Bachelor's Thesis

AUTOMATING WEBSITE DESIGN WITH A HUMAN IN THE LOOP

LUKAS ANSTETT

August 8, 2022

Advisors:
Kallistos Weis — Chair of Software Engineering
Christian Kaltenecker — Chair of Software Engineering

Examiners:
Prof. Dr. Sven Apel — Chair of Software Engineering
Prof. Dr. Anna Maria Feit — Computational Interaction Group

Chair of Software Engineering
Saarland Informatics Campus
Saarland University

UNIVERSITÄT
DES
SAARLANDES

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken,_____ _____

(Datum/Date) (Unterschrift/Signature)

ABSTRACT

Designing a website is a very creative but time-consuming process. Not only can the designer be distracted from the actual design work due to technical issues, it also often takes a significant amount of creative effort to to improve the current version until it fulfills the designer's goals.

A large amount of time can be saved by automating this iterative process. However, the criteria for what constitutes a "good" or "bad" design generally are subjective and depend on the designer and their opinions and wishes. As such, trying to partially automate a (website) design process poses a challenge, in that the design process requires cooperation between the human user and an algorithm.

In this thesis, we present a semi-automated design process where we incorporate both a human and an algorithm. To do this, we let the user (i. e., the human) rate generated designs shown to them by the algorithm and employ a genetic algorithm to further develop these designs. This results in a design loop where we include the human and their opinions.

To find out whether and to which extent our approach improves the design experience, we evaluated it using a website design test scenario and a user study. We found that it can produce satisfactory results in a fast and approachable manner. However, there are still lots of possibilities for further improvement and exploration of this approach in the future.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

HTML Hypertext Markup Language
CSS Cascading Style Sheets
XML Extensible Markup Language

1 INTRODUCTION

Design is ubiquitous in our world: every man-made thing has been designed by someone. In the process of design, one has to make many decisions about the final product, which might turn out to be good or bad in the end; this decision-making process is up to the designer who decides the criteria and priorities for a certain design project. These criteria can be entirely objective, but subjective criteria also play a role.

In this thesis, we look at website design in particular. Websites have become commonplace today, but they are also a product of a design process. Objective criteria for website designs are for example accessibility, responsiveness (websites working across different kinds of devices), or compatibility across browsers. Subjective criteria include for example ease of use, and aesthetics.

No matter which goals are important to the designer of the website, they are the one who takes the decisions for the final product. But it is impossible to create a perfect design right from the start, which is why most design processes work iteratively: Try out one design, evaluate it, modify it and try again. We also call this iterative design process the *design loop*.

In the context of website design, the designer usually creates the website design using a style sheet. Then, they render the website in a browser and look at it to determine which changes they would like to apply. This process will be repeated until they are satisfied with the design.

We want to enhance this iterative process by automating it. Thereby, we have two objectives. First, we want to ease the creation of an appealing website. Second, our aim is to speed up the creative process.

The largest challenge in doing so is optimizing subjective criteria. Coming up with general design rules that fit everybody's taste and can be followed by an algorithm is very difficult since multiple people often perceive the same design very differently and subsequently hold differing opinions when it comes to, for instance, aesthetics.

As such, we incorporate the human in an automated design process to evaluate the designs created by the algorithm, to combine objective knowledge contributed by the algorithm with subjective knowledge contributed by the human.

The goal of this thesis is to examine how a relatively naïve algorithm with little to no knowledge of website design can be used to augment the human design process, and to explore the concept of human-centered automated design.

Our approach consists of multiple steps. First, as a starting point we generate a number of arbitrary designs. These designs are shown to the user, who then gives ratings to them. These ratings are then used to generate a new set of designs using a genetic algorithm. These new designs are then once again rated by the user, completing the cycle. This process repeats until the user has found a satisfactory design.

This results in a relatively fast design loop where the user and the algorithm are alternating in taking action.

Overview

After this introduction, we first give an overview over different related work in Chapter 2 where we cover previous approaches to automating website design.

In Chapter 3, we cover underlying concepts. There, we give a brief overview on website design in the context of this thesis, and introduce the concept of feature models, which we use to model website designs. Finally, we explain the basics of the inner workings of genetic algorithms.

Following this chapter, we describe our approach in Chapter 4. There we explain how our automated design process process works and incorporates human feedback. In Chapter 5, we then showcase the website design test scenario that we developed for this thesis.

In Chapter 6, we discuss the way in which we evaluated our design process and the results of the study which we conducted. Finally, in Chapter 7 we give an outlook on further ideas and possibilities in this area of research before summarizing our findings in Chapter 8.

2 RELATED WORK

In this chapter, we give an overview about related topics and prior works. Doing so, we will showcase works used as inspiration and highlight the distinctive facets of this thesis.

Automating user interface and website design

Gajos and Weld [9] developed a way to algorithmically layout and render an interactive user interface on various devices. The resulting layouts are based on and can adapt to the device's characteristics and the usage patterns of the interface.

Based on this work, Oulasvirta et al. [16] formalized layout design problems as a combinatorial optimization problem, which they solve using integer programming. Laine et al. [12] applied this approach to website layouts, where they personalize a website layout for a specific end user based on their preferences.

As opposed to this line of work, in this thesis we consider the *user* to be the person in charge of designing the website rather than the reader of the finished website, and focus on their specific preferences to improve their manual design process.

Combining automated and human-centered design

To bring in a human user into an automated layout process, Todi, Weir, and Oulasvirta [24] developed a sketching tool with an included real-time layout optimizer. While the user is sketching a layout, it automatically proposes various improvements by inferring the designer's goals.

From a different perspective, Feit et al. [8] described the process of designing a new standard AZERTY keyboard for the French language. This was solved as a computational optimization problem, while at the same time incorporating feedback by users. The entire process took multiple years and took into account the preferences of many stakeholders on a nation-wide scale.

In this thesis, we focus on using only little domain knowledge in our automated optimization process. As such, we base the entire optimization process purely on the designer's opinions and do not assume any other criteria. Also, we put a larger emphasis on the iterative process and separate the automated and manual design parts clearly to the user. Finally, we also use a different method to automate the design process by using a genetic algorithm rather than a combinatorial approach.

