lack of availability of raw data and methodological details:* in order for scientists to be able to reproduce prior studies, they must be able to access the original data and protocols. Without these, reproduction is greatly hindered and researchers are forced to reinvent the wheel as they attempt to repeat previous work. Visual design experiments very rarely share their raw data and methodological details. Even if they do, reproducing the same experiment is not straightforward. This explains the large amount of one-shot experiments in GUI visual design.
- *Inability to manage complex datasets:* progress in technology has enabled the acquisition of extensive, complex, and very large datasets. However, many researchers do not have the knowledge nor the tools needed to manipulate the data correctly. Furthermore, new technologies or methodologies may yet to establish or standardize protocols, so variations and biases can be easily introduced, which in turn can hinder the replication of collected data. Some environments, such as Lab-in-the-Wild [125], produce such very large datasets and are therefore able to analyze visual design on a very large base. In contrast, many other

¹See <https://www.nature.com/articles/d42473-019-00004-y>

visual design experiments work on a dataset whose sampling is often criticized as not being representative enough of the whole population, particularly for its small size and gender bias.

- *Poor research practices and experimental design*: in many cases, the cause of non-reproducibility can be traced to poor practices in reporting research results, and poor experimental design [127]. Poorly designed studies without a core set of experimental parameters, whose methodology is not reported clearly, are less likely to be reproducible. Visual design experiments often rely on qualitative and quantitative measures, which is a good practice, but their operationalisation is debatable. Some measures are computed manually, which may lead to human errors, while some others are computed (semi-)automatically, which raises the question of software reliability.
- *A competitive culture that rewards novel findings and undervalues negative results*: in general, researchers are rewarded for publishing novel findings, even if they are not really interesting or feasible, but not for publishing negative or not significant results. Overall, reproducibility in research is hindered by under-reporting of studies that yield results deemed disappointing or insignificant. Human-Computer Interaction (HCI) does not really escape from this trend [61].

Based on the aforementioned observations, we want to focus on some concerns as follows:

- *There is a lack of common methodologies and availability of research material in experimental studies of GUI visual design*: Greiffenhagen & Reeves [54] observed that HCI privileges novelty over consolidation. In our context, this means that merely performing “yet another GUI visual design evaluation” would not be considered as a new contribution [124] as it would join a large pool of separate, independent experiments which are poorly related. Indeed, a significant body of papers exists that studies various facets of GUI visual design, but they are rarely, if ever, put in perspective with other papers, thus making them hard to compare. The lack of common open practices and methodologies as well as the lack of availability of raw data and research materials contribute to this concern.
- *There is no clear-cut definition of replicability, reproducibility and what a valid replication study actually is*: the importance of multiple studies corroborating a given result is acknowledged in virtually all fields of science. However, reproducibility and replicability have different meanings and uses across different domains of science and engineering,

which has led to confusion in understanding problems in reproducibility and replicability. For these reasons, ACM engaged in a continuous effort to define ACM badges.

- *Visual design experiments require time- and resource-consuming processes that hinder their replicability:* nearly all steps are highly repetitive and tedious, thus making them highly time- and resource-consuming, in particular when the dataset should be large enough to warrant a valid experiment.
- *There is a lack of flexibility and automation regarding the computation of GUI measures:* GUI measures that are automatically computed by software raise the question of their reliability (are the measures correctly computed?), their flexibility (how can we parameterize the computation?), and their automation (to what extent could we automate the computation and statistical handling of these data, in correlation with end users' data?)
- *There is a lack of flexibility and automation regarding the capture of GUIs:* GUIs serve as input stimuli for visual design experiments with a high degree of variability depending on their size, resolution, level of fidelity, orientation. When the GUI is represented (top part in Fig. 1.2), many formats are possible such as screenshots, mockups, drawings, prototypes, which increases even more the degree of variability encountered for the stimuli. Therefore, there is a need to support more flexibility in their handling.

1.3 Research Question and Working Hypotheses

Based on the concerns formulated previously, we would like to address the following research question:

How can we support a user interface evaluation method for a graphical user interface through an explicit, formal, calculable and interpretable quality model to make it reproducible?

To address this research question more specifically and to properly delineate the scope of this thesis, we pose a series of working hypotheses as follows (Fig. 1.5):

- H_1 . *We limit the investigation on GUI representations:* while real GUIs (bottom part in Whitefield *et al.*'s framework in Fig. 1.2) enable a wide spectrum of UEMs, the main requirement is that there should be a really working and properly running GUI. Conversely, when a GUI representation is solely required, more specific UEMs could be used and early

in the development life cycle. Since there is no need to have a really running GUI, a prototype, a sketch, or a mock-up could be convenient, thus enabling an early evaluation.

- H₂. We consider both represented and real end users:* when a GUI is represented, end users could be either real or represented (Fig. 1.2). To make the UEM flexible, the involvement of real end users could be required, but this requirement could be relaxed in setups where such users are hardly available or in setups where time and budget restrictions impose this relaxation.
- H₃. We focus on visual design:* this viewpoint is subject to a wide spectrum of experiments and is applicable in both cases: (1) when the end user is represented (in this case, the end user is replaced by an analytical model capturing its visual design, which will be referred to as a quality model) and (2) when the end user is real (in this case, the end user is in a position to express her perception of visual aspects since everybody supposedly possess this ability).
- H₄. We focus on preference:* performance measures are incompatible with represented GUIs since real GUIs are required to assess their performance. Moreover, performance essentially contributes to efficiency, one of the three dimensions of usability. Conversely, preference could be investigated by end users or not on the three usability dimensions: effectiveness, efficiency, and subjective satisfaction. But the restriction imposed by this hypothesis is therefore that only “what people say” measures can be involved, thus focusing on perceived aspects as opposed to real aspects (“what people do”). Since we do not aim at supporting performance measures, only self-reported measures can be collected from participants, whether they are represented or real.

1.4 Reading Map

To address our research question in the scope delineated by our working hypotheses, the remainder of this thesis is structured as follows:

Chapter 2 will discuss the state of the art in the area of GUI visual design experimentation by breaking this field as follows: (1) a non-exhaustive list of GUI Visual Design experiments will be drawn up to show their diversity; (2) a Systematic Literature Review (SLR) will be performed to determine which visual design measures are available today to evaluate GUI representations; (3) a survey of selected software for automating parts or whole of the GUI visual design experimentation. Based on these three facets (experiments, measures,

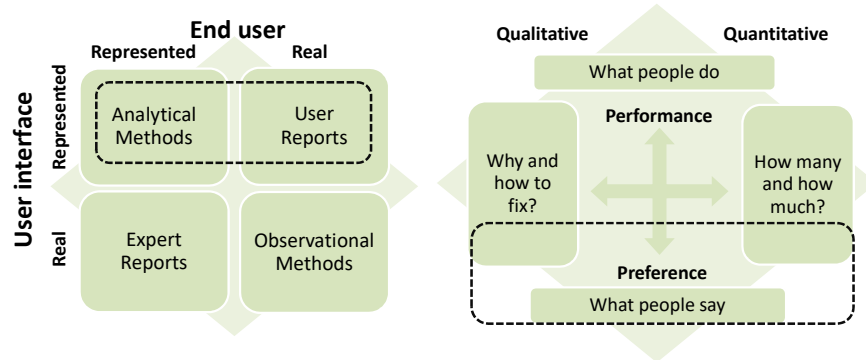


Figure 1.5: Scope of the research question.

and software tools), this chapter will highlight a set of limitations justifying the concerns raised in Section 1.2, then discusses the question of research reproductibility, progressively in science, in computer science, in HCI, and in our field of GUI visual design experimentation.

Chapter 3 will provide the fundamental contribution of this thesis by providing a formal definition of quality models. The definition will be generic enough to accommodate various types of quality models (*e.g.*, questionnaire-based models, machine-learning based models, etc.). The different properties of quality models will be discussed in the light of this formal definition. In addition to the formal definition of quality models, we will introduce a framework for replication and we will discuss how full replication of quality models can be achieved.

Chapter 4 will present `UILAB`, a software environment for (semi-) automating a series of processes implied in the conduction of experimental studies on GUI visual design and conception of quality models. A conceptual model of the tool will be first discussed in details. The remaining of the chapter will describe the walkthrough of the various parts of the software tool with implementation details provided throughout the different sections. The degree to which each of the tool features supports replicability will be discussed.

Chapter 5 will validate the methodology described in chapters 3 and 4 by defining, conducting, analyzing, and reporting results of a large-scale experiment in GUI visual design. The impact of the proposed methodology on interpretability of quality models will be discussed, along with its strengths and limitations.

Chapter 6 will summarize the contributions brought by this thesis, discuss their strengths and their limitations by revisiting the initial concerns raised in Section 1.2, and will present some future avenues to this thesis by relating them to the previously discussed limitations.

State of the Art

| 2

This chapter delivers an overview of the state of the art related to GUI visual design experimentation to support the various concerns raised in Section 1.2 of Chapter 1. This chapter is structured as follows: Section 2.1 draws up a non-exhaustive list of GUI Visual Design experiments to show their diversity; Section 2.2 performs a Systematic Literature Review (SLR) to determine which visual design measures are available today for evaluating GUI representations; Section 2.3 surveys selected software used to automate parts or whole of the GUI visual design experimentation process. Based on these three facets, *i.e.*, experiments, measures, and software tools, this chapter highlights a set of limitations justifying the concerns raised in Section 1.2, then discusses in Section 2.4 the question of research reproducibility, progressively in science, in computer science, in HCI, and finally in our field of GUI visual design experimentation.

2.1 GUI Visual Design Experiments

2.1.1 GUI Aesthetic Measures

There has been a long tradition in trying to capture what GUI aesthetics means by translating this subjective value into a set of qualitative and/or quantitative measures. Here is a series of chronologically sorted studies:

- Tullis [153] (1981), the pioneer in this area, conducted a series of experiments to compute six task-independent measures evaluating alphanumeric user interfaces displayed on Cathode-Ray Tube (CRT) screens consisting of 24 lines $\times$ 80 columns of characters: the percentage of the display used to present information (*overall density*), the percentage of the space used within each individual group of items (*local density*), the number individual of items on the screen, the number of groups of items on the screen, the average size of the groups of items, and how well items are aligned vertically and horizontally. His results indicated that a combination of these measures can be correlated with both search times and end-user preference ratings for information extraction tasks on alphanumeric displays. The most accurate model used the two density measures and the two grouping measures. These measures were

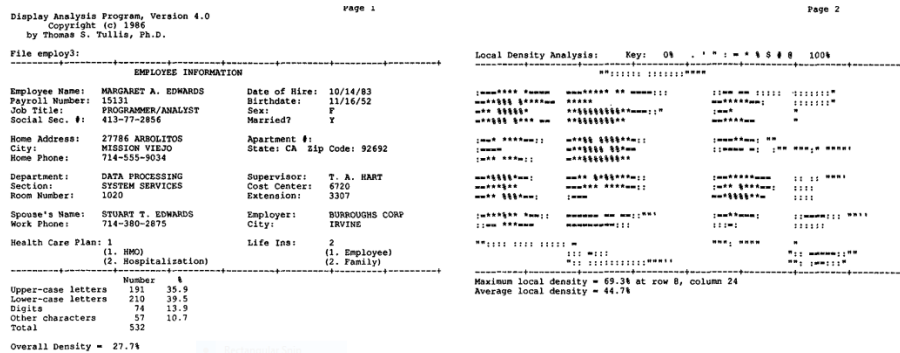


Figure 2.1: Density computation of an alphanumeric UI by DAP [152].

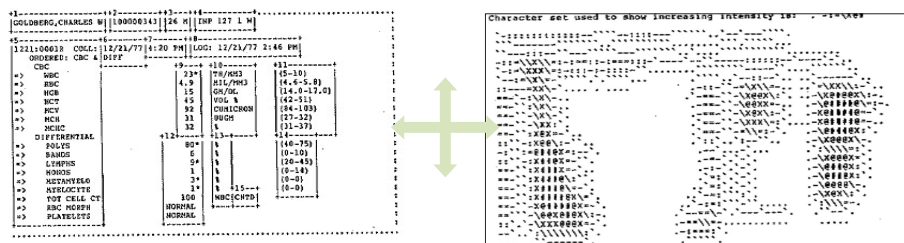


Figure 2.2: A boxing analysis (left) and a hot-spot analysis (right) of an alphanumeric display [145].

initially computed manually but, thanks to this correlation, a software called Display Analysis Program (DAP) was developed for this purpose in 1986 [152] (Fig. 2.1).

- Streveler & Wasserman [145] (1984) applied three types of analysis, *i.e.*, boxing analysis, hot-spot analysis, and alignment analysis, to determine the optimal value of physical measures (*i.e.*, pixel loading, field loading, and box loading), associative measures (*i.e.*, average distance between boxes and its ratio with the total number of boxes, average size of boxes), and aesthetic measures (*i.e.*, balance, symmetry, number of symmetry axes, and similarity of boxes). While these measures were manually computed on alphanumeric user interfaces like Tullis [153] (Fig. 2.2), the authors identify the merit and the need for such measures which should be natural, intuitive, and relatively easy to compute. Although these measures were applied on text screens, they could be generalized to GUIs, but this extension never appeared as such.
- Vanderdonck & Gillo [157, 156] (1994) defined a series of visual techniques for assessing a GUI according to visual techniques borrowed

from visual design [41]. These visual techniques fall into five categories: *physical techniques* (i.e., balance, symmetry, regularity, alignment, proportion, and horizontality), *composition techniques* (e.g., simplicity, economy, regularity), *association and dissociation techniques* (i.e., unity, repartition, grouping, and sparing) (Fig. 2.3), *ordering techniques* (e.g., consistency, predictability, sequentiality, continuity), and *photographic techniques* (e.g., sharpness, roundness). Although each visual technique is precisely defined in natural language (explicit), no procedure is provided to assess and to interpret them (informal). For example, “Simplicity is directness and singleness of layout, free from secondary complications or sophistications. Simplicity improves largely the ease of understanding the layout grid. Simplicity is guaranteed by placing GUI elements according to a logical and natural arrangement (e.g., by frequency, by a physical property) driven by the task’s semantics. Complexity, the opposite of simplicity, increases visual intricacy with too many units, forces and results and hinders any organization of the layout grid. Tiled GUI elements are considered as a simple layout while varying overlapping GUI elements are considered as a complex layout.” [156]. Each visual technique can play in favor of *harmony* (left part of each cell in Figs. 2.3 and 2.4), resp. *contrast* (left part of each cell in Figs. 2.3 and 2.4), when they are aimed at establishing an overall aesthetic value, resp. specifically intended values (e.g., to highlight some important parts).

- Ngo *et al.* [103] (2000) proposed a theoretical approach to quantify aesthetics of GUIs through measures. They introduced several aesthetic measures for GUIs along with their formulas. They carried a study demonstrating close relationships between perceived aesthetics and the proposed measures.
- Bauerly & Liu [14] (2006) investigated the effect of several elements on aesthetic judgement. Their findings indicate strong correlations (both positive and negative) between visual design elements and aesthetic ratings. They further demonstrated the importance of symmetry and the number of compositional elements in another study [16] by reporting studies measuring the effect of these two parameters on subjective appraisal of interface aesthetics.
- Zain *et al.* [168] (2008) developed an Aesthetic Measurement Application (AMA) to measure aesthetics and led an experiment comparing the measured aesthetics with the users’ perception of visual appeal for web pages. They conclude that the results provided by the application were congruent with the collected data about users’ perceptions of aesthet-

ics.

- Zheng *et al.* [173] (2009) introduced low-level features of images based on Ngo's work [103, 106] in the study of GUI visual design and measured their influence on users' perception of visual design aesthetics. They identified several high correlations between low-level features of interfaces and design-relevant dimensions.
- Zain *et al.* [97] (2011) introduced an application, Self-Developed Aesthetics Measurement Application (SDA), allowing the evaluation of web page interfaces aesthetics through the measurement of elements such as balance, equilibrium, symmetry, sequence, rhythm, as well as order and complexity. Results from the associated experiment showed that the ranking of the aesthetics values by users were aligned with the predictions of the application.
- Altaboli *et al.* [8] (2012) systematically studied the effect of the number of objects and the number of different sizes of objects on the perceived aesthetics of websites visual design in various symmetry conditions. The purpose of their work was to verify findings of earlier observational studies suggesting that the effect of unity of form on perceived aesthetics of websites visual design was more pronounced in high symmetry conditions. Their results showed that both factors have significant effects on perceived aesthetics only at high levels of symmetry.
- Hamborg *et al.* [55] (2014) further examined the effect of these interplay between usability and aesthetics on perceived qualities of a mobile phone prototype. They conducted an experiment with four versions of the prototype varying on two factors, (1) usability (high versus low) and (2) aesthetics (high versus low). Perceived usability, perceived beauty, hedonic experience and system's appeal were used as dependent variables. Result showed that the mobile phone's aesthetics does not affect its perceived usability, either directly or indirectly. Instead, results revealed an effect of usability on perceived beauty. Furthermore, effects of aesthetics and of usability on hedonic experience and appeal were found, indicating that both instrumental (usability) and non-instrumental (beauty) qualities contribute to a positive user experience.
- Zen & Vanderdonckt [170] (2016) compared the results issued from the inter-subjectivity of judgment of 15 participants evaluating four aesthetic measures, *i.e.*, symmetry, proportion, simplicity, and balance, on 10 web GUIs and the values of these measures computed semi-automatically by QUESTIHM [171]. Symmetry, proportion, and simplicity were

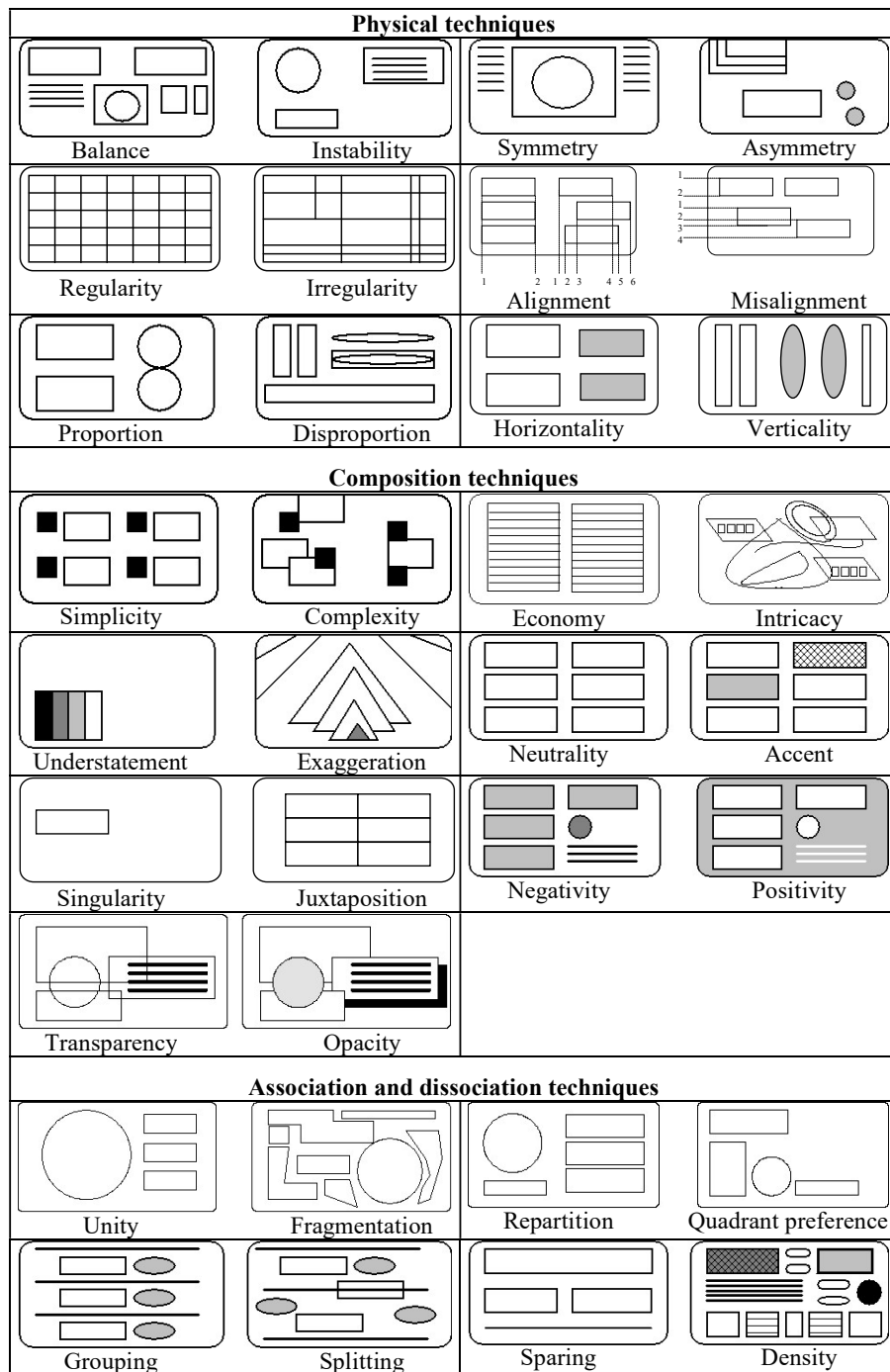


Figure 2.3: The three first categories of visual techniques: physical, composition, and association/dissociation.

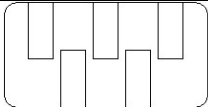
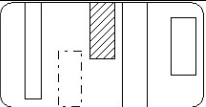
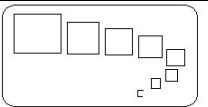
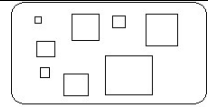
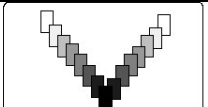
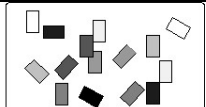
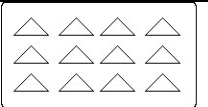
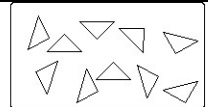
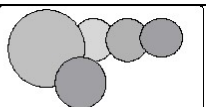
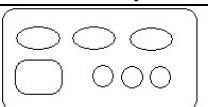
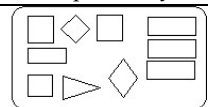
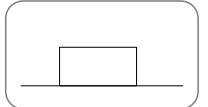
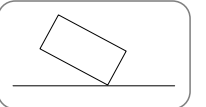
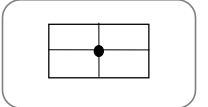
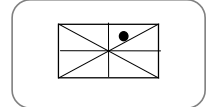
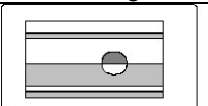
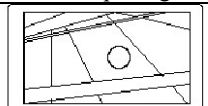
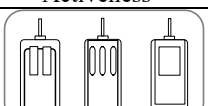
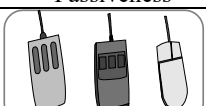
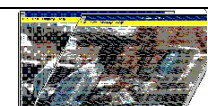
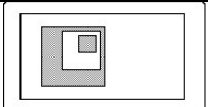
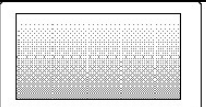
Ordering techniques			
			
Consistency	Variation	Predictability	Spontaneity
			
Sequentiality	Randomness	Continuity	Episodicity
			
Sharpness	Diffusion	Roundness	Angularity
Photographic techniques			
			
Stability	Stress	Levelling	Sharpening
			
Activeness	Passiveness	Subtlety	Boldness
			
Representation	Abstraction	Realism	Distortion
			
Flatness	Depth		

Figure 2.4: The two last categories of visual techniques: ordering and photographic techniques.

positively correlated with human judgment, while balance was not, but its decomposition into vertical and horizontal balances were again correlated with users perception. It was assumed that the various subjective evaluations performed by the participants would reach an objective evaluation, which is referred to as inter-subjective judgement.

Seffah *et al.* [137] created a generic environment called Quality in Use Integrated Management (QUIM), which consists of a repository of 127 usability measures to be used for assessing the overall quality of a user interface. These measures were gathered from various sources, such as user interface models, standards, style guides, and scientific literature. The repository was developed based on a unified hierarchical model that defines 10 factors, each of which represents a specific aspect of usability, similarly to ISO 25010 [113] factors that are decomposed into sub-factors. Also, these factors were decomposed into 26 criteria which define the 127 usability measures. WebTango [63] also gathered a set of 216 quality attributes, not quality measures, to assess the visual quality of a web page. These attributes are most of the time syntactical and low-level, such as the total number of fonts, the total number of font sizes, how many foreground and background colors. Ivory & Hearst [64] defined higher-level measures than these attributes, but computed based on them, thus making them an implicit quality model. They found out that these measures can predict the usability of web pages up to 80% by matching the results of expert evaluations of the same web pages.

Aesthetics and other facets of GUI visual design have been studied mainly on desktop configurations but recently, experimental studies started to investigate the effect of these facets on other platforms. The following series of chronologically sorted studies illustrate the spread of experimental studies on visual design of mobile GUIs:

- Alnanih & Ormandjieva [6] (2013) proposed a quality-in-use model for measuring UI design quality specifically intended for mobile devices. The proposed model, based on the international standard ISO 9126-4, uses existing quality-in-use factors and a new one which purpose is to measure users' cognitive load while interacting with mobile interfaces. Their work includes theoretical and empirical validation of the proposed measurement model.
- Harrison *et al.* [57] (2013) conducted a small review of mobile usability models and found that usability is usually measured in terms of three attributes; effectiveness, efficiency and satisfaction while other attributes, such as cognitive load, tend to be overlooked despite their likely impact on an application. They introduced a new usability model, named

PACMAD, which was designed to address the limitations of existing usability models when applied to mobile devices.

- Miniukovich & De Angeli [96] (2014) reported the results of 2 studies: the first study confirmed the persistence of first impression on mobiles interfaces evaluation, although suggesting that the exposure time might be longer. The second study extended previous work on automatic evaluation of GUIs from desktop to mobile interfaces. Using 6 automatic metrics, authors' model explained 40% of variation in subjective complexity score and 36% of variation in aesthetics score.
- Silvennoinen *et al.* [139] (2014) were interested in the role of visual usability and visual aesthetics in an experimental research setup. This study examined user experiences and preferences in relation to the visual elements of color and perceived dimensionality of two different mobile application contexts. Their results showed that colors and two-dimensionality of visual elements highly improve qualities of applications with task-oriented functionalities, as well as entertainment-oriented applications.
- Taba *et al.* [146] (2014) explored the relation between UI complexity and perceived quality in Android applications. They provided guidelines for the amount of UI complexity that maximizes perceived quality through an empirical study on 1,292 mobile applications belonging to eight different categories.
- Deka *et al.* [40] (2017) used a system that combines crowdsourcing and automation to scalably mine design and interaction data from Android apps at runtime and build a large repository of mobile app designs. The goal of the dataset is to support users and designers in several processes such as design search, GUI layout generation, GUI code generation, user interaction modeling and user perception prediction.
- Riegler & Holzmann [128] (2018) proposed an approach based on the computational analysis of an app's visual appearance to infer users' perceived usability. They introduced a set of complexity metrics partially based on existing metrics for GUI visual design and conducted a user study to evaluate the quality of these metrics. The results showed significant correlations between some of the metrics, the perceived usability and workload.
- Soui *et al.* [142] (2020) proposed a fully automated framework which combines quality attributes found in the literature with user profiles

to identify aesthetic defects of Mobile User Interfaces (MUIs) by considering the MUI evaluation as a multi-objective optimization problem. They conducted a comparative study of several algorithms in terms of accuracy of aesthetic defects identification and confirmed the efficiency of their approach.

2.1.2 Derivative Experiments

Beyond trying to capture the meaning of GUI aesthetics through measures, other studies investigated the degree of importance of various aesthetic aspects in GUI visual design, without necessarily computing it via measures.

- Zhang & Li [172] (2005) surveyed 194 students and asked them to explore a university website to assess its potential usefulness on their college life. Their hypothesis was that user's immediate and reflexive affective reaction to IT, namely the perceived affective quality (PAQ), has a positive impact on his / her consequent cognition-oriented evaluations such as perceived usability (PU) or perceived ease-of-use (PEOU). Their results showed that PAQ has effectively a positive impact on well-known predictors such as PU and PEOU.
- Lindgaard [85] (2006) conducted three studies to ascertain how quickly people form an opinion about visual appeal of web pages. In the first study, participants rated twice the visual appeal of web homepages with an exposure duration of 500 ms each. The second study was a replication of the first experiment in which participants also rated each web page on seven specific design dimensions. The third experiment replicated the 500 ms condition as well as adding a 50 ms exposure time condition using the same stimuli. Results showed that visual appeal ratings were highly correlated from one phase to the next as were the correlations between the 50 ms and 500 ms conditions, suggesting that users can assess visual appeal within 50 ms.
- Alsudani & Casey [7] (2009) conducted a study to ascertain the effect of several factors on users' perceived credibility. They first suggested a framework of factors to develop a method focusing on the users' actual perceptions in judging credibility based on aesthetics. This was done with 30 subjects viewing 13 pairs of images of recruitment agency web sites. They found that 'Unity' in design with its elements of balance, harmony, contrast and dominance was an effective aesthetic factor resulting in immediate judgments on web credibility.
- Sonderegger [140] (2009) asked 60 adolescents to complete a number of typical tasks of mobile phone users using two simulated devices

with different visual appearances (highly appealing vs not appealing). The purpose of the study was to determine the influence of appearance on perceived usability, performance measures and perceived attractiveness. Their results showed that participants using the highly appealing phone rated their appliance as being more usable than participants operating the unappealing model. Furthermore, the visual appearance of the phone had a positive effect on performance, leading to reduced task completion times for the attractive model.

- Reinecke *et al.* [126] demonstrated a way of predicting the initial impression of aesthetics based on perceptual models of a website's colorfulness and visual complexity. They collected ratings of colorfulness, visual complexity, and visual appeal of a set of 450 websites from 548 volunteers and successfully developed computational models that measure the perceived visual complexity and colorfulness of website screenshots.
- Trausan-Matu & Dathan [149] (2016) instructed participants in their study to rearrange a set of rectangles in a GUI to make it aesthetic and usable. These rectangles were first expressed as generic shapes in an unspecified image, then as GUI elements. Authors observed the variation of five aesthetic criteria between the two cases: vertical symmetry increased from 6.647 to 6.76 (on a 1..10 scale), horizontal symmetry increased from 3.441 to 4.2, repetition of shapes increased from 4.529 to 9.821, but figurativity decreased from 7.5 to 5.5, rhythm breaking was similar. This study suggests that aesthetic factors were even more important in the case of a GUI than in the case of an unspecified image. Figurativity decreased because it was perceived as less important than usability.
- Dupuy-Chessa *et al.* [44] (2016) asked participants to express pleasantness, playfulness, and perceived value on four GUIs according to two independent variables: usability (low vs. high) and aesthetics (low vs. high). GUIs were first represented (as in Fig. 1.2) during an exposure stage, then became real in the use stage. GUI aesthetics influenced the participants' evaluation at the exposure stage, but not at the use stage.

This non-comprehensive literature review illustrates the variety of experimental studies on GUI visual design in terms of variables of interest, platforms, stimuli, measures and other parameters of experimental setup. Table 2.1 highlights this variety by summarizing these studies by the mean of several indicators such as the type of stimuli used, the number of participants, the type of measures used in the study and if the computation of measures was either manual, (semi-) automatic or non-existing.

2.2 Quality Measures in GUI Visual Design

We can identify different types of measures used in the evaluation of GUI visual design, such as data-based measures (e.g questionnaire-based evaluation [125], eye-tracking evaluation [49]), formula-based measures [103], algorithm based measures (e.g [173]) and heuristic-based measures (e.g principle-based evaluation [121]). Facets of GUI visual design can be measured through measures of different types (e.g questionnaire-based & formula-based measures) or identical types (e.g Balance, [136] vs [103]). Aside of the type of measures used to quantify a given facet of visual design, the application of a measure in different contexts (e.g e-commerce websites on mobile vs desktop platforms) can lead to different interpretations.

Questionnaire-based methodologies are one way to quantify UI quality through the measurement of users' perception of various facets. Questionnaires such as UEQ+ [135] allows researchers to measure UX of a system based on various of its facets such as usefulness, trust or attractiveness. Researchers use the available scales to compose concrete questionnaires aimed at evaluating UX in specific scenarios. Another questionnaire-based methodology used to measure and evaluate UX is the Attrakdiff evaluation scale [59].

Unlike the non-exhaustive list of experiments drawn up, the remaining of this section aims at examining algorithm- and formula-based quality measures used in GUI visual design in a more systematic way, with an almost complete set of measures. This set of measures will never be complete since new measures regularly appear, new definitions and different interpretations are progressively published.

This section therefore presents the results of a Systematic Literature Review (SLR) performed on quality measures in GUI visual design. We hereby outline our method for conducting the SLR organized as a four-phase flow procedure consisting of Identification, Screening, Eligibility, and Inclusion phases, inspired by the approach of Liberati *et al.* [82]. We took specific care to implement each phase, so that our results could be readily reproduced. Fig. 2.5 shows an overview of our method represented in the form of a PRISMA¹ diagram [82].

Since the goal of our examination is to provide a consolidated overview of the research in GUI visual design experimentation, a SLR is a typical research instrument that suits well this purpose. Several SLR methods exist in the literature and they are often aimed at extensively covering large domains [82]. A standard SLR procedure was applied [24, 71] to the research question, resulting in the process depicted in Fig. 2.5, based on the PRISMA (Preferred Reporting Items for Systematic reviews and Meta-Analyses) four-phase flow diagram [82] used in healthcare domains.

¹Preferred Reporting Items for Systematic reviews and Meta-Analyses.

Reference (Year)	Stimuli	Participants	Measures	Computation
Ngo <i>et al.</i> [103] (2000)	7 (layout patterns)	6; 22	Balance; Equilibrium; Symmetry; Sequence; Order & Complexity	Manual
Zhang [172] (2005)	1 university website	194	Affective quality	Manual
Bauerly <i>et al.</i> [14] (2006)	30 images; 30 webpages	16	symmetry, balance and qty. of elements	Manual
Lindgaard [85] (2006)	100 ; 50 homepages	22; 31; 40	Visual appeal	None
Zain <i>et al.</i> [168]	12 (mandarin learning web pages)	Unknown	balance, equilibrium; symmetry; sequence; etc.	Manual
Zheng <i>et al.</i> [173] (2009)	30 (web pages)	22	symmetry; balance; equilibrium; number of quadtree leafs	Semi-auto.
Alsudani[7] (2009)	13 (recruitment agencies websites)	30	Credibility	None
Sonderregger [140] (2009)	2 (simulated smartphones)	60	Usability & performance	None
Altaboli <i>et al.</i> [8] (2012)	8 (web page designs)	17; 21	Unity; Symmetry ; Visual aesthetics	None
Alnanih <i>et al.</i> [6] (2013)	1 (mobile app)	20	Effectiveness; Productivity; Task Efficiency; etc.	None
Hamborg <i>et al.</i> [55] (2014)	4 (prototypes)	88	usability; aesthetics	None
Miniukovich <i>et al.</i> [96] (2014)	17+33 mobile apps; 51+99 screenshots	17; 20	Visual aesthetics; visual complexity	Automatic
Silvennoinen <i>et al.</i> [139] (2014)	8 (4 designs / 2 mobile apps)	37	Color ; Perceived dimensionality	None
Taba <i>et al.</i> [146] (2014)	1292 android apps	None	Perceived quality; Complexity (sub-measures)	Automatic
Dupuy-Chessa [44] (2016)	4 (travel agency webpages)	103	Pleasantness, playfulness, perceived value	Manual
Zen [170] (2016)	10 (webpages)	15	Balance, symmetry, proportion, simplicity	Semi-auto
Riegler <i>et al.</i> [128] (2018)	3 android apps	12	Complexity-related measures	Automatic

Table 2.1: GUI Visual Design: Comparison of Experimental Studies

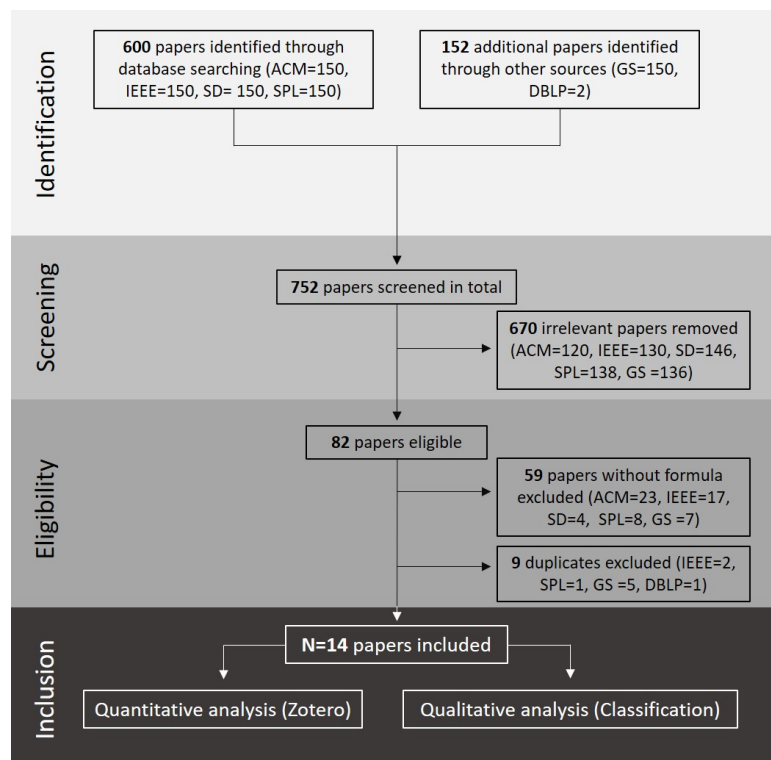


Figure 2.5: The four-phase flow diagram of our SLR examination.

The *number of references* (NoR) allows to rank the measures by popularity in the literature. Each measure is also associated to a *strength of evidence* (SoE) depending on the degree of verification of the associated formula. The SoE is defined as follows:

- If only a theoretical formula is given, SoE=1.
- If the formula is verified by at least one study with an example on a website, SoE=2.
- If the formula has been validated empirically with a correlation below 50%, SoE=3.
- If the formula has been validated empirically with a correlation between 50% and 90%, SoE=4.
- If the formula has been validated empirically with a correlation above 90%, SoE=5.

Table 2.2 summarizes the SLR performed on quality measures in GUI visual design by providing the number of references, number of formulas and the average strength of evidence for each measure. Details about the review procedure, the measures extracted from the literature and their corresponding formulas are given in appendix of this thesis.

2.3 Software for Experimental Study and Evaluation in GUI Visual Design

Considering the experimental study of GUIs visual design, there exists several tools allowing to partially automate of the dataset construction process. These tools mainly focus on either computing measures on GUIs or gathering data about users evaluation of GUIs and a lot of tedious tasks such as building experiments or capturing screenshots are left to the experimenter. Most of the tools available in the field of GUI visual design have been built with evaluation in mind, which makes them not fully suitable for experimental purposes.

In this section, we provide an analysis of of the different tools in the field of GUI visual design, their characteristics and the measures they implement. We evaluate the functionalities of these tools regarding a list of features provided in listing of section 2.3.15 and we summarize this evaluation in table 2.3. Table 2.4 summarizes the measures implemented by the different tools.

2.3.1 DON

DON [69, 70] (1990) is a software assisting designers in the generation of menu and dialog box layouts for their user interfaces. Users can specify a draft

Measure	Nb. of formulas	Nb. of references	Avg. SoE
Equilibrium	2	10	5
Economy	1	5	5
Homogeneity	1	5	5
Rhythm	1	6	5
Typography complexity	1	1	5
Element smallness	1	1	5
Inconsistency	1	1	5
Simplicity	2	9	4.5
Symmetry	5	17	4.4
Order and complexity	6	13	4
Sequence	3	11	4
Density	4	9	4
Size complexity	2	2	4
Edge congestion	1	1	4
Balance	9	21	3.6
Unity	2	9	3.5
Color contrast	2	2	3.5
Cohesion	3	7	3.3
Proportion	2	5	3
Visual coherence	1	2	3
Color complexity	1	1	3
Essential efficiency	1	1	3
Regularity	6	11	2.8
Composition	3	3	2.6
White-space fraction	1	1	1
Concentricity	1	1	1

Table 2.2: Summary table for Quality Measures.

or conceptual design of their UI in the application. The software then generates a realistic user interface automatically according to the design specified by the user. The tool also allows the computation of several visual measures for each generated user interface, such as symmetry, alignment, orientation, and proportion (Fig. 2.6). Aside of the computation of these measures, the tool does not allow any addition of new measures without recompilation neither user-related data collection.

2.3.2 AIDE

AIDE [136] (1995) is a software whose purpose is the semi-automatic design and evaluation of user interfaces. It evaluates GUIs aesthetics according to several visual measures such as efficiency, alignment, horizontal & vertical balance and constraint measures. The tool cannot be easily extended with new measures and the collection of user data is not possible.

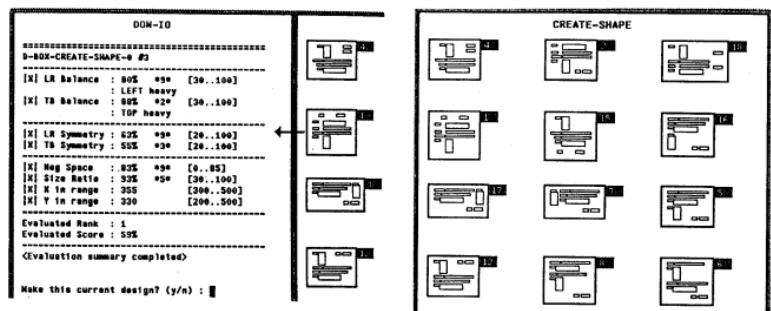


Figure 2.6: Layout GUIs generated by DON (right) and their automatic metric computation (left) [70].

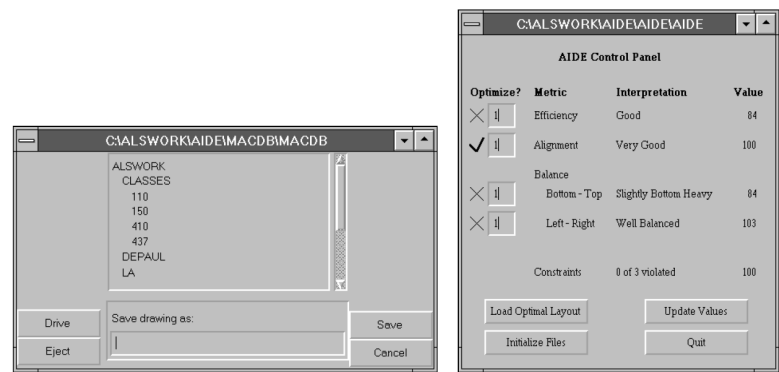


Figure 2.7: An optimal GUI layout computed by AIDE when only alignment is considered (left) with the AIDE Control Panel displaying values and interpretations for all measures [136].

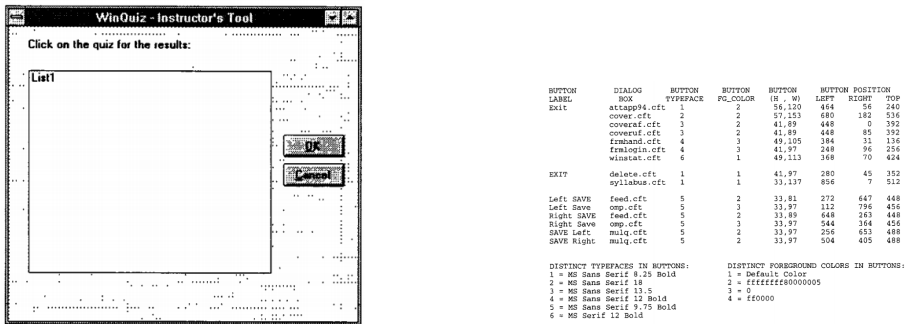


Figure 2.8: Screenshot of the SHERLOCK interface.

2.3.3 SHERLOCK

Mahajan [91] (1997) developed a set of consistency checking tools named SHERLOCK to facilitate the evaluation of GUIs. The tool uses visual measures such as number of widgets, aspect ratios, number of non-widgets areas, grideness, margin, balance, number of distinct typefaces, density, and distinct background / foreground colors. Upon evaluation of a graphical user interface, the tool returns a report with the computed measures. Extension of the tool with new measures cannot be done without modifying the source code and user data collection is not supported.

2.3.4 SEGUIA

SEGUIA [158] (1999) is a software allowing the automatic generation of alternative GUI layouts for a given task. The tool operates by optimizing several visual measures and their associated constraints. SEGUIA implements several algorithms for automatic generation; these algorithms mainly working on justification, centering, uniformity or equilibrium. As for other tools, the extension capabilities of this software are limited and it does not allow user data collection.

2.3.5 GUILayout++

GUILAYOUT++ [98] (2011) was developed by Montero as software tool to support the creation and evaluation of GUIs. The tools uses UsiXML [159] to generate abstract UIs and can evaluate real GUIs by computing measures such as unused space, interest regions or advertisement content. The evaluation of UIs is displayed to the user with the help of a report indicating a rating score for each of the computed measures. Again, this tool does not support the collection of user-related data and the extension of the tool with new measures cannot be done easily.

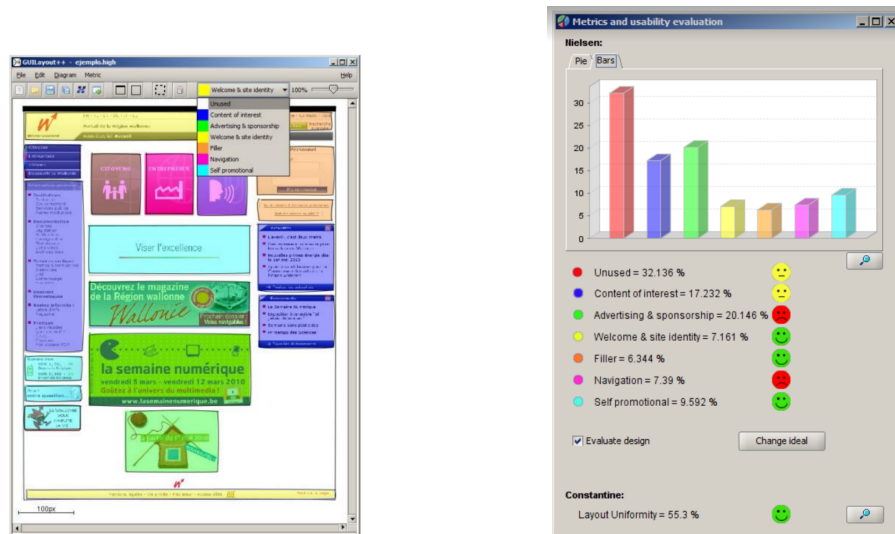


Figure 2.9: Screenshots of the GUILayout++ interface.

2.3.6 BaLOReS

BALORES [51, 87] (2012) is a framework composed of five structural principles and their associated measures. These principles help designers to structure their mockups and produce well-designed, pleasing GUIs to improve user's subjective satisfaction. The framework comes with a prototyping tool, BGLayout, that allows assessing the quality of interfaces design through metrics. BGLayout has been developed in Java and is presented as desktop application. BALORES builds a layout based on several "screen areas", which can be of different types (*e.g.*, text, image, form). The layout can be designed from scratch or based on a screenshot (Fig. 2.10). BGLayout allows computing five visual measures (*i.e.*, balance, regularity, linearity, sequentiality, and orthogonality) on the layout in order to quantify its different aspects and improve its design by putting it in correspondence with users' perceptions about the visual appearance. However, this comparison with users' perception must be done externally as the application does not support collecting user-related data. Moreover, the number of measures is fixed and the software does not accommodate any new measure in its current form.

2.3.7 QUESTIM

QUESTIM [171] (2014) is a web-based application whose purpose is to evaluate GUIs through measures computation. The tool is implemented in Java and is compiled for Javascript using Google Web Toolkit. In order to provide designers with an objective feedback regarding the visual design of their GUIs, such

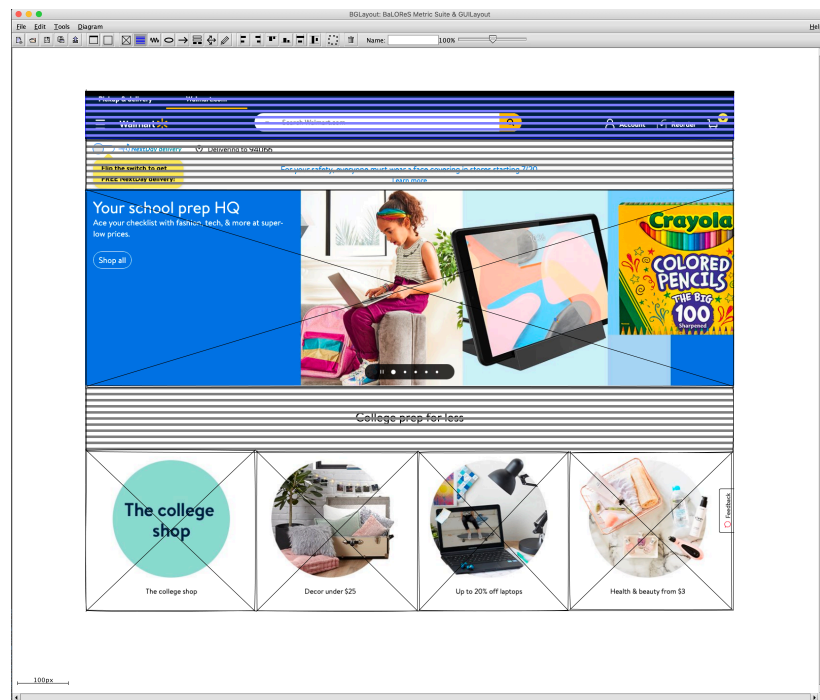


Figure 2.10: Screenshot of the BGLayout interface.

as for web pages, QUESTIM enables the end user to specify a website URL or upload a file containing any GUI artefact, such as a screenshot, a wireframe, a sketch, a picture, a prototype (Fig. 2.12). After defining graphical regions of interest (e.g., a widget, a group box, a menu, an image) by direct manipulation, visual design measures are automatically computed (Figure 2.11). Any change in the location, sizing, and arrangement of a region triggers the re-computation of the measures. Measures are displayed with the help of a color gradient whose formatting depends on the distribution of the measure. For example, the density measure will orient to red when the value is low or high, but becomes green slight above the middle. As measures included in Questim are coded in the core of the application, its extension requires modification of the source code and to redeploy the application. As it is the case for other application, user-related data collection is not implemented in Questim.

2.3.8 LabInTheWild

LabInTheWild [125] (2014) is a platform hosting various types of experiments, as represented on figure 2.13. It has been used, among other experiments, to collect user data about perceived visual aesthetics and to model several GUI features. This platform provides some support for collecting user-related dat. Participants can take part to any experiment deployed on the platform



Figure 2.11: QUESTIM: introductory screen

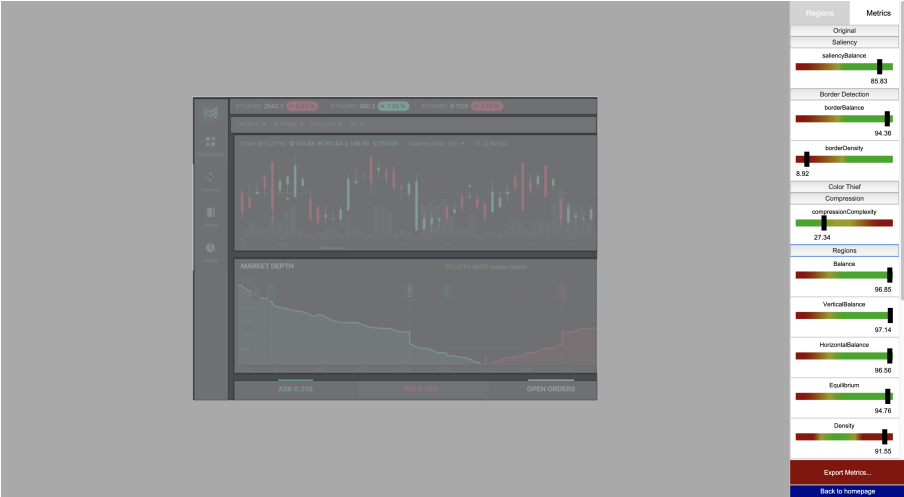


Figure 2.12: QUESTIM: evaluation screen.

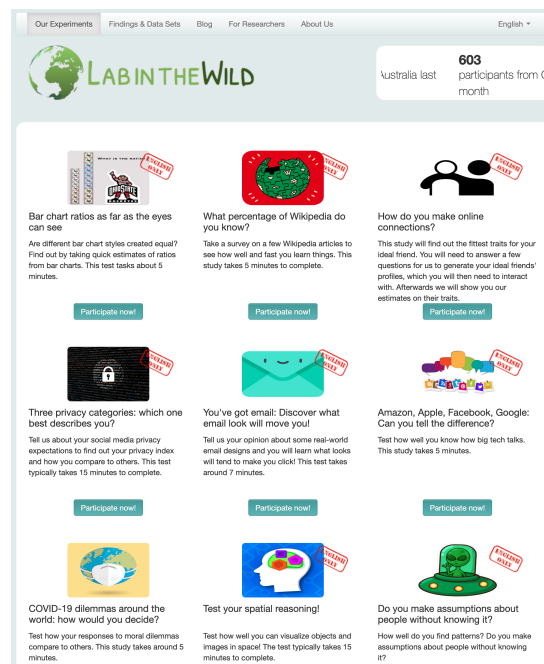


Figure 2.13: LabInTheWild - Home page with hosted experiments

and are instructed to fill in a pre-test survey about their socio-demographical profile before taking part to the experiment, as illustrated by figure 2.14. Given the variety of experiments available on the platform, the existence of a template-based experiment generation engine behind the platform seems unlikely. Such capability would reduce the variety of the experiments that can be deployed, but would increase the reproducibility of their setup and reduce their development and deployment time.

2.3.9 GUIEvaluator

GUIEVALUATOR [5] (2014) is a desktop application for evaluating the GUI complexity based on its structure. The application automatically computes five structural measures. The tool, developed in Visual Basic 2012, extracts the interface layout information using reflection techniques supported in the language. The tool can be used at design- or at run-time. The application does not allow collecting user-related data.

2.3.10 GUIExaminer

GUIEXAMINER [4] (2015) automatically computes the Screen Layout Cohesion (SLC) measure to predict its usability based on different aspects of the GUI.

Progress bar: Consent (✓), About you (2), Test Part 1, Test Part 2, Final questions, Results!

Please tell us a little bit about yourself.

These questions are **optional**, but if you could answer at least the first four, that would really help us. (Why?)

Have you taken this test before? (important)

How old are you? (important)

What is your gender? (important)

What is the highest level of education you have received or are pursuing? (important)

Imagine a 6-step ladder where on the first step stand the poorest people in your country, and on the sixth step stand the richest people in your country. On which step are you?

☐ Step 6 (the richest)

☐ Step 5

☐ Step 4

☐ Step 3

☐ Step 2

☐ Step 1 (the poorest)

☐ Prefer not to answer

How often do you use a computer?

In what country did you live most of your childhood?
(Please pick one that influenced you the most if you grew up in more than one)

In what country have you spent most of the past five years?

[Submit](#)

Figure 2.14: LabInTheWild - example of experiment

As for GUIEvaluator, the tool is developed in Visual Basic and is only applicable to Visual Basic GUIs since the source code of the interface is required to extract the widget properties and the layout type.

2.3.11 UI-CAT

UI-CAT [129] (2015) is a background service developed for the Android OS that allows the automatic computation of complexity metrics for Android applications. The tool records users' touch events and take screenshots of the opened application upon interaction without the need to modify the application's source code. Aside the user touch events, the application does not allow the collection of user data and is focused on measures computation.

2.3.12 PLAIN

PLAIN [143] (2017) is an Eclipse-based plugin that automatically computes eight measures, called *defects*, used to predict the usability of a mobile GUI: incorrect layout of widgets, overloaded GUI, complicated GUI, incorrect data presentation, incohesive GUI, difficult navigation, ineffective appearance of widgets, and imbalance. The measures are transposed from desktop GUIs to mobile GUIs based on Ngo's measures [106]. The source code for computing these measures is embedded in the plugin, thus requiring code modifica-

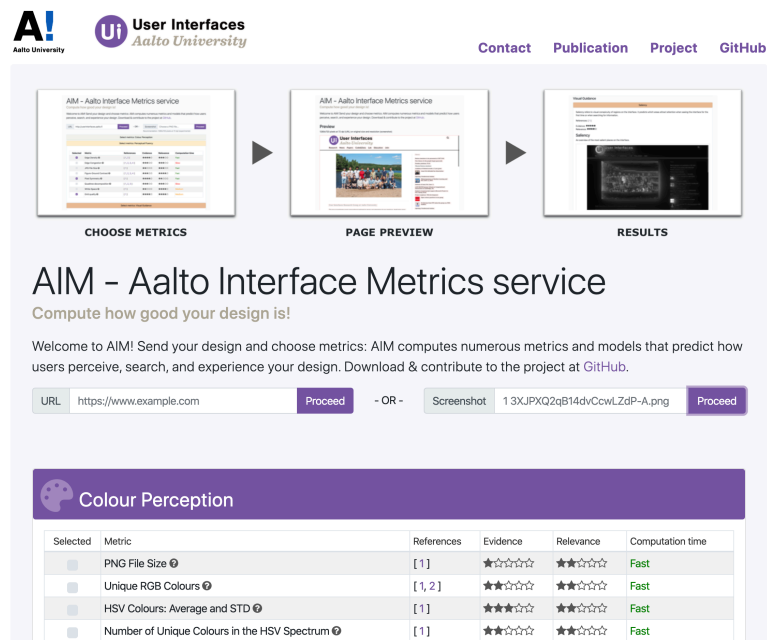


Figure 2.15: Aalto Interface Metrics (AIM): Measure selection

tion and re-compilation if any measure should be added, deleted, or modified. PLAIN is based on the genetic algorithm technique used for the generation of evaluation rules. PLAIN uses the GUI source code to perform the computation of the measures and the evaluation of the interface, contrarily to QUESTIM which uses a graphical representation of the GUI to compute the measures. Any user data collection has to be performed outside of the application. PLAIN proceeded later on to ADDET [19], which consists of two modules: a first module that statically computes thirteen visual design metrics and a second module that analyzes deviations from reference measures to detect defects in GUIs for mobile devices.

2.3.13 Aalto Interface Metrics (AIM)

AIM [115] (2018) is an online web application built for the computational evaluation of GUIs. The goal of the application is to facilitate the use and appropriation of computational methods in design practices. After specifying the screenshot or an URL (Fig. 2.15), the end user can choose the measures to be computed automatically on designated elements. The screen resulting from the computational evaluation (Fig. 2.16) appears with the values of the selected measures in real-time. For each selected measure, a detailed explanation is provided below the summary (Fig. 2.17).

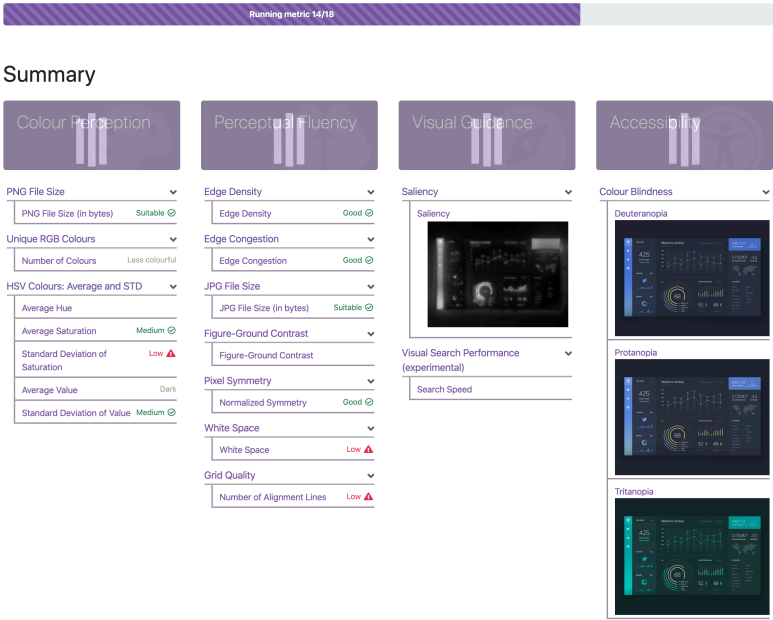


Figure 2.16: Aalto Interface Metrics (AIM): Summary of computed measures.

Results

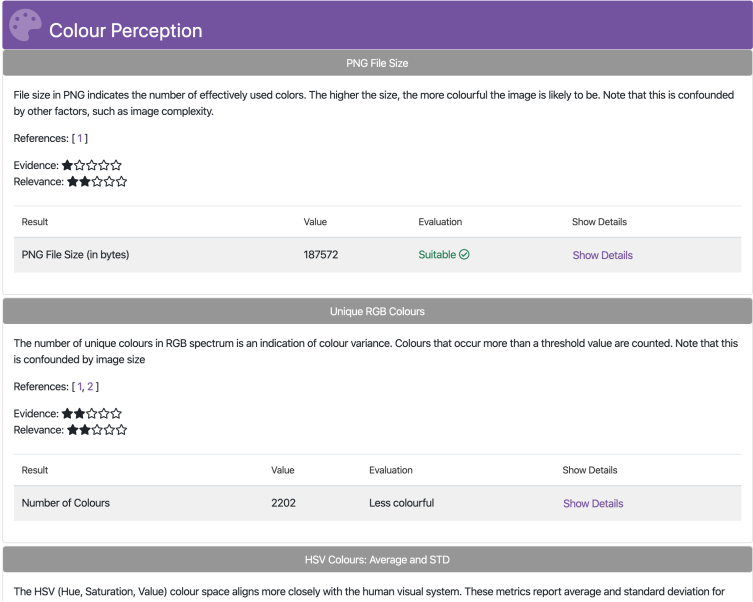


Figure 2.17: Aalto Interface Metrics (AIM): Computation details

2.3.14 Web UI Visual Analyzer (WUI)

WUI [12] (2019) consists in an online web application integrating measures computed from different providers for evaluating GUI visual design. The application is composed of a front-end and a back-end, as for AIM and QUESTIM. For this purpose, WUI is able to work with different remote services such as AIM as major providers for visual design measures. WUI allows the automatic capture of screenshots by specifying the URL or allows users to upload GUI screenshots, but in either ways, only one GUI can be analyzed at a time. As for AIM and QUESTIM, WUI only considers the computation of measures. To conduct the experiment they report in the the original paper, authors had to run WUI several times with different parameters. The computation of measures in WUI is similar to AIM as the user only has to select the measures he/she wants to compute on the specified interface. End-users can add new measures to the application by registering it in the system according to a specified procedure. The measures can be self-implemented or provided by a remote service. However, the whole system must be redeployed in order for changes to take place.

2.3.15 Comparative Analysis of Software

Table 2.3 compares the aforementioned tools and applications on the basis of a set of selected common functionalities and qualifies the extent to which eight comparison criteria are supported in the different tools via Harvey’s balls [67]. When a criterium is not applicable for comparison for a specific tool (e.g., the corresponding feature is not implemented), “NA” will be mentioned in the corresponding entry.

From Table 2.3, we identify several shortcomings from the existing tools and application in the field of GUI visual design. All of them process only a single type of input at a time, be it a screenshot or a piece of source code, thus repeating the same process for conducting an experiment where a large number of GUIs need to be measured is a time- and resource-consuming task that may be unfeasible. Not all aspects of GUI visual design can be evaluated or measured at once using a single type of representation of the interface. Such a restriction could be considered as an important limitation when studying GUI visual design. These applications do not process inputs concurrently or in batches (e.g., via a parallel computation of measures over a dataset of GUIs), thus lengthening the preparation phase. Perhaps this feature was not required for these applications as soon as they were primarily designed for evaluation and not for experimental research. If we consider experimental research of GUI visual design as a target, such applications are not suitable because they do not allow to process large datasets.

Inputs

1. Diversity: refers to the input types recognized by the application.
 - ○ - Low: the application is able to process only one type of input.
 - ● - High: the application can process several types of input.
2. Scalability: refers the the ability to process inputs concurrently.
 - ○ - Low: the application only allows the processing of one input at a time.
 - ● - High: the application allows the processing of multiples inputs concurrently.

User Data Collection

1. Diversity: refers to the variety of data types that can be collected.
 - ○ - Low: the application allows collecting only one type of user-related data
 - ● - High: the application allows the collection of a variety of user-related data
2. Customization: refers to the extent to which the data collection process is customizable.
 - ○ - Low: the data collection process accepts little or no parameters.
 - ● - High: the data collection process is highly parametrizable.
3. Automation: refers to the ease of deployment of the data collection process.
 - ○ - Low: the data collection process has to be deployed manually.
 - ① - Medium: the deployment of the data collection process is partially automated.
 - ● - High: the deployment of the data collection process is automated.

Measures computation

1. Diversity: refers to the variety of programming languages supported by the application for measures implementation.
 - ○ - Low: the application only supports one programming languages for the implementation of measures
 - ● - High: the application supports multiple languages for measures implementation.

2. Extensibility: refers to the ease of extensibility of the application with new measures.

- ○ - Low: the addition of new measures is time-consuming / difficult. The application must be rebuilt / redeployed.
- ● - High: The addition of new measures can be done without any change in the application nor rebuild / redeployment.

3. Automation: refers to the level of automation of measures computation.

- ○ - Low: the computation of measures is completely manual.
- ① - Medium: the computation of measures is partially automated but still requires human intervention.
- ● - High: the computation of measures is completely automated.

	Inputs		User data collection			Measures computation		
	Diversity	Scalability	Diversity	Customisation	Automation	Diversity	Extensibility	Automation
DON [69] (1990)	○	○	NA	NA	NA	○	○	●
AIDE [136] (1995)	○	○	NA	NA	NA	○	○	●
SHERLOCK [91] (1997)	○	○	NA	NA	NA	○	○	●
SEGUIA [158] (1999)	○	○	NA	NA	NA	○	○	●
GUILayout++ [98] (2011)	○	○	NA	NA	NA	○	○	●
BaLOReS [51, 87] (2012)	○	○	NA	NA	NA	○	○	○
QUESTIM [171] (2014)	○	○	NA	NA	NA	○	○	○
LabInTheWild [125] (2014)	NA	NA	●	●	○	NA	NA	NA
GUIEvaluator [5] (2014)	○	○	NA	NA	NA	○	○	●
GUIExaminer [4] (2015)	○	○	NA	NA	NA	○	○	●
UI-CAT [129] (2015)	○	○	○	○	●	●	○	●
PLAIN [143] (2017)	○	○	NA	NA	NA	○	○	●
AIM [115] (2018)	○	○	NA	NA	NA	○	○	●
WUI [12] (2019)	○	○	NA	NA	NA	○	○	●

Table 2.3: Level of support of visual variables expressed according to Harvey's Balls.

Only one application allows the collection of user-related data with a high degree of diversity. Again, this may not be part of requirements of application built for evaluation purpose but experimental studies on GUI visual design require user-related data to analyze any potential correlation between these data and values of measures. Most applications implement their measures by hard-coding them in the application via a single programming language. Consequently, the set of languages for implementing the computation of measures is de facto limited to the language used to develop the application itself. In some cases, it also limits the type of GUIs to be analyzed, such as for Visual Basic GUIs. Any modification of existing measures, any addition of a new measure, or any combination of existing measures imply direct manual programming of the source code of the application and the need to re-compile, re-build, and re-deploy it in order for changes to take place.

2.4 Reproducibility in (Computer) Science, HCI, and GUI Evaluation

Sections 2.1 and 2.2 revealed the heterogeneity of experiments conducted in the area of GUI Visual Design as well as the variety of existing measures allowing us to characterize GUI visual design in a quantitative way. Section 2.3 described several tools allowing the computation of measures on GUIs and compared them in table 2.3 according to the degree to which they implement the different features enumerated in listing of section 2.3.15.

The combination of various elements such as experiments heterogeneity, the large variety of measures and the limited capabilities of existing tools for measures computation limits the reproducibility of experimental studies on GUI visual design and hinders the scientific progress in this field of research.

Reproducibility is the cornerstone of cumulative knowledge and a requirement to confirm scientific truth [52, 132], where new results are confronted with prior findings and theories. Reproducibility is supported by clear-cut description of procedures, methods, experimental designs, and publicly available data. In particular, the lack of public data hinders the reproducibility and verifiability of experiments, since recollecting data may be a tedious endeavour [122]. The reproducibility crisis [118] denotes an actual situation in which scientific experiments are difficult, if not impossible, to reproduce [134].

Many attempts have been made to raise awareness of the importance of reproducibility, to formalize and structure reproducibility in science [52, 119, 120], including in Computer Science (CS) [50] and Human-Computer Interaction (HCI) [54, 62, 164, 165]. For example, Peng [120] considered that a paper is not reproducible unless its artifacts are made available. After that, the next steps require releasing source code [45], and both code and data to increase the potential for reproducibility. This continuum defines a gold standard for reproducible papers. Patil *et al.* [119] suggested a visual notation for expressing to what extent reproducibility is explicitly addressed in a scientific paper. In 2020, ACM [2] updated its definitions for “repeatability” (same team, same experimental setup), “reproducibility” (different team, same experimental setup), and “replicability” (different team, different experimental setup) to characterize and foster reproducibility in experimental CS, but without any operationalization.

The HCI community has equally acknowledged the need for reproducible research. For example, RepliCHI [62, 164, 165] defined *replication* as the “attempt to confirm, expand, or generalize an earlier study’s findings” (p. 3525). HCI also welcomes replication studies, *e.g.*, Claes *et al.* [35] replicated an in-the-wild study to evaluate the potential impact on ecological validity of a high fidelity prototype against the final user interface to find out that the prototype was very close to the final interface in terms of appreciation. How-

ever, according to the findings of Hornbæk *et al.* [62] on an outlet of 891 HCI papers, a replication rate of just 3% was found (28 studies). Most of the studies (22) replicated the work of other researchers, while the others replicated their own work (2) or did both (6 studies). By analyzing the nature of replications, Hornbæk *et al.* [62] revealed that earlier findings were confirmed, and that comparisons to prior work were often simple. In a follow-up, Hornbæk [61] encouraged the HCI community to embrace replication, especially when the outcomes are likely to question previous results, *e.g.*, “we must be more wrong in HCI research”.

Following RepliCHI [164, 165], the term “replicability” has been widely adopted in HCI where “reproducibility” has been used for the same purpose in general science [52]. Although different terminologies co-exist in HCI and other fields to designate reproducible research, this confusion is generalized across the entire science spectrum. Gomez *et al.* [50] reported 27 different frameworks to classify reproducibility.

For example, Hornbæk *et al.* [62] distinguish three replication types: a strict replication reuses the same variables as the original study to repeat it; a partial replication introduces any possible variation in these variables to investigate how the original results still hold or vary in a different setup (*e.g.*, such as in a different context of use); and a conceptual replication applies other measurement methods to the original data to detect any effect. Goodman [52] specifies that “reproducibility” refers to the ability of a researcher to duplicate the results of a prior experiment using the same materials as were used by the original investigator. Reproducibility is a minimum necessary condition for a finding to be believable and informative.’

Tsang & Kwan [150] classify replications into seven types along two dimensions: sources of data (same dataset, same population, and different population change in method same measurement and analysis, different measurement and/or analysis). From this perspective, we are considering the same measurement and analysis. The second dimension is about the sources of data, *e.g.*, reanalyze the data (checking of analysis), collect data again from the same population, or from a different population. While the aforementioned classifications are useful, they do not specify which artefact should vary in order to fulfill the needs of a particular replication type.

In the ACM definitions [2], the “same experimental setup” does not specify whether the same participants should be involved. If it is the case, a participant involved in the original study will never be the same in any further replication because of carry-over effects. In addition, the ACM definitions do not cover the case “same team, different setup”. The dimensions are not independent.

2.5 Summary

This chapter highlighted the large variety of experiments on GUIs visual design. Numerous studies have tried to capture the effect of visual design on the various facets of user experience [129, 7] or the effect of sub-factors of visual design on the quality of UIs itself [8]. Until now, the different studies lead in the field achieved their objectives with various degrees of success and reproducibility.

As in other research domains, HCI and in particular the study of GUI visual design suffer from a replicability crisis. This issue hindering scientific research has different sources, among which is the lack of research best practices (e.g sharing of data & methodology, etc.) in experimental studies led in the field of HCI and GUI visual design, as highlighted by section 2.1. corresponding table 2.1.

User Experience is a vast domain, influenced by a large number of factors such as UI quality or infrastructure. These factors also have influencing subfactors, such as depicted in figure 2.18. In the context of this thesis, we will focus on the formalisation of quality models for GUI visual design, along with its influencing factors and subfactors as illustrated in colors in figure 2.18. The next chapters will describe in details the support, both software and methodological, that we provide to make quality models explicit, formal, interpretable and in this way, reproducible.

	DON [69]	AIDE [136]	SHERLOCK [91]	SEGUIA [158]	GUILayout++ [98]	BaLOReS [51, 87]	QUESTIM [171]	LabInTheWild [125]	GUIEvaluator [5]	GUIExaminer [4]	UI-CAT [129]	PLAIN [143]	AIM [115]	WUI [12]
Balance	●	●	●	○	○	●	●	○	●	○	●	●	●	●
Equilibrium	○	○	○	●	○	○	○	○	○	○	○	○	●	●
Symmetry	●	○	○	○	○	○	●	○	○	○	○	●	●	●
Sequence	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Cohesion	○	○	○	○	○	○	○	○	●	●	●	○	○	○
Unity	●	○	○	●	●	○	○	○	○	○	●	○	○	○
Proportion	●	○	●	○	○	○	○	○	○	○	○	○	○	○
Simplicity	○	○	○	○	○	○	○	○	○	○	○	○	●	●
Density	○	○	○	○	○	○	○	○	●	○	●	●	●	●
Regularity	●	●	●	○	●	●	●	○	●	○	●	●	●	●
Economy	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Homogeneity	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Rhythm	○	○	○	○	○	○	○	○	○	○	○	○	○	○
White-space	●	○	●	○	●	○	●	○	○	○	●	○	●	●
Composition	○	○	○	○	○	○	○	○	○	○	○	●	○	○
Color contrast	○	○	●	○	○	○	●	○	○	○	○	○	●	●
Color complexity	○	○	○	○	○	○	●	○	○	○	○	○	●	●
Typography complexity	○	○	●	○	○	○	○	○	○	○	○	○	○	○
Efficiency	○	○	○	○	○	○	○	○	○	○	○	○	●	●
Size complexity	●	○	○	○	○	○	○	○	●	○	●	○	●	●
Coherence	●	○	○	○	○	○	○	○	○	●	○	●	○	○
Smallness	○	○	○	○	○	○	○	○	●	○	○	○	○	○
Edge congestion	○	○	○	○	○	○	○	○	○	○	○	○	●	●
Concentricity	○	○	○	○	○	○	●	○	○	○	○	○	○	○
Order & Complexity	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Inconsistency	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Others	●	●	●	●	●	●	●	○	○	○	●	●	●	●

Table 2.4: Implementation of measures by tools supporting GUI visual design evaluation

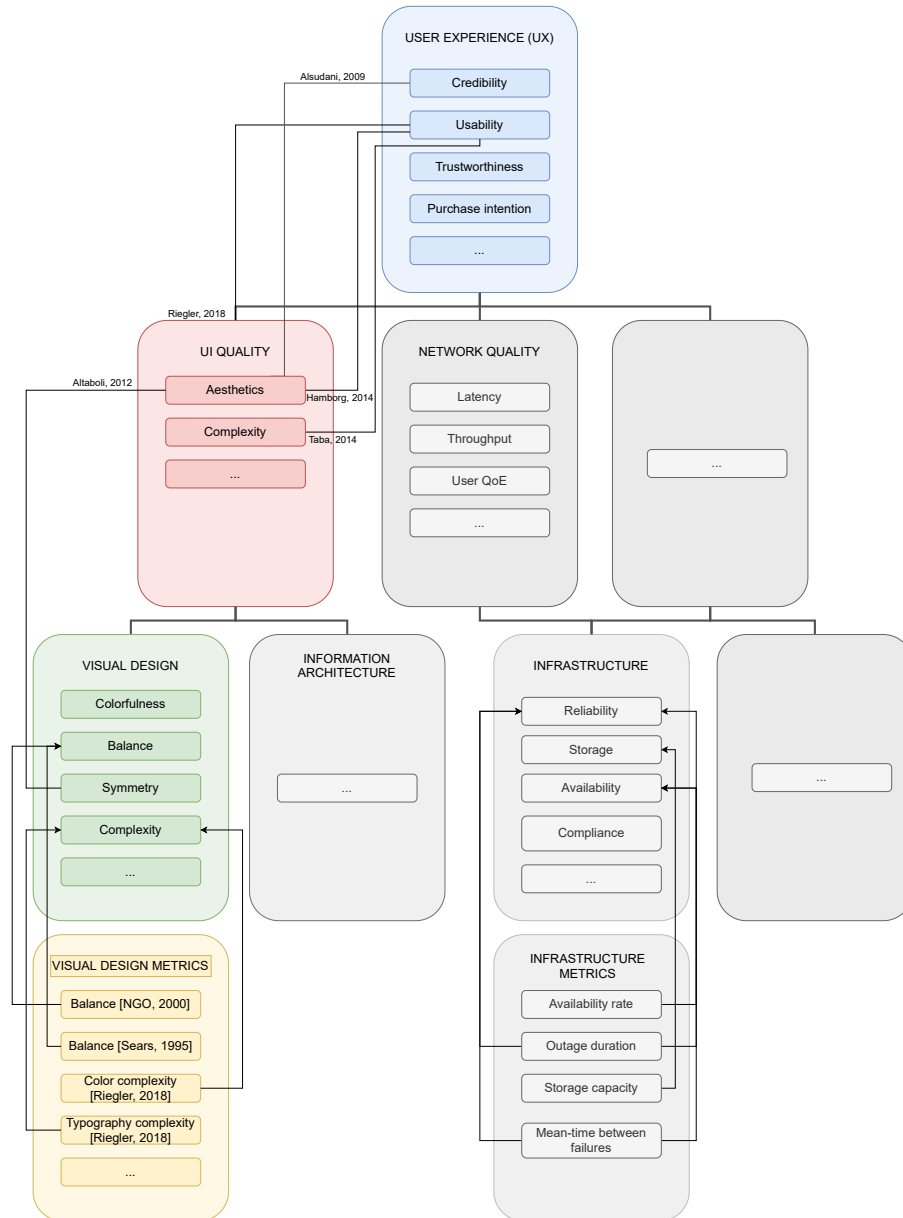


Figure 2.18: Influencing factors and sub-factors of User Experience

Quality Models for GUIs | 3

This chapter introduces, motivates, and defines the concept of a quality model for GUI evaluation so that this process can become reproducible, *i.e.*, explicit, formal, calculable, and interpretable We also introduce a framework for replicability with incremental criteria to make testable the replicability of quality models and to foster the replicability of experiment studies in the field of GUI visual design.

3.1 Related Work

3.1.1 The ISO 25010 Standard

Usability, as an ISO-25010 [113] software quality property, is defined as¹: “the degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use.” While *effectiveness* should evaluate how the UI supports users in completing actions accurately and *efficiency* should evaluate how end-users can perform tasks quickly through the UI, *satisfaction* should capture the subjective satisfaction with which end-users are carrying out their tasks.

The manual practice of GUI visual design experimentation may impedi- ment some fundamental ISO/IEC 25010 software quality properties [113] such as compatibility, maintainability, and portability, along with some of their sub-properties:

¹See <https://iso25000.com/index.php/en/iso-25000-standards/iso-25010?limit=3&start=3>

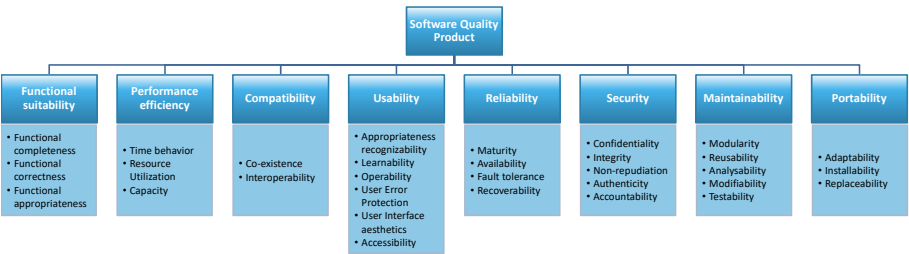


Figure 3.1: The ISO 25010 standard of software quality

1. *Compatibility*: “degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions while sharing the same hardware or software environment”. This characteristic is composed of the following sub-characteristics [113]:
 - *Co-existence*: “degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product”.
 - *Interoperability*: “degree to which two or more systems, products or components can exchange information and use the information that has been exchanged”.
2. *Maintainability*: “degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it or adapt it to changes in environment, and in requirements”. This characteristic is composed of some of the following sub-characteristics [113]:
 - *Modularity*: “degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components”.
 - *Reusability*: “degree to which an asset can be used in more than one system, or in building other assets”.
 - *Modifiability*: “degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality”.
 - *Testability*: “degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met”.
3. *Portability*: “degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another”. This characteristic is composed of the following sub-characteristics [113]:
 - *Adaptability*: “degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments”.
 - *Installability*: “degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment”.

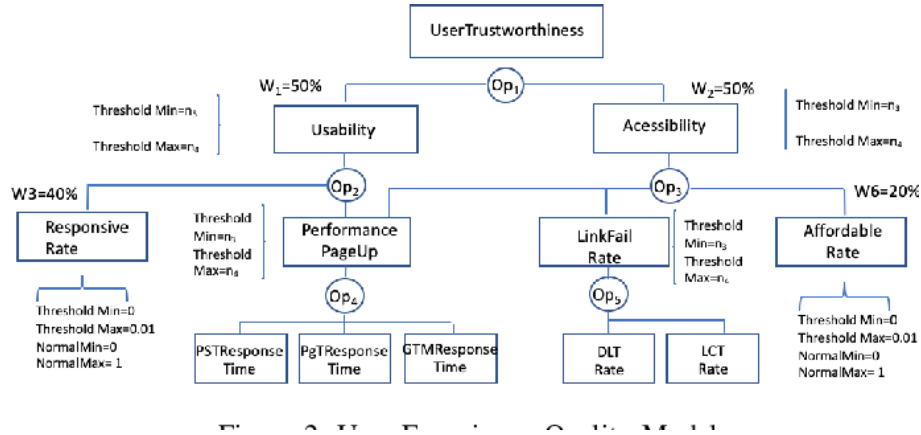


Figure 3.2: Quality Model for Interface Trustworthiness

- *Replaceability*: “degree to which a product can replace another specified software product for the same purpose in the same environment”.

3.1.2 Quality Models and Ontologies

Casare *et al.*² developed a series of quality models to assess the trustworthiness of e-commerce websites. Quality models are modeled as syntactic trees where attributes and sub-attributes are spread over several levels and linked by operators. An example of such syntactic tree representation of quality models is represented in figure 3.2. Authors empirically validated their quality models by acquiring user-related data about perceived trustworthiness and by determining the quality model parameters that achieved the best prediction performances.

Olsina *et al.* [110, 111] have developed a meta-model for expressing the fundamental concepts of a GUI quality model. They suggested an ontology of quality views to improve quality models based on System and Software Quality Requirements and Evaluation (SQuaRE) frameworks such as the ISO-25010 standard. The proposed ontology for quality views is depicted in figure 3.3.

Although the meta-model provides an explicit semantic for the considered domain and provide useful strategy patterns to evaluate the quality focuses in several situations, the contribution remains highly conceptual and does not allow to easily evaluate the quality of concrete user interfaces in a systematic fashion.

²<https://www.atmosphere-eubrazil.eu/>

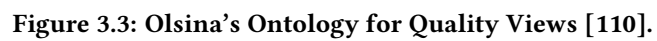


Figure 3.3: Olsina’s Ontology for Quality Views [110].

3.2 Explicit, Formal, and Calculable Definition of a Quality Model

In its generic form, a quality model can be described as a function expressed through the following formula:

$$Q_M : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{\mathbb{R}^l} = X \rightarrow Q \quad (3.1)$$

where the result returned by the quality model is a *quality function*, called Q and defined as follows:

$$Q : \mathbb{R}^l \rightarrow \mathbb{R} = x \rightarrow q \quad (3.2)$$

By definition, this formulation of quality models is formal and generic enough to encompass a large variety of existing models. Calculableness is a property that quality models defined by formula 3.1 can have, but it is not always guaranteed. We can indeed find quality models based on formula 3.1 that are not calculable, such as the following :

$$Q_M : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{\mathbb{R}^l} = X \rightarrow \begin{cases} Q_T & \text{if } Q_M \text{ halts on input } X \\ Q_F & \text{otherwise} \end{cases} \quad (3.3)$$

where Q_T and Q_F are two arbitrary quality functions following definition given by formula 3.2

Explicitness is not a property of the quality model as such but is rather linked to the whole process of model construction, including data collection and interpretation. The following sections provide several examples of quality models to illustrate the versatility of the proposed formulation.

3.2.1 Application to SUS Evaluation Method

Questionnaire-based methods such as SUS [25], CSUQ [78] or PSSUQ [79] can easily be expressed in terms of equations 3.1 and 3.2. For example, SUS methodology for a single questionnaire can be expressed as follows:

$$Q_{M-SUS} : \emptyset \rightarrow \mathbb{R}^{[0,7]^{10}} = \emptyset \rightarrow Q_{SUS} \quad (3.4)$$

where Q_{SUS} is the quality function defined as follows :

$$Q_{SUS} : [0, 7]^{10} \rightarrow \mathbb{R} = x \rightarrow q \quad (3.5)$$

The quality model Q_{M-SUS} does not take any argument as the quality function $Q_M - SUS$ is solely based on the questionnaire, hence the domain of the function defined as the empty set. The quality function Q_{SUS} is defined on the domain $\mathbb{N}^{10}$, corresponding to the feature space formed by the responses

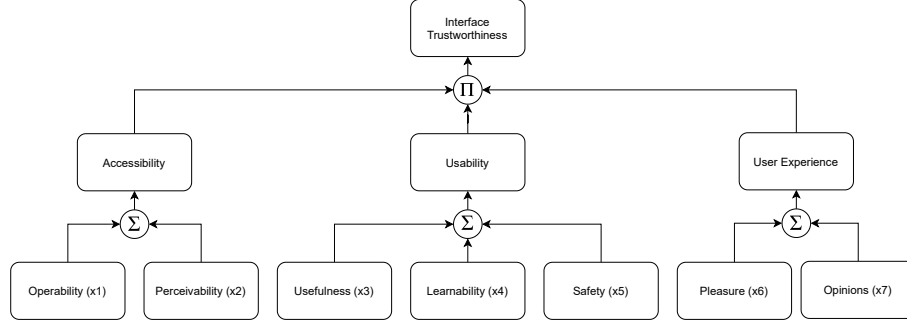


Figure 3.4: Example of SQuaRE-based quality model

to the 10 questions composing the SUS questionnaire. We can assume here a numerical encoding of the responses without loss of generality. The output of the quality function is a real number q corresponding to the score given by the SUS methodology.

The SUS methodology expressed in terms of equation 3.4 is formal by definition. SUS is also calculable as clear rules for computing the overall score are given in the methodology. Explicitness, however, is not guaranteed as this property depends on the documentation of the processes underlying the methodology. Explicitness of the SUS methodology based on a single questionnaire as in this example is trivial. However, we might find more complex situations in which explicitness might be harder to demonstrate.

3.2.2 Application to SQuaRE-based models

Quality models based on the approach proposed in the ISO-25010 standard can also be modeled in the formalism of equations 3.1 and 3.2. To illustrate this approach, we will consider a simplified version of the quality model of figure 3.2, depicted in figure 3.4.

Based on this, the SQuaRE-based model can be expressed as such :

$$Q_{M-SQuaRE} : \emptyset \rightarrow \mathbb{R}^{\mathbb{R}^7} = \emptyset \rightarrow Q_{SQuaRE} \quad (3.6)$$

where Q_{SQuaRE} is the quality function defined as follows :

$$Q_{SQuaRE} : \mathbb{R}^7 \rightarrow \mathbb{R} = (x_1, \dots, x_7) \rightarrow q = \sum_{i=1}^2 x_i \times \sum_{j=3}^5 x_j \times \sum_{k=6}^7 x_k \quad (3.7)$$

The SQuaRE-based model expressed in terms of equation 3.6 is formal by definition. It is also calculable as all the operands and operators of the syntactic tree are defined. However, in the most general case, we should ensure that all the operators implied in the model are calculable in order to achieve calculability of the model. Explicitness, again, is not guaranteed as this property

depends on the documentation of the processes underlying the methodology. Explicitness of the model given in figure 3.4 is again trivial as all the parameters and inputs of the model are defined, but more complex models could imply the explicit documentation of parameters and/or inputs of the model.

3.2.3 Application to machine-learning based quality model

Aside of questionnaire- and SQuaRE-based methodologies, other kinds of models can be expressed in formalism of equations 3.1 and 3.2, such as machine learning (ML) quality models. Let's consider a population of e-commerce websites from which we extract a sample of $N=50$ individuals. Let's also consider that we compute a series of 10 measures such as described in section 2.2 on the graphical representation of these 50 websites. We also gather and aggregate data about users' perception of visual appeal of these websites.

Based on this, a ML model can be expressed as follows:

$$Q_{M-ML} : \mathbb{R}^{50 \times 11} \rightarrow \mathbb{R}^{\mathbb{R}^{10}} = X_t \rightarrow Q_{ML} \quad (3.8)$$

where Q_{ML} is the quality function defined as follows:

$$Q_{ML} : \mathbb{R}^{10} \rightarrow \mathbb{R} = x \rightarrow q \quad (3.9)$$

In this case, the quality model represents the training procedure of a specific ML algorithm (i.e random forest, linear regression, etc). This procedure takes as argument a training set, denoted X_t and outputs a trained predictor, denoted Q_{ML} . The quality function Q_{ML} takes as input the feature vector of a particular sample point, which in this case would be the representation of a GUI in a feature space $\mathbb{R}^{11}$, and returns the predicted quality of the considered sample point. The ML quality model defined by equations 3.8 and 3.9 is formal and calculable. However, as for the questionnaire-based models, explicitness is not guaranteed by the sole formal definition of the model.

3.3 Our Framework for Experiment Replicability

Explicitness of quality models is not a property of the model itself or its formal definition. It is related to the clear definition and documentation of the various processes involved in the construction of the quality model and the interpretation of its results by the intermediate of a quality function. In that sense, explicitness is directly linked to the degree of replicability of quality models and their underlying processes.

Based on section 2.4 of chapter 2, it appears clearly that existing frameworks for classification of replicability have overlapping concepts that can't always be evaluated when applied to a given set of processes or studies.

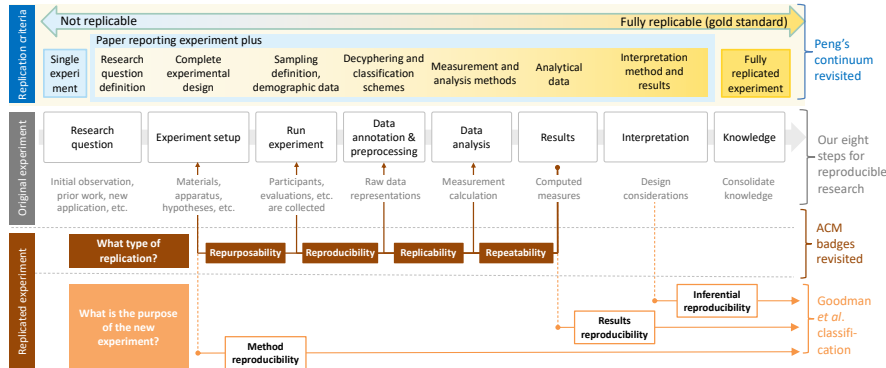


Figure 3.5: Our framework for experiment replicability.

Since the ACM definitions are not independent, we define four properties (see Figure 3.5, brown lane): “repeatability” (same team, same experimental setup), “replicability” (different team, different experimental setup), “reproducibility” (different team, same experimental setup), and “repurposability” (same team, different setup). Three original types of replication are recommended (see Figure 3.5, bottom part in brown): method reproducibility strives for making the raw data and the experimental setup totally explicit and accessible to repeat the same experiment; if the same results would be obtained by matching this setup, results reproducibility would be satisfied; and inferential reproducibility would reach to the same conclusion resulting from an independent replication or a reanalysis of the obtained results.

Finally, the upper blue part of Figure 3.5 depicts seven levels of possible replications by revisiting Peng’s continuum for fully replicable research [120]. Similarly, we argue that six types of experiment artefacts should become explicit and accessible to enable progressively more elaborated levels of replication, similarly to the three levels of W3C accessibility³. For this purpose, we introduce the following incremental *replicability criteria*:

- **Level A:** the complete experimental setup should be made explicit and accessible. Various techniques exist for this purpose, such as templates for research questions and hypotheses development, such as the FINER criteria in PICOT format [46] and preregistration of experiments [36].
- **Level B:** equals “Level A” plus the explicit definition of the population sampling, demographic data, the context of use, and the stimuli. All these data should be made available through electronic documents in

³See <https://www.w3.org/WAI/standards-guidelines/wcag/>

commonly used formats [34]. Instead of capturing data by picture or video (*e.g.*, taking a manual screenshot), it is advisable to actually record GUIs on the very device used, which may require a common, shareable format. In absence of a commonly used format, log files and raw data should be made available in any repository or in an accessible database.

- **Level C:** equals “Level B” plus the publicly available definition of any decyphering or classification scheme used for clustering GUIs into groups. The classification criteria and their potential values should be clearly defined without any overlapping. Again, a common software platform for managing GUIs and their evaluation would be welcome, like Galaxy for biomedical engineering [3], allowing the designer to semi- automatically explore and test alternate decyphering schemes, without the need to perform this classification manually.
- **Level D:** equals “Level C” plus the definition of the measurement and analysis methods used for analyzing data, *e.g.*, through the formalization of measurement methods, tools, formula, and units, a common practice in software quality assurance [17].
- **Level E:** equals “Level D” plus the accessibility of any manual or software tool for applying the measurement methods. For instance, a software could automatically compute all the required GUI measures and provide the designer with some interpretation, like in WebTango [63].
- **Level F:** equals “Level E” plus the interpretation scheme followed to derive the conclusion from analytical data. Note that a same measurement method, perhaps through different formula, may reach to comparable conclusions (confirmation) or not (disconfirmation).

A fully replicable quality model is obtained when “Level F” is reached, which is quite demanding but rewarding. Moreover, a quality model achieving full replicability is *de facto* explicit, formal, calculable and interpretable.

Software Support for Explicit Quality Models

4

4.1 Introduction

Replicability is a matter of habits and good practices on the part of the researcher, but it is also related to the tools and technologies that makes the processes efficient and realistic. Considering the experimental study of GUIs visual design, there exists several tools allowing to partially automate of the dataset construction process, as explained in section 2.3.15. These tools mainly focus on either computing measures on GUIs or gathering data about users evaluation of GUIs and a lot of tedious tasks such as building experiments, capturing screenshots or computing measures are left to the experimenter. Most of the tools available in the field of GUI visual design have been built with evaluation in mind, which makes them not fully suitable for to achieve full replicability of experimental studies and quality models.

As explained earlier, explicitness of quality models is a property related to the degree of documentation of their underlying processes for data collection and analysis. Explicitness of quality models implies a clear documentation but also a mean to easily apply this documentation in order to rebuild data collection and rerun analysis processes underlying quality models.