Genetic algorithms in design and art

We use a *genetic algorithm* in our design process. Genetic algorithms have shown to be able to produce good results for domains that are difficult to explore, like design in our case [19].

One practical application of genetic algorithms in the design space has been presented by Hornby et al. [11], where they show how they use them for designing antennae for satellites. For this, they calculated the fitness of each candidate antenna based on a simulation. With this approach, they were able to find novel and more effective designs for antennae.

However, the approach of using evolutionary algorithms also works very well for artistic and creative purposes [20]. For example, Sims [22] explored this concept in relation to computer graphics, by generating 3D plant structures, images, textures, and animations using artificial evolution. The evaluation for the genetic algorithm in this work was done completely manually.

Interactive genetic algorithms

To incorporate subjective preferences and creative ideas into a genetic algorithm, *interactive genetic algorithms* have been used successfully [4]. In an interactive genetic algorithm, the evaluation of the design is not calculated, but judged by a human. Cho [5] describes application of such an interactive genetic algorithm in the context of fashion design and image retrieval.

This approach has also been applied to the design of user interfaces. Quiroz et al. [18] used an interactive genetic algorithm to design a simple panel layout of user input elements.

Website design using interactive genetic algorithms

Finally, interactive genetic algorithms have also been used to create website designs. Monmarché et al. [14] built a simple version of a website design tool by applying a simple style sheet to a static website. Based on this work, Oliver, Monmarché, and Venturini [15] then applied the same concept to the website layout as well, on a grid-based system. In both of these works, they opted to let the user select a number of designs from a given set of possible options.

In this thesis, we give the user more influence over the genetic algorithm, not only by giving them more than a binary choice on whether they like a design or not, but also by letting them select their preferences on specific aspects of the website design if they choose to.

Sorn and Rimcharoen [23] took a very similar approach, in that they let the user rate a design in eight different categories with values from 1 to 5. However, they did not evaluate their approach with human feedback, but rather gave each design a deterministic rating based on its properties. Therefore, they did not account for potential inconsistencies in the ratings given to the algorithm. In contrast, we focused on the experience of the designer using our design tool, and conducted a user study to evaluate it.

3 BACKGROUND

In this chapter, we present all background information needed for understanding this thesis.

First, in Section 3.1 we explain the basics of website design in regard to this thesis. In later chapters, we model website designs as configurable systems or feature models, which we introduce in Section 3.2. Finally, in Section 3.3, we describe the fundamentals of genetic algorithms, which we use to improve our website designs algorithmically.

3.1 WEBSITE DESIGN

Websites are usually written in a markup language called Hypertext Markup Language (HTML) [26]. An HTML file defines a hierarchy of *elements*, which make up the content of the website. The syntax of HTML is very similar to XML; most elements are opened with a tag like `<html>` and are closed with the corresponding tag like `</html>`. In between these tags, the contents of the element are specified.

To design the website, these elements are then styled using a language called Cascading Style Sheets (CSS) [25]. A CSS file defines a set of *rules*. Each rule selects a number of elements and applies style properties to them. These properties include text and background colors, font family and size, but also properties that allow to change the layout of the site, e. g. positioning elements next to each other.

Figure 3.1a shows the source code for a simple website. Inside of the `<head>` element, there is invisible metadata about the website; in this case the encoding, the website title, and a reference to the CSS file responsible for styling the website. If there are no styles specified, the browser applies a simple default style sheet, as is shown in Figure 3.1b.

In contrast, all the visible elements are inside of the `<body>` element, which are then rendered by the browser depending on the currently active style sheet(s). In Figure 3.1a, there are three elements inside of the `<body>` element: a `<h1>` element for a first-level heading, and two `<div>` elements, which are used to group together two child elements each in a block. These two `<div>` elements also both have an *attribute* called `class`, which is used to be able to target the elements more easily using CSS.

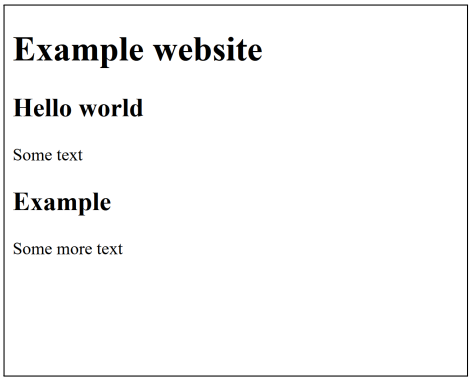
In Figure 3.1c, we show a simple CSS file with three rules. The first rule selects all `<h1>` elements and changes the font as well as the text color. The other two rules each select the elements by their `class` attribute instead, and change the color of the text and the background. In Figure 3.1d, these rules have been applied to the HTML page in Figure 3.1a.

Summing up, the HTML file defines the contents and general structure of the page, while the CSS file specifies how these contents are rendered to the screen. As such, a style sheet usually does not add any content by itself but relies on the HTML definitions for doing so. Nevertheless, by altering only the CSS file, it is possible to create a wide variety of designs for the same website.

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Example HTML website</title>
    <link rel="stylesheet" href="styles.css">
  </head>

  <body>
    <h1>Example website</h1>
    <div class="first">
      <h2>Hello world</h2>
      <p>Some text</p>
    </div>
    <div class="second">
      <h2>Example</h2>
      <p>Some more text</p>
    </div>
  </body>
</html>
```

(a) A simple HTML file



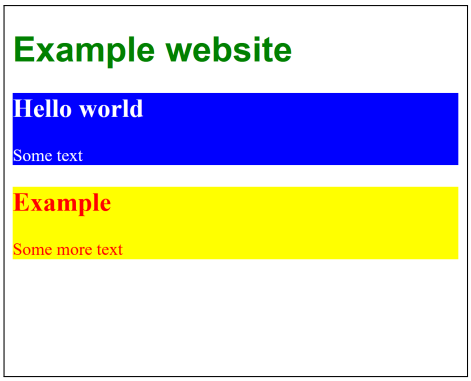
(b) The HTML file in (a) rendered with the browser's default styling

```css
h1 {
    font-family: sans-serif;
    color: green;
}

.first {
    color: white;
    background: blue;
}

.second {
    color: red;
    background: yellow;
}
```

(c) A simple CSS file



(d) The style sheet in (c) applied to the HTML file in (a)

Figure 3.1: A simple website

3.2 CONFIGURABLE SYSTEMS

A configurable software system is a software system with multiple configuration options that change certain aspects of the software [17]. These configuration options are also known as *features*. In this thesis, we consider two kinds of features: Binary features, which can be enabled or disabled, and numeric features, which have a numeric value in a specified range.

Formally, the set of features $\mathcal{F}$ is defined as $\mathcal{F} = \mathcal{F}_B \cup \mathcal{F}_N$ where $\mathcal{F}_B \cap \mathcal{F}_N = \emptyset$. $\mathcal{F}_B$ is the set of binary (or boolean) features, and $\mathcal{F}_N$ the set of numeric features.

A specific selection of features is called a *configuration*, which sets a value for each feature in the configurable system. We model configurations as a function $c : \mathcal{F} \to \mathbb{N} \cup \{\bot\}$. A binary feature f is either *enabled* in a configuration c if $c(f) = 1$, or *disabled* if $c(f) = 0$. Meanwhile, a numeric feature is enabled if it is set to a value $v \in \mathbb{N}$, i. e. $c(f) = v$. Otherwise, if it is disabled, we call it *undefined*, which we denote as $c(f) = \bot$.

The set of possible configurations for a given configurable system is the *configuration space*. Its size grows exponentially with the number of features. To restrict this configuration space to only useful configurations, we use *constraints* $\Phi_1, \Phi_2, \ldots, \Phi_n$ for excluding certain configurations. A constraint Φ_n is a propositional formula that defines which configurations are valid and which ones are not. For example, the constraint $\mathit{Alpha} \Rightarrow \mathit{Beta}$ means that the feature *Alpha* can only be enabled if the *Beta* feature is enabled as well.

Furthermore, we use a function $\mathsf{Dom} : \mathcal{F} \to 2^{\mathbb{N} \cup \{\bot\}}$ to define which values are valid for each feature. We define Dom as follows:

$$\mathsf{Dom}(f) = \begin{cases} \{0, 1\} & \text{if } f \in \mathcal{F}_B \\ \mathsf{Dom}_N(f) \cup \{\bot\} & \text{if } f \in \mathcal{F}_N \end{cases}$$

where the function $\mathsf{Dom}_N : \mathcal{F}_N \to 2^{\mathbb{N}}$ defines which values are valid for each numeric feature.

With this, we can define the set of valid configurations $\mathcal{C}$ for the configurable system. This set only includes the configurations c which assign valid values to every feature, and for which the constraints $\Phi = \Phi_1 \wedge \Phi_2 \wedge \cdots \wedge \Phi_n$ are satisfied.

$$\mathcal{C} = \{c : \mathcal{F} \to \mathbb{N} \cup \{\bot\} \mid \Phi(c) \wedge \forall f \in \mathcal{F}. c(f) \in \mathsf{Dom}(f)\}$$

As an example, consider a configurable system with two binary features *Sidebar* and *Images*, as well as two numeric features *Width* and *Height*. Also, we add a constraint saying that *Width* is defined if and only if the *Sidebar* feature is enabled. We model this configurable system as follows:

$$\begin{aligned} \mathcal{F}_B &= \{\mathit{Sidebar}, \mathit{Images}\} \\ \mathcal{F}_N &= \{\mathit{Width}, \mathit{Height}\} \\ \mathsf{Dom}_N &= \{\mathit{Width} \mapsto [1\,..\,12], \mathit{Height} \mapsto [1\,..\,8]\} \\ \Phi &\equiv \mathit{Width} \neq \bot \Leftrightarrow \mathit{Sidebar} \end{aligned}$$

One possible configuration for this configurable system would be:

$$\{\mathit{Sidebar} \mapsto 1, \mathit{Images} \mapsto 0, \mathit{Width} \mapsto 8, \mathit{Height} \mapsto 4\}$$

3.2.1 *Feature models and diagrams*

We can represent a configurable system using a *feature model* to visualize constraints. It places the features of the configurable system in a hierarchy that expresses the dependencies between them.

These feature models can be depicted in a tree-like structure using *feature diagrams*. This thesis uses a modified version of the feature diagram notation presented by Apel et al. [2].

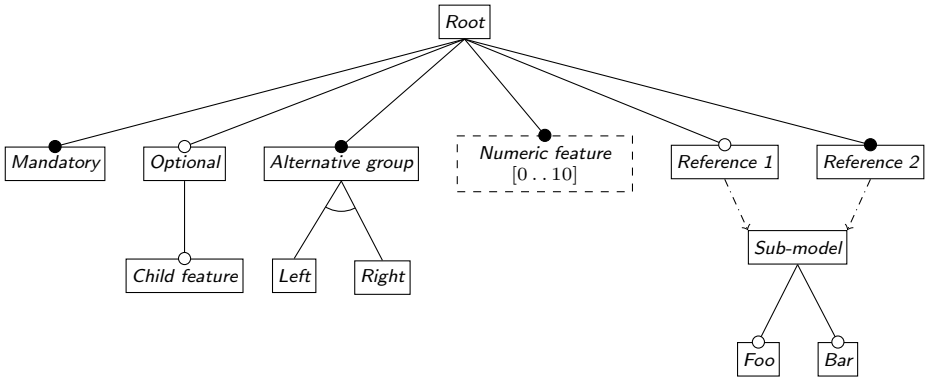


Figure 3.2: A feature diagram for showcasing feature model notation

In Figure 3.2, we showcase the different types of features that are used in this thesis. Each feature is notated as its own node, and the dependencies of the features are denoted by the edges between the nodes. This hierarchy is structured from top to bottom, meaning the features above are the *parent features* of the features below, which are their *child features*. We call features with no children *leaf features*.

For any feature to be enabled, its parent feature has to be enabled as well. For example, *Child feature* can only be enabled if *Optional* is enabled as well.

The *Root* feature has six child features. As the root of the tree structure, it is always enabled.