To that purpose, we developed a software tool, `UILAB`, aiming at (semi-)automating the dataset construction process for the establishment of quality models. The software allows the automated capture of graphical representation of UIs, the design and automated deployment of experiments related to the evaluation of GUIs and the automated computation of measures on GUIs in a flexible way by the mean of directed acyclic graphs (DAGs).

4.2 High-level Overview

`UILAB` is based on a hybrid architecture based on both client / server and serverless approaches and depicted in figure 4.1.

The back-end side of our application is built using Hasura¹, an "API-as-a-service" provider. The different concepts of figure 4.2 are implemented in the

¹<https://hasura.io/>

backend and are exposed through an GraphQL API.

The computation of measures is offloaded to the Amazon cloud by using the AWS Elastic Container Service² as well as AWS Lambda³. This services allows the on-demand creation of container-based computational instances. The front-end side of uiLAB is divided in two different applications. The first application allows experimenters to administrate their galleries of screen-shots, experiments, and workflows. The second application allows participants to take part to deployed experiments.

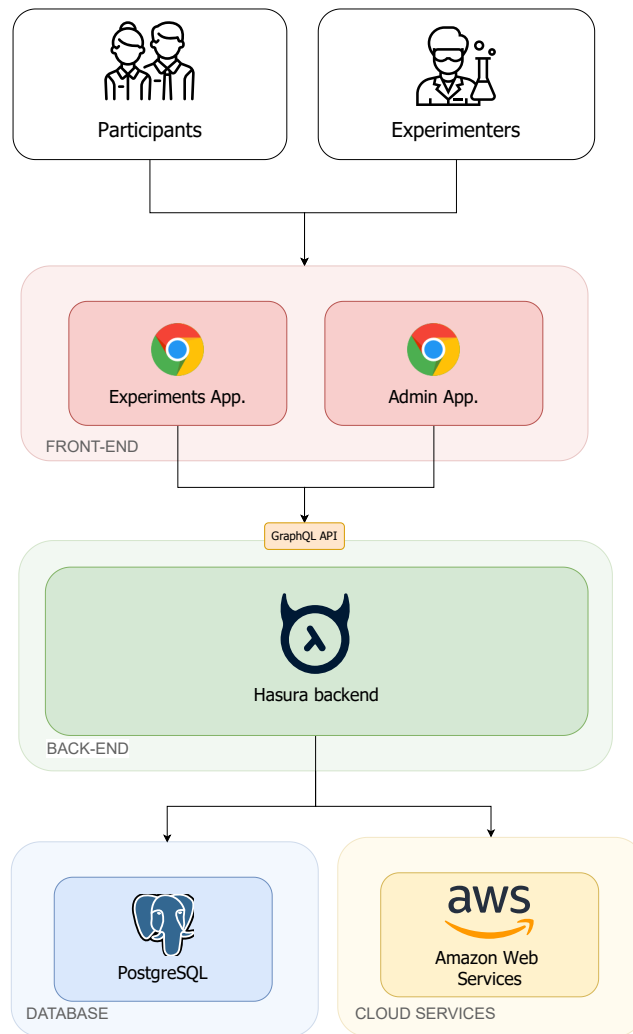


Figure 4.1: uiLAB- Overview of architecture.

²<https://aws.amazon.com/ecs/features/>

³<https://aws.amazon.com/fr/lambda/features/>

4.3 Conceptual Model

The application is designed based on a conceptual model, expressed as an Extended Entity-Relationship (EER) diagram [147], composed of several entities and concepts depicted in figure 4.2. A first set of entities is related to the concept of gallery and is represented in greater details in figure 4.3:

Gallery: A gallery is a set of screenshots that are logically grouped together. A gallery can be associated to 0, 1 or more experiments. Screenshots contained in the gallery will be subject to user evaluation in the experiments based on that gallery. Galleries can also be associated to workflows by the intermediate of “Run” entities. Galleries are characterized by the following attributes :

- *id (String)* : A randomly-assigned identifier.
- *title (String)* : A short text describing the gallery for quick identification.
- *description (String)* : A detailed textual explanation of the gallery and its content.

Screenshot: A screenshot is the captured graphical representation of a user interface. A gallery is composed of screenshots captured from different URLs and of various resolutions and densities. Screenshots are characterized by the following attributes :

- *id (String)* : A randomly-assigned identifier.
- *galleryId (String)* : Reference to the identifier of the gallery the screenshots belongs to.
- *websiteId (String)* : Reference to the identifier of the website associated to the screenshot
- *resolutionId (String)* : Reference to the identifier of the resolution associated to the screenshot
- *storageUrl (String)* : URL of the screenshot on the AWS S3 repository

Website: Website entities are simply representations of existing websites, allowing the creation of screenshots in association with resolution entities. Websites are characterized by the following attributes :

- *id (String)* : A randomly-assigned identifier.
- *name (String)* : Short textual description of the website.

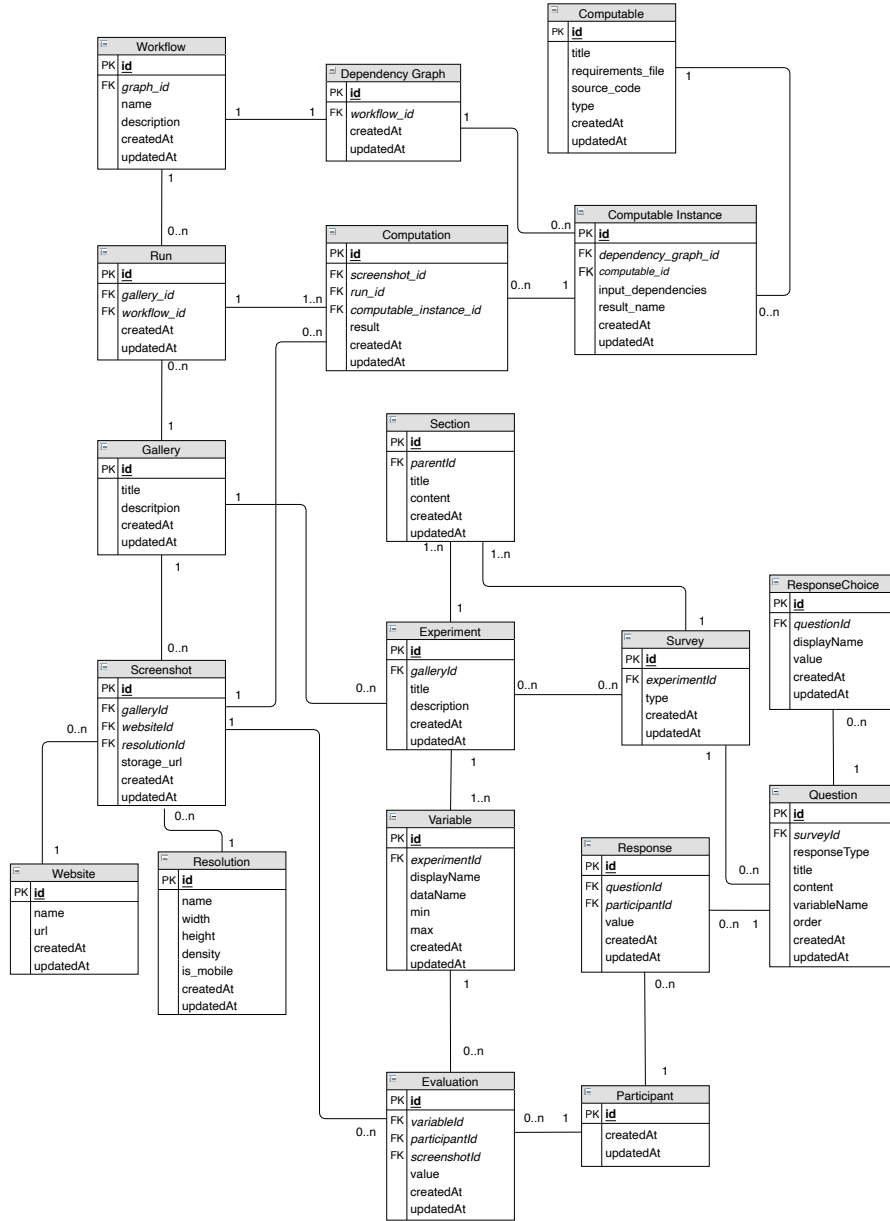


Figure 4.2: uiLAB- Extended Entity-Relationship (EER) conceptual model.

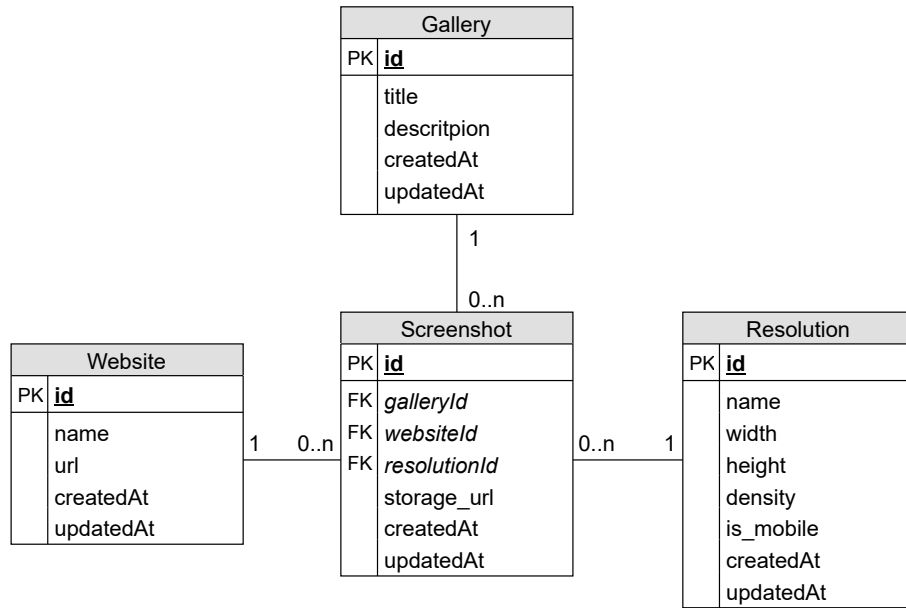


Figure 4.3: uiLAB- EER model of galleries and related concepts

- url (*String*) : A complete URL of the considered web page.

Resolution: Resolution entities are representations of resolutions, allowing the creation of screenshots in association with website entities. Resolutions are characterized by the following attributes :

- id (*String*) : A randomly-assigned identifier.
- name (*String*) : Short textual description of the resolution.
- width (*Integer*) : Width of the resolution (in pixels).
- height (*Integer*) : Height of the resolution (in pixels).
- density (*Float*) : Device Pixel Ratio (DPR) of the resolution.
- is_mobile (*Boolean*) : A boolean variable indicating if the resolution has to be captured with a mobile User-Agent or not.

Another logical set of entities in uiLAB is related to the concept of experiments and is represented in figure 4.4 :

Experiment: The experiment encompasses the elements that are required to design and deploy an online experiment allowing users to evaluate screenshots of the associated gallery. An experiment can be associated to different sections, surveys, and variables. An experiment must be

associated to one and only one gallery. The following attributes characterize an experiment in uILAB:

- `id (String)` : A randomly-assigned identifier.
- `galleryId (String)` : Reference to the identifier of the gallery associated to the experiment
- `title (String)` : Short textual description of the experiment.
- `description (String)` : A detailed textual explanation of the experiment and its content.

Section: A section is a paragraph of text displayed to the user during an experiment. It can be used to introduce the experiment, explain the objectives of the experiment to the participant, etc. A section must be part of an experiment or a survey. Sections are represented by the following attributes :

- `id (String)` : A randomly-assigned identifier.
- `parentId (String)` : Reference to the identifier of the parent (either experiment or survey) associated to the section.
- `title (String)` : Title of the section that will be displayed as header.
- `content (String)` : Content of the section that will be displayed as a paragraph of text.

Survey: A survey is a step of an experiment used to gather information about participants. For example, it can be used to collect socio-demographical information such as age, gender, etc. Experiments have 2 surveys (a pre-test survey occurring before the experiment and a post-test survey occurring after the experiment). A survey is characterized a set of sections and a set of questions to be asked to the participants, along with the following attributes :

- `id (String)` : A randomly-assigned identifier.
- `experimentId (String)` : Reference to the identifier of the experiment associated to the section.
- `type (String)` : Either “PRE_TEST” or “POST_TEST” to indicate where the survey is located in the course of an experiment.

Question: This entity represents a question asked to the participant in the context of a survey. It is characterized by the following attributes :

- `id (String)` : A randomly-assigned identifier.

- `surveyId (String)` : Reference to the identifier of the survey associated to the question.
- `title (String)` : Title of the section that will be displayed as header.
- `content (String)` : Content of the section that will be displayed as a paragraph of text.
- `variableName (String)` : Name of the variable captured by the question. This name should appear as field name when data are extracted from uILAB in JSON format.
- `order (Integer)` : The order of the question in the survey.

Response Choice: A response choice represents the different items proposed to the user as a possible answer in the context of a multiple-choice question, usually displayed as drop-down lists or radio buttons. A response choice is composed of :

- `id (String)` : A randomly-assigned identifier.
- `questionId (String)` : Reference to the identifier of the survey associated to the question.
- `displayName (String)` : A clear textual description of the response item (e.g., “Male”, “Strongly agree”, etc.)
- `value (String)` : The actual value representing the response that can either be textual or encoded in numerical format (e.g., M, 5, etc).

Response: A response corresponds to a participant’s answer to a question of a survey. It is represented by the following attributes :

- `id (String)` : A randomly-assigned identifier.
- `questionId (String)` : Reference to the identifier of the survey associated to the question.
- `participantId (String)` : Reference to the identifier of the participant associated to the question.
- `value (String)` : The actual value representing the response that can either be textual or encoded in numerical format (e.g., M, 5, etc).

Variable: This entity represents a variable that an experimenter wants to be evaluated by participants in the context of an experiment. Several variables may be evaluated for each screenshot during an experiment. For example, an experimenter could ask participants to evaluate both the complexity and the beauty of a user interface, both being represented

by a variable during the configuration of the experiment by the experimenter. Variables are displayed to participants as a Likert scale [83] with minimum and maximum values enforced by the application. Variables are characterized by the following attributes :

- `id (String)` : A randomly-assigned identifier.
- `experimentId (String)` : Reference to the identifier of the survey associated to the question.
- `displayName (String)` : A clear textual description of the variable (e.g., “Visual appeal”).
- `dataName (String)` : Name of the variable that should appear in datasets (e.g., “var_visual_appeal”).
- `minValue (Integer)` : Minimum value authorized for the variable in the context of an evaluation.
- `maxValue (Integer)` : Maximum value authorized for the variable in the context of an evaluation.

Participant: A participant represents an individual taking part to experiments in the application. As we do not establish user profile outside of data collected through surveys and experiments, the actual “Participant” entity is solely characterized by an “id” field.

Evaluation: An evaluation represents the evaluation of a specific screenshot by a participant regarding a specific variable in the context of an experiment. As experimenters can ask participants to evaluate several variables for each screenshot during an experiment, it can lead to several evaluations per screenshot, one for each variable evaluated. An evaluation is characterized by the following properties :

- `id (String)` : A randomly-assigned identifier.
- `variableId (String)` : Reference to the identifier of the variable associated to the evaluation.
- `userId (String)` : Reference to the identifier of the user associated to the evaluation.
- `screenshotId (String)` : Reference to the identifier of the screenshot associated to the evaluation.
- `value (Integer)` : Actual value given by the user to the considered screenshot for a particular variable.

The last set of entities in ULAB is related to the concept of workflow and computation. These entities are represented in figure 4.4 :

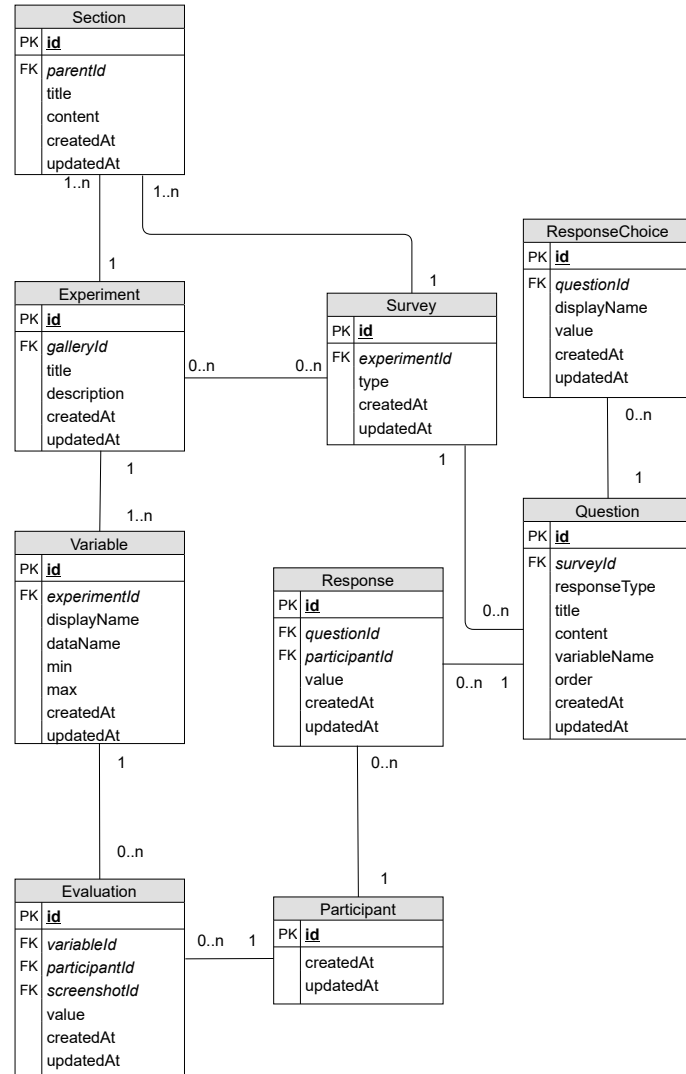


Figure 4.4: uLAB- EER model of experiments and related concepts.

Run: A run corresponds to the request of a user to compute a given workflow on a gallery. Each time a user asks the application to compute a workflow on a gallery, a run entity is created. A run is characterized by the following fields :

- `id (String)` : A randomly-assigned identifier.
- `galleryId (String)` : Reference to the identifier of the gallery associated to the run.
- `workflowId (String)` : Reference to the identifier of the workflow associated to the run.
- `status (String)` : Status of the run, being either “IDLE”, “IN_PROGRESS” or “FINISHED”.

Computation: A computation represents the actual computation of a computable instance on a specific screenshot. This entity is characterized by :

- `id (String)` : A randomly-assigned identifier.
- `screenshotId (String)` : Reference to the identifier of the gallery associated to the run.
- `runId (String)` : Reference to the identifier of the workflow associated to the run.
- `instanceId (String)` : Reference to the identifier of the workflow associated to the run.
- `status (String)` : Status of the run, being either “IDLE”, “CHECK”, “LAUNCHABLE”, “RUNNING”, “COMPLETED” or “ERROR”.
- `value (String)` : The actual value of the computation result.

Workflow: A workflow represents a set of computable instances grouped together that can be computed on screenshots belonging to a gallery. The computation of a workflow on a given gallery implies the computation of all the computable instances contained in the workflow on all screenshots contained in the gallery. For each screenshot, the order of execution of computable instances is determined by the dependency graph associated to the workflow. A workflow is composed of :

- `id (String)` : A randomly-assigned identifier.
- `name (String)` : Short textual description of the workflow.
- `description (String)` : A detailed textual explanation of the workflow.

Dependency Graph: A dependency graph is an entity containing the dependencies between the computable instances of a workflow. A dependency graph indicates which functions are to be computed on a given gallery and in which order these functions must be computed.

- `id (String)` : A randomly-assigned identifier.
- `workflowId (String)` : Reference to the identifier of the workflow associated to the dependency graph.

Computable: A computable represents an object that can be computed on screenshots or derived results. A computable can be associated to workflows by the intermediate of their dependency graph. A given dependency graph can be associated multiple times with the same computable by adding several instances of the computable to it. A computable can take several parameters as input and returns a result. This result will be passed to subsequent computable instances according to the dependency graph of the workflow. Not all computables are computed directly on screenshots as some may require intermediate computations in order to be executed with the rights parameters. A computable instance is characterized by :

- `id (String)` : A randomly-assigned identifier.
- `name (String)` : Short textual description of the computable.
- `description (String)` : A detailed textual explanation of the computable.
- `sourceCodeURL (String)` : A string containing the URL of the source code of the computable.

Computable Instance: A computable instance is an entity representing the actual integration of a computable into a workflow by the mean of a dependency graph. A dependency graph can have multiple computable instances. Computables can be related to multiple computable instances, in different dependency graphs but also inside the same dependency graph. A computable instance is represented by :

- `id (String)` : A randomly-assigned identifier.
- `dependencyGraphId (String)` : Reference to the identifier of the dependency graph associated to the instance.
- `computableId (String)` : Reference to the identifier of the computable associated to the instance.
- `resultName (String)` : Name under which the results of the computations associated to the instance should appear in a dataset.

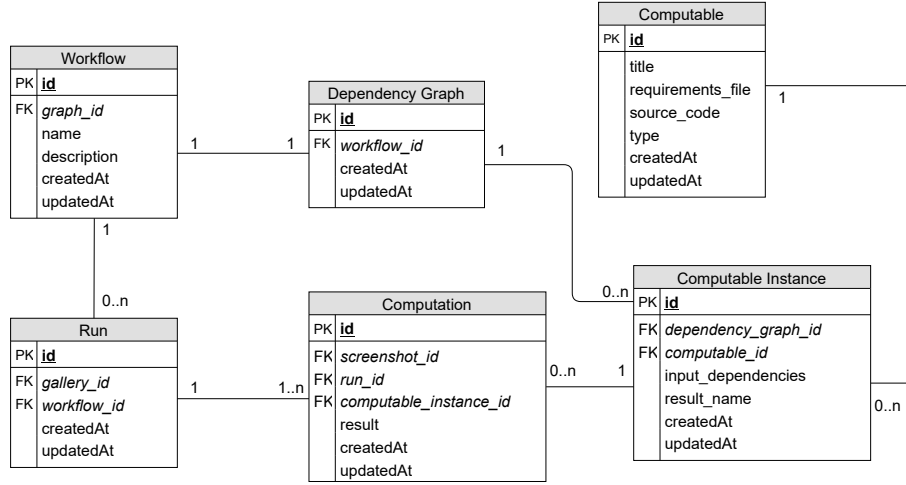


Figure 4.5: UILAB- Table for workflows and related concepts.

- *isTransient (Boolean)* : Boolean value indicating whether or not computations associated to the instance should be kept after run completion or not. This flag is typically used for intermediate results in order to delete them upon run completion.

4.4 System Walkthrough

4.4.1 Construction of Galleries: Automation of the Screenshots Capture Process

In UILAB, experimenters have to create a gallery prior to any experiment design or measure computation. Experimenters have to specify the title and description of the gallery (Figure 4.8). They also have to specify the set of websites and resolutions included in the gallery.

Upon validation by the user, the backend of UILAB creates the various entities (i.e gallery and screenshots) and triggers a task on the AWS ECS service for each screenshot in order to capture it. These tasks are docker containers running custom images illustrated by figure 4.6. This image is based on a basic NodeJS image (Figure 4.6, line 1) with Chromium⁴, Puppeteer⁵ and other required libraries installed (Figure 4.6, lines 5 to 11). Upon start, the container will run JavaScript code (Figure 4.6, line 17) depicted in figure 4.7.

The script imports modules and initializes a set of variables (Figure 4.7, lines 1-28) and launches a headless chromium window with the specified

⁴<https://www.chromium.org/>

⁵<https://pptr.dev/>

```
1 FROM node:alpine
2
3 WORKDIR /app
4
5 RUN apk add --no-cache udev ttf-freefont chromium git
6 ENV PUPPETEER_SKIP_CHROMIUM_DOWNLOAD true
7 ENV CHROMIUM_PATH /usr/bin/chromium-browser
8
9 COPY ./package*.json ./
10
11 RUN npm install
12
13 COPY . .
14
15 EXPOSE 3000
16
17 CMD ["npm", "run", "start"]
```

Figure 4.6: uILAB: Docker image for screenshot capture

User-Agent and resolution for screenshot capture (Figure 4.7, lines 30-44). It then navigates automatically to the specified URL and captures a screenshot of the website (Figure 4.7, lines 46-52). Then, the script uploads the captured screenshot to the AWS S3 repository of the corresponding gallery and updates the status and storage URL of the screenshot in the backend (Figure 4.7, lines 54-73). Once the screenshots are captured, the experimenter manages them in the corresponding gallery page (Figure 4.9b). Experimenters can update the gallery information, add or remove websites and resolution and they can export the gallery and associated screenshots into a single file in order to facilitate the sharing of information.

By automating the capture of screenshots and allowing the import / export of galleries, uILAB makes the process of gallery construction explicit. Indeed, experimenters can easily import galleries of other persons to replicate experimental conditions of screenshot capture, reducing errors and variability due human intervention in the screenshot capture process.

4.4.2 Design and Deployment of Experiments

uILAB implements a workflow that enables experimenters to design their experiments according to a generic template and deploy them automatically. Deployed experiments are readily available for users and data collection does not require long design and development phases. In addition to that, experimental setups can be saved for further reuse in experiments of similar configuration. Before designing an experiment, an experimenter must create one or more surveys to be incorporated in the experiment as pre- or post-test survey. The experimenter must provide the name, description of the survey

```

1  const puppeteer = require('puppeteer');
2  const AWS = require('aws-sdk');
3  const fetch = require('node-fetch');
4  const util = require('util');
5
6  const MOBILE_USER_AGENT =
7    'Mozilla/5.0 (iPhone; CPU iPhone OS 10_3 like Mac OS X) AppleWebKit/602.1.50 (KHTML, like Gecko)
8    CriOS/56.0.2924.75 Mobile/14E5239e Safari/602.1';
9
10 const DESKTOP_USER_AGENT = 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:69.0) Gecko/20100101 Firefox/69.0';
11
12 const updateScreenshotStatusMutation = `
13   mutation updateScreenshot($id: uuid!, $storage_url: String) {
14     update_screenshots_by_pk(pk_columns:{id: $id}, _set:{ status: "CAPTURED", storage_url: $storage_url}){
15       id
16     }
17   }
18 `;
19
20 const captureScreenshot = async () => {
21   const resolution = JSON.parse(process.env.RESOLUTION);
22   const url = process.env.WEBSITE_URL;
23   const galleryId = process.env.GALLERY_ID;
24   const screenshotId = process.env.SCREENSHOT_ID;
25   const waitTime = process.env.WAIT_TIME || 5000;
26   console.log('Creating credentials');
27   const credentials = new AWS.Credentials({ accessKeyId: process.env.AWS_ACCESS_KEY, secretAccessKey:
28     process.env.AWS_SECRET_KEY });
29   const s3 = new AWS.S3({ credentials });
30
31   console.log('Capturing screenshot');
32   const browser = await puppeteer.launch({
33     headless: true,
34     executablePath: process.env.CHROMIUM_PATH,
35     args: ['--no-sandbox', '--disable-setuid-sandbox', '--disable-gpu', '--disable-dev-shm-usage'],
36   });
37
38   const page = await browser.newPage();
39
40   await page.setUserAgent(resolution.is_mobile ? MOBILE_USER_AGENT : DESKTOP_USER_AGENT);
41
42   await page.setViewport({
43     width: resolution.width,
44     height: resolution.height,
45     deviceScaleFactor: resolution.density,
46   });
47
48   await page.goto(url);
49
50   console.log('Waiting for ${waitTime / 1000} seconds');
51   await page.waitForTimeout(waitTime);
52
53   const capturedScreenshot = await page.screenshot();
54
55   await browser.close();
56
57   console.log('Saving screenshot');
58   const screenshotData = Buffer.from(capturedScreenshot);
59
60   await s3
61     .putObject({ Bucket: process.env.S3_BUCKET, Key: `galleries/${galleryId}/${screenshotId}.png`, Body:
62       screenshotData, ACL: 'public-read-write' })
63     .promise();
64
65   fetch(process.env.HASURA_ENDPOINT + '/v1/graphql', {
66     method: 'POST',
67     body: JSON.stringify({
68       query: updateScreenshotStatusMutation,
69       variables: {
70         id: process.env.SCREENSHOT_ID,
71         storage_url: `https://uilab-hasura.s3.eu-west-1.amazonaws.com/galleries/${galleryId}/${screenshotId}.png`,
72       },
73       headers: { 'Content-Type': 'application/json', 'x-hasura-admin-secret': process.env.HASURA_ADMIN_SECRET },
74     }).then((res) => {
75       console.log(res);
76       return res;
77     });
78   captureScreenshot();

```

Figure 4.7: uiLAB: Javascript script for screenshot capture

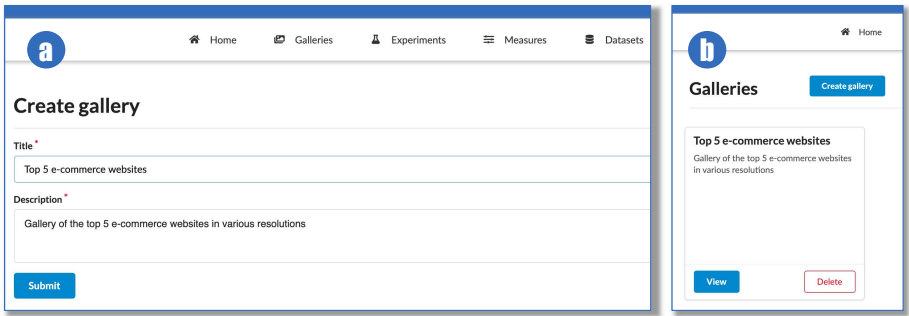


Figure 4.8: uiLAB: (a) Gallery creation screen, (b) Galleries overview screen.

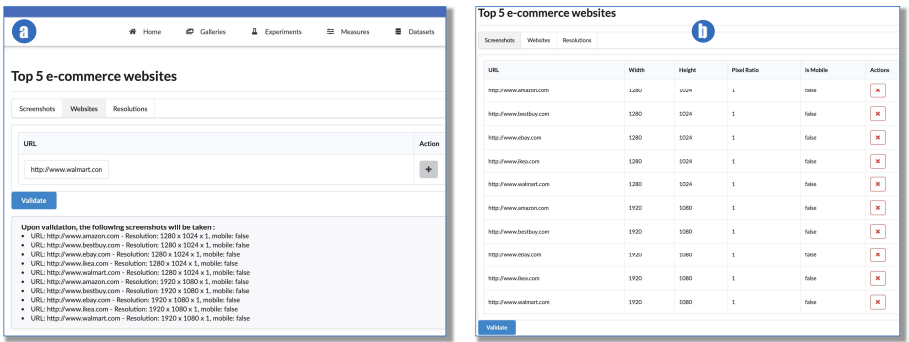


Figure 4.9: uiLAB: (a) Screenshots form screen, (b) Screenshots overview screen.

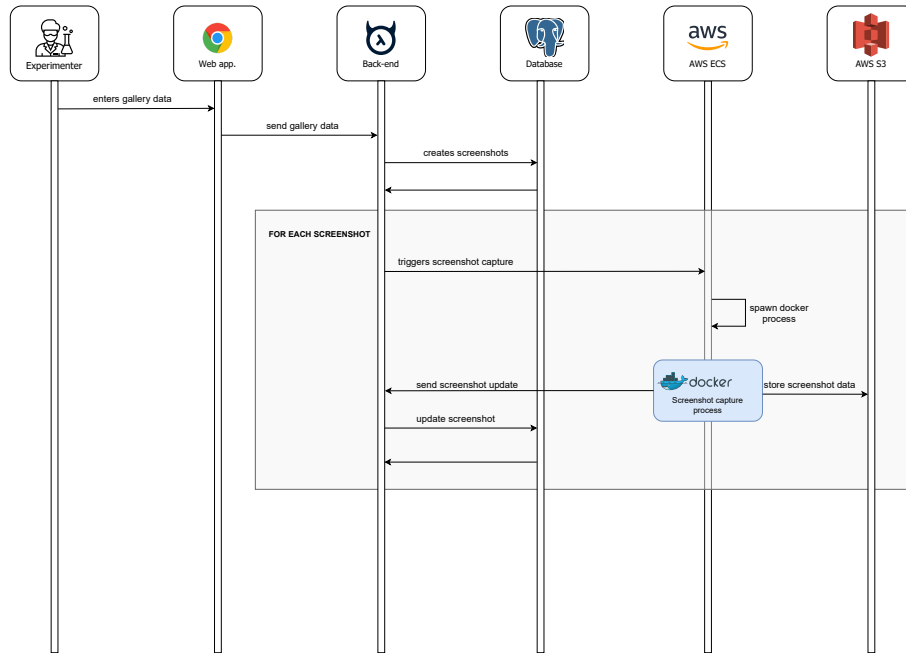


Figure 4.10: uILAB: Gallery creation and screenshot capture - sequence diagram

along with the set of sections and questions to be asked during the survey. In order to create an experiment, the experimenter has to provide all the details of the experiment, namely the title, the description, the gallery associated the experiment, the different sections of the experiment, the pre- and post-test surveys and finally the variables to be measured during the experiment (Figures 4.11a to 4.11g). Upon validation, the experiment data are sent to the back-end that creates the corresponding entities. After creation by the back-end, the experiment is deployed and ready to be shared with participants. A link to each experiment is provided to the experimenter on the experiments summary page (Figure 4.11h).

As explained in chapter 3, quality models often make use of user-related data to evaluate GUI visual design. By allowing users to design experiments in a straightforward way with the help of a predefined template and by allowing experimenters to share their experimental setup through metadata, uILAB improves the explicitness of user-data collection processes, which in turns helps improving the replicability of quality models.

4.4.3 Participation to Deployed Experiments

uILAB allows users to participate to experiments that have been designed and deployed by experimenters. Aside of the web application available for exper-

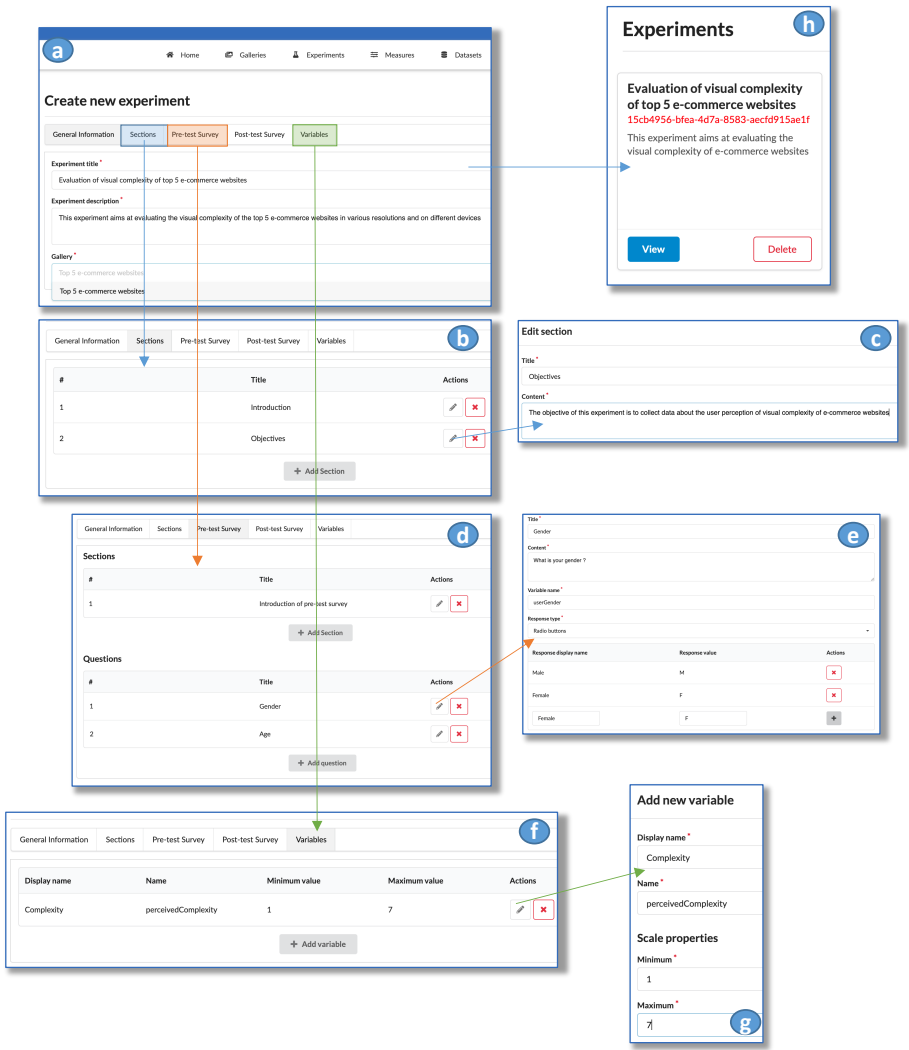


Figure 4.11: UiLAB: Experiment creation process.

imenters, there is an application dedicated to participants that allows them to take part in experiments (see figure 4.1). Each deployed experiment has its own URL and users only need the URL to participate to an experiment. Experiment flow in uILAB follows the one depicted in figure 4.12. When visiting the URL of an experiment, participants are provided with the introductory sections as specified by experimenter at design time (Figures 4.13a and b). Then, participants have the opportunity to answer the pre-test survey (Figures 4.13c to e). After answering the questions of the pre-test survey, participants are invited to take part in the experiment by evaluating the screenshots of the gallery specified by the experimenter (Figure 4.13f). Finally, once all screenshots have been evaluated, participants are invited to answer questions of the post-test.

4.4.4 Incremental Development of New Measures

uILAB allows users to define measures that can be computed over GUIs. Measures such as balance or equilibrium [106] require intermediate computations (e.g., the segmentation of a UI into zones) that are not measures, strictly speaking. For this reason, uILAB defines the broader concept of *computable*. Users can create computables in uILAB by providing the source code and dependencies file (Figures 4.14a and 4.14b). These measures must implement a uniform interface (such as measures in AIM [115]) and illustrated in figure 4.15.

Once created, a computable can be associated to a workflow for later computation on galleries of screenshots. However, computables are not directly added to workflows. Instead, uILAB uses the concept of *computable instance*. Computable instances are organized in a workflow by the intermediate of a dependency graph, which is a Directed Acyclic Graph (DAG) indicating the dependencies between the computable instances inside a workflow.

When creating a computable instance, the user has to specify the underlying computable, the name of the output (under which it should appear in a data set), and the list of dependencies with, for each dependency, the related parameter name in the source code of the computation instance being created. This parameter mapping is required due to the fact that several computable instances may have a particular computable instance as common dependency, while expecting the output of that dependency under different parameter names.

uILAB provides the ability to automatically compute an arbitrary number of measures that can be added at runtime without modifying the source of the application. It also allows to reuse measures created by other researchers provided that they implement a specific interface. These features improve the explicitness, efficiency and scalability of the measure computation process.

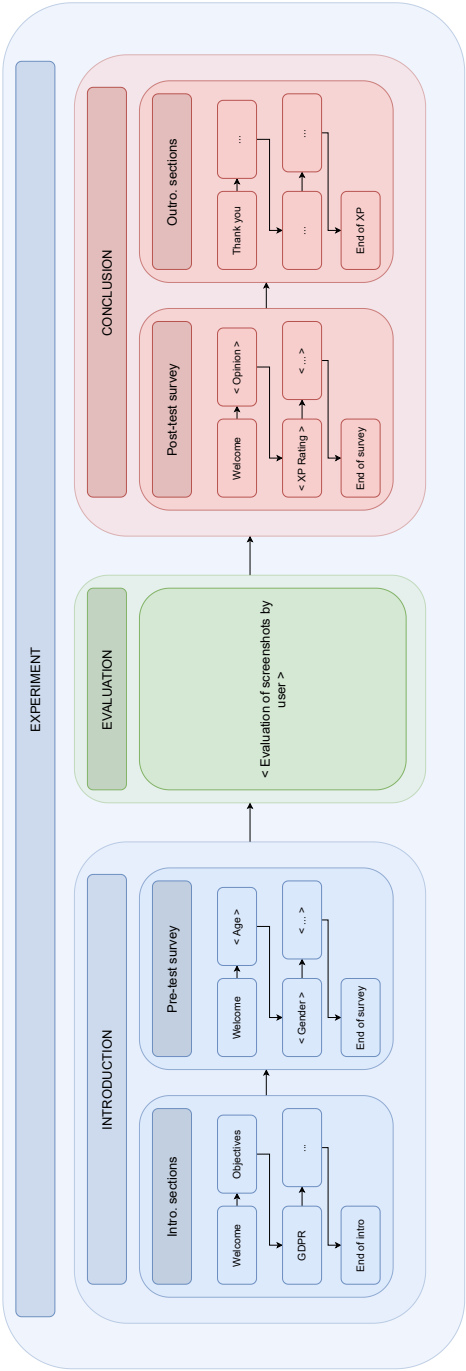


Figure 4.12: uiLAB- Example of experiment flow.

a

Introduction

Welcome to this experiment, please proceed to the next section.

Next

b

Objectives

The objective of this experiment is to collect data about the user perception of visual complexity of e-commerce websites

Back Next

c

Pre-test survey

Please answer the following questions about your profile.

Next

d

Age

What is your age?

Validate

e

Gender

What is your gender?

☐ Male

☐ Female

Validate

f

amazon

Today's Deals Help Registry Gift Cards Sell Your Amazon.com

Sign in Hello, Sign in Account & Lists Orders & Returns Today's Deals

We ship internationally. We're showing you items that ship to [Country] to see items that ship to a different country, change your delivery address. Additional language and currency settings are available. [Learn more](#)

Don't Change Change Address

Welcome to Amazon.com

We ship over 45 million products around the world.

Shop by Category

Computers & Accessories Video Games Baby Toys & Games

Shop now

Like-new Cellphones

See more

AmazonBasics

See more

Sign in for the best experience

Sign in securely

Discover Amazon [Click to learn more](#)

Deals and Shop in Shop in Secure Estimated Track your

Complexity

Validate

Figure 4.13: uILAB- Experiment participation process.

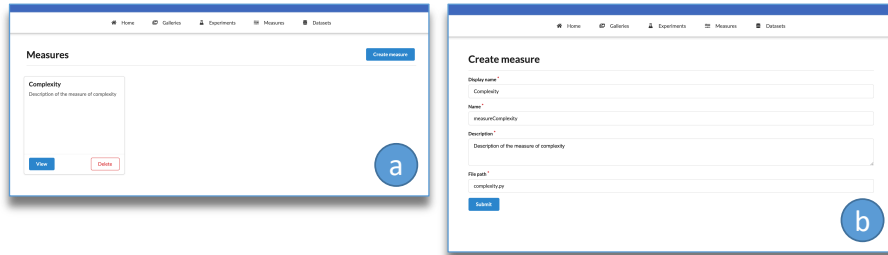


Figure 4.14: uiLAB: Measure creation process.

```

1 from skimage import util
2 import numpy as np
3 import base64
4 from PIL import Image
5 from io import BytesIO
6
7
8 def execute(**kwargs):
9     b64 = kwargs['b64_jpg']
10    b64 = base64.b64decode(b64)
11    b64 = BytesIO(b64)
12    img = Image.open(b64)
13    img = np.array(img)
14    img = util.img_as_byte(img)
15    img = img.reshape(-1, 3)
16    img = [tuple(l) for l in img]
17
18    lum = []
19    for pixel in img:
20        # Based on: https://en.wikipedia.org/wiki/Luma_(video)
21        y = 0.2126 * pixel[0] + 0.7152 * pixel[1] + 0.0722 * pixel[2]
22        lum.append(y)
23
24    result = np.std(lum)
25
26    return [result]

```

```

1 def execute(**kwargs):
2
3     b64 = kwargs['b64']
4     png_size = int(len(b64) * 0.75)
5
6     return [png_size]

```

```

1 def execute(**kwargs):
2
3     pngB64 = kwargs['b64']
4
5     pngB64 = base64.b64decode(pngB64)
6     pngB64 = BytesIO(pngB64)
7     img = Image.open(pngB64)
8     rgb_img = img.convert('RGB')
9     jpgBuffered = BytesIO()
10    rgb_img.save(jpgBuffered, format='JPEG')
11    jpg_encoded_data = base64.b64encode(jpgBuffered.getvalue())
12    jpgB64 = jpg_encoded_data.decode('utf-8')
13
14    return jpgB64

```

```

1 from skimage import util
2 import collections
3 import numpy as np
4 import base64
5 from PIL import Image
6 from io import BytesIO
7 import matplotlib.image as mpimg
8
9
10 def execute(**kwargs):
11
12     b64 = kwargs['b64']
13     type = int(kwargs['type'])
14
15     b64 = base64.b64decode(b64)
16     b64 = BytesIO(b64)
17     img = mpimg.imread(b64, format='PNG')
18     img = img[:, :, :3]
19     img = img.reshape(-1, 3)
20     img = [tuple(l) for l in img]
21
22     # Create histogram
23     hist = collections.Counter(img)
24
25     # Count the number of colors: only values that occur more than 5 (for web, type=0) or 2 (for mobile, type=1)
26     # times per image are counted
27     min_threshold = 5 if type == 0 else 2 if type == 1 else 0
28     count_rgb = [len({x: hist[x] for x in hist if hist[x] >= min_threshold})]
29
30     return count_rgb

```

Figure 4.15: uiLAB: Examples of measures: Luminance Std. dev (top-left); PNG to JPG (middle-right); PNG file size (top-right); Number of colors (bottom)

4.4.5 Computation of Workflows on Galleries

The computation of measures on GUIs in uILAB is performed through the creation of runs. When creating a run, the user has to specify the underlying workflow and gallery that will be associated to the run. The backend will then create a computation entity for each pair of computable instance and screenshot that will contain the computation result.

The life cycle of each computation can be described by a finite state machine such as illustrated in Figure 4.16(A). Figure 4.16 also illustrates the flow between computations as their state changes. The different states are as follows:

IDLE : When a run is first created, all the associated computations are set in the **IDLE** state. After creating all the required computations, the backend sets all of computations without dependencies to the **CHECK** state. The screenshot is provided as input to all the computations and is not considered as a dependency.

CHECK : When a computation enters the **CHECK** state, the uILAB backend verifies that all of its dependencies are in the **COMPLETED** state, if any. If so, the backend puts the computation in the **LAUNCHABLE** state, otherwise it puts the computation back in the **IDLE** state.

LAUNCHABLE : When a computation enters the **LAUNCHABLE** state, the backend aggregates the result of the computation of its dependencies in a file that is made available online on AWS S3. The backend then triggers the creation of an AWS ECS task. This task will run a Docker container that will gather all the required information for the computation (i.e the parameters previously aggregated, the considered screenshot and computable). When the computation finishes or if an error occurs, the backend will put the computation in either **COMPLETED** or **ERROR** state respectively, along with the result of the computation if applicable.

COMPLETED : When a computation is marked as **COMPLETED**, the backend of uILAB puts all of the computations depending on it in the **CHECK** state. This is illustrated by the green arrows of Figure 4.16.

ERROR : When a computation is marked as **ERROR**, the backend of uILAB puts all of the computations depending on it in the **ERROR** state, leading to a chain reaction of error propagation across computations. This is illustrated by the red arrows of Figure 4.16.