A feature is *mandatory* if its node in the feature diagram is marked with a filled-in circle. These features must be enabled if the parent feature is enabled as well. In contrast, a feature is *optional* if the circle is not filled in, in which case the feature can be disabled even if the parent feature is enabled. In both cases, a child feature cannot be enabled if its parent feature is disabled.

An *alternative group* is a set of child features with a common parent, whose nodes have been connected with an arc in the feature diagram. If the parent feature is enabled, exactly one of its children has to be enabled as well. Otherwise, none of the features are enabled. In Figure 3.2, either *Left* or *Right* must be enabled, but both cannot be enabled at the same time.

In this thesis, we notate numeric features like other features, but with a dashed outline and a specified range of allowed values defining $\mathrm{Dom}_N(f)$, i. e. which values are valid for the feature.

This notation defines the configuration space $\mathcal{C}$ fully, including the constraints. As an example, the feature diagram in Figure 3.2 defines these constraints:

$$\Phi_1 \equiv \mathit{Root} = 1$$
$$\Phi_2 \equiv \mathit{Mandatory} \Leftrightarrow \mathit{Root}$$
$$\Phi_3 \equiv \mathit{Optional} \Rightarrow \mathit{Root}$$
$$\Phi_4 \equiv \mathit{Child\ feature} \Rightarrow \mathit{Optional}$$
$$\Phi_5 \equiv \mathit{Alternative\ group} \Leftrightarrow \mathit{Root}$$
$$\Phi_6 \equiv \mathit{Alternative\ group} \Rightarrow \mathit{Left} \oplus \mathit{Right}$$
$$\Phi_7 \equiv \mathit{Numeric\ feature} \neq \bot \Leftrightarrow \mathit{Root}$$
$$\vdots$$

Cross-tree constraints are constraints that cannot be modeled using a strict hierarchy of features, and thus cannot be represented in a feature diagram. For simplification purposes, we only consider feature models without such constraints in this thesis.

Finally, in our feature diagrams, we use *references* as notation for inserting another feature model in place of the feature. This is used if multiple features have the same children. For example, in Figure 3.2, both *Reference 1* and *Reference 2* in effect have two child features each, namely *Foo* and *Bar*. In case of name collisions, we can disambiguate the conflicting features by using the names of their parent features, e. g. *Reference 1 Foo* or *Reference 2 Bar*.

Our notion of references is similar to and inspired by the concept of *dependent feature models* as described by Schröter et al. [21] and the feature model combination operators introduced by Acher et al. [1].

3.2.2 *Configuration notation*

We notate a configuration as a set of features that adhere to the constraints given by the feature model. A binary feature is enabled if it is a member of the set. Numeric features however have a value attached to them if they are enabled. For example, consider the following configuration for the feature model in Figure 3.2:

$$\{\mathit{Root},\ \mathit{Mandatory},\ \mathit{Alternative\ group},\ \mathit{Left},\ \mathit{Numeric\ feature} = 5,\ \mathit{Reference\ 1},\ \mathit{Reference\ 2}\}$$

This notation can quickly become unwieldy the more features there are, therefore we use a more compact notation in this thesis, in that we do not include features that are implied to be enabled. As an example, consider this configuration:

$$\{\mathit{Child\ feature},\ \mathit{Right},\ \mathit{Numeric\ feature} = 7,\ \mathit{Reference\ 1\ Foo}\}$$

From these enabled features, we can deduce that the features *Root*, *Mandatory*, *Optional*, *Alternative group*, *Reference 1* and *Reference 2* are enabled as well, either because their parent feature is enabled, or because one of their child features is.

3.2.3 *Modeling website design as a configurable system*

The visible elements of a website can be conceived as a giant configurable system. Every element has its own set of style properties which all have a number of different possible values they can be set to. Both the properties and values can be modeled as features. We can then reduce this large configuration space to a more reasonable size by only picking specific properties and values that we are interested in.

Figure 3.3 depicts a simple feature model for the website in Figure 3.1 for the styling properties and values used there.

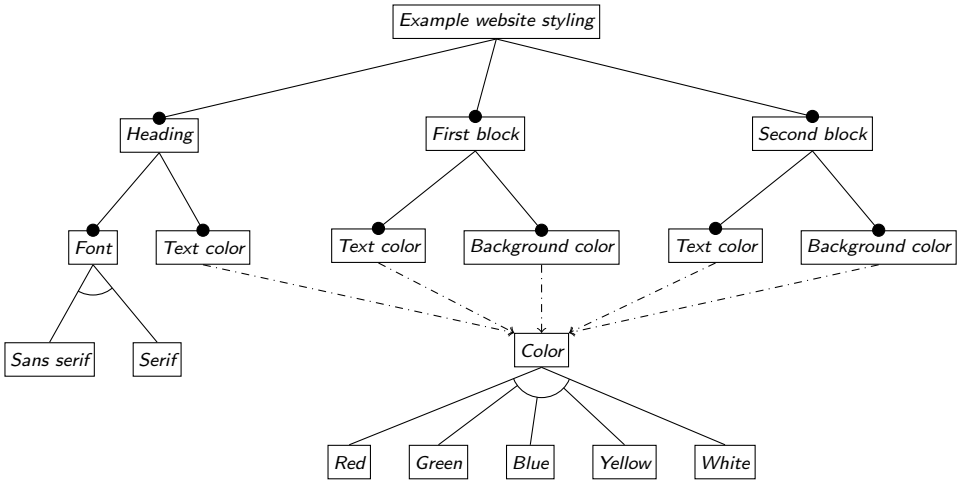


Figure 3.3: A feature model for designing the website from Figure 3.1

The specific website design configuration shown in Figure 3.1 can be expressed like this:

$$\left\{ \begin{array}{c} \textit{Heading font sans serif, Heading text color green,} \\ \textit{First block text color white, First block background color blue,} \\ \textit{Second block text color red, Second block background color yellow} \end{array} \right\}$$

Using these features, we can then generate the CSS that applies to our website. Using a CSS feature called "custom properties", also known as "variables", we can simplify the process of generating the website design style sheet based on a given configuration. Listing 3.1 shows such a style sheet, where the custom properties in lines 2 – 9 can be generated automatically based on the enabled features.