The computation of a measure on a screenshot is based on a custom Docker image which depends on the language used by the underlying mea-

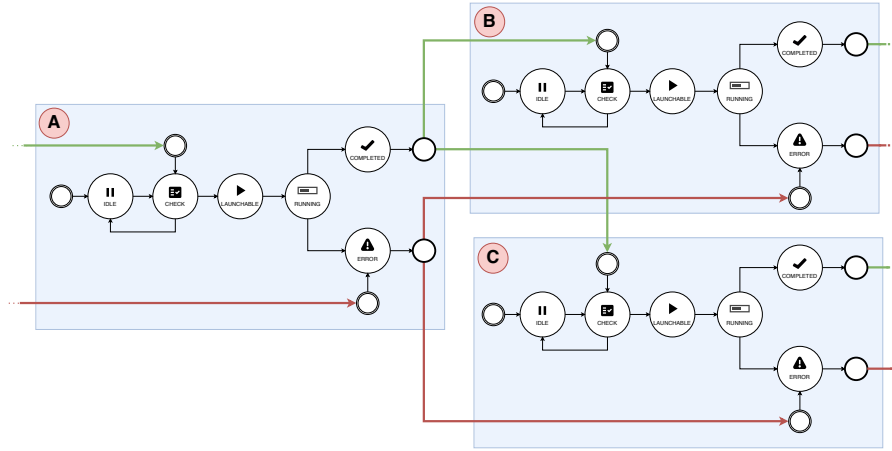


Figure 4.16: uILAB computations: Execution flow.

sure. The content of the Docker image for measures written in Python is depicted in figure 4.17. The script installs required packages (figure 4.17, lines 4-5), copies the script files used to launch the measure computation (figure 4.17, lines 7-11) and specifies the entry point of the image (figure 4.17, line 14).

The entry point of the docker image, *i.e.*, the command executed when the container starts, is the `main.sh` script depicted in figure 4.18. The script downloads the various resources required to compute the measure (figure 4.18, lines 7-10), It then uncompresses the source code of the measure, installs its dependencies and runs the `main.py` file illustrated by figure 4.19 (figure 4.18, lines 12-20)

The python script actually computing the measures starts by loading the parameters, the screenshot and the related properties (figure 4.19, lines 14-21). It then actually run the computation of the measure with the right parameters (figure 4.19, line 23) and updates the back-end with the result of the computation or an error message depending on the outcome of the computation.

As uILAB allows experimenters to upload any measure complying with a specific interface, it has not influence on the spatial or temporal complexity of the measure itself. Example of execution times for several measures compatible with uILAB can be found in [115].

```
1 FROM python:3.7
2
3 # Libraries required to run OpenCV
4 RUN apt-get update
5 RUN apt-get install ffmpeg libsm6 libxext6 -y
6
7 COPY ./main.sh .
8 WORKDIR /app
9 COPY ./app/main.py .
10 WORKDIR /
11 RUN chmod +x ./main.sh
12 EXPOSE 3000
13
14 CMD ["./main.sh"]
```

Figure 4.17: uiLAB- Docker image for measure computation

```
1 #!/bin/sh
2
3 cd /app
4
5 mkdir metric
6
7 curl -o /app/metric.zip $COMPUTABLE_SOURCE_URL
8 curl -o /app/screenshot.png $SCREENSHOT_URL
9 curl -o /app/parameters.json $PARAMETERS_URL
10
11 unzip metric.zip -d /app/metric
12
13 mv /app/metric/requirements.txt /app
14
15 touch /app/metric/__init__.py
16
17 rm /app/metric.zip
18
19 python -m pip install -r requirements.txt
20 python -m pip install requests
21
22 python main.py
```

Figure 4.18: uiLAB- main shell script for measure computation

```
1 import base64
2 from metric.metric import execute
3 import traceback
4 import os
5 import json
6 import requests
7
8 print("Python script starting")
9
10 with open("screenshot.png", "rb") as image_file:
11     with open("parameters.json", "r") as parameters_file:
12
13         try:
14             print("Reading params")
15             params = json.load(parameters_file)
16
17             print("Encoding file as base64")
18             encoded_string = base64.b64encode(image_file.read())
19
20             screenshot_data = os.environ['SCREENSHOT_DATA']
21
22             params['b64'] = encoded_string
23             params['screenshot'] = screenshot_data
24
25             print("Computing measure...")
26             r = execute(**params)
27
28             result = {}
29             result['data'] = json.dumps(
30                 {"id": os.environ['COMPUTATION_ID'], "result": "{}".format(r)})
31             result['status'] = 'FINISHED'
32             x = requests.post(os.environ['CALLBACK_URL'], json = result)
33
34         except Exception as err:
35             result = {}
36             result['data'] = json.dumps(
37                 {"id": os.environ['COMPUTATION_ID'], "result": "error"})
38             result['status'] = 'ERROR'
39             x = requests.post(os.environ['CALLBACK_URL'], json = result)
40
41     print(str(traceback.format_exc()))
```

Figure 4.19: uLAB- main python script for measure computation

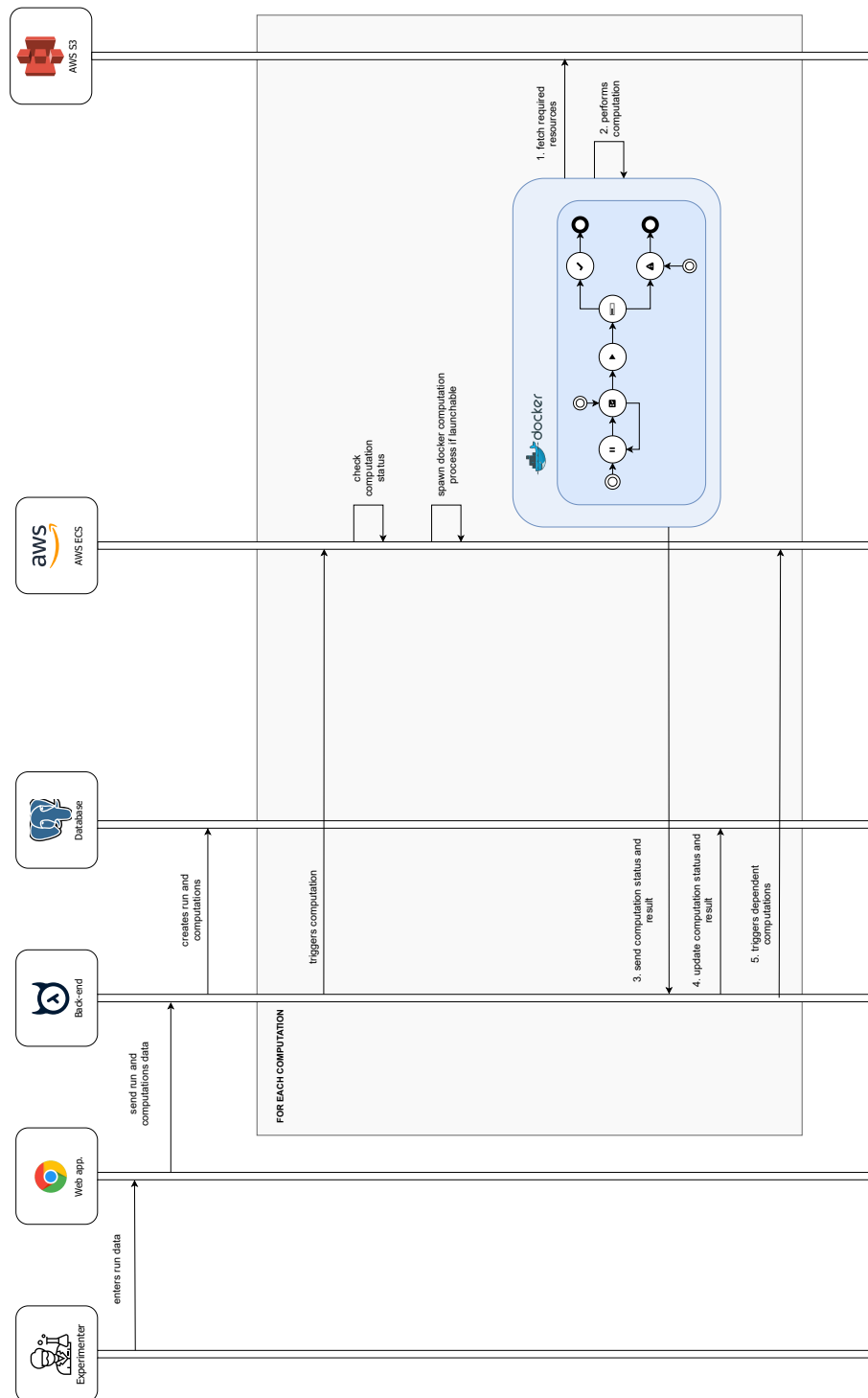


Figure 4.20: UILAB- Measures computation - sequence diagram

Experimental Studies

| 5

In the first chapter of this thesis, we defined several features of quality models that participate in their replicability, namely explicitness, formalness, calculableness and interpretability. In chapter 3, we provided a formal definition of quality models allowing to model a series quality models categories such as questionnaire-based models or machine-learning based models. We described why calculableness and explicitness could not be guaranteed by the sole formal definition of quality models. We then presented in chapter 4 a software tool which purpose is to improve the explicitness of experimental studies on GUI visual design and quality models by (semi-)automating a series of processes and workflows such as screenshot capture, experiment design or measure computation.

All the features of quality models listed previously contribute to the replicability of the latter. However, the interpretability of quality models has not yet been discussed. In this chapter, we tackle this issue by developing a methodology based on factor analysis and aiming at the improvement of the interpretability of quality models. This methodology is illustrated by an experimental study establishing the visual profile of popular websites on mobile platforms.

The decision of analyzing websites on mobile platforms comes from the lack of literature on this particular type of platform, such as explained in chapter 2. We also decided to establish the visual profile of these websites because, as explained in chapter 2, there already exists numerous experimental studies putting in a one-to-one relation influencing factors of UI quality or visual design and facets of User Experience. On the contrary, establishing the visual profile of websites based on quantified aspects of their UI is an issue that has little or not been tackled by the HCI community and could lead to interesting insights.

5.1 Methodology

The purposes of this experiment are manifold. First, the objective is to identify the latent factors underlying the low-level features of mobile websites GUIs quantified through measures. Another objective of this experiment is to determine the visual profile of popular websites based on the identified fac-

tors and to establish an explicit, formal, calculable and interpretable quality model based on the collected data. This section first describes the experimental setup used in this study, namely how the websites were selected, how screenshots of websites mobiles GUIs were collected and how measures were computed on the collected screenshots.

5.1.1 Experimental Setup

For this experiment, we collected the data of the 50 most popular websites of 22 top categories provided by SimilarWeb¹ using the SimilarWeb API². We used uILAB to capture automatically the screenshots of the selected websites by building 22 galleries, each corresponding to a category provided by SimilarWeb and containing the websites of the category. We captured the screenshot with a resolution of 414 (width) by 732 (height) with a Device Pixel Ratio (DPR) of 3, corresponding to the resolution of a high-end mobile device. Some screenshots had to be captured by hand and some websites were removed from the dataset because of various reasons such as request timeout or regional restrictions. After processing, the number of websites considered is N=1059. We used uILAB to automate the computation of measures implemented in AIM [115] on the set of 1059 screenshot captured. As some of the measures implemented in AIM return results composed of several metrics, we decomposed them to obtain 53 metrics.

5.1.2 Data Analysis

We first studied the univariate distribution of the 53 measures composing the dataset. Figure 5.1 provides a global overview of the distribution of variables. We can observe that the variables have very different distributions. Some variables have a clear bi-modal or multi-modal distribution such as the variables illustrated in figure 5.2. Multi-modal variables are mostly color-related variables linked to the angle (on the hue wheel) at which the best fit occurs between the screenshot color scheme and the color scheme template [37] represented by the variable. An image has one color scheme of each type that is the closest to it (in terms of angle and distance), but it does not mean that the image matches perfectly each of the color schemes. The best-matching type of color scheme must be considered for each image by looking at the distance variables for each of the color schemes. The smallest distance indicates the best matching scheme.

Other variables have a very strong skewness, as it can be observed on figure 5.4. This apparent skewness is due to statistical outliers which removal

¹<https://www.similarweb.com/fr/top-websites/>

²<https://data.similarweb.com/api/v1/data?domain=github.com>

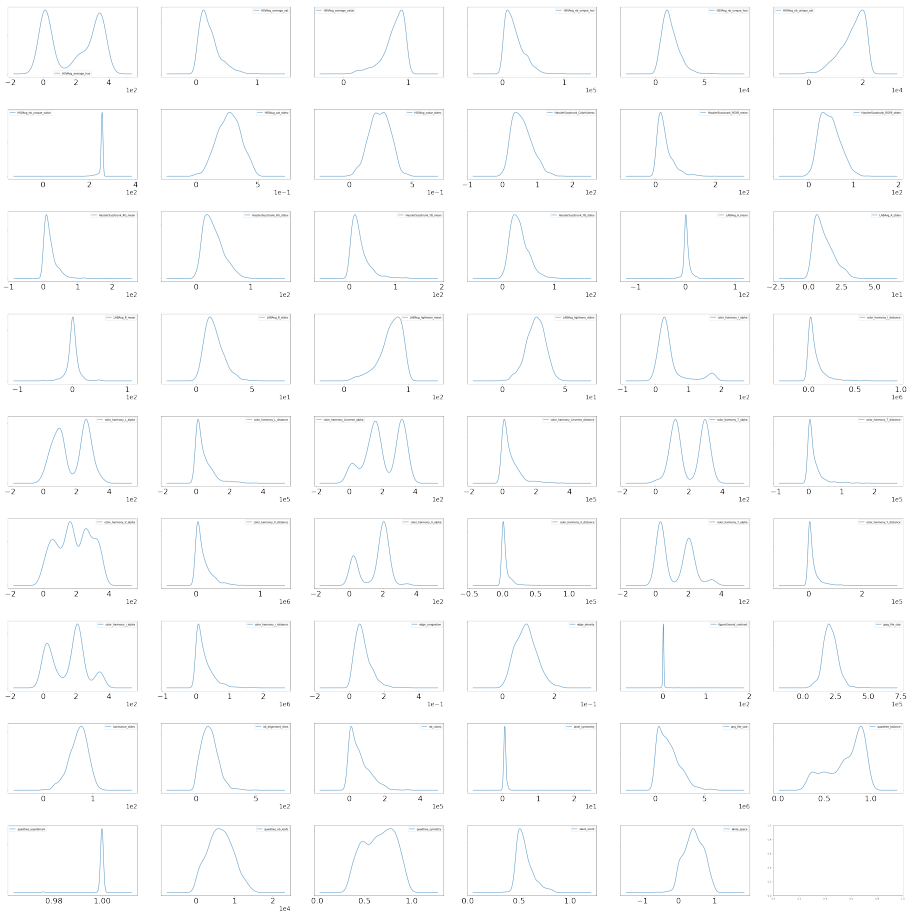


Figure 5.1: Screenshot measures - statistical distributions of variables

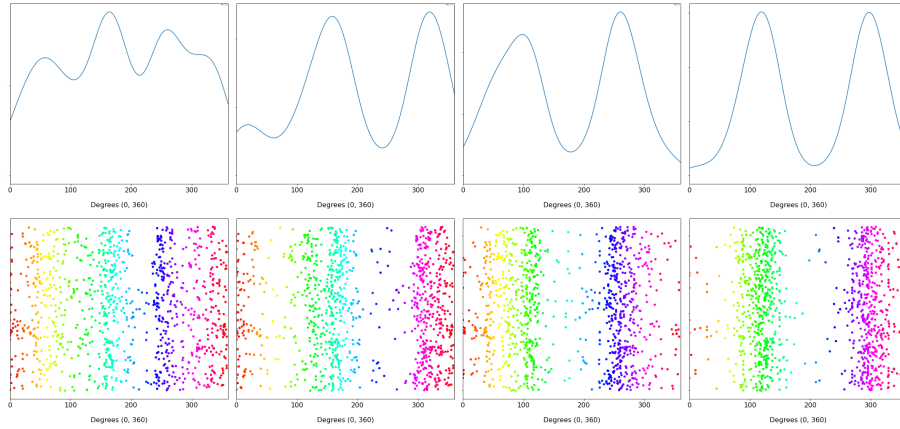


Figure 5.2: Density (top) and color hue (bottom) distributions for variables *color_harmony_V_alpha*, *color_harmony_Linverse_alpha*, *color_harmony_L_alpha* or *color_harmony_T_alpha* respectively (left to right)

may be required for further data analysis. However, outliers must be considered on a multivariate basis as some points might be outliers regarding one or few variables while taking values in a reasonable range for the other variables. Also, it is not acceptable to drop observations just because of their outlier nature. In the context of the dataset considered here, all screenshots have been verified manually and measures have been computed in a systematic manner. As a consequence, as all the observations are legitimate, we decided not to remove outliers.

The identification of latent factors makes sense only if variables or groups of variables are correlated to some extent, indicating the existence of underlying concepts. Figure 5.3 represents the correlation matrix between the 49 measures considered in this study. We can observe several clusters of highly-correlated variables suggesting that the original data could be summarized as a smaller set of latent variables.

To objectivate the suitability of the data for structure detection, we computed the Kaiser-Meyer-Olkin Measure of Sampling Adequacy [65] (KMO) for each variable and for the overall data. The KMO score is a statistic that indicates the proportion of variance in your variables that might be caused by underlying factors³. The test measures sampling adequacy for each variable in the model and for the complete model. The KMO score for the whole data was 0.52, meaning that the data is barely suitable for factor analysis. We

³<https://www.ibm.com/docs/en/spss-statistics/version-missing?topic=detection-kmo-bartlettstest>

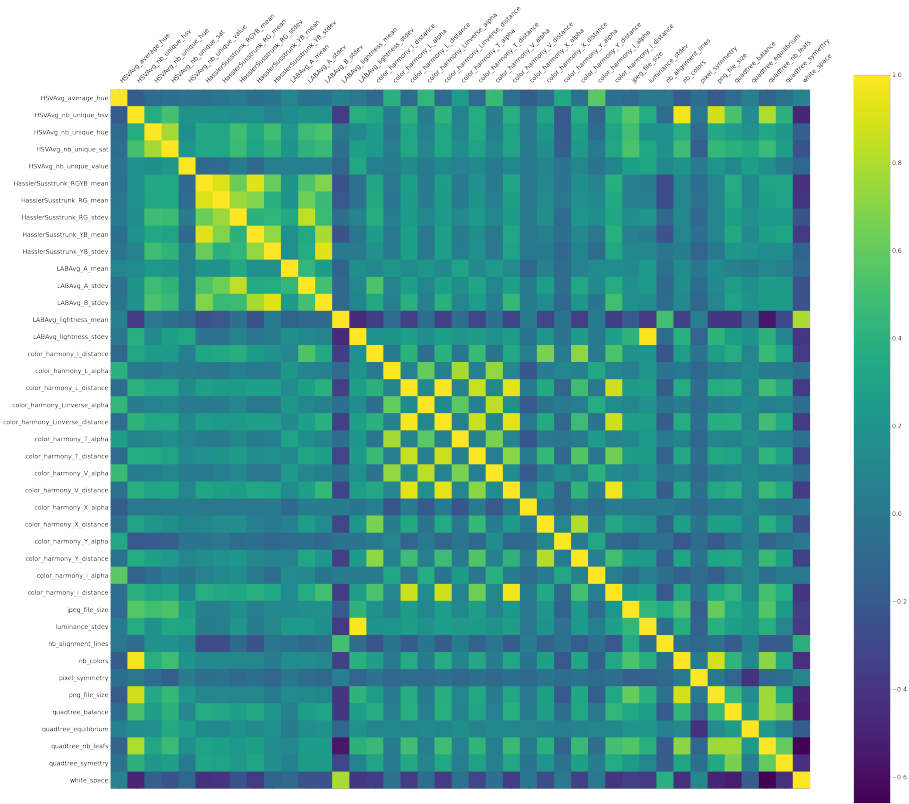


Figure 5.3: Screenshot measures - correlation matrix.

decided to remove the 12 variables with KMO score lower than 0.6, which improved the global KMO score to 0.8, which is "meritorious" according to Kaiser and indicating the suitability of data for factor analysis. We computed the Bartlett sphericity score of the resulting dataset (value=54491.7, p-value < 1e-3) which confirms the adequacy of the data for factor analysis.

We ran a Principal Component Analysis (PCA) to make a first selection of the number of factors to retain. We applied the Kaiser criterion [167] to determine the number of components to keep for further analysis. Figure 5.5 depicts the Scree plot [76] representing the amount of variance explained by each extracted factor (in decreasing order of eigenvalue). From the application of Kaiser criterion, the first 10 factors should be retained for a cumulated explained variance of 76.96%.

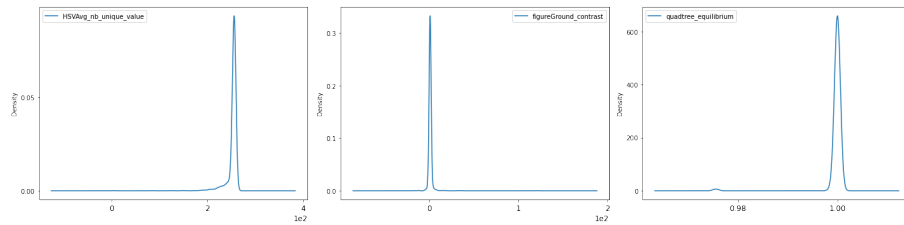


Figure 5.4: Screenshot measures - Examples of skewed variable distributions due to statistical outliers

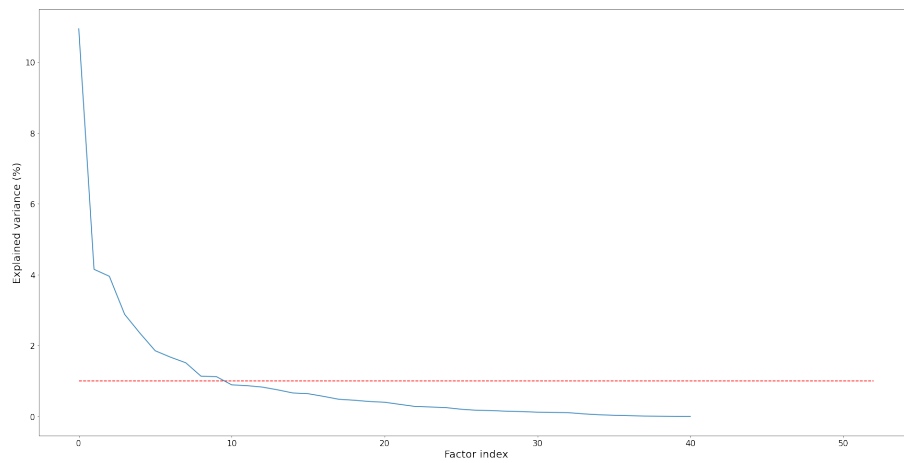


Figure 5.5: Scree plot of Principal Component Analysis

5.2 Results

5.2.1 Identification of latent factors

The intent of Principal Component Analysis is mainly to summarize features, reduce data dimensionality and to identify structure in data. However, interpreting principal components may be a challenging task. Indeed, the factors extracted through PCA with the highest explanatory power are usually "generic" factors loading most of the variables. These factors correlate to all the variables of the original dataset to some extent, making them difficult to interpret as they do not convey any particular information about the data.

For this reason, we used factor analysis (FA) to obtain a more insightful decomposition. Factor analysis operates on the notion that observable variables can be reduced to a smaller set of latent, unobserved variables. These variables, called factors, are not directly measured but are rather hypothetical constructs that are partially reflected through each of the observed variables. Exploratory Factor Analysis (EFA) is especially used to discover factors influencing variables and to detect clusters of related variables.

First, we had to determine what kind of rotation must be applied to the factor space. The goal of factor rotation is to improve the interpretability of the factor solution by reaching a simple structure [151].

Figure 5.6 depicts the factor loadings for various configurations with 10 factors extracted, namely PCA, Factor analysis with promax, oblimin, varimax and quartimax rotations.

From figure 5.6, we can see that the simplest structure is achieved by factor analysis with oblique rotations (promax and oblimin). PCA without rotation suffer from cross-loadings and genericity of the most explanatory factors, as explained earlier. In the case of factor analysis with orthogonal rotations, the extracted structures are more complex than with oblique rotations. However, we must take into account that in oblique rotations, factors can be correlated as they are not required to be orthogonal with each other. Oblique rotations can achieve simpler structures because part of the explained variance is shifted to inter-factors correlations.

For the interpretation of latent factors and further analysis, we decided to keep the configuration with oblimin rotation as the achieved structure is simple enough for interpretation. Moreover, it seems reasonable to expect the factors to be correlated to some extent with each other as they are representing different facets of the same image.

The decision to extract 10 factors was based on principal component analysis but this number may not apply to the factor analysis with oblimin rotation we chose to continue with. The application of Kaiser criterion and the analysis of the scree plot for this configuration indicates that 8 factors have

eigenvalues > 1 and thus should be retained. However, we ran a parallel analysis [84] to confirm that number and only the first 7 factors have an eigenvalue both greater than 1 and their random counterpart in parallel analysis. For this reason, we retained 7 factors for further analyses.

The factor loadings for factor analysis with 7 factors and oblimin rotation are indicated in table 5.1. They correspond to the factor loadings depicted in figure 5.7 for oblimin rotation. In order to interpret the factors and name them accordingly, we decided to consider only the variables with loadings > 0.4 in absolute value as influential for the corresponding factors. These variables are highlighted in table 5.1.

1. **Colorfulness:** The most important loadings are all related to colorfulness and color variation
2. **Color complexity:** Some of the contributive loadings are directly related to the number of colors (*HSVAvg_nb_unique_hsv*, *nb_colors*, etc.). The variables *png_file_size* and *jpeg_file_size* are also positively related to the number of color as when this number increases, the image compression becomes less efficient and the file size increases. The number of leafs in quadtree decomposition (*quadtree_nb_leafs*) is also related to the number of colors as the decomposition is based on color homogeneity/entropy. On average, a larger number of colors means larger number of leafs.
3. **Analogous color schemes dissimilarity:** This factor is positively related to the distance of the image color scheme to the analogous color schemes (V, i)[37] and to the T and L Matsuda color schemes [93] that can also be considered as analogous color schemes but with a higher degree of uncertainty. As described in [74], we can consider that the Matsuda color schemes can be described by 3 patterns - analog, complementary and triad. So the greater the value of this factor, the more dissimilar the color schema of the image is from analogous color schemes and the more they tend to have a larger number of dissimilar colors, being either complementary or triadic schemes.
4. **Primary hue:** This factor is related to the hue (i.e, angle on the hue wheel) at which analogous color schemes (V, L, L_inverse, T, i) are closest to the image color scheme. This can be interpreted as the main color component / main hue of the image. This is also confirmed by the loading of the variable *HSVAvg_average_hue*.
5. **Lightness deviation:** The most contributive loadings of this factor are related to the lightness deviation in the image around the lightness

average value. The greater the value of this factor, the more variation in luminosity can be observed in the image.

6. **Complementary colors dissimilarity:** This factor is positively related to the distance to complementary color schemes (the greater the value of this factor, the greater the distance to complementary color schemes). It means that image with large values for this factor are far from complementary schemes and tend to be closer to either analogous or triadic schemes.
7. **Structural simplicity :** this factor is related to the simplicity of the image from a structural perspective (*white_space*, *nb_alignment_lines* representing grid quality of the image). The average lightness of the image (*LABAvg_lightness_mean*) is positively related to the proportion of white space and thus contributes to improve the structural simplicity score. The higher the score of this factor, the simpler the structure of the interface.

Variable	Colorfulness	Color complexity	Analog. schemes. diss.	Primary hue	Lightness dev.	Compl. schemes diss.	Structural simplicity
HSVAvg_average_hue	-0.004	-0.201	-0.061	0.565	-0.017	0.049	-0.055
HSVAvg_nb_unique_hsv	-0.016	0.923	0.042	0.004	-0.020	0.014	0.030
HSVAvg_nb_unique_hue	0.299	0.299	0.091	-0.003	0.010	0.122	0.575
HSVAvg_nb_unique_sat	0.293	0.428	0.011	0.053	0.130	0.039	0.511
HSVAvg_nb_unique_value	-0.112	0.084	0.021	-0.031	0.335	0.003	0.168
HasslerSusstrunk_RGYB_mean	0.961	0.044	-0.004	0.002	-0.070	0.021	-0.135
HasslerSusstrunk_RG_mean	0.798	0.052	-0.077	0.040	-0.085	0.208	-0.096
HasslerSusstrunk_RG_stdev	0.621	-0.047	-0.088	0.088	0.075	0.337	0.244
HasslerSusstrunk_YB_mean	0.882	0.051	0.109	-0.034	-0.042	-0.156	-0.117
HasslerSusstrunk_YB_stdev	0.695	-0.116	0.208	-0.067	0.090	-0.171	0.283
LABAvg_A_mean	0.213	0.009	-0.026	0.239	0.131	0.083	0.124
LABAvg_A_stdev	0.518	-0.064	-0.053	0.075	0.089	0.471	0.249
LABAvg_B_stdev	0.713	-0.089	0.278	-0.025	0.092	-0.067	0.241
LABAvg_lightness_mean	-0.128	-0.223	-0.084	0.019	-0.338	-0.102	0.598
LABAvg_lightness_stdev	-0.092	-0.001	0.024	0.004	0.976	-0.015	-0.071
color_harmony_I_distance	0.128	0.030	0.124	-0.089	0.036	0.726	-0.060
color_harmony_I_alpha	-0.034	-0.027	0.020	0.795	0.064	-0.010	-0.029
color_harmony_L_distance	-0.031	0.021	0.974	0.011	0.005	0.028	0.001
color_harmony_Linverse_alpha	-0.014	0.044	0.076	0.813	-0.028	-0.072	0.009
color_harmony_Linverse_distance	-0.037	0.026	0.981	-0.009	0.005	0.023	0.006
color_harmony_T_alpha	0.021	0.068	-0.027	0.725	0.082	-0.055	0.071
color_harmony_T_distance	-0.109	0.032	0.729	-0.028	0.014	0.250	0.063
color_harmony_V_alpha	-0.029	0.036	0.036	0.917	-0.030	-0.021	-0.004
color_harmony_V_distance	0.061	-0.000	0.956	0.058	0.009	-0.074	-0.027
color_harmony_X_alpha	0.019	-0.042	0.035	-0.261	0.067	0.021	0.058
color_harmony_X_distance	-0.148	0.076	0.108	-0.039	0.026	0.740	-0.028
color_harmony_Y_alpha	-0.029	-0.230	-0.115	0.183	0.021	0.198	-0.166
color_harmony_Y_distance	-0.125	0.083	0.246	0.003	0.014	0.731	-0.022
color_harmony_Y_alpha	0.056	-0.203	-0.065	0.506	-0.009	0.193	-0.073
color_harmony_Y_distance	0.134	0.002	0.819	0.038	0.022	0.089	-0.056
jpeg_file_size	-0.057	0.555	-0.029	0.016	0.240	0.016	0.342
luminance_stdev	0.004	-0.020	-0.008	0.040	0.944	0.010	0.013
nb_alignment_lines	-0.238	0.068	-0.093	-0.003	-0.010	-0.103	0.603
nb_colors	-0.087	0.891	0.026	-0.002	0.012	-0.009	0.069
pixel_symmetry	-0.090	0.041	0.023	-0.074	-0.182	0.000	-0.083
png_file_size	-0.098	0.897	0.021	-0.008	0.055	-0.016	0.018
quadtree_balance	0.264	0.509	0.076	0.048	0.012	0.082	-0.164
quadtree_equilibrium	0.149	0.075	-0.063	0.109	0.247	0.011	0.142
quadtree_nb_leafs	0.114	0.775	0.089	0.008	0.049	0.121	-0.166
quadtree_symmetry	0.232	0.447	0.105	0.079	-0.162	0.088	-0.168
white_space	-0.319	-0.397	-0.015	0.019	-0.243	-0.074	0.466

Table 5.1: Factor loadings for factor analysis with oblimin rotation - 7 extracted factors

5.2.2 Visual profiles

In order to establish the visual profile of each category, we first analyzed the average profile of websites by considering the distribution of factors across all

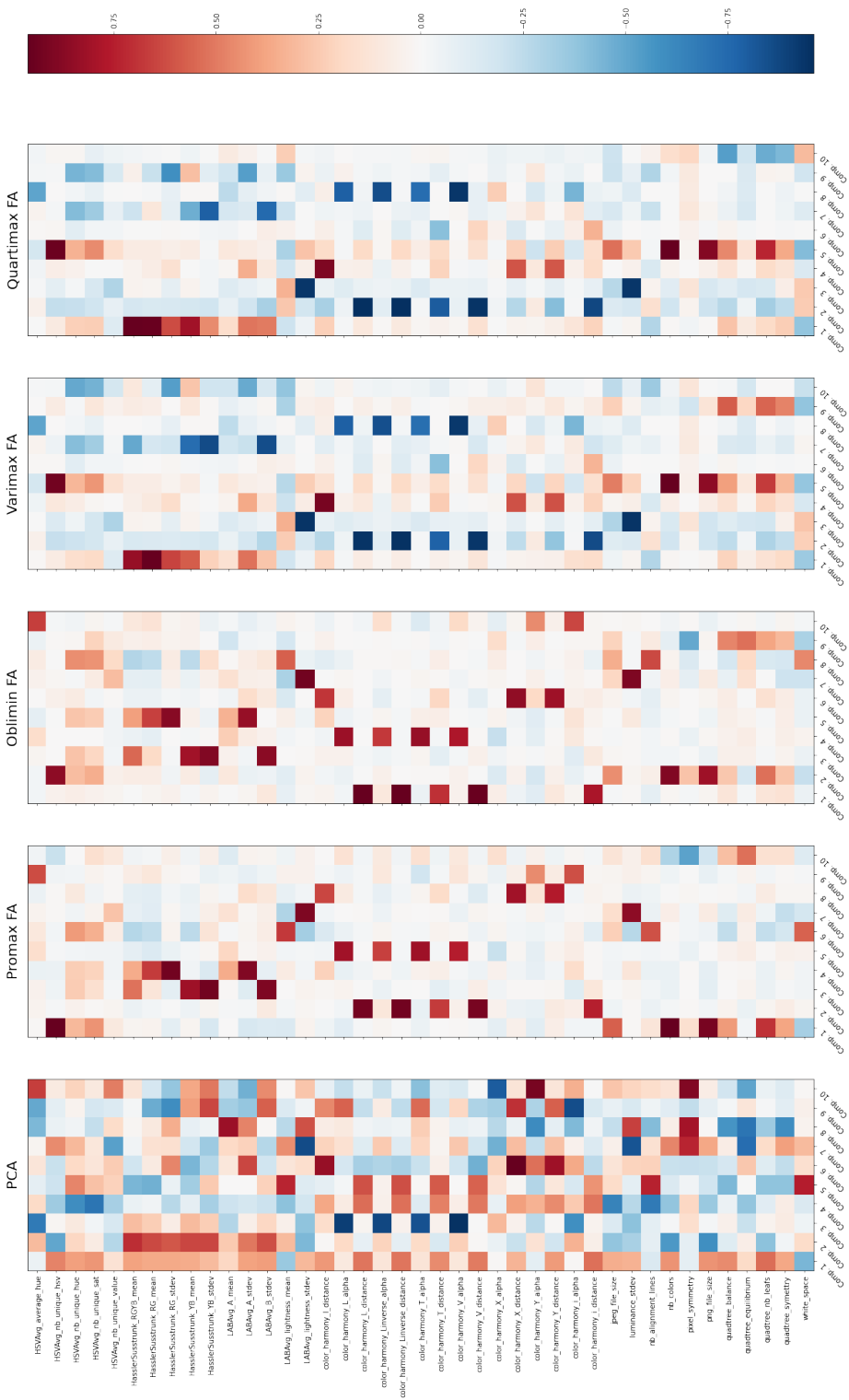


Figure 5.6: Comparison of the different factorial analysis rotations with 10 factors

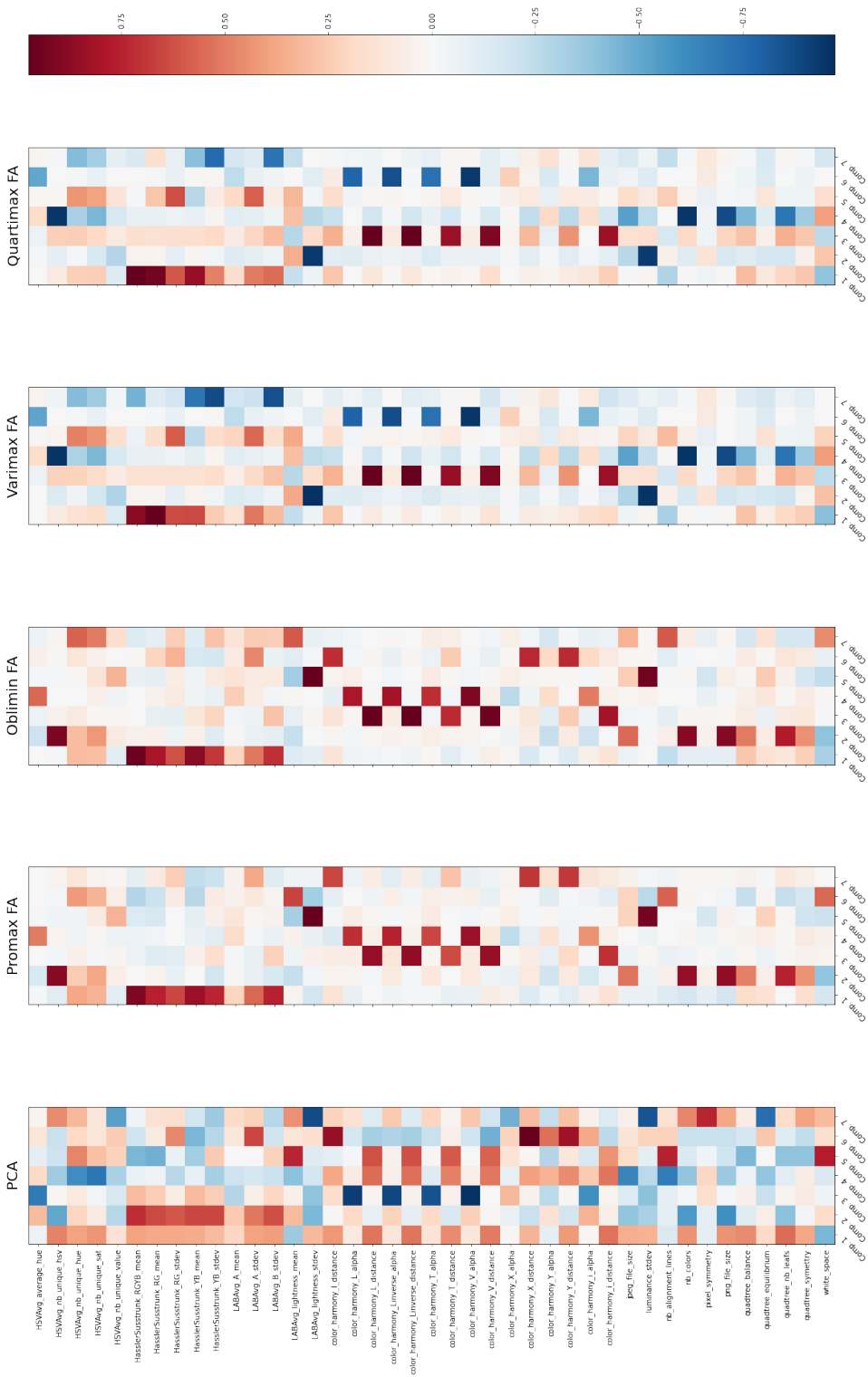


Figure 5.7: Comparison of the different factorial analysis rotations with 7 factors

categories. We compared the distribution of factors for each category with the global distribution of the same factors in order to detect significant deviations from the average profile.

All the factors are centered around zero. Some of them are evenly distributed around their mean while others are skewed to some extent, either positively or negatively.

The distribution of the 5th factor named *primary hue* displays an interesting shape. We know that this factor loads mainly the alpha variables of several color schemes. Most of these variables displays a bi-modal distribution with peaks located at different angles on the hue wheel. We can approximate the range of the distribution of the 5th factor as such, although it is not as precise as for single variables due to the superimposition of several alpha variables as well as other variables having minor impact.

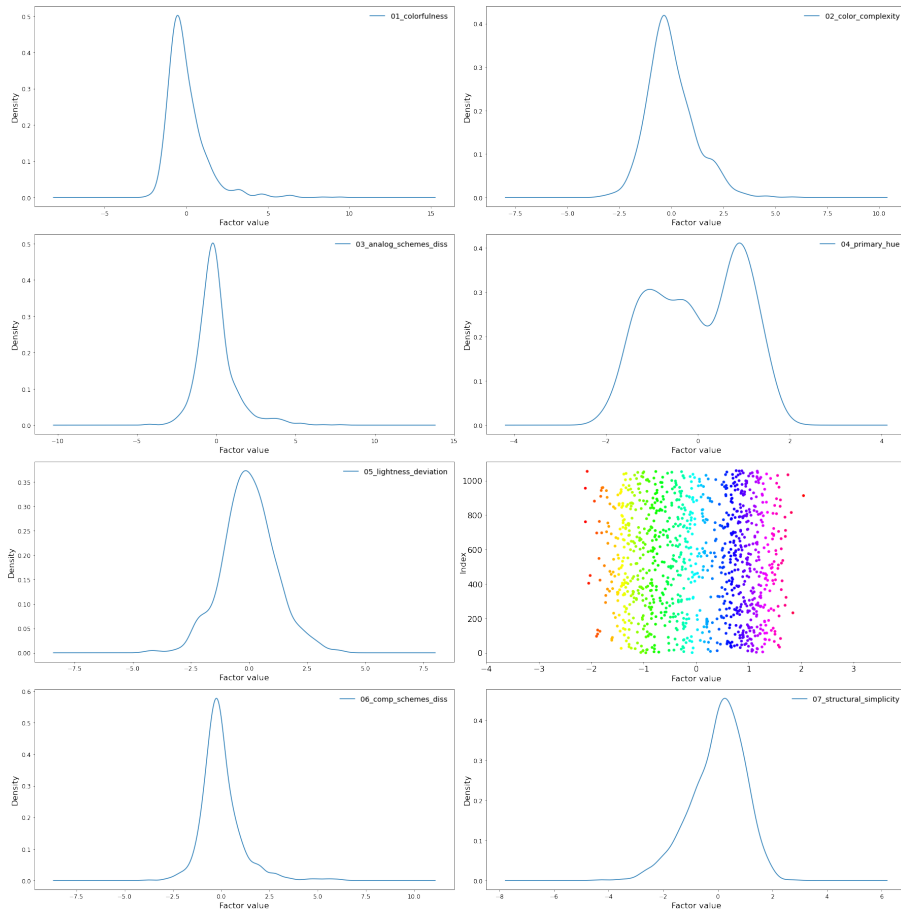


Figure 5.8: Distribution of factors across all categories

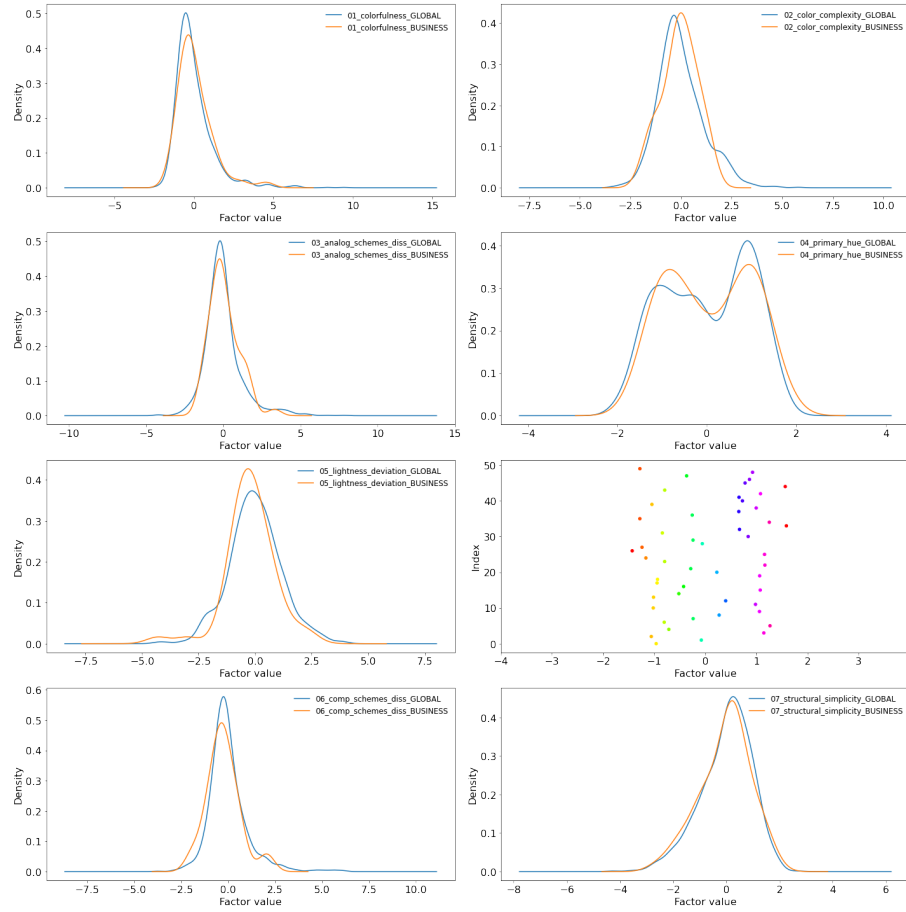


Figure 5.9: Distribution of factors for the category “Business and consumer services”

The visual profiles of each category and their deviations compared to average profile according to each factor are explained here under. We provide a complete example for the first category with the description of the visual profile and its associated figure (figure 5.9). For the other categories, only the description of the visual profile is provided in this section and the associated figures can be found in the appendix C.

- **Business and consumer services** (figures 5.9 and C.1) : There is no significant deviation from the average profile for most of the factors. Regarding the primary hue factor, the bi-modality of the distribution for that category is more pronounced than for the global distribution.
- **Arts and entertainment** (figure C.2): Regarding the complementary schemes dissimilarity, the distribution for this category is broader and

skewed towards positive values. This means that a larger proportion of websites from this category have a color scheme that distinguish itself from complementary color schemes. The distribution of primary hue for that category differs strongly from the average profile and does not follow a bi-modal distribution. Primary hue of websites of this category is mainly located in the blue-magenta region. Differences in the distribution of lightness deviation can also be observed. This factor has a higher standard deviation and is slightly shifted towards positive values, meaning that websites of this category display on average a greater variation of luminosity in the different parts of their interface than the average website profile.

- **Community and society** (figure C.3): The only major deviation from average websites for this category is related to the primary hue factor. The bi-modality of the distribution for that category is less pronounced than for the global distribution and its standard deviation is broader. It means that websites in this category have a greater variety of colors while still following the color distribution of the average website profile with a majority of websites in the blue-magenta region.
- **Computers Electronics and Technology** (figure C.4): Websites from this category have a color complexity lower than the average profile. On average, they have less variations in their lightness than other categories but the standard deviation of this factor is also larger than the one of the global distribution for the same factor, meaning that websites of this category are less uniform in their lightness deviation than on average across all categories. Primary hue displays the most important difference of distribution. The distribution of primary hue for this category is uni-modal with a peak in the blue region, corresponding to the local minimum of the global distribution of the same factor. This means that these websites strongly differ from the average profile from a color perspective.
- **E-commerce and Shopping** (figure C.5): Color complexity of these websites is significantly lower than other websites, we can see that the distribution of this factor is more concentrated around zero than the global distribution. The distribution of primary hue factor is closer to bi-modality than the global distribution, as it is the case for Business websites. The standard deviation of distributions regarding the distances to color schemes is also larger, meaning that it is harder to associate a particular color scheme to websites of this category. Another significant difference of this category compared to the average profile concerns the structural simplicity. E-commerce websites appear to be

much simpler from a structural point-of-view. This is probably due to the grid quality of most of the websites in this category, positively correlated to the visual simplicity of websites. This can be observed on the related chart where we can see a clear shift of the distribution towards positive values for this category.

- **News and Medias** (figure C.6): The primary hue distribution of this category strongly differs from the global distribution. It is similar to the primary hue distribution of the category “Community and Society” where the distribution is biased towards the positive values, corresponding approximately to the blue/magenta zone of the HSV color wheel. Websites of this category are also less uniform in their lightness, meaning that we can have stronger differences between dark and light zones of these websites than in the average website profile. Finally, they are structurally simpler than the average website. Similarly to e-commerce websites, we can attribute this simplicity to the grid-based representation of information in these websites.
- **Finance** (figure C.7): Color complexity mean of financial websites is lower than the average color complexity across all websites. Distribution of primary hue, with a peak in the green-blue region, is also different in this category compared to the global distribution.
- **Food and drinks** (figure C.8): This category has a lot of significant differences compared to the average website profile. Websites of this category are on average more colorful than other websites, making them also more complex from a color perspective. The primary hue distribution is also different from the global distribution, the vast majority of websites having a main color located in the red-green area of the hue wheel. Websites of this category have also less variation of lightness in their interfaces on average. Finally, these websites are simpler from a structural point-of-view than other websites.
- **Lifestyle** (figure C.9): Lifestyle websites have primary hue distribution that is more uniformly distributed than the global distribution. They are more complex than other websites from a color perspective, although this complexity is not reflected in the distribution of colorfulness. The distribution of distance to analog color schemes is also more spread than on average, meaning that some websites of this category will be more likely to match complementary or triadic color schemes while others may not have clear color scheme.
- **Games** (figure C.10): Gaming websites are more complex than the average website, both from color and structural point of views. The primary

hue is also distributed in a more uniform fashion compared to the distribution of the same factor across all categories. The spread of lightness deviation for this category is also larger than on average, meaning that we can find a larger variety of lightness differences in gaming websites than on average.

- **Health** (figure C.11): Health websites are more colorful than the average website and they are also slightly more complex from a color point-of-view. The structure of health websites is more complex than the structural profile of average websites. As for websites of the “Games” category, the primary hue of health websites is more homogeneous compared to the distribution of the same factor across all categories. Health websites also display less variation in the lightness of their interface.
- **Science and education** (figure C.12) These websites are simpler from a color point-of-view. However, they are structurally more complex. Their distance to analog and complementary color schemes displays more spread, meaning that it may be hard to attribute a precise color scheme to websites of this category. The distribution of primary hue for this category is similar to the distribution of the “Games” category with a peak around the blue / magenta region of hue wheel.
- **Heavy industry and engineering** (figure C.13): Industry websites are more complex from a color perspective, the spread of the distribution also being larger than the global distribution for the same factor. The primary hue distribution for this category is very close to the distribution of “Health” websites for the same factor. “Industry” websites also have a smaller lightness deviation in their interface than other websites and they are structurally more complex.
- **Hobbies and leisures** (figure C.14): Websites of this category are more complex from a structural perspective. Their primary hue distribution displays a slight, asymmetrical bi-modality shifted towards the green-yellow section of the HSV space. The distribution of lightness deviation also displays a larger spread than the global distribution.
- **Home and garden** (figure C.15): Their primary hue distribution also displays a slight, asymmetrical bi-modality shifted towards the green-yellow section of the HSV space as it is the case for the “Hobbies & Leisures” category. Websites of this category are not more colorful than other websites but their color complexity tend to be higher, which may seem contradictory.

- **Jobs and career** (figure C.16): Websites from this category are simpler from a color perspective. The distribution of primary hue strongly differs from the global distribution of that factor and is similar in shape to the distribution of the same factor for the “Computer Electronics & Technology” category, while having a larger standard deviation.
- **Law and government** (figure C.17): “Law and government” websites have a primary hue factor displaying a strong uni-modal distribution with a maximum density achieved in the blue/magenta region of the hue wheel. Aside of that, all the other factors do not display significant deviations from their corresponding global distributions.
- **Pets and animals** (figure C.18): Websites of this category have primary hue distribution somehow similar to the corresponding distribution of the “Food & Drinks” category. They are also simpler structurally speaking but more complex from a color point-of-view.
- **Reference materials** (figure C.19): Primary hue of websites of this category displays a wide uni-modal distribution, as it is the case for other categories, with a peak located in the green-cyan region of hue wheel. They are also less colorful and simpler from a color perspective. The lightness of websites from this category is on average more uniform than the average website profile (i.e the lightness deviation is smaller on average). These websites are also structurally simpler than other websites, although the difference between distributions is less pronounced than with other categories.
- **Sports** (figure C.20): The primary hue distribution of websites from this category is a bi-modal distribution with weak peaks located in the green-yellow and blue-magenta regions of the HSV space. The most significant difference lies in the lightness deviation, where a larger lightness deviation in the interfaces of websites from this category is observed.
- **Travel and tourism** (figure C.21): Websites from this category follow quite closely the average website profile, except for primary hue where the spread around the blue-magenta region is larger for this category.
- **Vehicles** (figure C.22): Websites from the “Vehicles” category are more complex, both from a structural and a color perspectives. The distribution of primary hue and other factors for this category does not differ much from the corresponding global distributions.

5.2.3 Quality models of aesthetics and complexity

After establishing the visual profile of the different categories, we used the collected data about user perception of visual appeal and visual complexity to build quality models based on these facets of GUI visual design. We performed a multiple linear regression with a L1-regularization algorithm (Lasso) to mitigate the risk of data over-fitting. To optimize model hyper-parameters (*i.e.*, alpha coefficient for Lasso regression), we used a grid search method with a 10-fold cross-validation and the alpha coefficient taking 50 different values ranging from 0.1 to 1. In order to evaluate the impact of the factor analysis approach on the predictive performances of the quality models, we built quality models using either the original set of measures and the extracted factors. The results are summarized in table 5.2.

We can observe that the various models display poor predictive performances, with a R^2 as low as 2.4% for the prediction of perceived visual complexity with factors as explanatory variables. This barely superior to the uniform model (*i.e.*, predictive model composed of a single constant explanatory variable). The loss of performances between the models using original variables and factors can be explained by the loss of explained variance due to the factor extraction procedure and the projection of a high-dimensional space onto a space of lower dimension. Aside of this loss of performances, the potential reasons explaining these globally poor performances are manifold :

- The perceived visual complexity and visual appeal are not influenced by the measured variables. This hypothesis seems unlikely as other studies have reported significant correlations between these perceived variables and various dimensions of visual design in similar contexts [125, 154, 173].
- The data collection process suffered from flaws and biases due to a wrong comprehension of the facets to be evaluated by users, a lack of serious of participants taking part in the experiment, a bad sampling of participants regarding socio-demographical variables, etc.
- The algorithms used to perform the modeling or the considered features are too simple. Linear regression may be too simple to represent the existing relationships between measures variables and highly subjective perception of facets of visual design. Moreover, higher order features of the original variables may be required, although the model using higher-order features of the collected measures performed worse than the simple, 1st-order features model depicted in table 5.2.
- The modeling problem is ill-defined. Another potential formulation of the problem would be to consider it as a classification problem and not

as as continuous regression problem.

However, regardless of the performances of the models and data engineering techniques that could be applied in this context, this underlines the need for replication of such experimental studies incorporating subjective collected data and modeling / processing decision during the analysis process. Indeed, a single experimental studies such as the one described in this chapter is of poor scientific value. Insightful conclusions can only come from a consensus of a large number of experiments studying the same phenomenon with slight variations to consolidate the scientific claims derived from the results.

	Complexity	Visual appeal
Original measures	5.3%	22.6 %
Factors	2.4%	9.3%

Table 5.2: Predictive performance (R^2) for quality models

5.2.4 Reproducibility of experimental studies

We showed in the last section that UILAB could be successfully used to build datasets used in the construction of quality models and foster their development. However, the reproducibility of experimental studies has yet to be proven. In [42], the dataset described in this section was used to carry out the construction of similar quality models, along with other datasets collected through UILAB but not described in this document. This independent study proves that UILAB can effectively be used to build datasets that can be shared or reused by researchers to reproduce or replicate experimental setups to some extent, that extent being formalized by the framework described in chapter 3.

	Experiment 1	Experiment 2	Experiment 3
Websites source	SimilarWeb	Alexa	SimilarWeb
Measures	QUESTIM + UILAB	QUESTIM + UILAB	UILAB
User ratings	/	Beauty (UILAB)	Beauty & Complexity (UILAB)

Table 5.3: Reproduced experimental studies including UILAB

Conclusion

| 6

In this concluding chapter, we assess the contributions brought by this thesis along with their strengths and limitations. We also discuss the potential improvements that can be brought to the proposed solutions, both in short- and long-term views.

6.1 Overview of Contributions

Existing quality models such as the ISO-25010 standard remain highly conceptual and cannot always be applied to evaluate the various aspects of concrete user interfaces. Moreover, their highly conceptual nature makes them hardly testable against replicability criteria. For this reason, we provided in chapter 3 a formal yet flexible definition of quality models that can accommodate a variety of existing quality models such as questionnaire-based models or SQuaRE-based models. In addition to the highly conceptual nature of quality models, such as depicted in section 3.1 of chapter 3, existing frameworks for replicability are often issued from domains that may be different from Human-Computer Interactions. These differences can make these framework unsuitable for the evaluation of replicability of experimental studies in HCI. These frameworks are also based on overlapping concepts that do not always allow to derive clear and testable criteria to evaluate the replicability of experimental studies. For this reason, we proposed a framework for replicability composed of incremental criteria that allow to evaluate replicability of experiments in GUI visual design. The proposed framework is generic enough to be instantiated to other fields of HCI, such as Gesture recognition [90, 89, 116, 161].

Explicitness is a property of the quality models related to the degree to which their underlying processes are documented, both from a methodologically speaking but also from a data management and sharing point of view. Explicitness is a direct influencing factor of replicability, which is a matter of good practices and habits on the part of the researchers from a methodological, scientific and documentation point of views. However, replicability also depends on the tools and technologies that make the processes efficient and realistic. To that purpose, we built a software tool aimed at improving the explicitness, and in this way the replicability, of experimental studies and qual-

ity models on GUI visual design by (semi-)automating a series of processes that were until now resource- and time-consuming as well as error-prone due to their manual nature. The tool allows the automation of the screenshot capture process, the design and automated deployment of experiments and the computation of measures on previously capture screenshots.

In chapter 5, we reported the results of the analysis of a dataset collected with uILAB about the perceived visual appeal and complexity of popular websites, along with a series of measures computed on those websites. The purposes of this experimental study were manifold. First of all, one of the goals was to demonstrate the feasibility of the construction of large-scale datasets by using uILAB. Second, the goal was to derive insights about the visual profile of popular websites on mobile platforms, as to the best of our knowledge, this kind of experimental studies had not yet been performed. Finally, one of the purposes of this endeavour was to demonstrate the improvement in interpretability of results achieved through the use of factor analysis in the study of GUIs visual design. Also, the collected dataset underlying this experimental study can be considered as a contribution on its own.

6.2 Strengths and Limitations

Regarding the definition of quality model of chapter 3, the formulation generic enough to encompass a large variety of existing quality models, such as the ones given in example in the same chapter. However, it may still be too theoretical or not flexible enough to accommodate some other contexts of use. The replicability framework provided in the same chapter also covers a large number of concepts from other frameworks while being generic enough to fit in contexts of use different from GUI visual design, such as gesture recognition. However, this framework may suffer from the same flaws as other frameworks, although to a lesser extent, as it was first designed to fit in a Human-Computer Interactions context and may not be applied to particular fields of research having different requirements for experimental studies.

uILAB allows experimenters to save time and resources while building large-scale datasets on GUI visual design by (semi-)automating several processes :

- *Screenshot capture:* uILAB allows to capture screenshot according to various resolutions, orientations, platforms, and display sizes and does it without human intervention during the capture process.
- *Experiment design and deployment:* uILAB allows users to design their experiments in the application according to a specified template. Upon design completion, the experiments are readily available for users with-

out the need to deploy it through online web services or specific infrastructure.

- *Measures computation:* uILAB allows users to compose workflows of measures and compute them automatically on previously captured screenshots.

All data generated through these automated processes can be downloaded and shared with other users to foster replication of experimental studies on GUI visual design. However, uILAB comes with several limitations that make it unsuitable for several contexts of use and situations:

- uILAB allows automatic capturing of only screenshots. Wireframes, sketches and other types of GUI representations must be uploaded manually by users. Also, uILAB does not allow capturing dynamic aspects of end-user interactions, which limits the analysis to static views of user perception without any interaction trace.
- The software only supports the capture of websites UIs. Other types of UIs such as mobile apps, games, smartwatches, etc. are not supported by the software.
- Only one kind of experimental workflow (rating scale) is supported by the software. Other types of experiments (comparison, ranking, etc) are not currently supported by uILAB.
- Only the Python language is currently supported for computing measures. Although the Python language is versatile and complete enough to implement a large number of measures, uILAB might not support some measures using specific libraries only available in other languages such as Java or C#.

The experimental study reported in Chapter 5 of this thesis brought interesting insights about the visual profile of popular websites on mobile platforms. Through this experiment, we also demonstrated the usefulness of the approach based on factor analysis to improve the interpretability of results compared to the basic analysis of low-level measures such as the ones computed by uILAB. However, the identification and the use of latent factors often implies a loss of information, and as such, of predictive performances such as we described in Chapter 5. Also, a number of subjective decisions were made during this experimental study that might have an impact on the results and their interpretation: removal of outliers, naming of extracted factors, number of factors to retain, choice of rotation to apply, etc. All these decision points

have an impact on the interpretation of results and considering the high importance of replication in scientific research, these decisions call for a detailed and precise documentation on the part of the researchers.

6.3 Future Work

Future work can be organised to unlock uILAB limitations reported in Section 6.2 and to leverage its strengths to new capabilities. For this purpose, we may want to relax progressively constraints imposed by the four working hypotheses stated in Section 1.3:

H_1 . We limit the investigation on GUI representations.

H_2 . We consider both represented and real end users.

H_3 . We focus on visual design.

H_4 . We focus on preference.

6.3.1 Relaxing the GUI representation (H_1)

Other representations. While uILAB automatically captures GUI screenshots of websites in various conditions, a quality model can be computed on any other GUI representation when manually uploaded. For example, GUI sketches, mockups, prototypes, pictures, or any other GUI representation produced by a third-party software could be used as stimuli, thereby addressing comparison questions such as: what is the impact on the quality model when a GUI evolves from early sketches to a final version, or when a GUI evolves over time, a question relevant to software evolution [32], what is the impact on the quality model when a GUI evolves from a low level of fidelity (like in a sketch) to a medium level of fidelity (like in a mockup) to a high level of fidelity (like in a final prototype). Multi-fidelity prototyping [39] of GUIs has been expressed as an interesting approach for this purpose.

Other capturing methods. All these cases cover a static evaluation of a GUI through a quality model. This type of evaluation could gracefully evolve to a more “dynamic” evaluation. First, by automatically collecting a suite of GUI screenshots by browsing recursively web pages of a website (e.g., scan the home page, crawl it and discover included URL to further explore other web pages). Second, by automatically collecting a sequence of GUI screenshots while the end user is interacting with the system, thereby capturing a simplified form of interaction trace or history. Furthermore, instead of a sequence of screenshots, a whole interaction session could be captured in a video that will be further analysed by signal processing techniques, such as video analysis and segmentation. Finally, a whole interaction trace could be captured

through code instrumentation [117] by using dynamic program analysis tools to capture user interactions without the need to capture graphical representations of the UI.

Other types of interactive systems. The GUI is also not restricted to a GUI of a website and can cover any GUI in principle: a GUI of a desktop, a stand-alone application, a commercial software, or for any other type of system, such as virtual reality, augmented reality, and extended reality. For example, a screenshot, a series of screenshots or a video could be captured while the end-user interacts with a 3D scene in these contexts.

Other GUI formats. Other formats could be also accepted as input to uILAB, such as direct HTML for a website, a X3D file for a virtual reality scene, or even a UsiXML file describing a GUI, which has been achieved in AesthetiXML [94]. In this way, a quality model could be obtained even before a real GUI is delivered. The possibility to accept several formats as input would also allow the evaluation of a GUI at different development stages if multiple formats are use throughout the development process. Finally, Large User Interfaces (LUI) [43] are an interesting GUI format that could be evaluated with the help of uILAB.

Other interaction modalities. Any interaction modality involving the visual sense of the end user could be subject to a quality model. Beyond virtual reality and associated interaction modalities, gesture interaction can be considered: 2D stroke gesture interactions [89] produce strokes on a surface, which could be analyzed through many measures such as those computed by Omni Praedictio [77]; 3D trajectories could be projected on places and become also subject of measures [116].

In sum, as long as a measure exists to calculate or estimate any aspect of any particular GUI representation, a quality model can be obtained and deployed via uILAB. In particular, the density measure can be computed for any GUI representation.

6.3.2 Relaxing the represented and real users (H_2)

On the one hand, uILAB supports quality models involving measures coming from both end users (such as rating, ranking, preference) and represented users (through automatically computed measures or models). On the other hand, the user interface subject to the quality model is only represented. uILAB can be extended to work hand-in-hand with a real user interface of an interactive application at run-time, but this requires code instrumentation [11]. Beyond merely collecting a sequence of screenshots or even a video of an interactive session, the real user interface could be probed for various measures, which opens the door to relaxing the next working hypothesis.

6.3.3 Relaxing the visual design (H_3)

One can imagine measure other formulas and quality models on user interfaces than those belonging to the field of visual design. For instance, Uribe *et al.* [154] compute various measures based on the JPEG vector of parameters of a screenshot stored in this format. Some of these measures are non-visual (e.g., the compression rate) or do not belong to the visual design (e.g., the spectrum of colors reduced). If a real user interface is instrumented with special code, other measures could be computed at run-time, such as performance measures. Various measures could then be aggregated into one or several different quality models relating particular aspects of the user interface. As soon as a set of identical measures is computed for a sequence of screenshots or others, global measures could be computed.

Let us imagine $S = s_{i=1,...,n}$ a set of n screens captured via any mean and $M = m_{j=1,...,N}$ a set of N metrics computed for all these screens. The values are calculated for each screen s_i and compared by calculating the mean, the standard deviation, the median. In order to normalize the result in a $[0,1]$ range, the result of the standard deviation calculation is divided by x_{max} , which denotes the maximum value for a certain measure over all screens in order to compute the inconsistency (I) as follows:

$$I = \frac{1}{N} \sum_{n=1}^N \frac{1}{x_{max_n}} \sqrt{\frac{1}{J-1} \sum_{j=1}^J (x_{j_n} - \bar{x}_n)^2} \in [0, 1] \quad (6.1)$$

Where x_{j_n} denotes the measure value x on screen j and measure i . N is the number of measures per screen and $\bar{x}_n$ is the mean complexity value for measure n over all screens. In this way, a quality model could be computed for estimating the inconsistency I and the consistency $C = 1 - I$ for a series of measures and screens.

6.3.4 Relaxing the preference option (H_4)

Measures acquired in ULAB regarding real end-users are always self-reported measures, such as ratings, rankings, orders of preference, which are typically expressed through closed Likert scales. End users' answers to open questions, like "what do you think this user interface?", "what are the disadvantages of this screen?", could be analysed today with Natural Language Processing (NLP), such as sentiment analysis or opinion mining [30] to determine the level of acceptance of a particular user interface. Self-reported questions always suffer from intrinsic bias like self-selection, perception bias, discrepancy between what the end user perceives and what is the real situation, etc. The aforementioned extensions could open the door to capture in principle any measure from the end user and incorporate it in a quality model. For example,

measure acquired through captors and sensors, like head-mounted displays, eye trackers, muscle sensors, EEG signals, but this is another story...

Systematic Literature Review on Visual Design Measures

A.1 Identification Phase

Initial Search

The following semantic query has been introduced in different digital libraries to identify potentially relevant papers to the field of experimental studies on GUI visual design:

$$Q = ("Visual" \text{ AND } "Metric" \text{ AND } "User" \text{ AND } "Interface")$$

The semantic query was ran on both single-publisher libraries (*e.g.*, ACM Digital Library) and multi-publisher engines (*e.g.*, Google Scholar). The 6 digital libraries selected to carry out the search process of this SLR are the following:

- ACM Digital Library [ACM]: <http://portal.acm.org/>.
- IEEE Xplore [IEEE]: <http://ieeexplore.ieee.org/>.
- Science Direct [SD]: <http://www.sciencedirect.com/>.
- SpringerLink [SPL]: <https://link.springer.com/>.
- Google Scholar [GS]: <http://scholar.google.com/>.
- DBLP Complete Search [DBLP]: <http://www.dblp.org/search/>.

The selected digital libraries use their own syntax and have their own advanced search options, therefore the chosen search string has been adapted for each search engine as shown in Table A.1.

All searches were performed between 24th March 2020 and 20th April 2020. The search results were sorted in decreasing order of relevance. For

Digital library	Adapted query string	Advanced search options
ACM	Visual AND Metric AND User AND Interface	Search items from: ACM Guide to Computing Literature Search within: Anywhere Publication date: All dates
IEEE	Visual AND Metric AND User AND Interface	Each term was searched in: Full text & metadata
SD	Visual AND Metric AND User AND Interface	Find articles with the terms of the query
SPL	Visual AND Metric AND User AND Interface	Find resources with all of the words of the query
GS	Visual AND Metric AND User AND Interface	Find articles containing all of the words of the query
DBLP	Visual Metric User Interface	Combined DBLP search

Table A.1: Syntax and advanced search option used for each selected digital library (adapted from [112])

each search, the first 150 results were inspected to form the initial set. Supposing that the accuracy of each search engine ranking algorithm is effective, we can assume that considering the first 150 results yields a relevant and representative result set ([31]). However, it should be noted that for the DBLP digital library, only 2 results were found which have both been included for further screening. The total number of references found after the identification phase is therefore equal to 752 (150 papers on 5 digital libraries + 2 from DBLP).

A.2 Screening Phase

The initial set of publications contained irrelevant results that had to be removed. Each paper was evaluated with respect to its relevance to visual design using criteria related to form and content, as follows:

Form : Only the papers written in English that underwent a peer-review process and for which the full text was available were retained. Therefore, research papers published in peer-reviewed journals, conferences, symposiums and workshops were included, while books, Ph.D. theses, master theses and patents were excluded.

Content : Only the papers that explicitly introduced a metric for GUI visual design or discussed about GUI aesthetics and usability were retained. Publications linked to the visual aspect of the interface or to the user interface layout were also included for screening. All off-topic articles were therefore excluded.

After excluding articles according to form and content criteria, a total number of 670 irrelevant articles were excluded, leaving $752 - 670 = 82$ papers. These 82 papers were therefore examined in the third phase of the Systematic Literature Review.

A.3 Eligibility Phase

Among the screened papers, only those that had a formula for a metric related to visual design were retained. For this purpose, the full text of the 82 articles was reviewed in search of a formula or an algorithm. After the addition of this eligibility criterion, 59 publications that did not have a formula related to a visual design metric were therefore excluded, leaving to 23 the number of remaining publications.

At the end of this phase, 9 duplicates coming from similar results on different digital libraries were eliminated, leaving therefore 14 publications. The detailed process is described by a PRISMA diagram [82] as shown in the Figure A.1.

From these 14 articles, a selective backward snowballing was carried out in order to potentially discover additional references that meet all the previously defined criteria.

A.4 Backward Snowballing

In order to complete our set of articles, a backward snowballing was carried out. Indeed, since the semantic search was $Q = ("Visual" \text{ AND } "Metric")$

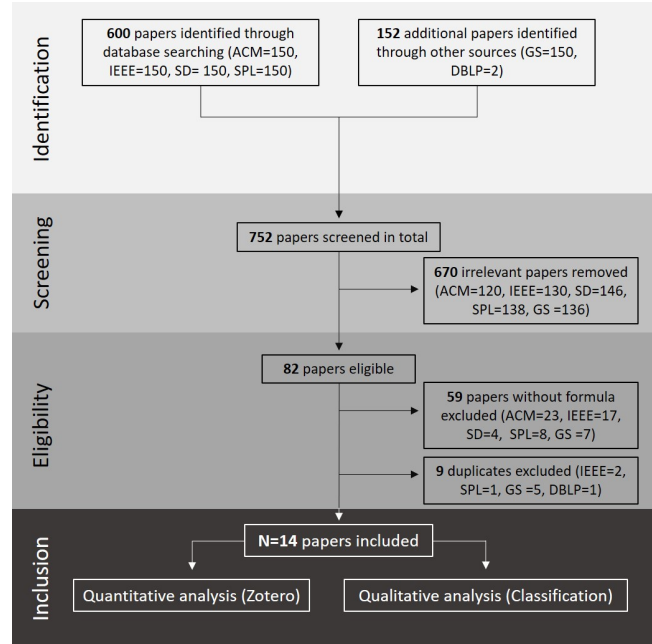


Figure A.1: The four-phase flow PRISMA diagram of our SLR examination [161]

AND "User" AND "Interface"), other relevant references using a synonym for visual design like aesthetics might not have been presented by the search engine algorithms. For these reasons, a selective backward snowballing was carried out by screening through the reference list of each of the 14 previously selected articles. This backward snowballing is selective because only the sources containing the word “aesthetics” have been screened.

It is noteworthy that to be rigorous, it would have been necessary to screen all the sources of the set of selected articles as well as the realization of a forward snowballing, which is a time- and resource-consuming endeavour. For this reason, only a selective backward snowballing will be performed.

The search for new sources on the basis of the word “aesthetics” identified 51 new articles to be screened. Among them, 36 papers were not eligible according to the conditions defined in section A.2 and A.3. In addition to this, 4 publications were also excluded because they were duplicates of our original articles set. Therefore, this selective backward snowballing allowed us to identify 11 new references. This step is identified as the 1st iteration of our snowballing process.

A second iteration of the selective backward snowballing was carried out on newly discovered references. For each new article discovered, the references were studied in order to screen new publications containing the word

“aesthetics”. With this second iteration, a total of 135 papers were screened, 51 were not relevant according to the criteria of content or form, 52 did not contain a formula, 24 duplicates were found as well as 8 new articles eligible for the study. A third and last iteration was done but no new papers were found.

At the end of this Systematic Literature Review, $14 + 11 + 8 = 33$ references introducing a formula related to GUI visual design were identified and downloaded for further analysis.

A.5 Inclusion Phase

The quantitative aspect of our corpus of papers was verified by using Zotero⁶ and by generating summary statistics.

A collection of papers was created on Zotero and each reference was compiled in the online library with the full-text PDF of each source attached. Then, the metadata from each source has been completed in order to analyze them statistically. The 33 references that have been included are available online in the Zotero library¹

The quality of each source will be discussed with more details in the next chapter. Different qualitative aspects will be discussed, such as the number of authors per article or the degree of validation of the different formulas presented in the papers.

A.6 Summary of the Systematic Literature Review

The evolution of the number of references according to the different steps of the SLR is presented in the summary table below (Table A.2).

⁶<https://www.zotero.org/>

¹https://www.zotero.org/groups/2463417/visual_design_metrics/library

Source	Query	Screened	Excluded	Duplicates	Included
1. ACM	17,406	150	143	0	7
2. IEEE	23,504	150	147	2	1
3. SD	14,119	150	150	0	0
4. SPL	28,197	150	146	1	3
5. GS	248 000	150	143	5	2
6. DBLP	2	2	0	1	1
Total before snowballing	331230	752	729	9	14
7. Backward Snowballing 1 st iteration	-	51	36	4	11
8. Backward Snowballing 2 nd iteration	-	135	103	24	8
Total after snowballing	-	938	868	37	33

Table A.2: Summary table of the number of references analyzed at the different steps of the SLR (adapted from [161])

Visual Design Metrics

B

The Systematic Literature Review performed previously allows us to study, characterize, compare and find perspectives for the different GUI visual design metrics found in the articles. The quality of the sources will also be discussed through a meta-analysis of the papers.

For each metric, the number of citations per metric (NoC) is computer. The number of citations allows to rank the metrics by popularity in the literature. Each formula is also accompanied by a strength of evidence level (SoE) depending on the degree of verification of the formula.

The SoE is defined below:

- If only a theoretical formula is given, SoE=1.
- If the formula is verified by at least one study with an example on a website, SoE=2.
- If the formula has been validated empirically with a correlation below 50%, SoE=3.
- If the formula has been validated empirically with a correlation between 50% and 90%, SoE=4.
- If the formula has been validated empirically with a correlation above 90%, SoE=5.

B.1 Balance

Balance (NoC=21) can be defined as the distribution of optical weight in a picture. In screen design, balance is achieved by providing an equal weight to screen elements, left and right, top and bottom ([103]). The different formulas existing in the literature for balance are detailed here after.

A first formula (SOE=5) of balance (BM) is given below (equation B.1) [103, 102, 101, 105, 33, 9, 123, 66, 169, 106, 97, 171]:

$$BM = 1 - \frac{|BM_{vertical}| + |BM_{horizontal}|}{2} \in [0, 1] \quad (B.1)$$

where $BM_{vertical}$ and $BM_{horizontal}$ designate respectively the vertical and horizontal balances. These terms depend on the visual weight of the interface elements [103] which, in turn, depends on the area, color and shape of the various objects on the interface. The distribution of the visual weight in the different quadrants influences the overall balance of the interface.

A second formula (SoE=5) exists in order to measure GUI visual balance. The formula is given by equation B.2 [103, 100]:

$$BM = (w_L - w_R, w_T - w_B) \quad (B.2)$$

Where BM is the measure of balance, w_L , w_R , w_T and w_B are respectively the weights on the left, right, top and bottom sides of the interface. The weights depend on the area of the interface objects and the distance between their vertical/horizontal central line and the vertical/horizontal axis respectively. One advantage of this formula is the ability to compare the balance on each sides of the interface. However, this formula only considers the area of the different objects as influencing factor of balance, without considering any influence of color and shape.

Another formula to compute balance is proposed by [136] and is given by equation B.3 (SoE=2):

$$Balance = 200 \times \frac{\text{weight of side one}}{\text{weight of side one} + \text{weight of side two}} \quad (B.3)$$

The next formula (SoE=4) consists in a low-level approach by comparing the balance of an image pixel by pixel, where black pixels are given a mass of one and white pixels given a mass of zero. The equation of balance (b) is given by equation B.4 [14]:

$$b = \left(\frac{x_b}{w}, \frac{y_b}{h} \right) \quad (B.4)$$

Where b is a set of normalized coordinates between zero and one, w is the image width (in pixels), (x_b, y_b) is the center of balance and h is the image height (in pixels). An image with a perfectly centered balance has a b value equal to (0.5, 0.5). This formula has been validated empirically with a correlation of 0.79.

The fifth formula of balance (SoE=5) was developed for the particular case of text-overlaid images. Text-overlaid image consists of a large-size background image with a small number of texts overlaid on it. This formula is derived from the fourth formula of balance (equation B.4) but takes into account the color of the pixels via the formula presented in section B.16. It is decomposed in horizontal (B_H) and vertical balance (B_V) as shown below (equations B.5 and B.6) [73]:

$$B_H = 1 - \left| 2 \times \frac{x_b}{w} - 1 \right| \quad (B.5)$$

$$B_V = 1 - \left| 2 \times \frac{y_b}{h} - 1 \right| \quad (B.6)$$

B_H and B_V take values between 0 and 1. The closer to 1 the value of B_H (B_V), the better the horizontal (vertical) visual balance obtained. w is the image width in pixels, (x_b, y_b) is the center of balance (depending on color) and h is the image height in pixels.

In this sixth formula (SoE=4), the number and size of objects for each quarter of the screen are computed to obtain the Total Balance Complexity (TBC) value. The TBC formula is given in equation B.7 [5]:

$$TBC = 1 - (0.5 * BQn + 0.5 * BQs) \quad (B.7)$$

Where BQn and BQs are the number and size of objects for each quarter of the screen. These terms are defined below:

$$BQn = \frac{\sum_{k=1}^6 \frac{BQn_i}{BQn_j}}{6}$$

$$BQs = \frac{\sum_{k=1}^6 \frac{BQs_i}{BQs_j}}{6}$$

BQn_i and BQn_j represent the number of objects in the i^{th} and j^{th} quarters and BQs_i and BQs_j represent the sum of sizes of objects in the i^{th} and j^{th} quarters. The range of BQn is [0,1] where 0 means unbalanced and 1 means fully balanced in terms of number of objects and the range of BQs is [0,1] where 0 means unbalanced and 1 means fully balanced in terms of object size. This formula has been empirically validated with a correlation of 0.825.

Another balance formula is specifically applied to Android applications [128]. The metric calculates the margins between GUI elements and their neighboring elements or borders. For each GUI element, the distances to the four horizontal and vertical neighbors are considered. The neighbor is defined as the closest GUI element or group border in the respective direction (*i.e.*, left, right, top and bottom).

In order to compute the balance value (B_s), the formula given by equation B.8 uses the mean horizontal ($\overline{balh_s}$) and vertical margins ($\overline{balv_s}$) and weights them equally:

$$B_s = \frac{\overline{balh_s} + \overline{balv_s}}{2} \quad (B.8)$$

The eighth formula (SOE=1) to measure balance is proposed by [145]. The formula is computed as a comparison between the GUI objects center of mass and the screen center and is given by equation B.9 [171]:

$$BM = 1 - \frac{2 * \sqrt{(X_{CS} - X_{CM})^2 + (Y_{CS} - Y_{CM})^2}}{\sqrt{W^2 + H^2}} \quad (B.9)$$

Where W and H are the width and height of the interface, (X_{CS}, Y_{CS}) are the coordinates of the center of the screen and (X_{CM}, Y_{CM}) are the coordinates of the center of mass of the GUI objects.

The last formula for balance (SoE=2) is inspired by the concept of an equilibrium between two areas separated by a vertical line. Balance is achieved by giving similar visual weight to both sides. Since the quantification of this weight is subjective, a heuristic is used and considers weight as the accumulated height of the components on each side. The formula is given by equation B.10 [87]:

$$Ba = \frac{n_{hac}}{n_c} \left(1 - \frac{|h_l - h_r|}{h_l + h_r} \right) \quad (B.10)$$

Where n_{hac} is the number of components horizontally aligned, n_c is the number of components in the interface, h_l is the accumulated height of left side components and h_r is the accumulated height of right side components.

B.2 Equilibrium

Equilibrium (NoC=10) on a screen is accomplished by centering the layout itself. A layout is in equilibrium when its center coincides with the center of the frame [103, 105].

The first formula (SoE=5) to compute equilibrium is computed as the difference between the center of mass of the displayed elements and the physical center of the screen and is given by the equation B.11 ([103, 102, 101, 173, 105, 66, 106, 97]):

$$EM = 1 - \frac{|EM_x| + |EM_y|}{2} \in [0, 1] \quad (B.11)$$

where EM_x and EM_y are the equilibrium components along the x-axis and y-axis respectively.

The equilibrium components depend on the coordinates of the centers of the interface objects (x_i, y_i) , the center of the frame (x_c, y_c) , the surface area of the objects (a_i) , the number of object on the frame (n) and the width and height of the frame $(b_{frame}$ and h_{frame} respectively).

The second formula (SoE=5) to measure equilibrium is given by the equation B.12 ([100, 103]):

$$\begin{aligned} EM &= (x_c, y_c) - (x_0, y_0) \\ \text{or} \\ EM &= (x_c - y_0) - (y_c - y_0) \end{aligned} \quad (B.12)$$

where (x_0, y_0) is the center of the layout, and (x_c, y_c) is the center of the frame. The layout is composed of a set of objects of areas a_1, a_2, a_3 , etc. at points $(x_1, y_1), (x_2, y_2), (x_3, y_3)$, etc. in the $X-Y$ plane. By using this formula, the equilibrium is reached when $EM = (0,0)$.

B.3 Symmetry

Symmetry (NoC=17) can be represented as an axial duplication: when an element of screen design is present on one side of the center line, it is exactly replicated on the other side. The center line can be either vertical, horizontal or radial. Radial symmetry consists of equivalent elements balanced about two or more axes that intersect at a central point (figure B.1).

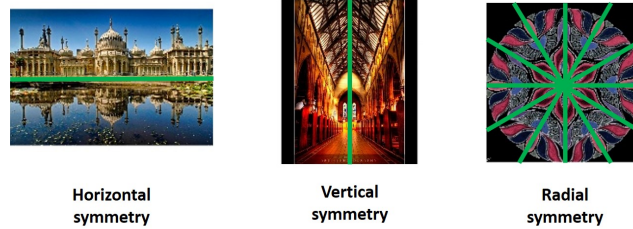


Figure B.1: Example of axes of symmetry [162]

The first formula for symmetry (SoE=5) decomposes the global symmetry into 3 components: vertical, horizontal, and diagonal. The formula is given by the equation B.13 ([104, 102, 142, 173, 105, 66, 106, 97, 131]):

$$SYM = 1 - \frac{|SYM_{vertical}| + |SYM_{horizontal}| + |SYM_{radial}|}{3} \in [0, 1] \quad (B.13)$$

$SYM_{vertical}$, $SYM_{horizontal}$ and SYM_{radial} are respectively the vertical, horizontal, and radial symmetries with:

$$\text{SYM}_{\text{vertical}} = \frac{|X'_{UL} - X'_{UR}| + |X'_{LL} - X'_{LR}| + |Y'_{UL} - Y'_{UR}| + |Y'_{LL} - Y'_{LR}| + |H'_{UL} - H'_{UR}| + |H'_{LL} - H'_{LR}| + |B'_{UL} - B'_{UR}| + |B'_{LL} - B'_{LR}| + |\Theta'_{UL} - \Theta'_{UR}| + |\Theta'_{LL} - \Theta'_{LR}| + |R'_{UL} - R'_{UR}| + |R'_{LL} - R'_{LR}|}{12}$$

$$\text{SYM}_{\text{horizontal}} = \frac{|X'_{UL} - X'_{LL}| + |X'_{UR} - X'_{LR}| + |Y'_{UL} - Y'_{LL}| + |Y'_{UR} - Y'_{LR}| + |H'_{UL} - H'_{LL}| + |H'_{UR} - H'_{LR}| + |B'_{UL} - B'_{LL}| + |B'_{UR} - B'_{LR}| + |\Theta'_{UL} - \Theta'_{LL}| + |\Theta'_{UR} - \Theta'_{LR}| + |R'_{UL} - R'_{LL}| + |R'_{UR} - R'_{LR}|}{12}$$

$$\text{SYM}_{\text{radial}} = \frac{|X'_{UL} - X'_{LR}| + |X'_{UR} - X'_{LL}| + |Y'_{UL} - Y'_{LR}| + |Y'_{UR} - Y'_{LL}| + |H'_{UL} - H'_{LR}| + |H'_{UR} - H'_{LL}| + |B'_{UL} - B'_{LR}| + |B'_{UR} - B'_{LL}| + |\Theta'_{UL} - \Theta'_{LR}| + |\Theta'_{UR} - \Theta'_{LL}| + |R'_{UL} - R'_{LR}| + |R'_{UR} - R'_{LL}|}{12}$$

where $X'_j, Y'_j, H'_j, B'_j, \Theta'_j$ and R'_j are respectively the normalised values of $X_j, Y_j, H_j, B_j, \Theta_j$ and R_j with:

$$X_j = \sum_i^{n_j} |x_{ij} - x_c| \quad j = UL, UR, LL, LR$$

$$Y_j = \sum_i^{n_j} |y_{ij} - y_c| \quad j = UL, UR, LL, LR$$

$$H_j = \sum_i^{n_j} h_{ij} \quad j = UL, UR, LL, LR$$

$$B_j = \sum_i^{n_j} b_{ij} \quad j = UL, UR, LL, LR$$

$$\Theta_j = \sum_i^{n_j} \left| \frac{y_{ij} - y_c}{x_{ij} - x_c} \right| \quad j = UL, UR, LL, LR$$

$$R_j = \sum_i^{n_j} \sqrt{(x_{ij} - x_c)^2 + (y_{ij} - y_c)^2} \quad j = UL, UR, LL, LR$$

UL, UR, LL, and LR stand for upper-left, upper-right, lower-left, and lower-right respectively. (x_{ij}, y_{ij}) and (x_c, y_c) are the coordinates of the centers of object i on quadrant j and the frame. b_{ij} and h_{ij} are the width and height of the object and n_j is the total number of objects on the quadrant [104].

Overall, the symmetry depends on the coordinates on the x and y-axis of the center of the different objects in relation to the center of symmetry, the size of the objects (their width and height) and the total number of objects present in each quadrant of the interface.

Another formula (SoE=5) for measuring symmetry is given by equation B.14 [103, 100]:

$$SM = \{[g_{UL} - g_{LL}], [g_{UR} - g_{LR}], [g_{UL} - g_{UR}], [g_{LL} - g_{LR}], [g_{UL} - g_{LR}], [g_{UR} - g_{LL}]\} \quad (B.14)$$

where g_j is the weight of the quadrant j . The weight of a layout is the algebraic sum of the weight of its components:

$$g = \sum_i (|x_i - x_c|, |y_i - y_c|, l_i, b_i)$$

l_i is the length of an object, b_i is the breadth of the object, (x_i, y_i) is the center of the object, and (x_c, y_c) is the center of the frame.

The vertical symmetry is obtained if: $g_{UL} - g_{UR} = (0,0,0,0)$ and $g_{LL} - g_{LR} = (0,0,0,0)$. The horizontal symmetry is obtained if: $g_{UL} - g_{LL} = (0,0,0,0)$ and $g_{UR} - g_{LR} = (0,0,0,0)$. Finally, the radial symmetry is obtained if: $g_{UL} - g_{LR} = (0,0,0,0)$ and $g_{UR} - g_{LL} = (0,0,0,0)$ [103].

A third formula (SoE=4) for computing symmetry of a layout considers the similarity of pixels on opposite sides of an axis of reflection. This approach compares each half of an image pixel by pixel [14]. The pixels that are close to the axis of reflection have a higher influence on the overall impression of symmetry than the ones which are further away. The formula is given by equation B.15 [14, 15, 16]:

$$s = \frac{2}{3mn} \sum_{i=1}^m \sum_{j=1}^n X_{ij} \left(1 + \frac{j-1}{n-1}\right) \quad (B.15)$$

Where m is the pixel length of the image dimension that is parallel to the axis of reflection, n is the number of comparisons required in each row or column of pixels, X_{ij} is a binary variable which is equal to one when the pixel pairs are the same and zero when they are opposite, i is the pixel row and j is the pixel column. This formula has been validated empirically with a correlation of 0.78. The ratings of the subjects of the study corresponded with the s values of the images presented in the study [14].

The fourth formula of symmetry (SoE=5) is applied to an interface composed of text-overlaid images. This formula is derived from the third formula of symmetry (equation B.15) but takes into account the color attributes of the pixels via the formula presented in section B.16. Two pixel blocks with a small difference in the HSV space are considered to be more symmetrical than two

pixels with a large HSV difference. Therefore, when comparing two corresponding pixel blocks for measuring symmetry, the difference value of symmetry is considered to be proportional to the HSV (hue, saturation, and value) color difference between the two pixel blocks. The computational models for the horizontal symmetry (S_H), vertical symmetry (S_V), and radial symmetry (S_R) is given by equation B.16, B.17 and B.18 respectively [73]:

$$S_H = \frac{2}{3w(h/2)} \sum_{i=1}^w \sum_{j=1}^{h/2} (1 - \Delta C_{ij-i'j'}) \left(1 + \frac{(h/2) - j}{(h/2) - 1} \right) \quad (\text{B.16})$$

$$S_V = \frac{2}{3w(h/2)} \sum_{i=1}^h \sum_{j=1}^{w/2} (1 - \Delta C_{ij-i'j'}) \left(1 + \frac{(w/2) - j}{(w/2) - 1} \right) \quad (\text{B.17})$$

$$\begin{aligned} S_R = & \frac{8}{6wh} \sum_{i=1}^{h/2} \sum_{j=1}^{w/2} (1 - \Delta C_{ij-i'j'}) \left(1 + \left(\frac{(w/2) - j}{(w/2) - 1} + \frac{(h/2) - i}{(h/2) - 1} \right) / 2 \right) \\ & + \sum_{i=(h/2)+1}^h \sum_{j=1}^{w/2} (1 - \Delta C_{ij-i'j'}) \left(1 + \left(\frac{(w/2) - j}{(w/2) - 1} + \frac{i - 1 - (h/2)}{(h/2) - 1} \right) / 2 \right) \end{aligned} \quad (\text{B.18})$$

Where w and h are respectively the width and height of the image in pixel blocks, (i', j') is the coordinates of a pixel on the opposite side of a given axis of reflection for a pixel situated in (i, j) and ΔC is defined in the section B.16. These formula have been validated with a correlation of 0.976 for S_V , 0.849 for S_H and 0.833 for S_R .

Miniukovich *et al.* [95] developed a formula to measure vertical symmetry at the pixel scale (SoE=3). First, an algorithm is applied to detect the image contours. Then, the algorithm keeps only the vertical component of detected contours. The number of pixels composing the contour is reduced by taking a contour pixel and dismissing others in the 3-pixel radius and taken as key points. For each key point, the algorithm looks for a match in the 4-pixel radius area across the central axis. Then, the ratio of matches k_{sym} to all key points k_{all} is taken and normalized by the probability of key point match due to chance. The probability of incidental match depends on the number of all key points, the size of match-search area (S_a , which is a 4-pixel radius area) and the size of the screenshots (S_s). The normalized ratio (Sym_{norm}) considered as a measure of symmetry and is expressed through equation B.19 [95]:

$$Sym_{norm} = \frac{k_{sym}}{k_{all}} * \left(\frac{(k_{all} - 1) * S_a}{S_s} \right)^{-1} \quad (\text{B.19})$$

B.4 Sequence

In visual design, the sequence (NoC=11) refers to the arrangement of objects in a layout in a way that facilitates the movement of the eye through the displayed information. Usually, the reader's eye starts from the upper left and moves back and forth across the display to the lower right. Moreover, the eyes tends to move from big objects to small objects, from bright colors to subdued colors, from color to black and white, and from irregular shapes to regular shapes (Figure B.2).

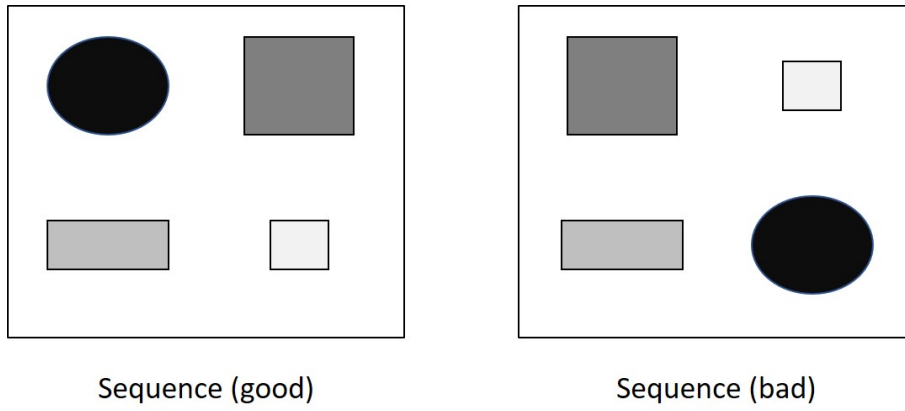


Figure B.2: Comparison between a good and a bad sequence [104]

The first measure for sequence (SoE=5) expresses how information in a display is ordered according to a reading pattern that is common in Western cultures and is given by the equation B.20 [104, 102, 105, 9, 66, 106, 97, 131]:

$$SQM = 1 - \frac{\sum_{k=A,C,S} \sum_{j=UL,UR,LL,LR} |q_j - v_j^k|}{8} \in [0, 1] \quad (\text{B.20})$$

with $(q_{UL}, q_{UR}, q_{LL}, q_{LR}) = (4, 3, 2, 1)$: the weights of the quadrant as it should be in order to achieve the best sequence

$$v_j^k = \begin{cases} 4 & \text{if } w_j^k \text{ is the largest in } w^k \\ 3 & \text{if } w_j^k \text{ is the 2}^{nd} \text{ largest in } w^k \\ 2 & \text{if } w_j^k \text{ is the 3}^{rd} \text{ largest in } w^k \\ 1 & \text{if } w_j^k \text{ is the smallest in } w^k \end{cases}$$

$$j = UL, UR, LL, LR ; k = A, C, S$$

v_j^k is the effective weighting of each quadrant with:

$$w_j^A = q_j \sum_i^{n_j} a_{ij} \quad j = UL, UR, LL, LR$$

$$w_j^C = q_j \sum_i^{n_j} |c_{ij} - c_{frame}| \quad j = UL, UR, LL, LR$$

$$w_j^S = q_j \sum_i^{n_j} s_{ij} \quad j = UL, UR, LL, LR$$

$$w^k = (w_{UL}^k, w_{UR}^k, w_{LL}^k, w_{LR}^k)$$

where UL, UR, LL, and LR stand for upper-left, upper-right, lower-left, and lower-right (the 4 quadrants of the interface). A, C and S are respectively the size, the color, and the shape of the objects. a_{ij} , c_{ij} , and s_{ij} are respectively the area, color, and shape of object i on quadrant j [104].