```css
:root {
    --heading-font: sans-serif;
    --heading-color: green;

    --first-color: white;
    --first-bg: blue;

    --second-color: red;
    --second-bg: yellow;
}

h1 {
    font-family: var(--heading-font);
    color: var(--heading-color);
}

.first {
    color: var(--first-color);
    background: var(--first-bg);
}

.second {
    color: var(--second-color);
    background: var(--second-bg);
}
```

Listing 3.1: A CSS file using variables for the website in Figure 3.1

3.3 GENETIC ALGORITHMS

The purpose of a genetic algorithm is to find an optimal solution to a problem. They are inspired by the concept of natural selection from biology, in that they simulate an evolutionary process.

Genetic algorithms have been first described by John Holland [10] in the 1960s. The appeal of using them is that they are able to discover innovative solutions that would be hard to find by hand. Additionally, genetic algorithms can adapt to external changes in a graceful way, for example if the criteria for the optimization process change over time [13].

In our design process, we use such a genetic algorithm, which we explain in Section 4.4.2. For the purposes of this thesis, we use a slightly simplified notion of such an algorithm, which we explain in this section.

On a high level, a genetic algorithm simulates the evolution of a *population*. With natural selection, or also "survival of the fittest", this population converges towards the global optimum over time [3].

The members of this population are called *individuals* or also *phenotypes*. In a genetic algorithm, these individuals can be generally anything, they are the subject that is to evolve over time. An individual can be encoded genetically as a *genotype*, however in this thesis we do not distinguish between phenotype and genotype for simplification purposes.

The algorithm works in multiple phases, called *generations*. Each generation captures a population in its current state, that is evolved into a new population which makes up the next generation.

The first generation's population needs to be initialized for the algorithm to start. Usually it is randomly generated to cover the search space as widely as possible.

Then, the current generation's population is evaluated using a *fitness function*. This function takes an individual as input and returns a *fitness value* as output. This fitness value evaluates the individual; usually there are fixed maximum and minimum fitness values and the higher the value is, the better.

Afterwards, a number of individuals from the current generation are selected, in a process called *selection*. This selection process is biased towards individuals with high fitness values.

The selected individuals are then used to create new individuals using a number of *genetic operators*. In this thesis, we consider the two operators of *crossover* and *mutation*. During crossover, a number of selected individuals are combined to produce a new individual; essentially, a new "child" individual is created from multiple "parents". Then, during mutation, certain aspects of individuals are sometimes changed randomly. These selected, combined and mutated individuals then make up the population of the next generation.

This process is repeated until certain termination criteraia are reached, for example, when a certain number of generations is reached or when an individual with certain properties is found. The result of the genetic algorithm then is the last generation's population.

4 IMPLEMENTATION

In this chapter, we describe our method for automating a website design process while giving the user the ability to influence it.

For our design process, we have several prerequisites. For one, we assume that we have an HTML file that we want to create a website design for.

Second, we need a feature model for designing this website. We require a way for each configuration to be translated into a CSS file that is then applied to the website. In Section 3.2.3, we showed an example of such a feature model and how we can generate a style sheet based on it.

Additionally, we also have to manually select a subset of configurations beforehand as representatives of our design space. This is needed as a starting point for our process. We explain this in more detail in Section 4.1.2.

The goal of our design process is to find a configuration that the user is satisfied with.

We showcase the website and feature model used as a test scenario for this thesis in Chapter 5. Nevertheless, the following sections explain our implementation in a general notion that can in theory be applied to any feature model. For explanation purposes, we use the small feature model in Figure 4.1 as an example in the following sections.

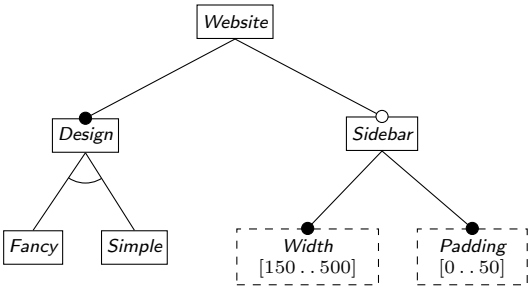


Figure 4.1: A small example feature model

The broad idea of our approach is to let the user rate a set of designs, which we describe in Section 4.2. These ratings then are saved and used as knowledge by the algorithm for approximating the user's subjective preferences. Section 4.3 describes how these ratings are stored and used to evaluate designs based on these approximated references.

This approximate evaluation is needed for an algorithm responsible for creating new random designs for the user to rate. Section 4.4 describes this algorithm and how it works.

Finally, Section 4.5 explains the full design process at large, from the user initializing the algorithm, over the design loop where the website design is iterated over, to the selection of the final result. We provide an overview of the entire design process as a diagram in Figure 4.8.

4.1 WEBSITE DESIGNS