Another study by [103] (SoE=5) defined sequence as an arrangement of objects in a layout in a way that facilitates the eye movement. In order to improve the sequence, this study suggests to manipulate the user's attention by using eye attractors and visual paths. The formula of sequence depends on the visual paths and is given by the equation B.21 [103, 100]:

$$SQM = (p, d) \quad (B.21)$$

with the following definitions:

$$p = \begin{cases} 1 & \text{If a path exists} \\ 0 & \text{Otherwise} \end{cases}$$

$$d = n_e$$

This third formula (SoE=2) to measure sequence is used in the context of mobile applications. The proposed formula to calculate this metric is given by equation B.22 [142]:

$$SQM = 1 - \frac{\sum_j (q_j \sum_{i=1}^n N_{i,j})}{4n} \in [0, 1] \quad j = UL, UR, LL, LR \quad (B.22)$$

where $N_{i,j}$ is the number of object on the quadrant j and n is the total number of objects on the mobile user interface. Each quadrant is given a weighting with $q_{UL} = 4$, $q_{UR} = 3$, $q_{LL} = 2$, $q_{LR} = 1$.

B.5 Cohesion

The cohesion (NoC=7) is linked to the aspect ratio, which refers to the relationship of width to height. In screen design, similar aspect ratios promote cohesion. Changing the aspect ratio of a visual field may affect eye movement patterns sufficiently to account for some of the performance differences. The aspect ratio of a visual field should stay the same during the scanning of a display [104].

Cohesion is a measure of how cohesive the screen is and a first formula for cohesion is given by equation B.23 (SoE=5) [104, 102, 105, 106, 131]:

$$CM = \frac{|CM_{fl}| + |CM_{lo}|}{2} \in [0, 1] \quad (\text{B.23})$$

CM_{fl} is a relative measure of the ratios of the layout (l) and the frame of the screen (f) with:

$$CM_{fl} = \begin{cases} t_{fl} & \text{if } t_{fl} \leq 1 \\ \frac{1}{t_{fl}} & \text{Otherwise} \end{cases}$$

CM_{lo} is a relative measure of the ratios of the objects (o) and layout (l) with:

$$CM_{lo} = \frac{\sum_i^n f_i}{n}$$

with:

$$t_{fl} = \frac{\frac{h_{layout}}{b_{layout}}}{\frac{h_{frame}}{b_{frame}}}$$

$$f_i = \begin{cases} t_i & \text{if } t_i \leq 1 \\ \frac{1}{t_i} & \text{Otherwise} \end{cases}$$

with:

$$t_i = \frac{\frac{h_i}{b_i}}{\frac{h_{layout}}{b_{layout}}}$$

where b_i and h_i , b_{layout} and h_{layout} , and b_{frame} and h_{frame} are the widths and heights of object i , the layout, and the frame, respectively and n is the number of objects on the frame [104].

The second formula (SoE=4) measures the screen layout cohesion (SLC). The screen layout cohesion is composed of 4 metrics of cohesion based on 17

aesthetic and structural attributes (GLC, UGLC, WLC and SR). The 4 metrics are described in greater details in [4]

The formula of SLC is given by equation B.24:

$$SLC = \frac{GLC * W1 + UGLC * W2 + WLC + SR * W1}{2 + W1} \quad (B.24)$$

Where *GLC* is the Group Layout Cohesion, *UGLC* is the UnGroup Layout Cohesion, *WLC* is the Widget Layout Cohesion and *SR* is the Semantic Relatedness (SR). The values of *GLC* and *UGLC* are weighted based on the number of grouped and ungrouped widgets to the total number of widgets on a given user interface (*W1*, *W2*), respectively. In addition, *SR* is weighted by *W1* because *SR* represents the semantic relatedness of widget groups on a given UI.

In a study performed by Kokol *et al.* [72], cohesion was measured in terms of relationship between data on a screen. If all data on the screen belongs to the same entity, the cohesion is high. If data on one screen are attributes of two or more related entities, cohesion is considered as medium and if data do not relate directly, cohesion is considered as low. This is presented by equation B.25(SoE=1):

$$COH_i = \frac{(W_e N_e + W_{Re} N_{Re} + W_u N_u)}{(N_e + N_{Re} + N_u)} \quad (B.25)$$

where COH_i is the cohesion for one entry screen, N_e is the number of data belonging to the same entity, N_{Re} is the number of data (together) which belong to related entities, N_u is the data which belongs to unrelated entities. W are the weights (arbitrary set by the researchers). Finally the value of cohesion of an entire interface is given as:

$$COH = \sum (COH)_i$$

B.6 Unity

According to [104], unity (NoC=9) allows the elements of visual design to seem like they belong together, to dovetail so completely that they are seen as one thing. Unity is achieved by using similar sizes, shapes, or colors for related information and leaving less space between elements of a screen than the space left at the margins.

The first formula (SoE=5) considers unity as the extent to which the screen elements seem to belong together and is given by equation B.26 [104, 102, 101, 105, 123, 9, 106, 131].

$$UM = \frac{|UM_{form}| + |UM_{space}|}{2} \in [0, 1] \quad (B.26)$$

Where UM_{form} is the extent to which the objects are related in size with:

$$UM_{form} = 1 - \frac{n_{size} - 1}{n}$$

Ngo [104] went further in the definition of UM_{form} where it depends on the size, shape, and color of objects:

$$UM_{form} = 1 - \frac{n_{size} + n_{color} + n_{shape} - 3}{3n}$$

UM_{space} is a relative measure of the space between groups and that of the margins:

$$UM_{space} = 1 - \frac{a_{layout} - \sum_i^n a_i}{a_{frame} - \sum_i^n a_i}$$

Where a_i , a_{layout} , and a_{frame} are the areas of object i , the layout, and the frame, respectively. n_{size} , n_{color} , and n_{shape} are the numbers of sizes, colors, and shapes used respectively, and n is the number of objects on the frame.

The second formula (SoE=2) is proposed by [142] in the context of mobile applications. As the screen size of the smartphone is not the same as of a computer, it is necessary to adapt the information quantities and the form of information. Unity is the extent to which the components are correlated in size and number of space between groups and margins. A good unity is obtained by using the optimum number of size component (minimize the uses of different sizes in the mobile interface) and leaving less space between objects. The formula is given by equation B.27 :

$$UM = 1 - \left(0.5 \left[\frac{|N_{size} - 1|}{n} + \frac{|a_{sc} + \sum_i^n a_i|}{2a_{MUI}} \right] \right) \in [0, 1] \quad (B.27)$$

where N_{size} is the number of various sizes belong into used objects, n is the number of objects, a_{MUI} is the area of the mobile interface, a_{sc} is the area of the screen and a_i is the area of the interactive object i .

B.7 Proportion

In screen design, aesthetically pleasing proportions should be considered for major components of the screen, including windows and groups of data and text in order to improve the user experience.

The following shapes (with their proportions) are considered as aesthetically pleasing [106]:

- Square (1:1)

- Square root of two (1:1.414)
- Golden rectangle (1:1.618)
- Square root of three (1:1.732)
- Double square (1:2)

Proportion (NoC=5) is the comparative relationship between the dimensions of the screen components and proportional shapes and is given by the equation B.28 (SoE=5) [104, 102, 105, 106]:

$$PM = \frac{|PM_{object}| + |PM_{layout}|}{2} \in [0, 1] \quad (B.28)$$

PM_{object} is the difference between the proportion of the objects and the closest proportional shapes. PM_{layout} is the difference between the proportion of the layout and the closest proportional shape. Overall, proportion depend on the width and height of the interface objects, the width and height of the layout and the proportion of the interface shapes.

A second formula (SoE=1) of proportion (V_{ar}) proposed by [56] is given below (equation B.29):

$$V_{ar} = 1 - \frac{|Z_i - R|}{R} \in [0, 1] \quad (B.29)$$

with Z_i : the ratio of width and height of the major content elements and groups of elements on a page and R : the golden ratio between width and height and equal to $2/(1 + \sqrt{5}) = 0.618$.

B.8 Simplicity

Simplicity (NoC=9) is a combination of elements that results in ease on comprehension of the meaning of a pattern. Simplicity in screen design is achieved by optimizing the number of elements on a screen and minimizing the alignment points.

It can be easily measured by the equation B.30 (SoE=5) [104, 101, 102, 105, 33, 123, 106, 48]):

$$SMM = \frac{3}{n_{vap} + n_{hap} + n} \in [0, 1] \quad (B.30)$$

Where n_{vap} and n_{hap} are the numbers of vertical and horizontal alignment points and n is the number of objects on the frame.

Another formula (SoE=4) measures the vertical and horizontal alignment of objects in two levels: a group level (Local Alignment) and a screen level

(Global Alignment). These two alignment levels are then combined to calculate the Total Screen Alignment Complexity (TAC) (equation B.31) [5]:

$$TAC = AC * weight1 + SA * weight2 \quad (B.31)$$

where AC is the Alignment Complexity, SA is the Screen Alignment, *weight1* is the ratio of the number of grouped-objects to the total number of objects on the screen and *weight2* is the ratio of the number of ungrouped-objects to the total number of objects on the screen. The simplicity can be obtained by minimizing the TAC value.

B.9 Density

Density (NoC=9) is the extent to which the screen is covered with objects. Density is achieved by minimizing screen density levels. The density (DM) can be measured by the percentage of object on the entire frame by the following formula (equation B.32)(SoE=5) [104, 101, 102, 105, 106, 48]:

$$DM = 1 - 2 \left| 0.5 - \frac{\sum_i^n a_i}{a_{frame}} \right| \in [0, 1] \quad (B.32)$$

where a_i and a_{frame} are the areas of object i and the frame; and n is the number of objects on the frame.

A second formula (SoE=2) to measure density in the context of mobile applications has been proposed [142]. They found that in order to keep a user with a low motivation continually interested and inclined to make an effort to achieve a goal, mobile user interface with a low-density level was preferable on other types of interfaces. Density is computed by equation B.33:

$$DM = 0.5 \left| \frac{\sum_i^n a_i}{a_{MUI}} + \frac{a_{MUI}}{a_{sc}} \right| \quad (B.33)$$

where n is the number of interactive objects, a_{MUI} is the area of the mobile interface, a_{sc} is the area of the screen and a_i is the area of the interactive object i .

The third formula (SoE=4) of density developed by ([5]) takes into account the local density (LD_j) of a group of object j and the global density (GD) to obtain the Density-Complexity (DC) formula given below (equation B.34):

$$DC = \left(\frac{\sum_{j=1}^n LD_j}{n} \right) * W1 + GD * W2 \quad (B.34)$$

where W1 is the ratio of the area of groups to the screen area and W2 is the ratio of ungrouped area to the screen area. LD_j and GD are defined below:

$$LD_j = \frac{\sum_{i=1}^{\text{groupedobjects}} \text{Size of object } i \text{ in a group } j}{\text{Area of group } j}$$

$$GD = \frac{\sum_{k=1}^{\text{ungroupedobjects}} \text{Size of object } k}{\text{Area of ungrouped}}$$

This fourth formula (SoE=5) of density is specifically applied to Android applications. The density metric (D_s) shows how cluttered or empty the user interface is. The equation B.35 shows that the areas $A_{u,s}$, of all GUI elements U_s are summed up and related to the area A_s of the entire screen s , resulting in the density D_s [128]:

$$D_s = \frac{1}{A_s} \sum_{u=1}^{U_s} A_{u,s} \quad (\text{B.35})$$

B.10 Regularity

Regularity (NoC=11) is an uniformity of elements based on some principle or plan. Regularity can be achieved by establishing standard and consistently spaced horizontal and vertical alignment points for screen elements and by minimizing the alignment points ([104]). Experience in practical comparisons of real-world designs suggests that height and width, as well as bottom and right edge alignments are also factors in subjective judgments of visual regularity. A way to improve the usability is to improve the regularity because highly chaotic arrangements are more difficult to use [108].

It is possible to measure how regular the screen is thanks to the formula of regularity (RM) given by equation B.36 (SoE=5) ([104, 101, 102, 105, 106, 169, 131]):

$$RM = \frac{|RM_{alignment}| + |RM_{spacing}|}{2} \in [0, 1] \quad (\text{B.36})$$

$RM_{alignment}$ is the extent to which the alignment points are minimized and $RM_{spacing}$ is the extent to which the alignment points are consistently spaced. Overall, they depend on the numbers of vertical and horizontal alignment points, the number of distinct distances between column and row starting points and the number of objects on the frame.

A second formula (SoE=2) to measure regularity (RM) is given, in the context of mobile applications, by the equation B.37 ([142]):

$$RM = 1 - \frac{N_{av} + N_{ah} + N_{sp}}{3} \in [0, 1] \quad (\text{B.37})$$

where N_{av} is the number of vertical alignment points (number of rows), N_{ah} : the number of horizontal alignment points (number of columns), N_{sp} is the number of distinct distances between column and row starting points and n is the number of component of the mobile user interface.

Regularity is a structural metric that measures the uniformity or orderliness of the user interface layout. It is based on the principle that ease of use is hampered by very chaotic arrangements. The uniformity of the layout focuses on the spatial arrangement of the interface components. A user interface with a regularity score of 0.6 to 0.9 is considered as a good interface (see equation B.38) (SoE=2) [75]:

$$\text{Regularity} = 1 - \frac{(N_{size} + N_{top} + N_{left}) - A}{3C - A} \quad (\text{B.38})$$

where C is the number of components on the screen, N_{size} , N_{top} and N_{left} are the number of different sizes, top edge alignments and left edge alignments of components respectively. $A = \lceil 2\sqrt{C} \rceil + 1$ is an adjustment for the minimum number of possible alignments and sizes.

The fifth formula (SoE=2) to measure regularity is expressed as the layout uniformity of the interface in the paper of Noble and Constantine [108]. The layout uniformity is a synonym of regularity because they measure the same things (alignment and uniformity/orderliness of an user interface layout). They defined the layout uniformity (LU) as expressed in equation B.39 [108]:

$$LU = 100 * \left(1 - \frac{(N_h + N_w + N_t + N_l + N_b + N_r) - M}{6 * N_c - M} \right) \in [0, 100] \quad (\text{B.39})$$

where N_c is the total number of components on the screen, dialogue box, or other interface composite, and N_h, N_w, N_t, N_l, N_b and N_r are, respectively, the number of different heights, different widths, different top edge alignments, left edge alignments, bottom edge alignments, and right edge alignments of visual components. M is an adjustment for the minimum number of possible alignments and sizes that makes the value of LU range from 0 to 100:

$$M = 2 + 2 * \lceil 2\sqrt{N_{total}} \rceil$$

with " $\lceil \quad \rceil$ " denoting « smallest integer greater than ». Layout Uniformity is based on the argument that highly chaotic arrangements are more difficult to use. However, complete regularity is not the goal. Indeed, moderately high regularity characterizes good designs (LU between 0.5 and 0.9) ([108]).

In their paper, Riegler *et al.* describe the misalignment metric (SoE=4) [128]. This metric formula is specifically applied to Android applications.

In order to calculate the misalignment metric M_s , the mean number of horizontal ($\overline{numh_s}$), vertical ($\overline{numv_s}$) and central alignments ($\overline{numc_s}$) of pairs of neighboring elements of a screen s is calculated as follows (equation B.40) :

$$M_s = 1 - (\overline{numh_s} + \overline{numv_s} + \overline{numc_s}) \quad (B.40)$$

From this formula, it is possible to measure alignment or regularity, by calculating $1 - M_s$. The calculation of the terms of M_s is described below:

$$\overline{numh_s} = \frac{1}{A_s} \sum_{g=1}^{G_s} A_{g,s} \frac{numh_{g,s}}{num_{g,s}}$$

where $A_{g,s}$ is the area of a group g of GUI elements on the screen s , A_s is the area of the screen, G_s is the weighted arithmetic mean of the width and height elements over all groups, $numh_{g,s}$ is the number of horizontal alignments in a group g and $num_{g,s}$ is the number of possible alignments (horizontal, vertical and central) in a group. The mean number of vertical and central alignments $\overline{numv_s}$ and $\overline{numc_s}$ are calculated in a similar way.

Gonzales *et al.* [87] proposed formulas (SoE=2) for regularity and alignment consisting in 3 different alignment formulas which calculates the horizontal, vertical and perpendicular alignment separately and one formula of regularity. The first one, called linearity, measures the horizontally aligned arrangement of the element in the interface. Equation B.41 shows the linearity metric (Li):

$$Li = \frac{n_{hac} \times n_{va}}{n_c^2} \quad (B.41)$$

where n_{hac} is the number of components horizontally aligned with other components, n_{va} is the number of different vertical alignments and n_c is the number of components in the interface.

The second one, called sequentiality (Se), is the vertically analogous metric to linearity (equation B.42):

$$Se = \frac{n_{vac} \times n_{ha}}{n_c^2} \quad (B.42)$$

where n_{vac} is the number of components vertically aligned with other components, n_{ha} is the number of different horizontal alignments and n_c is the number of components on the interface.

The last formula of alignment proposed is called orthogonality. It considers the perpendicular alignment of elements of different shapes and sizes and ensures that every component is horizontally and vertically aligned. The expression of this metric (Or) is shown below (equation B.43):

$$Or = \frac{n_{vac} \times n_{hac}}{n_c^2} \quad (B.43)$$

Where n_{hac} is the number of components horizontally aligned with other components, n_{vac} is the number of components vertically aligned with other components and n_c is the number of components in the interface.

The formula of regularity proposed is shown below (equation B.44):

$$Re = \frac{n_{vac} \times n_{hac} \times (n_c - n_s + 1)}{n_c^3} \quad (B.44)$$

Where n_{vac} is the number of components vertically aligned with other components, n_{hac} is the number of components horizontally aligned with other components, n_c is the number of components in the interface and n_s is the number of different shapes.

B.11 Economy

The economy (NoC=5) consists in using as few display elements as possible. The objective is to convey a message in the simplest possible way. This can be done with minimal use of colors, technical displays and font style or visual design elements ([104]).

The formula of economy (SoE=5) is given below (equation B.46) ([101, 102, 105, 106, 131]):

$$ECM = \frac{1}{n_{size}} \in [0, 1] \quad (B.45)$$

Ngo went further in the definition of economy (ECM) with this formula [104]:

$$ECM = \frac{3}{n_{size} + n_{color} + n_{shape}} \in [0, 1] \quad (B.46)$$

with: n_{size} , n_{color} , and n_{shape} , the numbers of sizes, colors and shapes used respectively.

B.12 Homogeneity

The relative degree of homogeneity (NoC=5) of a composition is determined by how evenly the objects are distributed among the four quadrants of the screen. The degree of evenness is a matter of the quadrants that contain nearly equal numbers of objects ([104]).

Homogeneity (HM) is a measure of how evenly the objects are distributed among the quadrants and is given by equation B.47 (SoE=5) [104, 102, 105, 106, 142]:

$$HM = \frac{W}{W_{max}} \in [0, 1] \quad (B.47)$$

W is the number of different ways a group of n objects can be arranged for the four quadrants when n_j is the total number of objects in quadrant j .

$$W = \frac{n!}{\prod_{j=UL,UR,LL,LR} n_j} = \frac{n!}{n_{UL}!n_{UR}!n_{LL}!n_{LR}!}$$

W is maximum when the n objects are evenly allocated to the various quadrants of the screen and therefore:

$$W_{max} = \frac{n!}{\left(\frac{n}{4}\right)! \left(\frac{n}{4}\right)! \left(\frac{n}{4}\right)! \left(\frac{n}{4}\right)!} = \frac{n!}{\left(\frac{n}{4}\right)!^4}$$

with n_{UL} , n_{UR} , n_{LL} , and n_{LR} the numbers of objects on the upper-left, upper-right, lower-left, and lower-right quadrants, respectively and n the number of objects on the frame ([104]).

B.13 Rhythm

Rhythm (NoC=6) in visual design refers to regular changes in the elements used. It combines order with variation to make the appearance exciting. Rhythm is accomplished through variation of arrangement, dimension, number and form of the elements.

Rhythm is the extent to which the objects are systematically ordered and is given by the equation B.48 (SoE=5) [104, 102, 105, 66, 106, 97]:

$$RHM = 1 - \frac{|RHM_x| + |RHM_y| + |RHM_{area}|}{3} \in [0, 1] \quad (B.48)$$

The rhythm components depend on the coordinates of the center of the frame, the coordinates of the centers of the interface objects, the area of the interface objects and the total number of objects on the quadrant.

B.14 White-space fraction

A study by Harrington *et al.* (NoC=1) defined a design rule: white space (including margins) should total about half of the total page area. The non white-space area can be estimated by totaling the areas of the content objects [56].

The formula proposed by [56] to measure the white-space fraction (V_{ws}) is given by equation B.49 (SoE=1):

$$V_{ws} = 1 - 4 \left(\frac{\sum A_i}{A_p} - 0.5 \right)^2 \quad (B.49)$$

Where A_i is the area of object i and A_p is the area of the page.

B.15 Composition

Composition (NoC=3) is used in the context of mobile applications and serves to increase the visual clarity of the GUI by the meaningful arrangement of the interface components. The objective of this metric is the visual and semantic grouping of the components that are related in the same border (the line, the color, the shape, etc.). This metric counts the number of objects that have a clear boundary by the group. In general, a composition close to 1 helps younger or older users with low computer skills to interact with the system. To measure this, we use the following formula (equation B.50) (SoE=2) [142]:

$$COM = 1 - \frac{G + UG}{2n} \in [0, 1] \quad (B.50)$$

Where G is the number of groups with clear boundary by line, background, color, or space, UG is the number of ungrouped objects and n is the number of component of the mobile user interface.

A second formula for composition, called « Grouping metric» in the paper written by Alemerien *et al.* [5], measures the number of objects that have a clear boundary by line, background, color, space, or size. The formula of composition calculates the percentage of grouped and ungrouped objects (G and UG respectively). Finally, the grouping complexity (GT) is obtained as given by equation B.51 (SoE=3):

$$GT = UG + GC \quad (B.51)$$

with :

$$UG = 1 - \frac{\sum_{i=1}^N GW}{N}$$

$$GC = \frac{G}{M} * \text{Weight}$$

where N is the total number of objects on the screen, G is the ratio of the number of different object types to the total number of objects (M) in all groups. Weight is the ratio of total number of grouped objects to total number of objects on the screen. GW is a variable that indicates whether the object is grouped or not. The value of GW equals 1 if the object exists in a group,

otherwise GW equals 0. The formula of the grouping complexity has been empirically validated with a low correlation of 0.462.

Under the grouping technique, elements with similar functions or information are surrounded by a boundary by line, background color and/or space. Grouping makes it easier for the users to extract the information assigned to the group. The method for calculating the grouping complexity is computed as follows (equation B.52, SoE=3) ([48]):

$$CG = \frac{g_i}{g} \in [0, 1] \quad (B.52)$$

Where g_i is the number of groups with clear boundary by line, background, color, or space and g is the total number of groups. This formula has been empirically validated but no correlation value was given. The study only concluded that the proposed formula is useful.

B.16 Color contrast

Contrast (NoC=2) refers to the juxtaposition of visually dissimilar elements in order to convey the fact that these elements are different. It provides the eye with a noticeable difference between two objects or two sets of objects. In this case, we will talk of the color contrast which can provoke an emotion to the user or emphasize/de-emphasize text elements.

The color model used to compute this metric is the HSV model (hue, saturation, and value). Hue varies from 0° to 360° while the corresponding colors vary from red through yellow, green, cyan, blue, magenta, and back to red. Saturation varies from 0 (unsaturated) to 1 (fully saturated). Value varies from 0 to 1 with the corresponding colors becoming increasingly brighter (figure B.3).

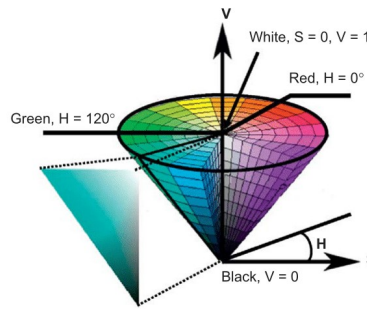


Figure B.3: The coordinate system for the HSV color space [73]

The color difference between two colors, (H_1, S_1, V_1) and (H_2, S_2, V_2) , is therefore given by equation B.53 (SoE=5) [73]:

$$\Delta C = \frac{1}{\sqrt{4}} \sqrt{(V_1 + V_2)^2 + (V_1 S_1 \cos H_1 - V_2 S_2 \cos H_2)^2 + (V_1 S_1 \sin H_1 - V_2 S_2 \sin H_2)^2} \quad (\text{B.53})$$

A human perceives quite differently two colors with a ΔC close to 1 while the ΔC of two perceptually similar colors is close to 0. Therefore, a pixel block with a large ΔC against the color of the background region should appear to be more contrasted and visually heavier than one with a small ΔC .

In their paper, Miniukovich *et al.* calculated the figure-ground contrast (SoE=2). They defined this term as the difference in color or brightness between two adjacent areas. This difference forms an edge and the magnitude of this difference defines the strength of the edge ([96]). They used a microscopic approach (pixel scale) to generate a formula for this metric.

In their study, they analyzed screenshots of web pages. The edges of two adjacent areas were detected by using an algorithm which requires two input thresholds in the range of 0 to 1: low and high. Lower input thresholds detects more edges while higher thresholds detects fewer edges. In the paper of Miniukovich and De Angeli, the threshold varied from 0.1 to 0.7 with a step of 0.1, which gave seven levels (l) of edge strength (E_l). The edge pixels were counted for each level and the difference between successive levels was computed. Then, the differences were weighted and summed up. The weakest edges received the highest weight, since they contribute the most to the difficulty of visual differentiation. The last step was to normalize the sum by the total number of edge pixels. The figure-ground contrast metric (E_{norm}) is finally computed as follows (equation B.54) ([95]):

$$E_{norm} = \frac{\sum_{l=1}^6 ((E_l - E_{l+1}) * (1 - \frac{(l-1)}{6}))}{E_1 - E_6} \quad (\text{B.54})$$

B.17 Color complexity

The use of color in an user interface has been shown to improve performance and visual search time, to facilitate organizing information and to aid memory. Color can also create positive user perceptions and is preferred to monochromatic screens. A colored interface is seen less monotonous, reduce eye strain, and is more globally more pleasant [128]. Research has found that if the number of colors on a display is increasing, the response time to a single color rises and color confusion is more likely [88]. A study found that the maximum number of colors that a person can handle is in the range of 4 to 10, while lower numbers should be encouraged [26].

A formula (SoE=3) of color complexity (NoC=1) has been proposed by Reigler *et al.* [128] and is applied specifically to Android applications. The metric

is based on the number of dominant colors. Dominant colors are occurring more frequently on a pixel-basis than non-dominant colors. The algorithm also computes the difference in terms of HSV value for the dominant colors to obtain the formula of color complexity shown in equation B.55 :

$$C_s = \frac{\frac{numc_s}{numcmax} + (1 - \overline{ccm_s}) + c_{h_s, h_{s+1}}}{3} \quad (B.55)$$

Where $numcmax$ denotes the maximum number of dominant colors on a certain screen of the whole user interface, $numc_s$ is the number of dominant colors $c_1, c_2, \dots$ on screen s , $\overline{ccm_s}$ is the average color combination match of the dominant colors and $c_{h_s, h_{s+1}}$ is the color combination match of the current screen's color histogram h_s and the succeeding screen's color histogram h_{s+1} .

B.18 Typography complexity

Typography (NoC=1) plays an important role in application design. Smaller fonts lead to a higher fixation duration and thus reduce the reading performance. On the other side, choosing a larger font size increases legibility. The color contrast can also influence the legibility of the text [128].

The formula of typography complexity (SoE=5) presented below is applied in the context of Android applications. Typographic aspects that are analyzed in this formula include the number of different text sizes, or heights of text elements, the average text size and the average text foreground / background color combination match. The formula of typography complexity (T_s) is given below equation B.56:

$$T_s = \frac{\frac{numt_s}{numtmax} + \left(1 - \frac{\overline{t_s}}{tmax}\right) + (1 - \overline{ccm_s})}{3} \quad (B.56)$$

Where $numt_s$ is the number of different text sizes, or heights of text elements, $numtmax$ denotes the maximum number of text sizes and $tmax$ denotes the maximum text size, both on a certain screen of the whole user interface. $\overline{t_s}$ is the average text size which is determined by calculating the average height of all text elements found on the screen s and $\overline{ccm_s}$ is the average text foreground/background color combination match.

B.19 Essential efficiency

A well designed user interface should have as few steps as possible to achieve the user's goal. Essential efficiency (NoC=1) is the ratio of the number of steps in the essential use case to those that are required to perform the task with a given user interface. It measures how close a given design is to the essential

use case model, in terms of the number of steps involved to perform a task or collection of tasks (equation B.57, SoE=3) [75]:

$$\text{Essential Efficiency} = \frac{\text{Essential Length}}{\text{Operational Length}} \quad (\text{B.57})$$

Where Essential Length represents the essential minimum number of steps from users' perspective. Essential efficiency computed for a given user interface design with a value of 1 or close to 1 indicates that the design is efficient in performing the given tasks [75].

B.20 Size complexity

The size complexity (NoC=2) can influence the global complexity of the interface. Alemerien *et al.* [5] have developed a formula to measure the Object Size Complexity (SC_k) and the overall size complexity (SC). The formula is given by equation B.58(SoE=5):

$$SC = \frac{\sum_{k=1}^{W_i} SC_k * \text{Weight}(k)}{W_i} \quad (\text{B.58})$$

with:

$$SC_k = \frac{\sum_{j=1}^N S_j}{N}$$

where N is the number of objects in type k th and S_j is the number of different sizes, S_j is 1 if the object size is not counted before and 0 if the object size is counted. W_i is the number of objects types and $\text{Weight}(j)$ is the number of objects in category j^{th} divided by the total number of objects on the screen. The formula of the overall size complexity has been empirically validated with a correlation of 0.943.

Size complexity involves the categorization of elements into groups according to actual physical size and variation in those sizes. It is based partially on information order, and is calculated for each type of information so that the complexity don't increase significantly when different information appear in different sizes. In terms of the magnitude size of the size complexity (CS), objects are related to size as follows (equation B.59, SoE=3)) [48]:

$$SC = 1 - \frac{\sum_I^{type} (n_{size} - 1)}{n} \in [0, 1] \quad (\text{B.59})$$

where n_{size} is the number of sizes and n is the total number of objects. This formula has been empirically validated but no correlation value was given. The study only concluded that the proposed formula is useful.

B.21 Visual coherence

Visual Coherence (NoC=2) measures how closely visual organization matches the semantic relationships among concepts associated with components. It is based on the principle that well-structured interfaces group semantically related components together.

Visual Coherence of a visual group is the ratio of the total “relatedness” of pairs of visual components to the total number of pairs. Total VC for a screen or dialogue box is the sum of the VC for each level of grouping (equation B.60, SoE=3)) ([108, 38]):

$$VC = 100 * \left(\frac{\sum_{\forall l} G_l}{\sum_{\forall l} N_l * (N_l - 1) / 2} \right) \quad (B.60)$$

with $G_l = \sum_{\forall i,j | i \neq j} R_{i,j}$

where N_l is the number of components in group l , $R_{i,j}$ is the semantic relatedness between components i and j . In practice, semantic relatedness can be simplified to $R_{i,j} = 1$ if components i and j are « substantially related », $R_{i,j} = 0$ otherwise. This formula correctly favors organizing user interface components into subgroups, but only so long as the groups make sense, that is, they enclose substantially related components ([108]).

B.22 Element smallness

The screen smallness E_s (NoC=1) is a measure of the size of GUI elements on a certain screen s . This formula is specifically applied to Android applications. Firstly, the screen is divided into groups. A group g is defined as a region of interest on the screen, which contains one or more GUI elements belonging together. There are various criteria for the formation of groups such as common background color, position on the screen or functionality. Then, the mean element width and height of each group g of a screen s ($\overline{width}_s$ and $\overline{height}_s$ respectively) is calculated from the $width_{u,g,s}$ and $height_{u,g,s}$ of each element u in this group. Finally, $\overline{width}_s$ and $\overline{height}_s$ are combined to a single metric : E_s , the screen smallness (equation B.61, SoE=5) [128]:

$$E_s = \frac{\left(1 - \frac{\overline{width}_s}{width_s} \right) + \left(1 - \frac{\overline{height}_s}{height_s} \right)}{2} \quad (B.61)$$

where $width_s$ and $height_s$ are the width and height of the screen s . Smaller objects tend to be harder to identify and interact with, so this metric needs a higher score for smaller elements and smaller score for higher elements.

B.23 Edge congestion

Edge congestion (NoC=1) relies on the notion of critical spacing / minimal distance between objects at which the user starts having difficulties differentiating the objects .

Miniukovich *et al.* [95] developed an algorithm to detect edges on a screenshot of a web page. This metric is applied in a microscopic approach by comparing the position of the different pixels of the screenshot. If at least two pixels of two different edges are in the same 20-pixel vicinity, they are marked as congested. The marking is done in both horizontal and vertical directions. Finally, the edge congestion metric is defined as the ratio between congested pixels (p_c) to all edge pixels (p_a) (equation B.62, SoE=4)) :

$$cong = \frac{p_c}{p_a} \quad (B.62)$$

B.24 Concentricity

The concentricity metric (NoC=1) measures how the interface objects are gathered in the center of the interface or rather kept in the corners [171]. Equation B.63(SoE=1) gives the formula of concentricity (C):

$$C = \frac{\bar{d}}{d_{diag}} \quad (B.63)$$

B.25 Order and complexity

By compiling some of the formulas described above, it is possible to create a scale of order or complexity (NoC=13) with extreme complexity at one end and minimal complexity (order) at the other ([104]).

The order and complexity scale (OM) is measured as shown below (equation B.64, SoE = 5)) ([104, 101, 102, 105, 66, 106, 97]):

$$OM = \frac{\sum_i^{13} \alpha_i M_i}{13} \in [0, 1], \quad 0 \leq \alpha_i \leq 1 \quad (B.64)$$

with $M = (M_1, M_2, M_3, M_4, M_5, M_6, M_7, M_8, M_9, M_{10}, M_{11}, M_{12}, M_{13}) = (BM, EM, SYM, SQM, CM, UM, PM, SMM, DM, RM, ECM, HM, RHM)$

where BM, EM, SYM, SQM, CM, UM, PM, SMM, DM, RM, ECM, HM and RHM is given by equation B.1, B.11, B.13, B.20, B.23, B.26, B.28, B.30, B.32, B.36, B.46, B.47 and B.48 respectively. The weighting component α is assumed to be a constant. Aesthetic measure M_i has its own weighting component α_i . The choice of weights is an optimization problem that needs to be solved with mathematical algorithms ([104]).

The strength of evidence level is 5 for this formula. Each metric listed above has been calculated and applied to more than 50 different screens. Then, an empirical study has been performed with a high correlation (0.930) between the relative ratings by the viewers and those obtained using the computational measures. A further study [102] showed that BM, SQM, UM, PM, DM and HM account for most of the variability in overall ratings, but the others metrics have less impact.

This second formula of order and complexity (SoE=5) is derived from the Birkhoff's formula [20] of the measure of order and complexity (M): $M = O/C$. Where O is the order and C the complexity. The formula proposed by Ngo computes 4 different metrics defined previously and is given by the equation B.65 [103, 100]:

$$M = \frac{O_{BM} + O_{EM} + O_{SM} + O_{SQM}}{C} \in [0, 3] \quad (C > 0) \quad (\text{B.65})$$

where O_{BM} is a measure of balance, O_{EM} is a measure of equilibrium, O_{SM} is a measure of symmetry, and O_{SQM} is a measure of sequence. The complexity C of a layout is defined as the number of its components.

This formula has been verified empirically by comparing the order value obtained from the formula B.65 to the order perceived by the subjects of the study. The correlation was very high (0.97).

Song [141] developed an algorithm to measure complexity and visual segmentation of Web pages (SoE=3). The formula of complexity is given below (equation B.66):

$$C = \frac{\sum C_k S_k}{S_k} \quad (\text{B.66})$$

where C_k and S_k are the complexity and size of sub-block k .

Overall, the algorithm analyzes the interface to generate blocks that are hierarchically arranged. The formula is then used in the hierarchy of blocks from bottom to top. The lowest level blocks are assigned a complexity equal to the number of objects they enclose. The estimated complexity of the whole web page is finally calculated with the complexity of all first level blocks.

This formula has been validated empirically. However, the correlation of the estimated complexity of the first level of blocks with the perceived complexity was low (0.433).

Soui *et al.* proposed that novice users usually prefer an interface with a low level of complexity while users having higher education level prefer Mobile User Interfaces with a high level of complexity. The complexity metric is therefore calculated as follows (equation B.67 , SoE=3) [142]:

$$CM = \frac{N_{vap} + N_{hap}}{2n} \in [0, 1] \quad (B.67)$$

Where N_{vap} is the number of vertical alignment points, N_{hap} is the number of horizontal alignment points and n is the number of object on the mobile user interface.

In the paper of Alemerien *et al.*, 5 metrics (SoE=4) contributing to the overall complexity of the screen layout were defined [5]. These 5 metrics are the Total Balance Complexity (TBC) (equation B.7), the Total Screen Alignment Complexity (TAC) (equation B.31), the Density Complexity (DC) (equation B.34), the Grouping Complexity (GT) (equation B.51) and the Size Complexity (SC) (equation B.58). The Overall Screen Layout Complexity (LC) formula is therefore given below (equation B.68):

$$LC = ((TAC * w1 + TBC * w2 + DC * w3 + SC * w4 + GT * w5) / 5) * 100\% \quad (B.68)$$

All the complexity metrics have been empirically validated and a weight for each metric was calculated based on the ratings of the study participants. $w1$, $w2$, $w3$, $w4$ and $w5$ are the weight for the TAC, TBC, DC, SC, and GT metrics respectively. The values of those weights are 0.84, 0.76, 0.80, 0.72, and 0.88, respectively. The correlation between the user ratings and the computational value of the overall screen layout complexity is 0.804.

The sixth formula of complexity (SoE=4) proposed by Stickel *et al.* [144] is called the XAOS metric. The formula for visual complexity (X) is shown below (equation B.69) :

$$X = A * O * S \quad (B.69)$$

A is the number of possible interactions, which can be considered as functional elements or just actions. O is the number of higher level structures or gestalt groups, in short organizational elements and S is the summed entropy of RGB values. This formula has been empirically validated with a correlation of 0.77.

B.26 Inconsistency

Inconsistency (NoC=1) is applied in the specific case of Android applications. According to Nielsen, consistency is one of the most important characteristics of usability in user interfaces ([107]). Consistency refers primarily to action sequences, terms, units, layouts, colors, typography, etc within an application state. A clear and clean organization that uses consistency makes it easier to

recognize the main objects on the screen and to ignore secondary information when necessary.

The inconsistency metric defined below aims to determine if the entire user interface is visually aesthetic. Therefore, it computes the metric values of the :

1. Number of GUI elements
2. Element smallness
3. Misalignment
4. Density
5. Imbalance
6. Color complexity
7. Typographic complexity

These metric values are calculated for each screen s and compared by calculating the standard deviation. In order to normalize the result in a $[0,1]$ range, the result of the standard deviation calculation is divided by x_{max} , which denotes the maximum value for a certain metric over all screens in order to obtain the inconsistency (I) as shown below (equation B.70, SoE=5) ([128]):

$$I = \frac{1}{N} \sum_{n=1}^N \frac{1}{x_{max_n}} \sqrt{\frac{1}{J-1} \sum_{j=1}^J (x_{j_n} - \bar{x}_n)^2} \in [0, 1] \quad (B.70)$$

where x_{j_n} denotes the metric value x on screen j and metric n . N is the number of metrics per screen and $\bar{x}_n$ is the mean complexity value for metric n over all screens. This formula has been empirically validated with a high degree of statistical significance ($P \leq 0.05$ for 3 different applications).

Visual profiles distributions

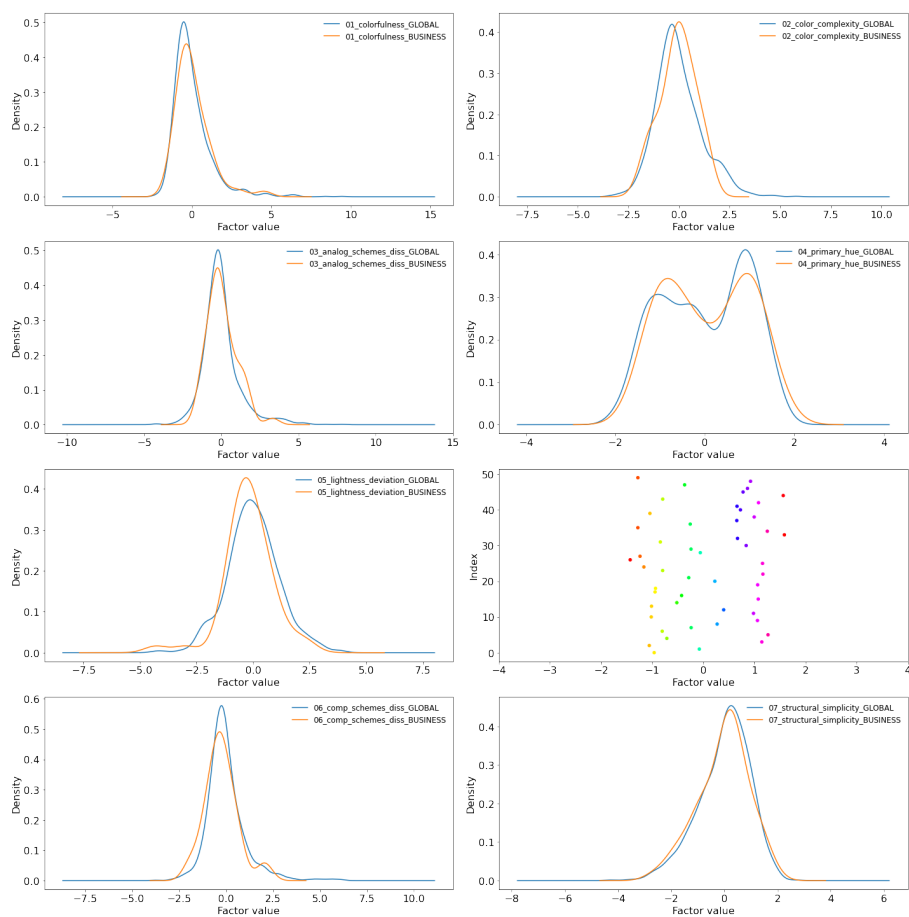


Figure C.1: Distribution of factors for the category “Business and consumer services”

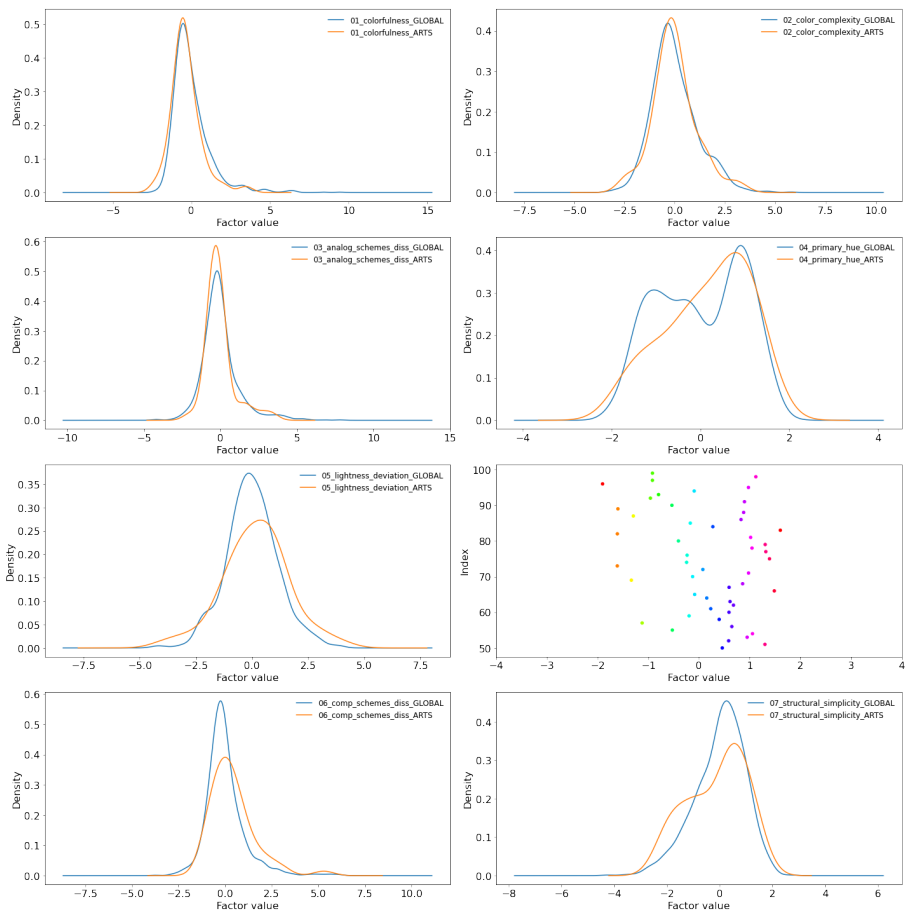


Figure C.2: Distribution of factors for the category “Arts & entertainment”

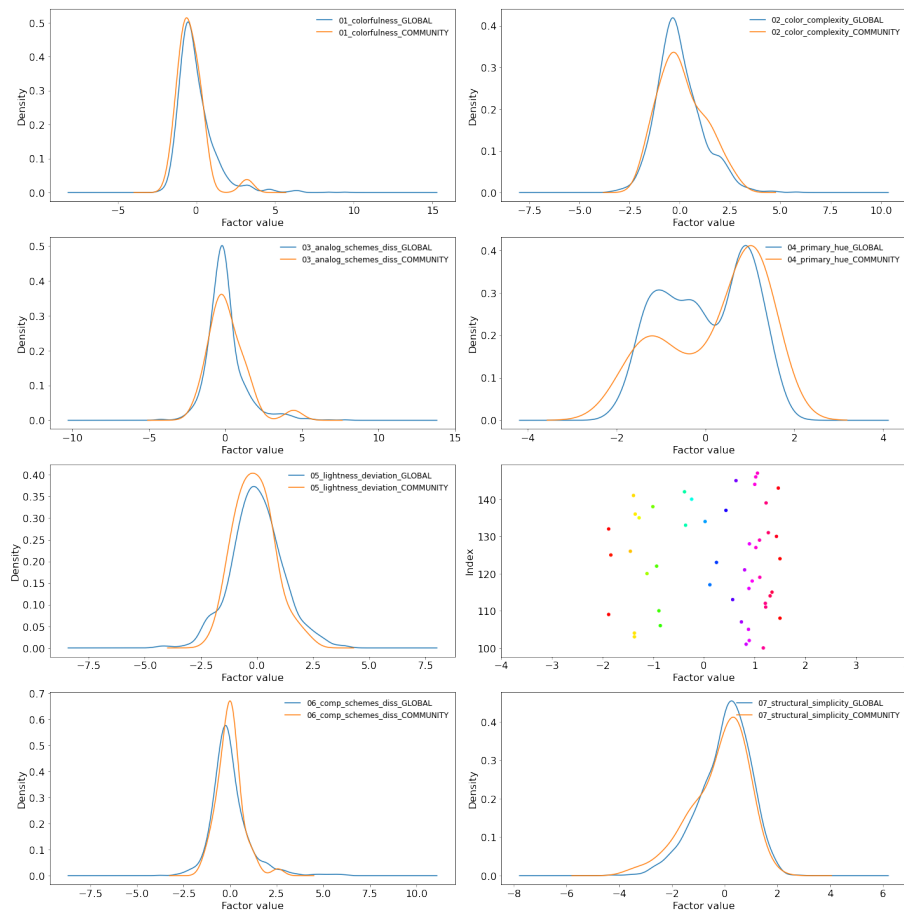


Figure C.3: Distribution of factors for the category “Community & Society”