We can represent a specific website design as a configuration of our feature model. For example, consider the following configuration for the feature model from Figure 4.1:

$$\{\mathit{Fancy},\ \mathit{Sidebar},\ \mathit{Width} = 250,\ \mathit{Padding} = 25\}$$

We can use the enabled features from this configuration to create a design as we describe in Section 3.2.3. We can use this to generate configurations which then cover a wide range of the configuration space and also, depending on the feature model, a large design space.

In the following sections, we go over a few methods for creating and modifying new designs which we rely on in later sections. These are needed for our genetic algorithm which we showcase in Section 4.4.2.

4.1.1 *Generating random designs*

From a feature model, we can easily generate random configurations by recursively going through the features of the model. We always enable mandatory features we come across and enable optional features randomly with a certain chance. For alternative groups, we randomly pick one possible feature and for numeric features, we randomly pick one value in the range.

It is possible to assign different probabilities for different features, but the simplest method is to have the same probability for all possible children of a feature.

This method of generating random designs always generates configurations which fulfill the constraints given by the feature model. However, this would not be the case if we allowed the feature model to have cross-tree constraints.

4.1.2 *Presets*

Because randomly created designs can often create unexpected results, we provide the option to generate a new design from a limited subset of the whole design space.

For this, we manually select a set of *presets*, which are specific configurations for certain parts of the feature model, which we then combine randomly.

As an example, for the feature model in Figure 4.1, we could choose the following presets for *Design*:

$$\{\mathit{Fancy}\}\,;\ \{\mathit{Simple}\}$$

and the following presets for *Sidebar*:

$$\{\mathit{Width} = 200,\ \mathit{Padding} = 20\}\,;\ \{\}$$

Combining these, we get the following four presets for the entire feature model:

$$\{\mathit{Fancy},\ \mathit{Width} = 200,\ \mathit{Padding} = 20\}$$
$$\{\mathit{Simple},\ \mathit{Width} = 200,\ \mathit{Padding} = 20\}$$
$$\{\mathit{Fancy}\}$$
$$\{\mathit{Simple}\}$$

4.1.3 *Mutation*

We additionally define a mutating operation for designs. The idea of this is to slightly modify an existing design, such that it is similar but still different. This way, our design process should also have the ability to tweak properties slightly, instead of changing them altogether.

To do this, we iterate over each feature of the original configuration, and with a certain chance decide to change it. For numeric features, we select a random value inside of a smaller range around the current value of the feature. For other kinds of features, mutation works similarly to generating a fully random design.

As an example, consider the following configuration for the feature model in Figure 4.1:

$$\{\mathit{Fancy},\ \mathit{Width} = 200,\ \mathit{Padding} = 20\}$$

This configuration has many different possible mutations. For example, since mutation is random, it might not change at all. Or, with a certain chance, the *Fancy* feature could change to a *Simple* feature. For the numeric features, the values might change slightly as well. Additionally, the *Sidebar* feature could be randomly disabled.

$$\{\mathit{Fancy}, \mathit{Width} = 200, \mathit{Padding} = 20\}$$
$$\{\mathit{Simple}, \mathit{Width} = 200, \mathit{Padding} = 25\}$$
$$\{\mathit{Fancy}, \mathit{Width} = 215, \mathit{Padding} = 18\}$$
$$\{\mathit{Fancy}\}$$

4.1.4 *Combination*

For the crossover operation of the genetic algorithm, we also provide the option to combine multiple designs of the same feature model. This is similar to mutation, but we have more than one design that we create a new design out of. For this, we iterate over their common features, and randomly decide which design's sub-features we pick.

For instance, consider the following three configurations:

$$\{\mathit{Fancy},\ \mathit{Width} = 200,\ \mathit{Padding} = 20\}$$
$$\{\mathit{Simple},\ \mathit{Width} = 300,\ \mathit{Padding} = 50\}$$
$$\{\mathit{Fancy}\}$$

These three configurations can be combined in various ways to create new configurations. The possible results of this combination operator could be any of the following configurations, based on random chance:

$$\{\mathit{Fancy},\ \mathit{Width} = 200,\ \mathit{Padding} = 20\} \quad \{\mathit{Fancy},\ \mathit{Width} = 200,\ \mathit{Padding} = 50\}$$
$$\{\mathit{Fancy},\ \mathit{Width} = 300,\ \mathit{Padding} = 20\} \quad \{\mathit{Fancy},\ \mathit{Width} = 300,\ \mathit{Padding} = 50\}$$
$$\{\mathit{Simple},\ \mathit{Width} = 200,\ \mathit{Padding} = 20\} \quad \{\mathit{Simple},\ \mathit{Width} = 200,\ \mathit{Padding} = 50\}$$
$$\{\mathit{Simple},\ \mathit{Width} = 300,\ \mathit{Padding} = 20\} \quad \{\mathit{Simple},\ \mathit{Width} = 300,\ \mathit{Padding} = 50\}$$
$$\{\mathit{Fancy}\} \qquad \{\mathit{Simple}\}$$

4.2 RATINGS

At the core of our semi-automated design process, we let the user rate different design configurations. To do so, the user selects a natural number from 0 to 250, where 0 is the worst and 250 the best possible rating value.

For the user to be able to give more nuanced ratings, we give them the ability to rate different groups of features separately. We call these feature groups *categories*. We assume that a category can be identified by a single root feature and all its members are this feature and all its child features.