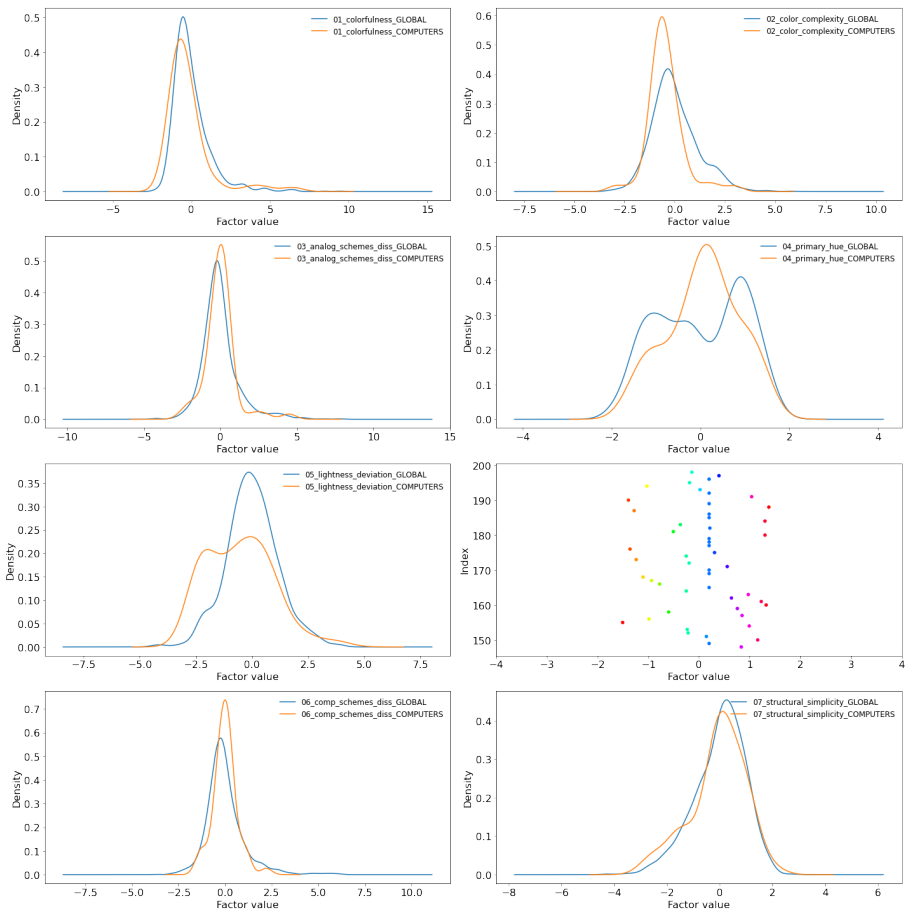


Figure C.4: Distribution of factors for the category “Computers Electronics & Technology”

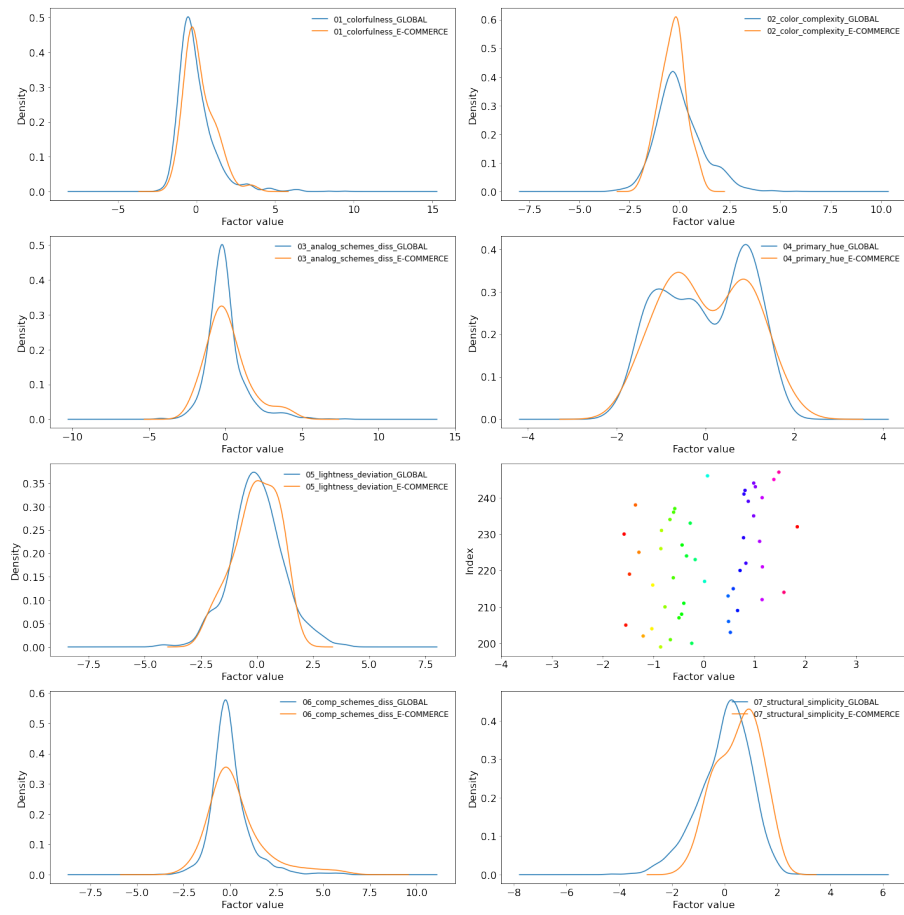


Figure C.5: Distribution of factors for the category "E-commerce & Shopping"

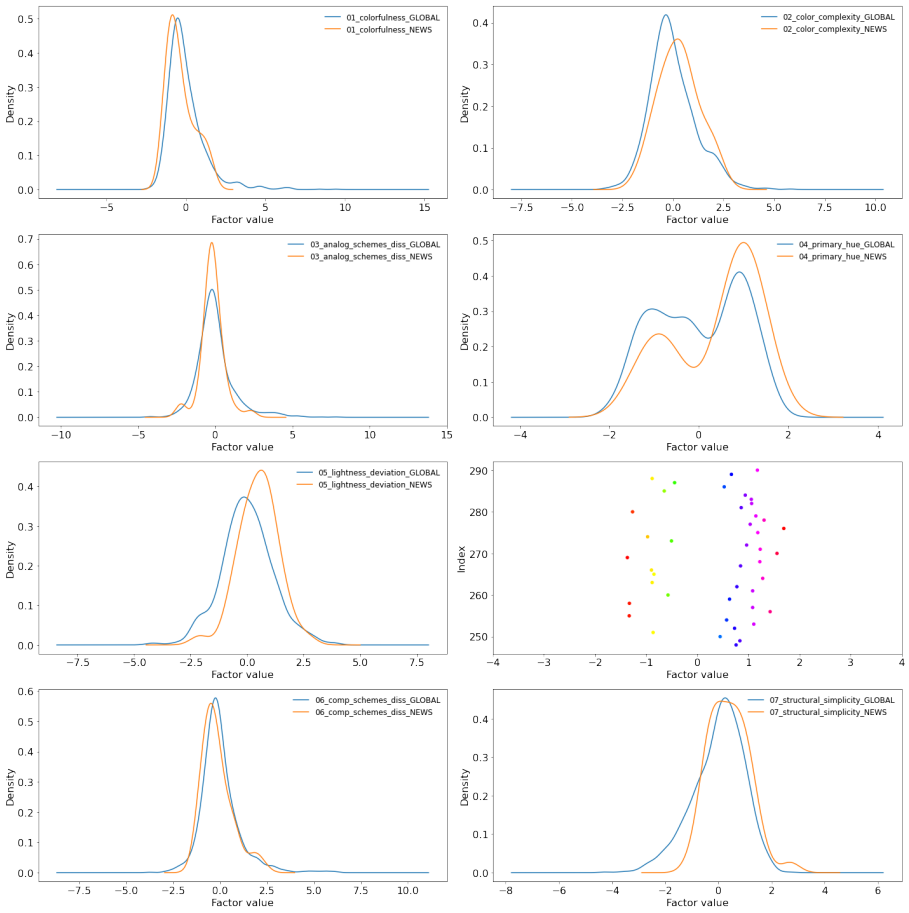


Figure C.6: Distribution of factors for the category “News & Media”

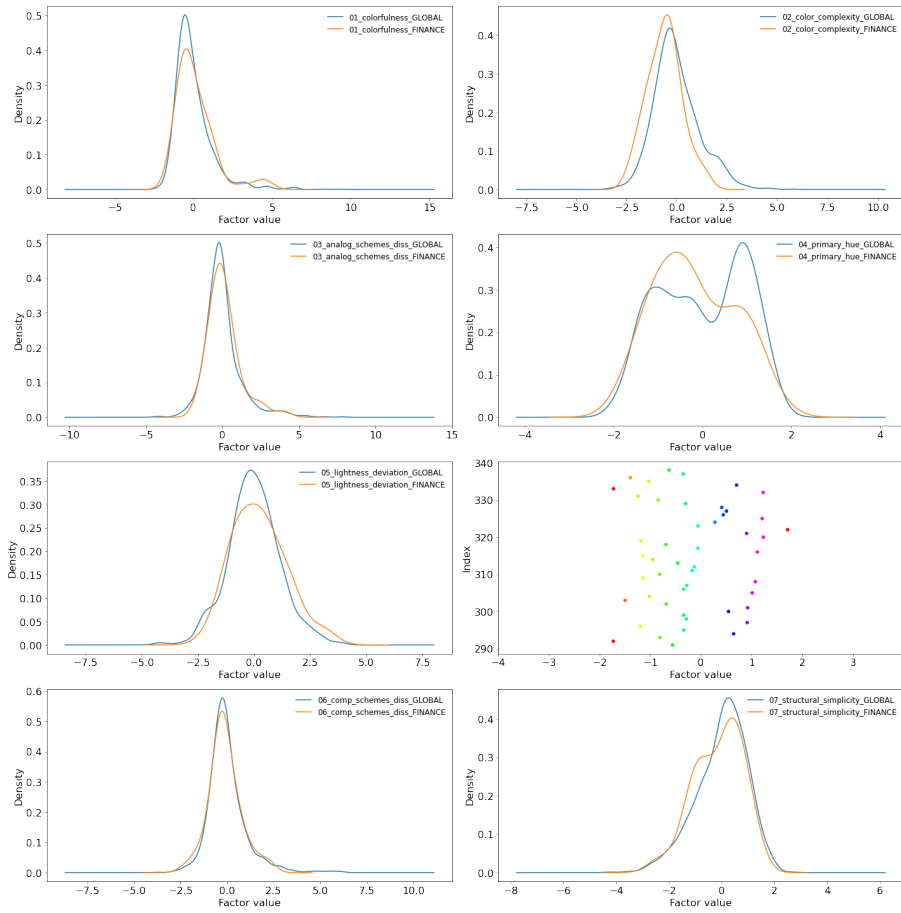


Figure C.7: Distribution of factors for the category "Finance"

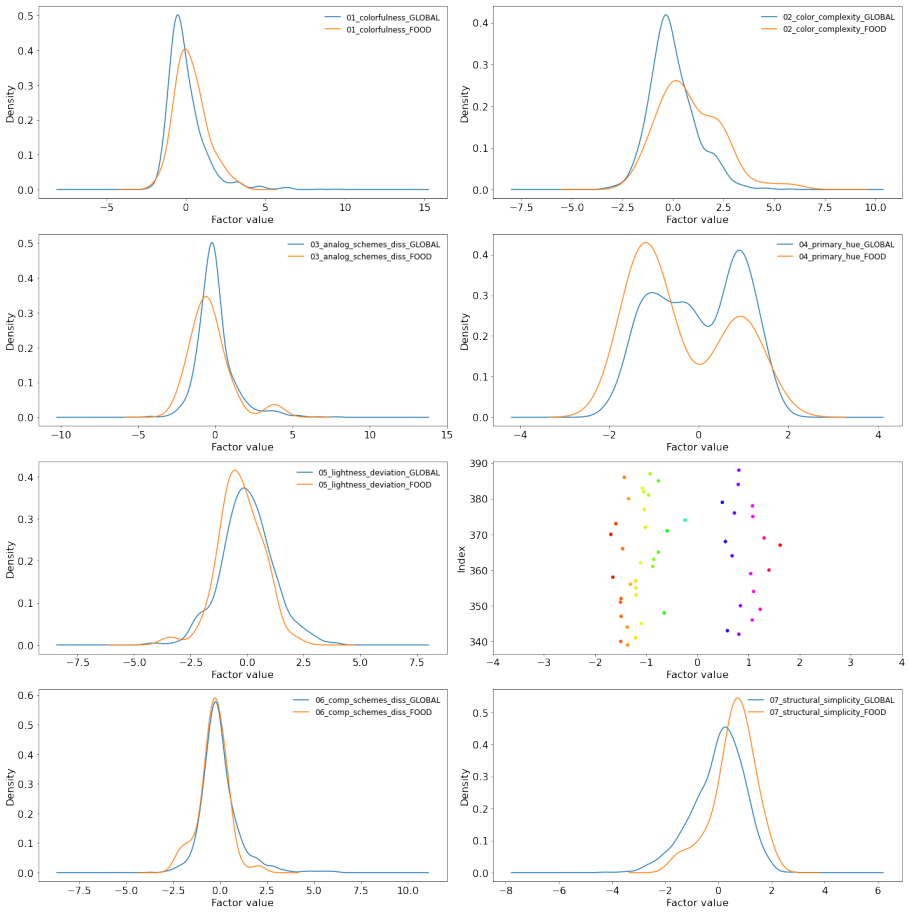


Figure C.8: Distribution of factors for the category “Food & Drinks”

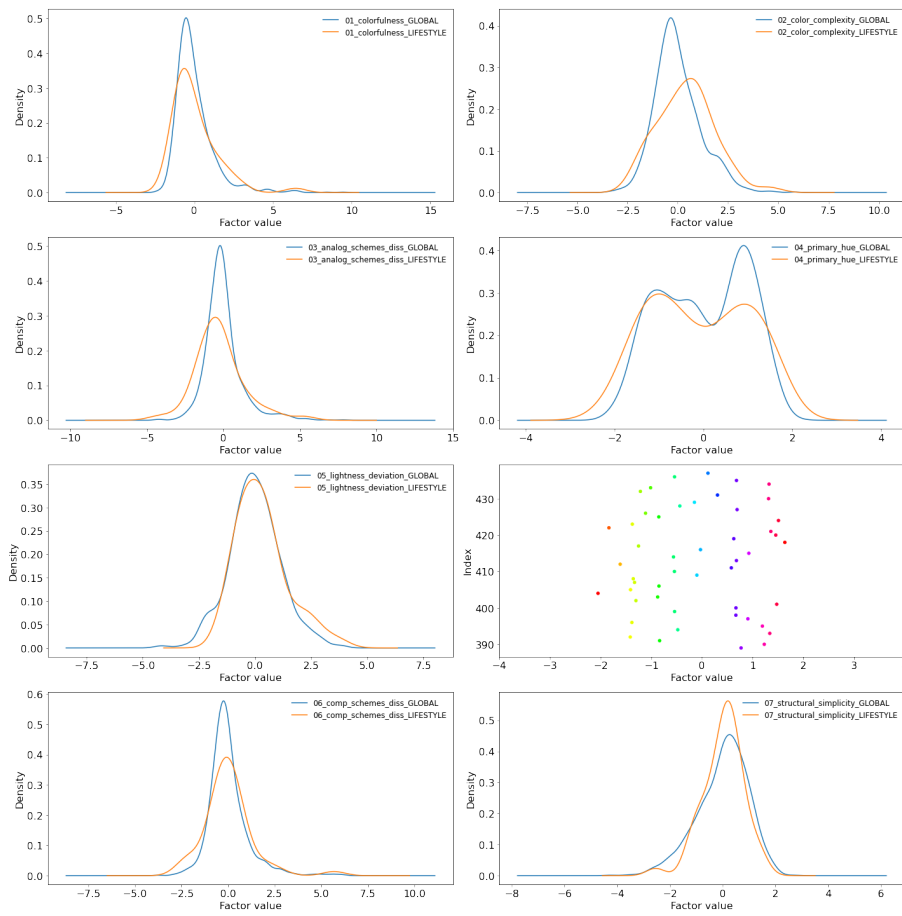


Figure C.9: Distribution of factors for the category "Lifestyle"

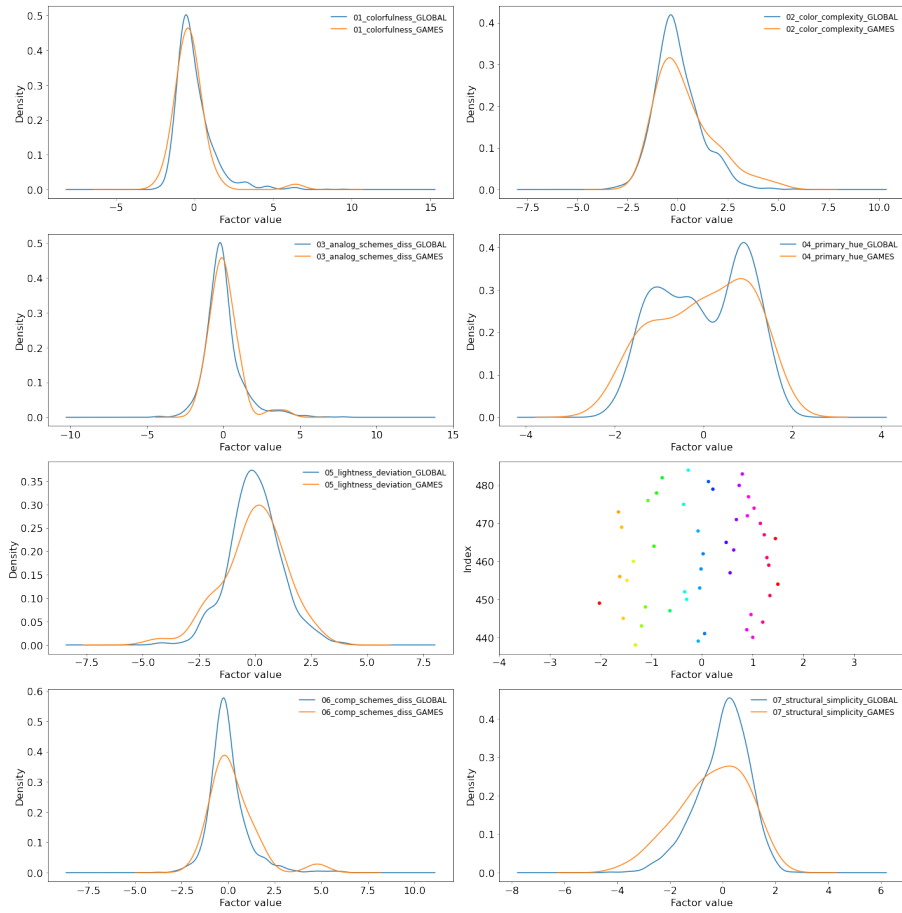


Figure C.10: Distribution of factors for the category "Games"

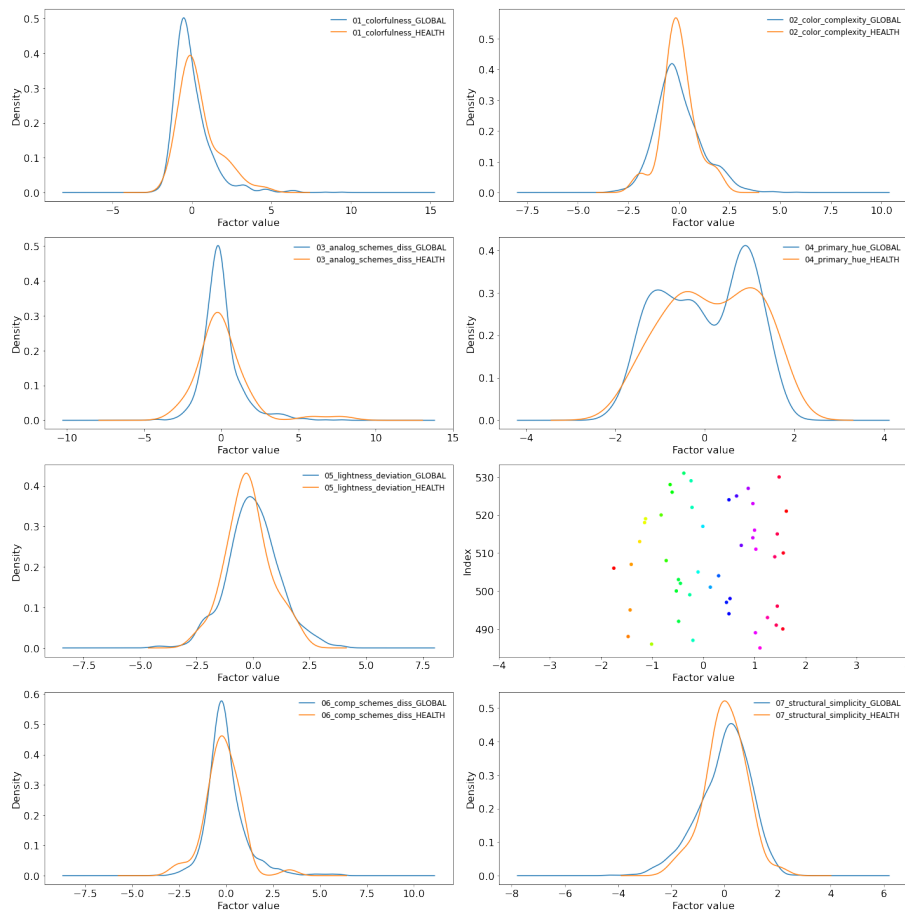


Figure C.11: Distribution of factors for the category "Health"

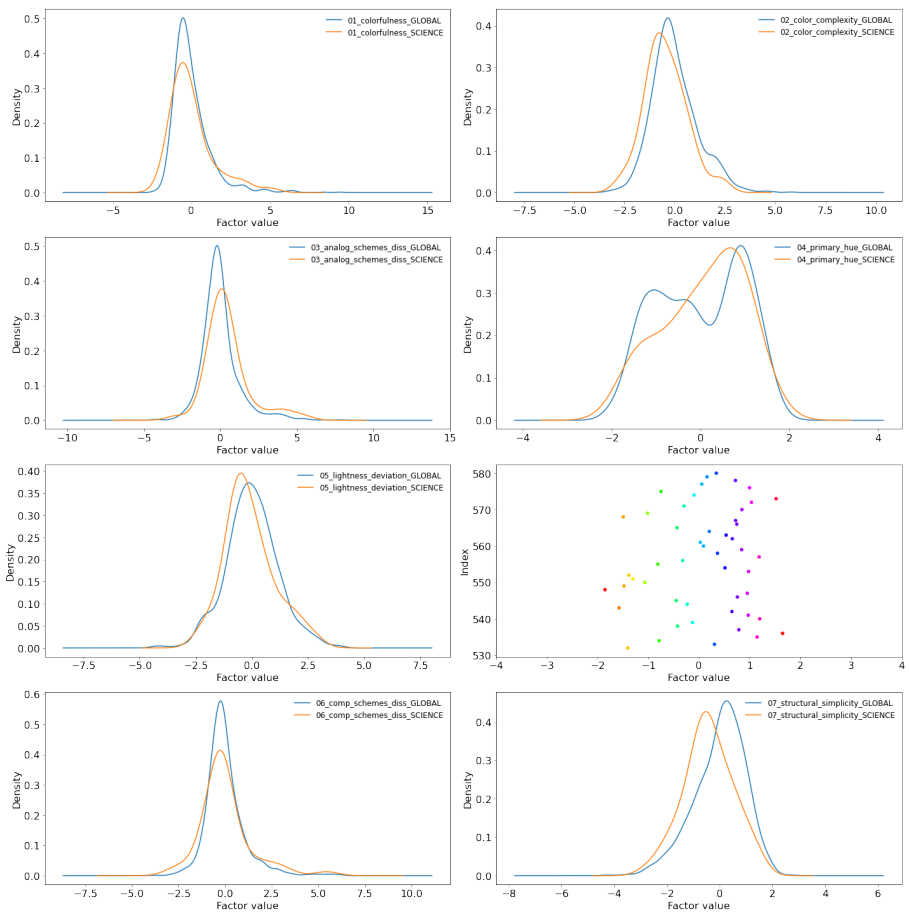


Figure C.12: Distribution of factors for the category “Science & Education”

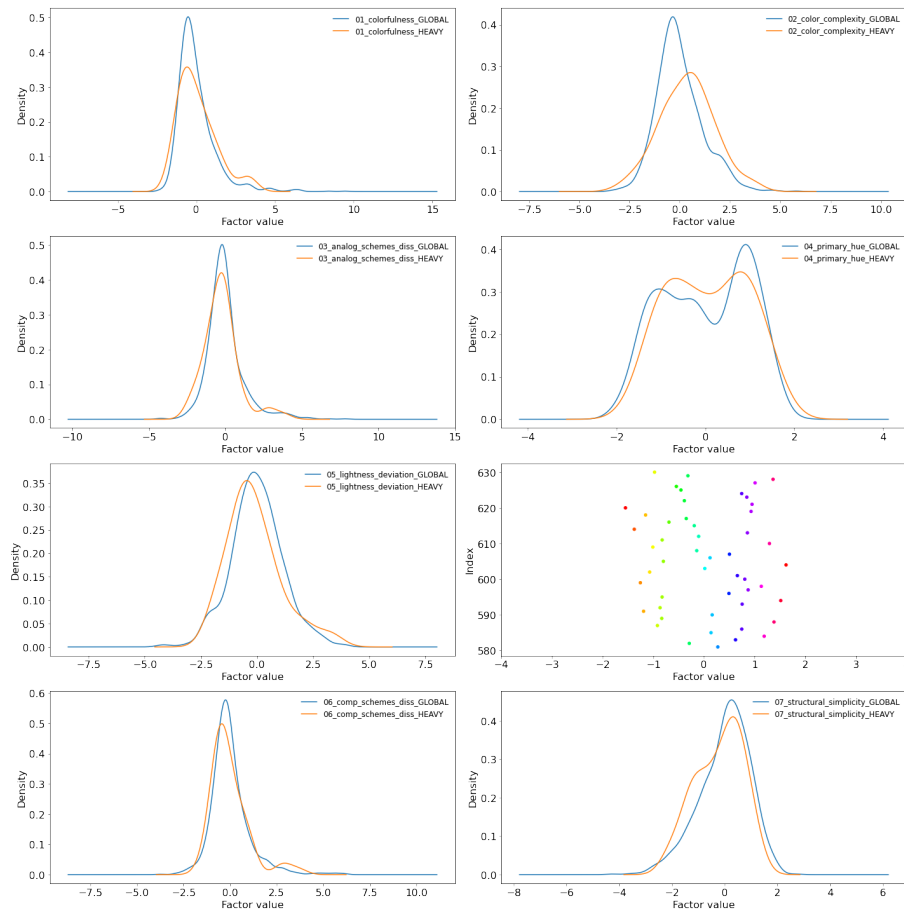


Figure C.13: Distribution of factors for the category "Heavy industry & Engineering"

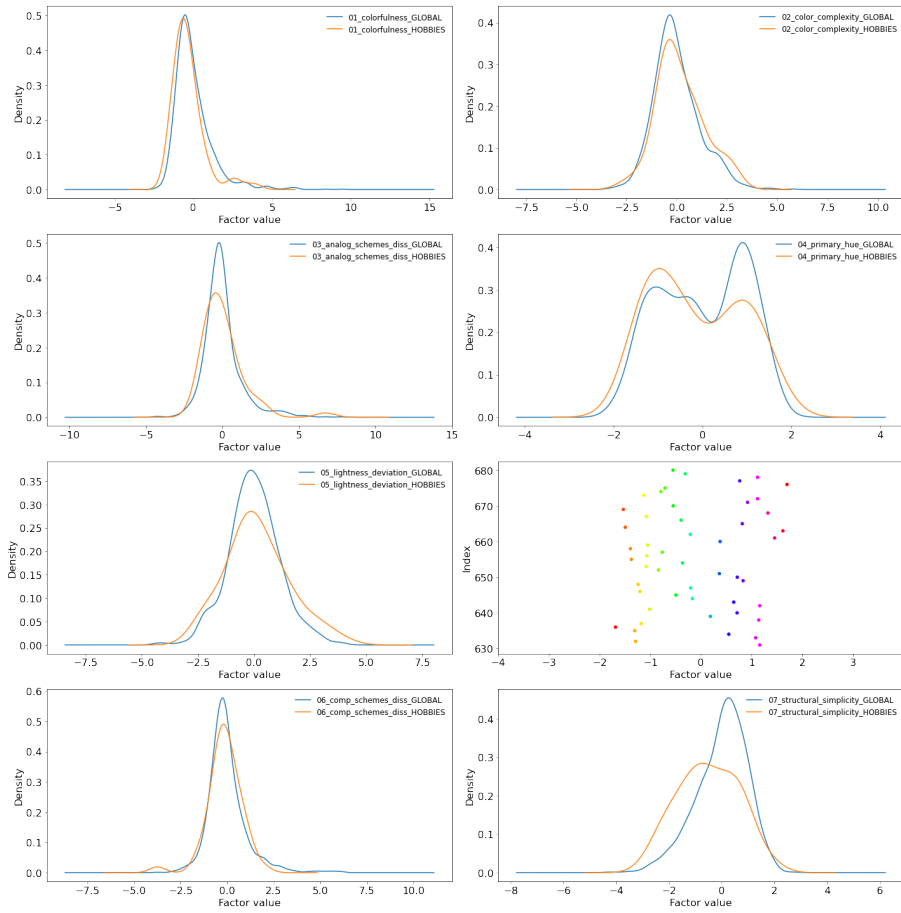


Figure C.14: Distribution of factors for the category “Hobbies & Leisures”

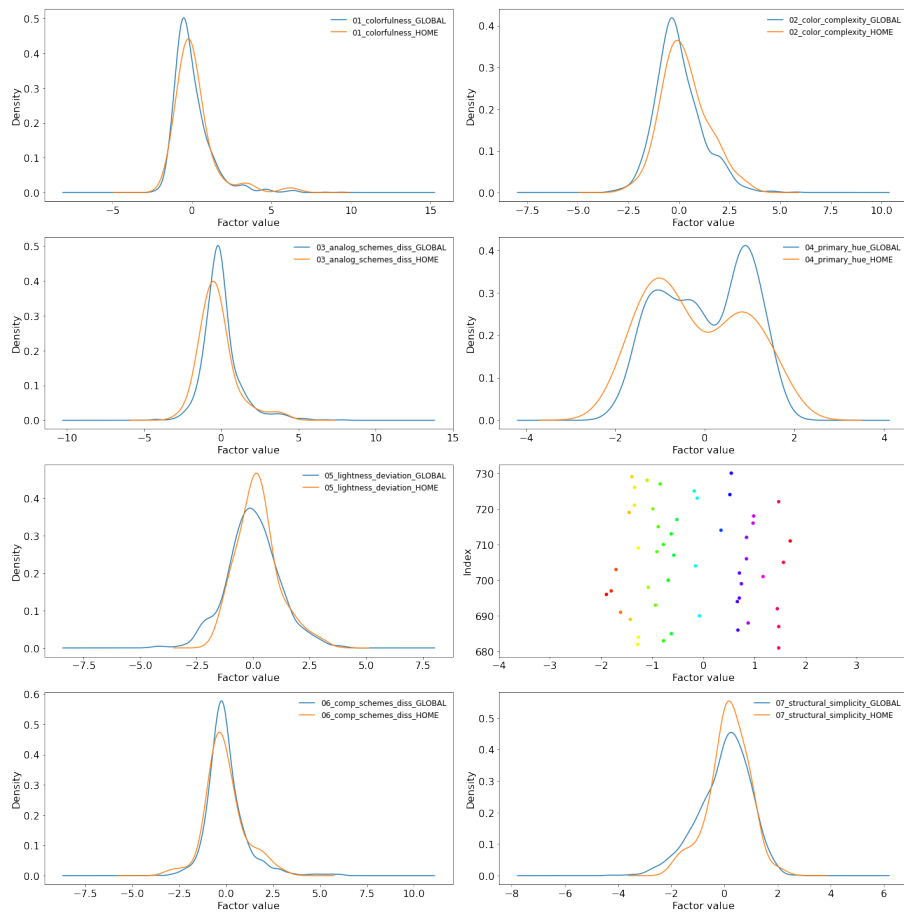


Figure C.15: Distribution of factors for the category "Home & Garden"

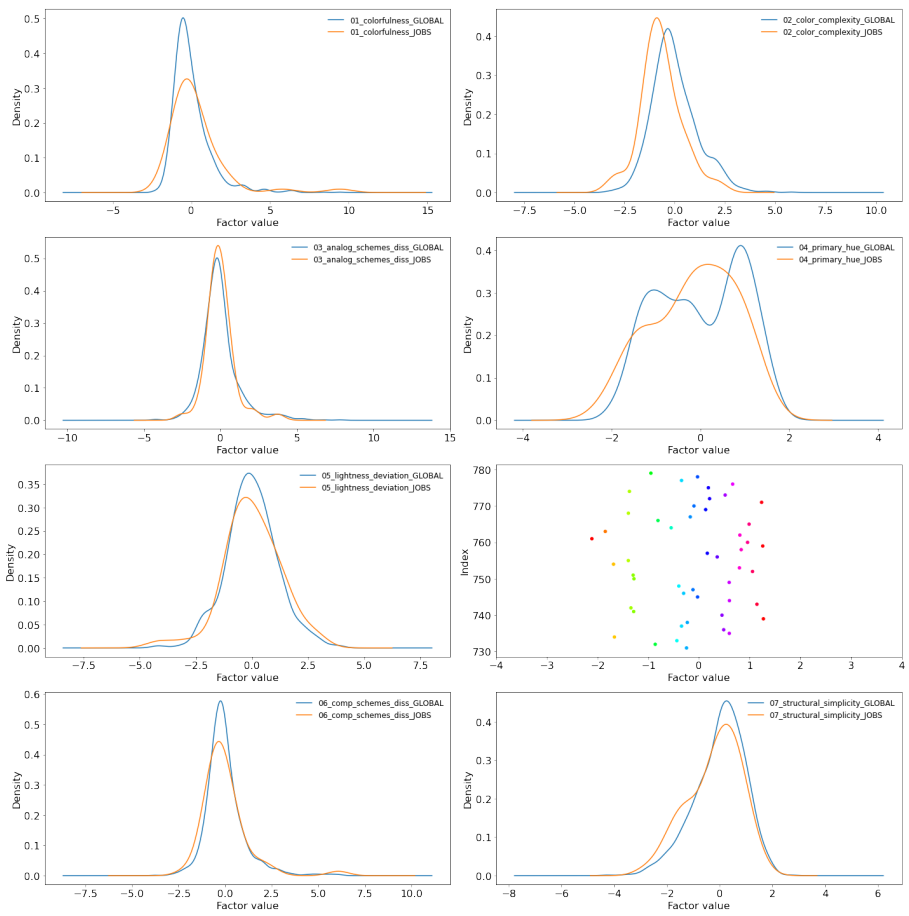


Figure C.16: Distribution of factors for the category “Jobs & Career”

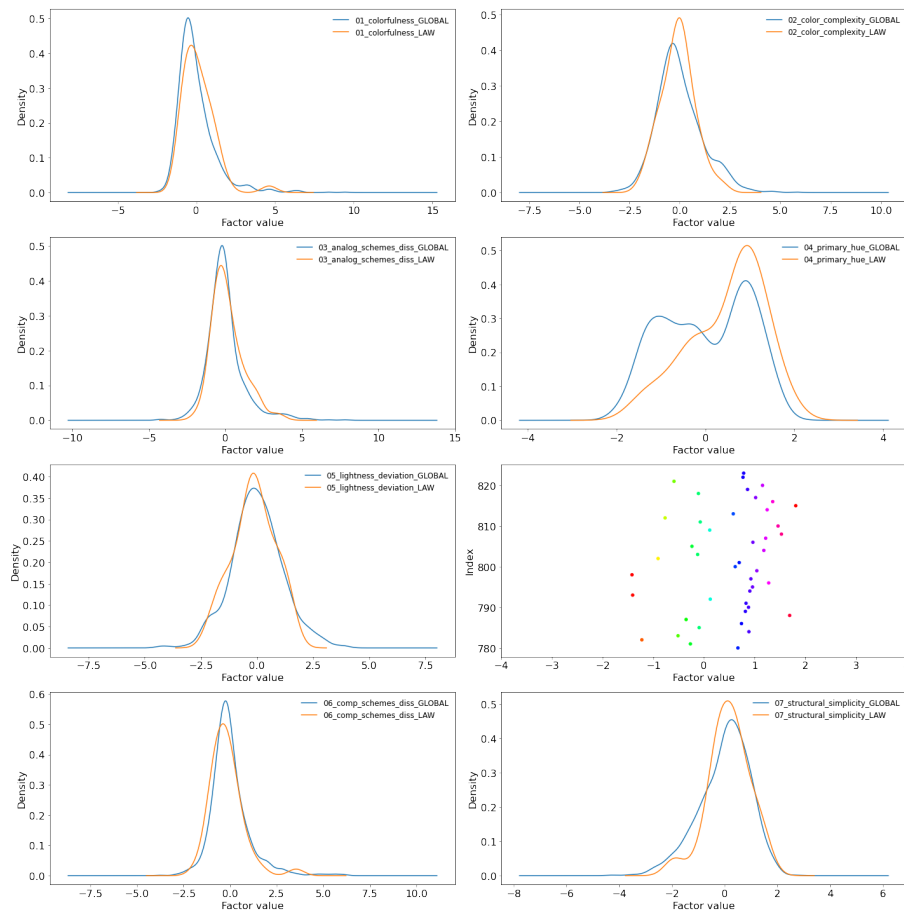


Figure C.17: Distribution of factors for the category “Law & Government”

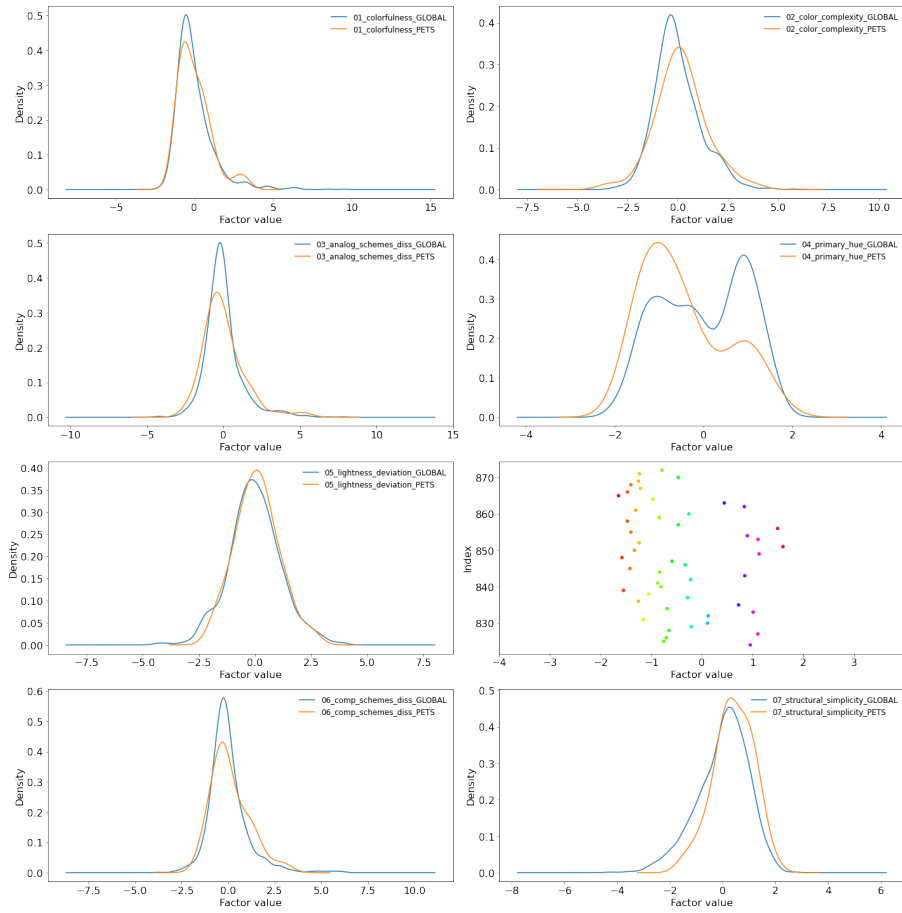


Figure C.18: Distribution of factors for the category “Pets & Animals”

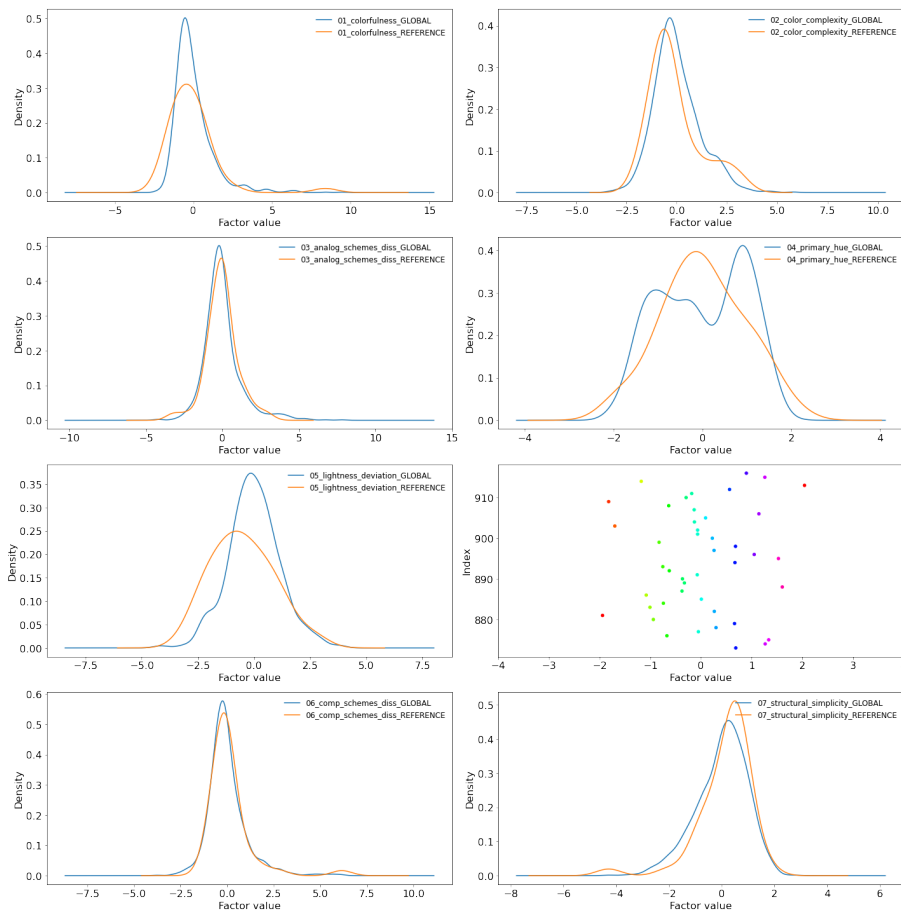


Figure C.19: Distribution of factors for the category “Reference Materials”

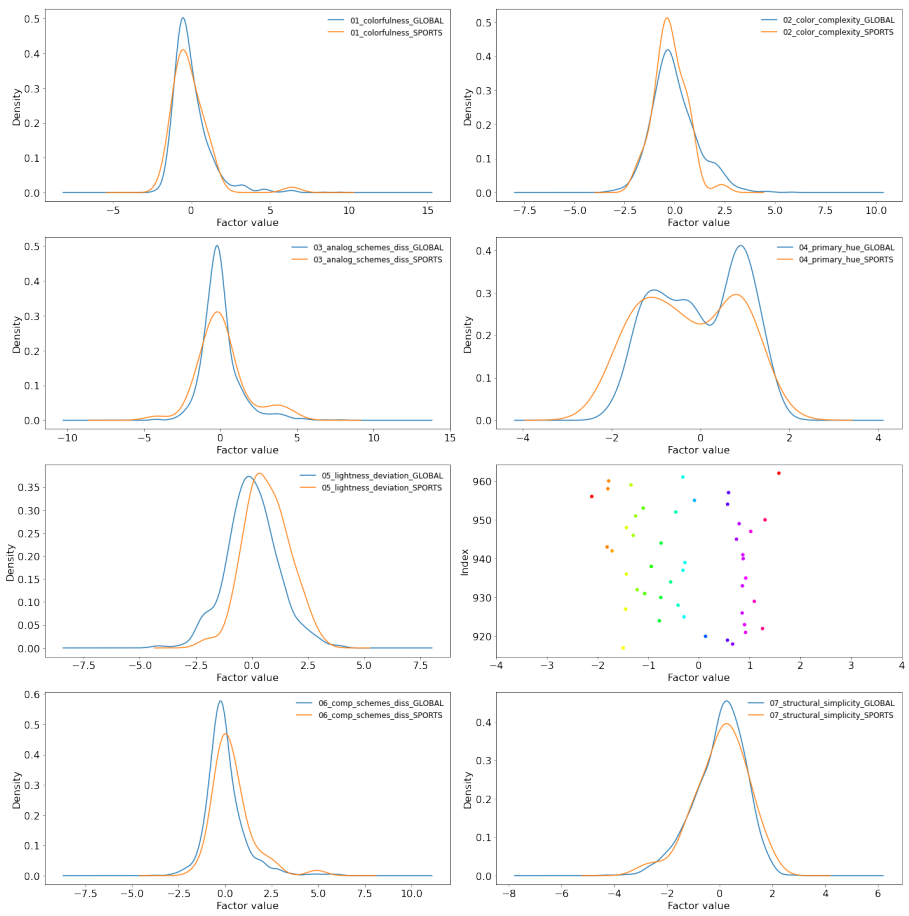


Figure C.20: Distribution of factors for the category "Sports"

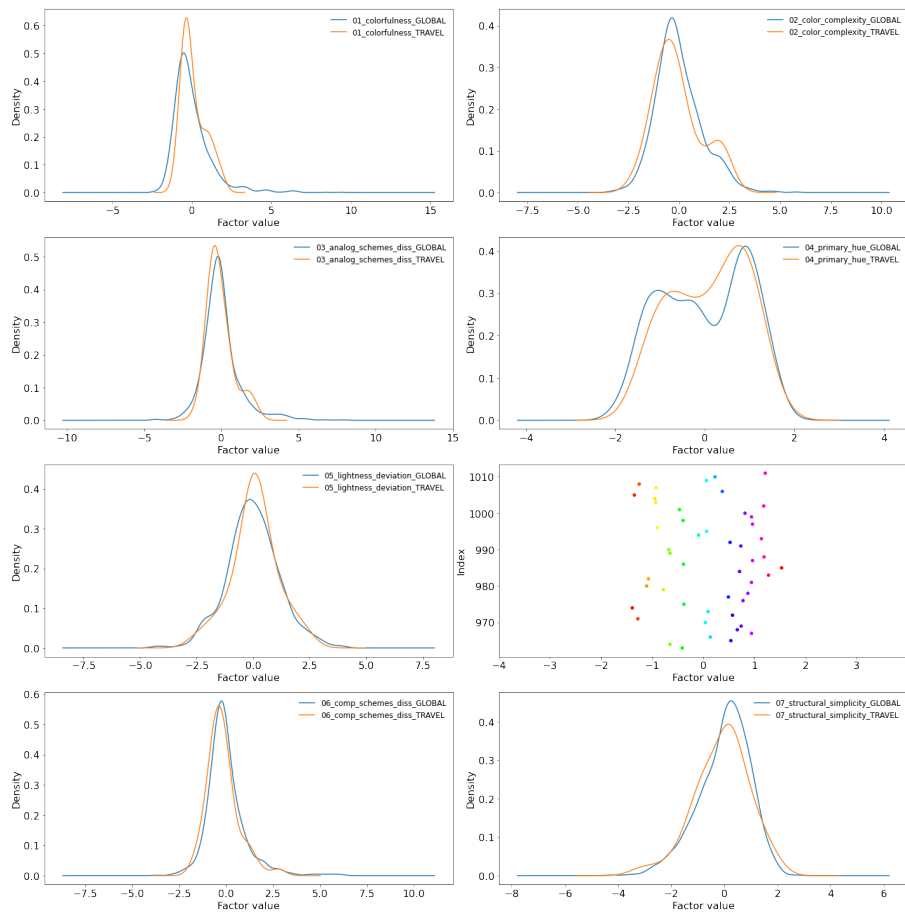


Figure C.21: Distribution of factors for the category "Travel & Tourism"

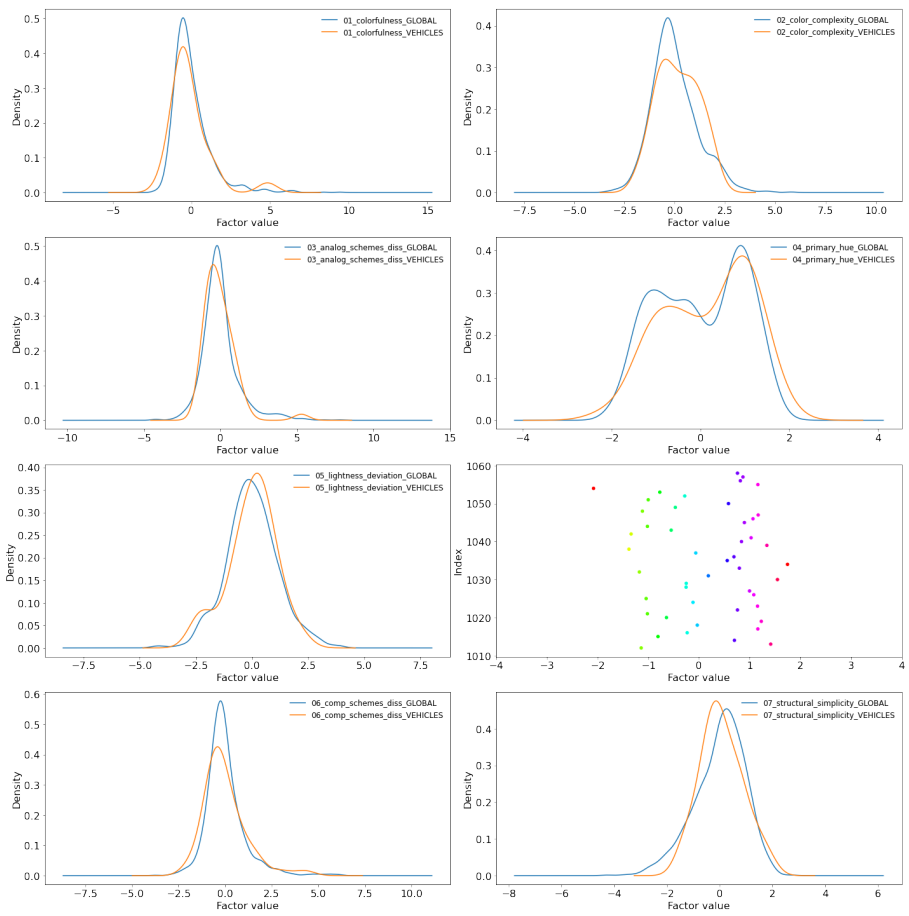


Figure C.22: Distribution of factors for the category "Vehicles"

Bibliography

- [1] Alain Abran et al. “Usability Meanings and Interpretations in ISO Standards”. In: *Software Quality Journal* 11.4 (Nov. 2003), pp. 325–338. ISSN: 0963-9314. DOI: 10.1023/A:1025869312943.
- [2] ACM. *Artifact Review and Badging*. Apr. 2018.
- [3] Enis Afgan et al. “The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update”. In: *Nucleic Acids Research* 46.W1 (July 2018), W537–W544. ISSN: 0305-1048. DOI: 10.1093/nar/gky379.
- [4] Khalid Alemerien and Magel Kenneth. “SLC: a visual cohesion metric to predict the usability of graphical user interfaces”. In: Apr. 2015, pp. 1526–1533. DOI: 10.1145/2695664.2695791.
- [5] Khalid Alemerien and Kenneth Magel. “GUIEvaluator: A Metric-tool for Evaluating the Complexity of Graphical User Interfaces”. In: *SEKE*. 2014.
- [6] Reem Alnanih and Olga Ormandjieva. “A New Quality-in-Use Model for Mobile User Interfaces”. In: Oct. 2013. DOI: 10.1109/IWSM-Mensura.2013.32.
- [7] Farah Alsudani and Matthew Casey. “The Effect of Aesthetics on Web Credibility”. In: *Proceedings of the 23rd British HCI Group Annual Conference on People and Computers: Celebrating People and Technology*. BCS-HCI ’09. Cambridge, United Kingdom: British Computer Society, 2009, pp. 512–519.
- [8] Ahamed Altaboli and Yingzi Lin. “Effects of Unity of Form and Symmetry on Visual Aesthetics of Website Interface Design”. In: vol. 56. Sept. 2012. DOI: 10.1177/1071181312561152.
- [9] Ahamed Altaboli and Yingzi Lin. “Investigating effects of screen layout elements on interface and screen design aesthetics”. In: *Advances in Human-Computer Interaction* 2011 (2011).
- [10] Joanna E Anderson et al. “Estimating the reproducibility of psychological science”. In: *Science* 349.6251 (2015).