For example, for the feature model in Figure 4.1, there could be two categories: One for the *Design* feature and all its child features, and one for the *Sidebar* feature and all its side features. The user then has the option to give the same design two different ratings. For instance, if they like the design but not the sidebar, they could give a high rating to the former and a low rating to the latter.

4.3 RATING STORAGES

We need a data structure for storing knowledge about the configuration space of our feature model. We call such a structure *rating storage*.

The general idea of a rating storage is to store for each feature which ratings have been given to designs with the given feature. When a user has rated a design, the rating is added to the rating storage with the context of the rated configuration.

Similar to feature models, a rating storage is a hierarchic data structure, and can have parents and children. We define four kinds of rating storages:

- *Group storages* always have at least one child storage. Their purpose is to group together multiple storages that are related to each other.
- *Alternative storages* also always have at least one child storage. They correspond to alternative groups from feature models, and as such conceptually only one child storage is relevant for each configuration.
- *Leaf storages* do not have child storages. We divide them into unit and numeric storages.
  - *Unit storages* have a multiset of ratings attached to them.
  - *Numeric storages* instead have a map attached to them. This map assigns each natural number $n \in \mathbb{N}$ a multiset of ratings. By default, this multiset is empty.

We call a leaf storage *empty* if it is a binary feature and its multiset is empty, or if it is a numeric feature and its map is empty. A group or alternative storage is empty if all its child storages are also empty.

4.3.1 *Converting a feature model into a rating storage*

The rating storage we use during the design process is closely based on the original feature model. In this section, we show how a feature model can be converted into an empty rating storage.

For this, we first have to simplify it into a form where we only have the root feature, mandatory features, and alternative groups.

First, we transform optional features. They can be replaced with a mandatory feature by adding an alternative group with two child features: a binary feature *Disabled* with no children, and *Enabled* with the children of the original feature. The result of this operation for the feature model in Figure 4.1 is shown in Figure 4.2.

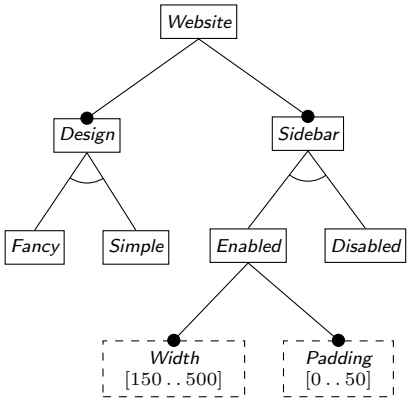


Figure 4.2: The feature model from Figure 4.1 without optional features

The resulting feature model is equivalent to the original one because the new *Enabled* feature in the new model is present if and only if its parent feature was enabled in the original model. Otherwise, the *Disabled* feature denotes the absence of the parent feature in the original model.

Then, we replace references in the feature model by the feature models they reference. After these two simplification steps are done, we can recursively convert a feature model into a rating storage, starting at the root feature.

Each feature is converted as follows:

- If the feature only has mandatory features as children, we replace it with a group storage whose children are the converted child features.
- If the children of the feature are an alternative group, we replace it with an alternative storage whose children are the converted alternative group.
- Otherwise, we have a leaf feature, for which we create a leaf storage:
  - For binary features, we create an empty unit storage.
  - For numeric features, we create an empty numeric storage.

Figure 4.3 shows the result of this conversion for our example feature model.

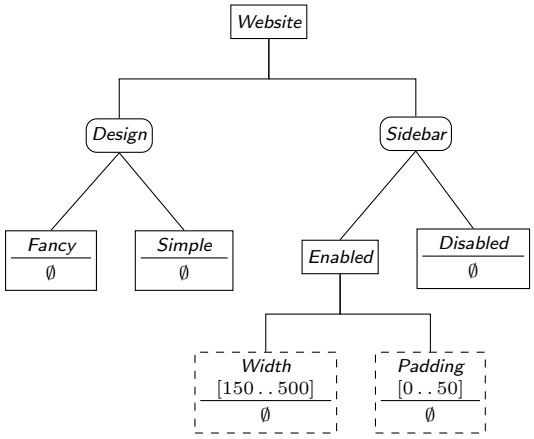


Figure 4.3: An empty rating storage for the feature model from Figure 4.2

4.3.2 *Notation*

We notate rating storages in this thesis in a similar fashion to feature models. Group storages have rectangular nodes and point to their children with straight angular lines, whereas alternative groups have rounded nodes and point to their children with diagonal lines. Unit and numeric features additionally are annotated with their multiset or map of ratings, respectively.

4.3.3 *Adding a rated configuration*

Rating storages correspond to their original feature model closely. As such, it is possible to select all rating storages that a specific configuration belongs to.

Thus, when adding a rated configuration to a rating storage, the rating value is simply added to all the leaf storages whose corresponding feature is enabled in the configuration. For unit storages, the rating value is simply added to the multiset of the storage. However, for numeric storages, it is instead inserted into the map for the value the numeric feature is set to in the configuration.

As an example, consider the following configuration for the feature model in Figure 4.2:

$$\{\mathit{Fancy},\ \mathit{Enabled},\ \mathit{Width} = 250,\ \mathit{Padding} = 25\}$$

If the user rates this configuration with a rating value of 200, adding this rated configuration to the rating storage from Figure 4.3 results in the rating storage from Figure 4.4. The enabled features of the rated configuration are highlighted in gray.

The multisets in the rating storage grow over time the more configurations and ratings are added to it. Figure 4.5 shows the rating storage from Figure 4.4 after five more rated configurations have been added to it.

When adding ratings for multiple categories as described in Section 4.2, the ratings are not added to the root rating storage, but rather to the rating storage corresponding to the root feature of each category instead.

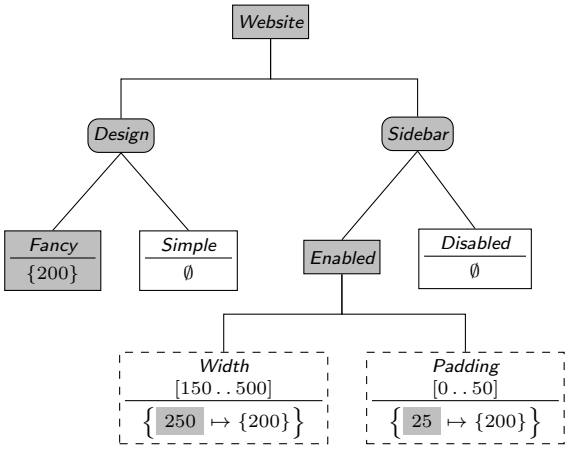