- [11] Richard Atterer, Monika Friedemann, and Albrecht Schmidt. "Knowing the user's every move: User activity tracking for website usability evaluation and implicit interaction". In: Jan. 2006, pp. 203–212. DOI: 10.1145/1135777.1135811.
- [12] Maxim Bakaev et al. "Auto-Extraction and Integration of Metrics for Web User Interfaces". In: *Journal of Web Engineering (JWE)* 17 (Mar. 2019), pp. 561–590. DOI: 10.13052/jwe1540-9589.17676.
- [13] Aaron Bangor, Philip T. Kortum, and James T. Miller. "An Empirical Evaluation of the System Usability Scale". In: *International Journal of Human-Computer Interaction* 24.6 (2008), pp. 574–594. DOI: 10.1080/10447310802205776.
- [14] Michael Bauerly and Yili Liu. "Computational modeling and experimental investigation of effects of compositional elements on interface and design aesthetics". In: *International Journal of Human-Computer Studies* 64 (Aug. 2006), pp. 670–682. DOI: 10.1016/j.ijhcs.2006.01.002.
- [15] Michael Bauerly and Yili Liu. "Development and validation of a symmetry metric for interface aesthetics". In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. Vol. 49. 5. SAGE Publications Sage CA: Los Angeles, CA. 2005, pp. 681–685.
- [16] Michael Bauerly and Yili Liu. "Liu, Y.: Effects of Symmetry and Number of Compositional Elements on Interface and Design Aesthetics. In: *International Journal of Human-Computer Interaction* 24(3), 275–287". In: *Int. J. Hum. Comput. Interaction* 24 (Mar. 2008), pp. 275–287. DOI: 10.1080/10447310801920508.
- [17] Pablo Becker, Philip Lew, and Luis Olsina. "Specifying Process Views for a Measurement, Evaluation, and Improvement Strategy". In: *Advanced Software Engineering* 2012 (2012), 949746:1–949746:28. DOI: 10.1155/2012/949746.
- [18] C. Glenn Begley and Lee M Ellis. "Raise standards for preclinical cancer research". In: *Nature (London)* 483.7391 (2012), pp. 531–533. ISSN: 0028-0836.
- [19] Narjes Bessghaier et al. "On the Detection of Structural Aesthetic Defects of Android Mobile User Interfaces with a Metrics-Based Tool". In: 11.1 (2021). ISSN: 2160-6455. DOI: 10.1145/3410468.
- [20] George David Birkhoff. *Aesthetic Measure by George D. Birkhoff*. Harvard University Press, 1933.

- [21] Elodie Bouzekri. “Model-Based Approach to Design and Develop Usable and Dependable Recommender Systems”. In: *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems, EICS 2018, Paris, France, June 19-22, 2018*. ACM, 2018, 17:1–17:7. DOI: 10.1145/3220134.3220147.
- [22] Giorgio Brajnik. “A comparative test of web accessibility evaluation methods”. In: *Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility, ASSETS 2008, Halifax, Nova Scotia, Canada, October 13-15, 2008*. Ed. by Simon Harper and Armando Barreto. ACM, 2008, pp. 113–120. DOI: 10.1145/1414471.1414494.
- [23] Giorgio Brajnik. “Comparing accessibility evaluation tools: a method for tool effectiveness”. In: *Univers. Access Inf. Soc.* 3.3-4 (2004), pp. 252–263. DOI: 10.1007/s10209-004-0105-y.
- [24] Pearl Brereton et al. “Lessons from applying the systematic literature review process within the software engineering domain”. In: *Journal of Systems and Software* 80.4 (2007). Software Performance, pp. 571–583. ISSN: 0164-1212. DOI: <https://doi.org/10.1016/j.jss.2006.07.009>.
- [25] John Brooke. “SUS-A quick and dirty usability scale”. In: *Usability evaluation in industry* (1996), pp. 189–194.
- [26] Richard Brooks. “Search time and color coding”. In: *Psychonomic Science* 2.1-12 (1965), pp. 281–282.
- [27] Nicolas Burny and Jean Vanderdonckt. “UiLab, a Workbench for Conducting and Reproducing Experiments in GUI Visual Design”. In: *Proc. ACM Hum.-Comput. Interact.* 5.EICS (May 2021). DOI: 10.1145/3457143.
- [28] Gaëlle Calvary et al. “A Unifying Reference Framework for Multi-target User Interfaces”. In: *Interacting with Computers* 15.3 (2003), pp. 289–308. DOI: 10.1016/S0953-5438(03)00010-9.
- [29] Murilo C. Camargo, Rodolfo M. Barros, and Vanessa T. O. Barros. “Visual Design Checklist for Graphical User Interface (GUI) Evaluation”. In: *Proceedings of the 33rd Annual ACM Symposium on Applied Computing, SAC ’18*. Pau, France: Association for Computing Machinery, 2018, pp. 670–672. ISBN: 9781450351911. DOI: 10.1145/3167132.3167391.
- [30] Erik Cambria et al. “New Avenues in Opinion Mining and Sentiment Analysis”. In: *IEEE Intelligent Systems* 28.2 (2013), pp. 15–21. DOI: 10.1109/MIS.2013.30.

- [31] Sven Casteleyn, Irene Garrig'os, and Jose-Norberto Maz'on. "Ten Years of Rich Internet Applications: A Systematic Mapping Study, and Beyond". In: *ACM Trans. Web* 8.3 (July 2014). issn: 1559-1131. doi: 10.1145/2626369.
- [32] Ned Chapin et al. "Types of software evolution and software maintenance". In: *Journal of Software Maintenance and Evolution: Research and Practice* 13.1 (2001), pp. 3–30. doi: <https://doi.org/10.1002/smr.220>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/smr.220>.
- [33] Neila Chettaoui and Med Salim Bouhlel. "I2Evaluator: An Aesthetic Metric-Tool for Evaluating the Usability of Adaptive User Interfaces". In: *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2017*. Ed. by Aboul Ella Hassanien et al. Cham: Springer International Publishing, 2018, pp. 374–383. isbn: 978-3-319-64861-3.
- [34] Jon F. Claerbout and Martin Karrenbach. "Electronic documents give reproducible research a new meaning". In: *SEG Technical Program Expanded Abstracts 1992*. 2005, pp. 601–604. doi: 10.1190/1.1822162. eprint: <https://library.seg.org/doi/pdf/10.1190/1.1822162>.
- [35] Sandy Claes and Andrew Vande Moere. "Replicating an In-The-Wild Study One Year Later: Comparing Prototypes with Different Material Dimensions". In: *Proceedings of the 2017 Conference on Designing Interactive Systems*. DIS '17. Edinburgh, United Kingdom: Association for Computing Machinery, 2017, pp. 1321–1325. isbn: 9781450349222. doi: 10.1145/3064663.3064725.
- [36] Andy Cockburn, Carl Gutwin, and Alan Dix. "HARK No More: On the Preregistration of CHI Experiments". In: *Proceedings of the ACM International Conference on Human Factors in Computing Systems*. CHI '18. Montreal QC, Canada: Association for Computing Machinery, 2018. isbn: 9781450356206. doi: 10.1145/3173574.3173715.
- [37] Daniel Cohen-Or et al. "Color Harmonization". In: 25.3 (July 2006), pp. 624–630. issn: 0730-0301. doi: 10.1145/1141911.1141933.
- [38] Larry L Constantine. "Visual coherence and usability: a cohesion metric for assessing the quality of dialogue and screen designs". In: *Proceedings Sixth Australian Conference on Computer-Human Interaction*. IEEE. 1996, pp. 115–121.

- [39] Adrien Coyette, Suzanne Kieffer, and Jean Vanderdonckt. “Multi-fidelity Prototyping of User Interfaces”. In: *Human-Computer Interaction - INTERACT 2007, 11th IFIP TC 13 International Conference, Rio de Janeiro, Brazil, September 10-14, 2007, Proceedings, Part I*. Ed. by Maria Cecilia Calani Baranauskas et al. Vol. 4662. Lecture Notes in Computer Science. Springer, 2007, pp. 150–164. DOI: 10 . 1007 / 978 - 3 - 540 - 74796 - 3_16.
- [40] Biplab Deka et al. “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* (2017).
- [41] D.A Dondis. *A Primer of Visual Literacy*. MA, USA: The MIT Press, 1973.
- [42] Zoé Dubois. “Evaluating the Visual Design of Graphical User Interfaces on Mobile Devices: A Family of Experiments”. MA thesis. Louvain-la-Neuve: UCLouvain, 2022.
- [43] Catherine Dumas et al. “Usability testing of LUI: A new human-computer interface for large displays”. In: *Proceedings of the Association for Information Science and Technology* 56.1 (2019), pp. 645–647. DOI: <https://doi.org/10.1002/pra2.118>. eprint: <https://asistdl.onlinelibrary.wiley.com/doi/pdf/10.1002/pra2.118>.
- [44] Sophie Dupuy-Chessa, Yann Laurillau, and Eric Céret. “Considering Aesthetics and Usability Temporalities in a Model Based Development Process”. In: *Actes de La 28ième Conference Francophone Sur l’Interaction Homme-Machine. IHM ’16*. Fribourg, Switzerland: Association for Computing Machinery, 2016, pp. 25–35. ISBN: 9781450342438. DOI: 10 . 1145/3004107 . 3004122.
- [45] Florian Echtler and Maximilian Häuundefinedler. “Open Source, Open Science, and the Replication Crisis in HCI”. In: *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI EA ’18. Montreal QC, Canada: Association for Computing Machinery, 2018. ISBN: 9781450356213. DOI: 10 . 1145/3170427 . 3188395.
- [46] Patricia Farrugia et al. “Practical tips for surgical research: Research questions, hypotheses and objectives”. In: *Canadian Journal of Surgery* 53.4 (Aug. 2010), pp. 278–281.
- [47] Leonard P. Freedman, Iain M. Cockburn, and Timothy S. Simcoe. “The Economics of Reproducibility in Preclinical Research”. In: *PLoS Biology* 13.6 (2015). DOI: 10 . 1371/journal.pbio.10021659.

- [48] Fongling Fu, Shao-Yuan Chiu, and Chiu Hung Su. “Measuring the screen complexity of web pages”. In: *Symposium on Human Interface and the Management of Information*. Springer. 2007, pp. 720–729.
- [49] Joseph Goldberg and Anna Wichansky. *Eye tracking in usability evaluation: A practitioner’s guide*. Jan. 2003, pp. 493–516. ISBN: 0444510206.
- [50] Omar S. Gómez, Natalia Juristo, and Sira Vegas. “Replications Types in Experimental Disciplines”. In: *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. ESEM ’10. Bolzano-Bozen, Italy: Association for Computing Machinery, 2010. ISBN: 9781450300391. DOI: 10 . 1145 / 1852786 . 1852790.
- [51] Salvador González, Francisco Montero, and Pascual González. “BaLOReS: A Suite of Principles and Metrics for Graphical User Interface Evaluation”. In: *Proceedings of the 13th International Conference on Interacción Persona-Ordenador*. Interaccion ’12. Elche, Spain: Association for Computing Machinery, 2012. ISBN: 9781450313148. DOI: 10 . 1145 / 2379636 . 2379645.
- [52] Steven N. Goodman, Daniele Fanelli, and John P.A. Ioannidis. “What does research reproducibility mean?” In: *Science Translational Medicine* 8.341 (June 2016), 341ps12. DOI: 10 . 1126/scitranslmed . aaf5027.
- [53] Wayne D. Gray and Marilyn C. Salzman. “Damaged Merchandise? A Review of Experiments That Compare Usability Evaluation Methods”. In: *Human-Computer Interaction* 13.3 (1998), pp. 203–261. DOI: 10 . 1207/s15327051hci1303_2. eprint: https://doi.org/10.1207/s15327051hci1303_2.
- [54] Christian Greiffenhagen and Stuart Reeves. “Is Replication important for HCI?” In: *Proceedings of the CHI2013 Workshop on the Replication of HCI Research*. Vol. 976. CEUR Workshop Proceedings. Paris, France, Apr. 2013, p. 6.
- [55] Kai-Christoph Hamborg, Julia Hülsmann, and Kai Kaspar. “The Interplay between Usability and Aesthetics: More Evidence for the “What is Usable is Beautiful” Notion”. In: *Adv. in Hum.-Comp. Int.* 2014 (Jan. 2014). ISSN: 1687-5893. DOI: 10 . 1155/2014/946239.
- [56] Steven J Harrington et al. “Aesthetic measures for automated document layout”. In: *Proceedings of the 2004 ACM symposium on Document engineering*. 2004, pp. 109–111.

- [57] Rachel Harrison, Derek Flood, and David Duce. "Usability of mobile applications: Literature review and rationale for a new usability model". In: *Journal of Interaction Science* 1 (Jan. 2013). DOI: 10.1186/2194-0827-1-1.
- [58] Edward Hartono and Clyde W. Holsapple. "Website Visual Design Qualities: A Threefold Framework". In: *ACM Trans. Manage. Inf. Syst.* 10.1 (Apr. 2019). ISSN: 2158-656X. DOI: 10.1145/3309708.
- [59] Marc Hassenzahl, Michael Burmester, and Franz Koller. "AttrakDiff: Ein Fragebogen zur Messung wahrgenommener hedonischer und pragmatischer Qualität". In: *Mensch & computer 2003*. Springer, 2003, pp. 187–196.
- [60] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. 3rd ed. USA: Pearson Education, 2013. ISBN: 978-1292039053.
- [61] Kasper Hornbæk. "We Must Be More Wrong in HCI Research". In: *Interactions* 22.6 (Oct. 2015), pp. 20–21. ISSN: 1072-5520. DOI: 10.1145/2833093.
- [62] Kasper Hornbæk et al. "Is Once Enough? On the Extent and Content of Replications in Human-Computer Interaction". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '14. Toronto, Ontario, Canada: Association for Computing Machinery, 2014, pp. 3523–3532. ISBN: 9781450324731. DOI: 10.1145/2556288.2557004.
- [63] Melody Y. Ivory and Marti A. Hearst. "Statistical Profiles of Highly-rated Web Sites". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '02. Minneapolis, Minnesota, USA: ACM, 2002, pp. 367–374. ISBN: 1-58113-453-3. DOI: 10.1145/503376.503442.
- [64] Melody Y. Ivory, Rashmi R. Sinha, and Marti A. Hearst. "Empirically validated web page design metrics". In: *Proceedings of the CHI 2001 Conference on Human Factors in Computing Systems, Seattle, WA, USA, March 31 - April 5, 2001*. Ed. by Julie A. Jacko and Andrew Sears. ACM, 2001, pp. 53–60. DOI: 10.1145/365024.365035.
- [65] Henry Kaiser. "An index of factorial simplicity". In: *Psychometrika* 39.1 (Mar. 1974), pp. 31–36. DOI: 10.1007/BF02291575.
- [66] Tridib Kumar Kar and Jasni Mohd Zain. "Measuring Aesthetics of Mandarin Learning Web Pages : Are Users ' Perceptions Congruent with the Measured Values of Aesthetics-Measurement Application (AMA) ?" In: 2007.

- [67] Vince Kellen. *Business Performance Measurement – At the Crossroads of Strategy, Decision-Making, Learning and Information Visualization*. DePaul University, Feb. 2003.
- [68] Suzanne Kieffer. “ECOVAL: Ecological Validity of Cues and Representative Design in User Experience Evaluations”. In: *AIS Transactions on Human-Computer Interaction* 9.2 (2017), pp. 149–172.
- [69] Won Chul Kim and James D. Foley. “DON: user interface presentation design assistant”. In: *Proceedings of the 3rd Annual ACM Symposium on User Interface Software and Technology, Snowbird, Utah, USA, October 3-5, 1990*. Ed. by Scott E. Hudson. UIST 1990. ACM, 1990, pp. 10–20. DOI: 10.1145/97924.97926.
- [70] Won Chul Kim and James D. Foley. “Providing high-level control and expert assistance in the user interface presentation design”. In: *Human-Computer Interaction, INTERACT '93, IFIP TC13 International Conference on Human-Computer Interaction, 24-29 April 1993, Amsterdam, The Netherlands, jointly organised with ACM Conference on Human Aspects in Computing Systems CHI'93*. Ed. by Bert Arnold, Gerrit C. van der Veer, and Ted N. White. ACM, 1993, pp. 430–437. DOI: 10.1145/169059.169346.
- [71] Barbara Kitchenham et al. “Systematic literature reviews in software engineering – A tertiary study”. In: *Information and Software Technology* 52.8 (2010), pp. 792–805. ISSN: 0950-5849. DOI: <https://doi.org/10.1016/j.infsof.2010.03.006>.
- [72] Peter Kokol, Ivan Rozman, and Vlado Venuti. “User interface metrics”. In: *ACM SIGPLAN Notices* 30.4 (1995), pp. 36–38.
- [73] Chien-Yin Lai et al. “Computational models and experimental investigations of effects of balance and symmetry on the aesthetics of text-overlaid images”. In: *International Journal of Human-Computer Studies* 68.1 (2010), pp. 41–56. ISSN: 1071-5819. DOI: <https://doi.org/10.1016/j.ijhcs.2009.08.008>.
- [74] Carlos Lara-Alvarez and Tania Reyes. “A Geometric Approach to Harmonic Color Palette Design”. In: *Color Research Application* 44 (Sept. 2017). DOI: 10.1002/co1.22292.
- [75] Judy Lau and David N Wilson. “Experience with Software Quality/Usability Metrics”. In: *Software Quality Management VI*. Springer, 1998, pp. 53–69.

- [76] Rubén Ledesma, Pedro Valero-Mora, and Guillermo Macbeth. “The Scree Test and the Number of Factors: a Dynamic Graphics Approach”. In: *The Spanish Journal of Psychology* 18 (June 2015). DOI: 10.1017/sjp.2015.13.
- [77] Luis A. Leiva et al. “Omnis Prædictio: Estimating the full spectrum of human performance with stroke gestures”. In: *Int. J. Hum. Comput. Stud.* 142 (2020), p. 102466. DOI: 10.1016/j.ijhcs.2020.102466.
- [78] James R Lewis. “IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use”. In: *International Journal of Human-Computer Interaction* 7.1 (1995), pp. 57–78.
- [79] James R Lewis. “Psychometric evaluation of the PSSUQ using data from five years of usability studies”. In: *International Journal of Human-Computer Interaction* 14.3-4 (2002), pp. 463–488.
- [80] James R Lewis. “Sample sizes for usability tests: mostly math, not magic, interactions, v. 13 n. 6”. In: *November + December* 13.6 (2006), pp. 29–33.
- [81] James R. Lewis and Jeff Sauro. “The Factor Structure of the System Usability Scale”. In: *Human Centered Design*. Ed. by Masaaki Kurosu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 94–103. ISBN: 978-3-642-02806-9.
- [82] Alessandro Liberati et al. “The PRISMA statement for reporting systematic reviews and meta-analyses of studies that evaluate health care interventions: explanation and elaboration”. In: *PLoS Medicine* 6.7 (July 2009), pp. 1–22. DOI: 10.1371/journal.pmed.1000100.
- [83] Rensis Likert. “A technique for the measurement of attitudes”. In: *Archives of Psychology* 22.140 (1932), pp. 55–.
- [84] Sangdon Lim and Seungmin Jahng. “Determining the number of factors using parallel analysis and its recent variants.” In: *Psychological Methods* 24.4 (Aug. 2019), pp. 452–467. DOI: 10.1037/met0000230.
- [85] Gitte Lindgaard et al. “Attention web designers: You have 50 milliseconds to make a good first impression! Behaviour and Information Technology, 25(2), 115-126”. In: *Behaviour IT* 25 (Mar. 2006), pp. 115–126. DOI: 10.1080/01449290500330448.
- [86] Adriana Lopes et al. “Evaluating HCI Design with Interaction Modeling and Mockups - A Case Study”. In: *ICEIS 2015 - Proceedings of the 17th International Conference on Enterprise Information Systems, Volume 3, Barcelona, Spain, 27-30 April, 2015*. Ed. by Slimane Hammoudi, Leszek A. Maciaszek, and Ernest Teniente. SciTePress, 2015, pp. 79–87. DOI: 10.5220/0005374200790087.

- [87] Salvador González López, Francisco Montero Simarro, and Pascual González López. “BaLOReS: A Framework for Quantitative User Interface Evaluation”. In: *New Trends in Interaction, Virtual Reality and Modeling*. Ed. by Victor M.R. Penichet, Antonio Peñalver, and José A. Gallud. London: Springer London, 2013, pp. 127–143. ISBN: 978-1-4471-5445-7. DOI: 10.1007/978-1-4471-5445-7_10.
- [88] SM Luria, David F Neri, and Alan R Jacobsen. “The effects of set size on color matching using CRT displays”. In: *Human factors* 28.1 (1986), pp. 49–61.
- [89] Nathan Magrofuoco, Paolo Roselli, and Jean Vanderdonckt. “Two-dimensional Stroke Gesture Recognition: A Survey”. In: *ACM Comput. Surv.* 54.7 (2022), 155:1–155:36. DOI: 10.1145/3465400.
- [90] Nathan Magrofuoco and Jean Vanderdonckt. “Gelicit: A Cloud Platform for Distributed Gesture Elicitation Studies”. In: *Proc. ACM Hum.-Comput. Interact.* 3.EICS (June 2019). DOI: 10.1145/3331148.
- [91] Rohit Mahajan and Ben Shneiderman. “Visual and Textual Consistency Checking Tools for Graphical User Interfaces”. In: *IEEE Trans. Software Eng.* 23 (1997), pp. 722–735.
- [92] Anna Beatriz Marques, Simone D. J. Barbosa, and Tayana Conte. “A comparative evaluation of interaction models for the design of interactive systems”. In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing, Pisa, Italy, April 4-8, 2016*. Ed. by Sascha Ossowski. ACM, 2016, pp. 173–180. DOI: 10.1145/2851613.2851679.
- [93] Yutaka Matsuda. “Color design”. In: *Asakura Shoten* 2.4 (1995), p. 10.
- [94] Patrick Mbenza and Nicolas Burny. “Computing aesthetics of concrete user interfaces”. In: *EICS '20: ACM SIGCHI Symposium on Engineering Interactive Computing Systems, Sophia Antipolis, France, June 23-26, 2020*. Ed. by Judy Bowen, Jean Vanderdonckt, and Marco Winckler. ACM, 2020, 4:1–4:8. DOI: 10.1145/3393672.3398493.
- [95] Aliaksei Miniukovich and Antonella De Angeli. “Quantification of interface visual complexity”. In: *Proceedings of the 2014 international working conference on advanced visual interfaces*. 2014, pp. 153–160.
- [96] Aliaksei Miniukovich and Antonella De Angeli. “Visual Impressions of Mobile App Interfaces”. In: *Proc. of the 8th Nordic Conf. on Human-Computer Interaction*. NordiCHI '14. Helsinki, Finland: ACM, 2014, pp. 31–40. ISBN: 9781450325424. DOI: 10.1145/2639189.2641219.

- [97] Jasni Mohamad Zain, Mengkar Tey, and Yingsoon Goh. "Probing a Self-Developed Aesthetics Measurement Application (SDA) in Measuring Aesthetics of Mandarin Learning Web Page Interfaces". In: *Computing Research Repository - CORR* 8 (Jan. 2011).
- [98] Francisco Montero and Víctor López. "GUILayout++: Supporting Prototype Creation and Quality Evaluation for Abstract User Interface Generation". In: 2011.
- [99] Kevin Mullet and Darrell Sano. *Designing Visual Interfaces. Communication Oriented Techniques*. USA: Pearson Education, 1995.
- [100] David Chek Ling Ngo and John G Byrne. "Aesthetic measures for screen design". In: *Proceedings 1998 Australasian Computer Human Interaction Conference. OzCHI'98 (Cat. No. 98EX234)*. IEEE. 1998, pp. 64–71.
- [101] David Chek Ling Ngo and John G Byrne. "Another look at a model for evaluating interface aesthetics". In: *International Journal of Applied Mathematics and Computer Science* 11 (2001), pp. 515–535.
- [102] David Chek Ling Ngo and John G Byrne. "Application of an aesthetic evaluation model to data entry screens". In: *Computers in Human Behavior* 17.2 (2001), pp. 149–185.
- [103] David Chek Ling Ngo, Azman Samsudin, and Rosni Abdullah. "Aesthetic measures for assessing graphic screens". In: *Journal of Information Science and Engineering* 16.1 (2000), pp. 97–116. DOI: 10.6688/JISE.2000.16.1.4.
- [104] David Chek Ling Ngo, Lian Seng Teo, and John G Byrne. "A mathematical theory of interface aesthetics". In: *Visual mathematics*. Vol. 2. 4. Mathematical Institute SASA. 2000.
- [105] David Chek Ling Ngo, Lian Seng Teo, and John G Byrne. "Formalising guidelines for the design of screen layouts". In: *Displays* 21.1 (2000), pp. 3–15.
- [106] David Chek Ling Ngo, Lian Seng Teo, and John G. Byrne. "Modelling Interface Aesthetics". In: *Inf. Sci.* 152.1 (June 2003), pp. 25–46. ISSN: 0020-0255. DOI: 10.1016/S0020-0255(02)00404-8.
- [107] Jakob Nielsen and Jonathan Levy. "Measuring Usability: Preference vs. Performance". In: *Communications of the ACM* 37.4 (Apr. 1994), pp. 66–75. ISSN: 0001-0782. DOI: 10.1145/175276.175282.
- [108] James Noble and Larry L Constantine. "Interactive design metric visualization: Visual metric support for user interface design". In: *Proceedings Sixth Australian Conference on Computer-Human Interaction*. IEEE. 1996, pp. 213–220.

- [109] Raquel Oliveira, Sophie Dupuy-Chessa, and Gaëlle Calvary. “Plasticity of User Interfaces: Formal Verification of Consistency”. In: *Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. EICS '15. Duisburg, Germany: Association for Computing Machinery, 2015, pp. 260–265. ISBN: 9781450336468. DOI: 10.1145/2774225.2775078.
- [110] Luis Olsina et al. “Updating Quality Models for Evaluating New Generation Web Applications”. In: *J. Web Eng.* 11.3 (2012), pp. 209–246.
- [111] Luis Olsina et al. “Using Web Quality Models and a Strategy for Purpose-Oriented Evaluations”. In: *J. Web Eng.* 10.4 (2011), pp. 316–352.
- [112] Karla Ordoñez, José Hilera, and Samanta Cueva. “Model-Driven Development of Accessible Software: A systematic literature review”. In: 2020.
- [113] International Standard Organization. *ISO/IEC 25010:2011, Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuARE) — System and software quality models*. 2011.
- [114] International Standard Organization. *ISO/IEC 9421-210:2019, Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems*. 2019.
- [115] Antti Oulasvirta et al. “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*. UIST '18 Adjunct. Berlin, Germany: ACM, 2018, pp. 16–19. ISBN: 978-1-4503-5949-8. DOI: 10.1145/3266037.3266087.
- [116] Mehdi Ousmer et al. “Recognizing 3D Trajectories as 2D Multi-stroke Gestures”. In: *Proc. ACM Hum. Comput. Interact.* 4.ISS (2020), 198:1–198:21. DOI: 10.1145/3427326.
- [117] Mr DS Panchal and SN Deshmukh. “Code Instrumentation using Dynamic Program Analysis Tools”. In: *International Journal of Engineering Research* 3.4 (2014).
- [118] Harold Pashler and Eric-Jan Wagenmakers. “Editors’ Introduction to the Special Section on Replicability in Psychological Science: A Crisis of Confidence?” In: *Perspectives on Psychological Science* 7.6 (2012). PMID: 26168108, pp. 528–530. DOI: 10.1177/1745691612465253. eprint: <https://doi.org/10.1177/1745691612465253>.
- [119] Prasad Patil, Roger D. Peng, and Jeffrey T. Leek. “A visual tool for defining reproducibility and replicability”. In: *Nature Human Behaviour* 3.7 (2019), pp. 650–652. DOI: 10.1038/s41562-019-0629-z.

- [120] Roger D. Peng. “Reproducible Research in Computational Science”. In: *Science* 334.6060 (2011), pp. 1226–1227. issn: 0036-8075. doi: 10 . 1126/science.1213847. eprint: <https://science.sciencemag.org/content/334/6060/1226.full.pdf>.
- [121] Helen Petrie and Nigel Bevan. “The Evaluation of Accessibility, Usability, and User Experience”. In: *The Universal Access Handbook*. 2009.
- [122] Stefan Pröll, Kristof Meixner, and Andreas Rauber. “Precise Data Identification Services for Long Tail Research Data”. In: (Jan. 2016). doi: 10 . 6084/M9 . FIGSHARE . 3847632.
- [123] Helen C Purchase et al. “Investigating objective measures of web page aesthetics and usability”. In: *Proceedings of the Twelfth Australasian User Interface Conference-Volume 117*. 2011, pp. 19–28.
- [124] Ghulam Jilani Quadri and Paul Rosen. *You Can’t Publish Replication Studies (and How to Anyways)*. 2019. arXiv: 1908 . 08893 [cs . HC].
- [125] Katharina Reinecke and Krzysztof Z. Gajos. “Quantifying Visual Preferences Around the World”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’14. Toronto, Ontario, Canada: ACM, 2014, pp. 11–20. isbn: 978-1-4503-2473-1. doi: 10 . 1145/2556288 . 2557052.
- [126] Katharina Reinecke et al. “Predicting Users’ First Impressions of Website Aesthetics with a Quantification of Perceived Visual Complexity and Colorfulness”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’13. Paris, France: Association for Computing Machinery, 2013, pp. 2049–2058. isbn: 9781450318990. doi: 10 . 1145/2470654 . 2481281.
- [127] David B. Resnik and Adil E. Shamoo. “Reproducibility and Research Integrity”. In: *Accountability in Research* 24.2 (2017). PMID: 27820655, pp. 116–123. doi: 10 . 1080 / 08989621 . 2016 . 1257387. eprint: <https://doi.org/10.1080/08989621.2016.1257387>.
- [128] Andreas Riegler and Clemens Holzmann. “Measuring Visual User Interface Complexity of Mobile Applications With Metrics”. In: *Interacting with Computers* 30.3 (Apr. 2018), pp. 207–223. issn: 0953-5438. doi: 10 . 1093/iwc/iwy008. eprint: <https://academic.oup.com/iwc/article-pdf/30/3/207/24805365/iwy008.pdf>.
- [129] Andreas Riegler and Clemens Holzmann. “UI-CAT: Calculating User Interface Complexity Metrics for Mobile Applications”. In: *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia*. MUM ’15. Linz, Austria: Association for Computing Machinery,

- 2015, pp. 390–394. ISBN: 9781450336055. DOI: 10 . 1145 / 2836041 . 2841214.
- [130] Christian Rohrer. *When to Use Which User-Experience Research Methods*. 2014. URL: <https://www.nngroup.com/articles/which-ux-research-methods/> (visited on 10/12/2014).
 - [131] Carolyn Salimun et al. “The effect of aesthetically pleasing composition on visual search performance”. In: *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*. 2010, pp. 422–431.
 - [132] Geir Kjetil Sandve et al. “Ten Simple Rules for Reproducible Computational Research.” In: *PLoS Computational Biology* 9.10 (2013).
 - [133] Jeff Sauro. *A practical guide to measuring usability: 72 answers to the most common questions about quantifying the usability of websites and software*. Measuring Usability LCC, 2010.
 - [134] Jonathan W. Schooler. “Metascience could rescue the “replication crisis””. In: *Nature* 515.9 (Nov. 2014), p. 9. ISSN: 0028-0836. DOI: 10 . 1038/515009a.
 - [135] Martin Schrepp and Jörg Thomaschewski. “Design and Validation of a Framework for the Creation of User Experience Questionnaires.” In: *International Journal of Interactive Multimedia & Artificial Intelligence* 5.7 (2019).
 - [136] Andrew Sears. “AIDE: a step toward metric-based interface development tools”. In: *Proceedings of the 1995 8th Annual Symposium on User Interface Software and Technology, UIST’95* ; Conference date: 14-11-1995 Through 17-11-1995. Dec. 1995, pp. 101–110.
 - [137] Ahmed Seffah et al. “Usability measurement and metrics: A consolidated model”. In: *Software Quality Journal* 14 (June 2006), pp. 159–178. DOI: 10 . 1007/s11219-006-7600-8.
 - [138] Bernard Senach. *Evaluation ergonomique des interfaces homme-machine : une revue de la littérature*. Research Report RR-1180. INRIA, 1990.
 - [139] Johanna Silvennoinen, Marlene Vogel, and Sari Kujala. “Experiencing Visual Usability and Aesthetics in Two Mobile Application Contexts”. In: *J. Usability Studies* 10.1 (Nov. 2014), pp. 46–62. ISSN: 1931-3357. DOI: 10 . 5555/2817310 . 2817314.
 - [140] Andreas Sonderegger and Jürgen Sauer. “The influence of design aesthetics in usability testing: Effects on user performance and perceived usability”. In: *Applied ergonomics* 41 (Nov. 2009), pp. 403–10. DOI: 10 . 1016/j . apergo . 2009 . 09 . 002.

- [141] Guangfeng Song. “Analysis of web page complexity through visual segmentation”. In: *International Conference on Human-Computer Interaction*. Springer. 2007, pp. 114–123.
- [142] Makram Soui et al. “Assessing the quality of mobile graphical user interfaces using multi-objective optimization”. In: *Soft Comput.* 24.10 (2020), pp. 7685–7714. DOI: 10.1007/s00500-019-04391-8.
- [143] Makram Soui et al. “PLAIN: PLugin for predicting the usAbility of mobile user INterface”. In: *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2017) - Volume 1: GRAPP, Porto, Portugal, February 27 - March 1, 2017*. Ed. by Ana Paula Cláudio, Dominique Bechmann, and José Braz. SciTePress, 2017, pp. 127–136. DOI: 10.5220/0006171201270136.
- [144] Christian Stickel, Martin Ebner, and Andreas Holzinger. “The XAOS metric—understanding visual complexity as measure of usability”. In: *Symposium of the Austrian HCI and Usability Engineering Group*. Springer. 2010, pp. 278–290.
- [145] Dennis J. Streveler and Anthony I. Wasserman. “Quantitative measures of the spatial properties of screen designs”. In: *Proceedings of the IFIP TC13 First International Conference on Human-Computer Interaction*. Ed. by B. Shackel. Interact ’84. Amsterdam: Elsevier Science Publishers B.V. (North Holland), 1984, pp. 81–89.
- [146] Seyyed Ehsan Salamati Taba et al. “An Exploratory Study on the Relation between User Interface Complexity and the Perceived Quality”. In: *Web Engineering*. Ed. by Sven Casteleyn, Gustavo Rossi, and Marco Winckler. Cham: Springer International Publishing, 2014, pp. 370–379. ISBN: 978-3-319-08245-5.
- [147] Toby J. Teorey, Dongqing Yang, and James P. Fry. “A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model”. In: *ACM Comput. Surv.* 18.2 (June 1986), pp. 197–222. ISSN: 0360-0300. DOI: 10.1145/7474.7475.
- [148] N Tractinsky, A.S Katz, and D Ikar. “What is beautiful is usable”. In: *Interacting with Computers* 13.2 (2000), pp. 127–145. ISSN: 0953-5438. DOI: [https://doi.org/10.1016/S0953-5438\(00\)00031-X](https://doi.org/10.1016/S0953-5438(00)00031-X).
- [149] Stefan Trausan-Matu and Brahma Dathan. “Perceived aesthetics of user-modifiable layouts: a comparison between an unspecified design and a GUI”. In: *Proc. of 13th International Conference on Human Computer Interaction*. RoCHI ’16. Iasi, Romania: Matrix ROM, 2016, pp. 22–25.

- [150] Eric W.K. Tsang and Kai-Man Kwan. "Replication and Theory Development in Organizational Science: A Critical Realist Perspective". In: *The Academy of Management Review* 24.4 (Oct. 1999), pp. 759–780. doi: 10.2307/259353.
- [151] Ledyard R Tucker. "The objective definition of simple structure in linear factor analysis". In: *Psychometrika* 20.3 (Sept. 1955), pp. 209–225. doi: 10.1007/bf02289018.
- [152] Thomas S. Tullis. "A System for Evaluating Screen Formats". In: *Proceedings of the Human Factors Society Annual Meeting* 30.12 (1986), pp. 1216–1220. doi: 10.1177/154193128603001218. eprint: <https://doi.org/10.1177/154193128603001218>.
- [153] Thomas S. Tullis. "An Evaluation of Alphanumeric, Graphic, and Color Information Displays". In: *Human Factors* 23.5 (1981), pp. 541–550. doi: 10.1177/001872088102300504. eprint: <https://doi.org/10.1177/001872088102300504>.
- [154] Silvia Uribe, Federico Álvarez, and José Manuel Menéndez. "User's Web Page Aesthetics Opinion: A Matter of Low-Level Image Descriptors Based on MPEG-7". In: *ACM Trans. Web* 11.1 (Mar. 2017), 5:1–5:25. ISSN: 1559-1131. doi: 10.1145/3019595.
- [155] Jean Vanderdonckt. "Accessing Guidelines Information with Sierra". In: *Human—Computer Interaction: Interact '95*. Ed. by Knut Nordby et al. Boston, MA: Springer US, 1995, pp. 311–316. ISBN: 978-1-5041-2896-4. doi: 10.1007/978-1-5041-2896-4_52.
- [156] Jean Vanderdonckt. *Guide ergonomique des interfaces homme-machine*. Namur, Belgique: Presses Universitaires de Namur, 1994.
- [157] Jean Vanderdonckt and Xavier Gillo. "Visual Techniques for Traditional and Multimedia Layouts". In: *Proceedings of the ACM Int. Conf. on Advanced Visual Interfaces*. Ed. by Maria Francesca Costabile et al. AVI '94. Bari, Italy: ACM, 1994, pp. 95–104. doi: 10.1145/192309.192334.
- [158] Jean Vanderdonckt, Missiri Ouedraogo, and Banta Ygueitengar. "A Comparison of Placement Strategies for Effective Visual Design". In: (Sept. 1999).
- [159] Jean Vanderdonckt et al. "USIXML: a User Interface Description Language for Specifying Multimodal User Interfaces". In: *W3C Workshop on Multimodal Interaction* (Jan. 2004).

- [160] Arnold P. O. S. Vermeeren et al. "User Experience Evaluation Methods: Current State and Development Needs". In: *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*. NordiCHI '10. Reykjavik, Iceland: Association for Computing Machinery, 2010, pp. 521–530. ISBN: 9781605589343. DOI: 10.1145/1868914.1868973.
- [161] Santiago Villarreal-Narvaez et al. "A Systematic Review of Gesture Elicitation Studies: What Can We Learn from 216 Studies". In: *Proceedings of ACM Int. Conf. on Designing Interactive Systems (DIS'20)*. 2020.
- [162] Walleriusz. *Principles of design*. 2015. URL: <https://fr.slideshare.net/walleriusz/principles-of-design-44496102/4> (visited on 07/14/2020).
- [163] Andy Whitefield, FRANK Wilson, and John Dowell. "A framework for human factors evaluation". In: *Behaviour & Information Technology* 10.1 (1991), pp. 65–79. DOI: 10.1080/01449299108924272. eprint: <https://doi.org/10.1080/01449299108924272>.
- [164] Max Wilson et al. "RepliCHI SIG: From a Panel to a New Submission Venue for Replication". In: *CHI '12 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '12. Austin, Texas, USA: Association for Computing Machinery, 2012, pp. 1185–1188. ISBN: 9781450310161. DOI: 10.1145/2212776.2212419.
- [165] Max L. Wilson et al. "RepliCHI: The Workshop II". In: *CHI '14 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '14. Toronto, Ontario, Canada: Association for Computing Machinery, 2014, pp. 33–36. ISBN: 9781450324748. DOI: 10.1145/2559206.2559233.
- [166] Claes Wohlin et al. *Experimentation in Software Engineering: An Introduction*. International Series in Software Engineering. Berlin: Springer, 2012.
- [167] Keith A. Yeomans and Paul A. Golder. "The Guttman-Kaiser Criterion as a Predictor of the Number of Common Factors". In: *Journal of the Royal Statistical Society. Series D (The Statistician)* 31.3 (1982), pp. 221–229. ISSN: 00390526, 14679884.
- [168] Jasni Mohamad Zain, Mengkar Tey, and Goh Ying Soon. "Using Aesthetic Measurement Application (AMA) to Measure Aesthetics of Web Page Interfaces". In: *Proc. of 4th Int. Conf. on Natural Computation*. ICNC '08. IEEE Computer Society, 2008, pp. 96–100. DOI: 10.1109/ICNC.2008.764.

- [169] Mathieu Zen. “Metric-based evaluation of graphical user interfaces: model, method, and software support”. In: *Proceedings of the 5th ACM SIGCHI symposium on Engineering interactive computing systems*. 2013, pp. 183–186.
- [170] Mathieu Zen and Jean Vanderdonckt. “Assessing User Interface Aesthetics Based on the Inter-Subjectivity of Judgment”. In: *HCI 2016 - Fusion! Proceedings of the 30th International BCS Human Computer Interaction Conference, BCS HCI 2016, Bournemouth University, Poole, UK, 11-15 July 2016*. Ed. by Shamal Faily et al. Workshops in Computing. BCS, 2016.
- [171] Mathieu Zen and Jean Vanderdonckt. “Towards an evaluation of graphical user interfaces aesthetics based on metrics”. In: *Proc. of IEEE 8th International Conference on Research Challenges in Information Science*. Ed. by Marko Bajec, Martine Collard, and Rébecca Deneckère. RCIS '14. Marrakech, Morocco: IEEE, 2014, pp. 1–12. DOI: 10 . 1109/RCIS . 2014 . 6861050.
- [172] Ping Zhang and Na Li. “The importance of affective quality”. In: *Commun. ACM* 48 (Sept. 2005), pp. 105–108. DOI: 10 . 1145 / 1081992 . 1081997.
- [173] Xianjun Sam Zheng et al. “Correlating Low-level Image Statistics with Users - Rapid Aesthetic and Affective Judgments of Web Pages”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '09. Boston, MA, USA: ACM, 2009, pp. 1–10. ISBN: 978-1-60558-246-7. DOI: 10 . 1145 / 1518701 . 1518703.