Figure 4.4: Adding a rated configuration to the rating storage from Figure 4.3

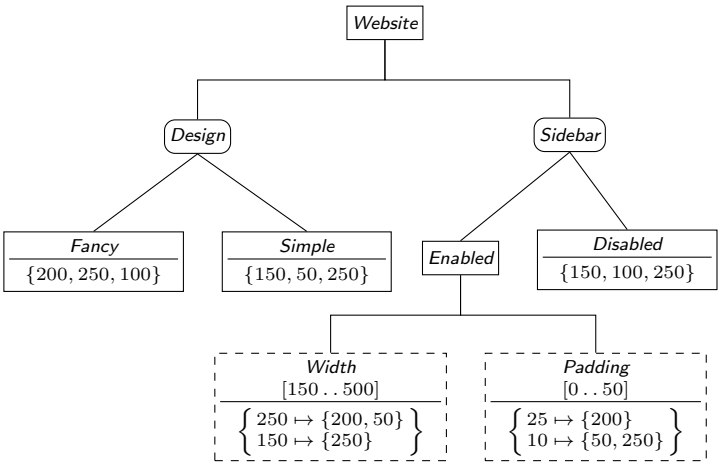


Figure 4.5: The rating storage from Figure 4.4 after five more ratings have been added to it

4.3.4 *Estimating the rating of a configuration*

Using a rating storage, the rating of a configuration can be estimated based on prior ratings by the user. This is the primary function of a rating storage: it approximates the subjective opinions of the user about a specific design that they might not have seen yet by extrapolating from the ratings that have already been stored in the rating storage.

To estimate a rating, first all leaf storages corresponding to features that are enabled in the configuration are collected, similar to adding a configuration to the rating storage.

As a next step, we ignore all empty leaf storages, since we do not yet have enough information about the specific feature to be able to rate it.

Then, we evaluate each of these leaf storages separately. For unit storages, we simply determine the average of all the ratings in the multiset. However, for numeric storages, this calculation is not as straightforward. This is because there are a lot of possible values that the feature can have, and ignoring all values that have not yet been rated is not feasible. Instead, we approximate the ratings by interpolating between the averages of each value with a fifth-degree polynomial.

This method is not optimal and can occasionally produce unexpected results, but it works well enough for roughly interpolating between the existing data points.

In Figure 4.6, we see a graph showcasing the interpolated numeric storage. The red dots represent ratings that have been given by the user, whereas the black dots represent the averages between ratings for the same value. For example, the value 48 had three different ratings: 200, 150 and 50. The average of these three values is $133.\overline{3}$, as indicated by the black dot. The blue line shows the fifth-degree polynomial used to determine the estimated ratings for each value. For the value 48, this estimated rating is 117.

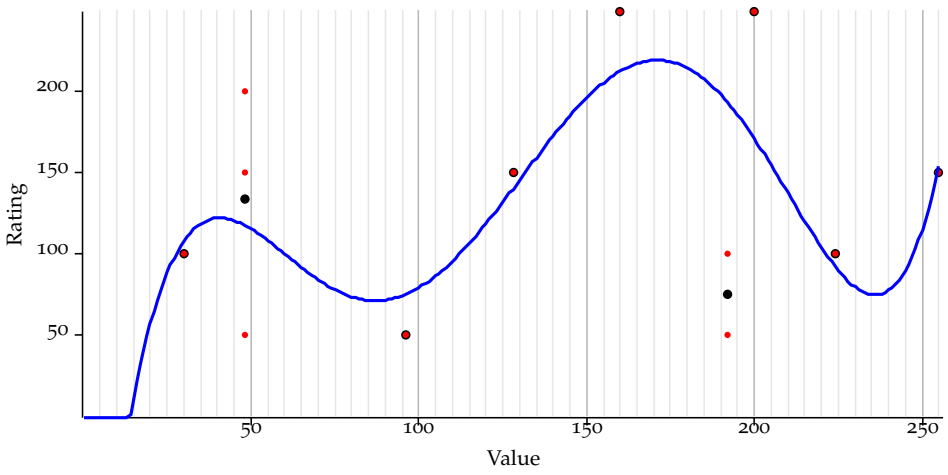


Figure 4.6: Interpolation between different ratings in a numeric rating storage

Finally, after having evaluated each leaf storage individually, we determine the average of all these results as the final estimated rating for the configuration.

4.3.5 *Determining the best configuration*

For the final product of our design process, it is important to retrieve the best possible configuration from a rating storage. This is not necessarily the configuration with the best estimated rating, but rather the configuration with the best average ratings by the user.

To find this configuration, we first calculate the best average ratings for each leaf storage. For unit storages, we simply calculate the average of the multiset. For numeric storages however, we calculate the average for each multiset in the map, and pick the highest.

We then propagate these average ratings up to the group and alternative storages. For group storages, we calculate the average of all child storages' averages, and for alternative storages, we only pick the best average of all child storages.

Afterwards, we can pick features based on these average ratings. Whereas the feature to be picked for unit storages is trivially the feature it corresponds to, for numeric storages, we pick the value with the highest average.

As the best features for a group storage, we pick the best feature configuration for each child storage. For alternative storages, we only pick the sub-storage with the highest average rating.

As an example, consider the rating storage from Figure 4.5. In Figure 4.7, all storages are annotated with their average best rating in angle brackets, and the picked features are highlighted in gray.

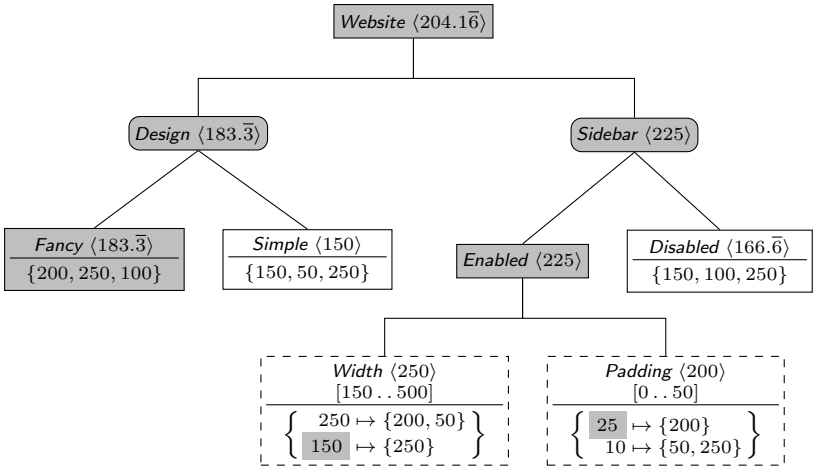


Figure 4.7: Determining the best configuration from the storage in Figure 4.5

During this process of determining the best configuration, there may be ties between two different features. In practice, these ties only occur rarely. In our implementation, we resolve them arbitrarily but deterministically, i. e. in case of a tie between two features, we always pick a particular predetermined feature.

4.3.6 *Generating configurations for empty rating storages*

Since empty rating storages are not very useful for estimating ratings, and we want to cover as much of the configuration space as possible using the rating storage, we want to fill a rating storage as evenly as possible. To do so, we provide a way to generate a random design with a particular bias towards features that have not been rated yet.

This process is similar to finding out the best design, however the criteria for picking features is different: instead of determining which feature to pick based on the average best rating, we pick the feature with the least amount of ratings. For unit storages, this is simply the cardinality of the multiset; for numeric storages, it is the cardinality of the smallest multiset in the map. In case of a tie, we pick a random feature to select out of the tied features.

4.4 THE ALGORITHM

The purpose of the rating storage is for the design tool to be able to evolve the designs automatically in a way that corresponds to the prior ratings given by the user. This section explains the algorithm that is responsible for doing this.

As an input, the algorithm receives a number of rated styles, and it returns three new designs, as well as the currently best design.

The algorithm has an internal state, which in our case consists of a rating storage for our feature model, and a list of configurations from the previous iteration of the algorithm.

First, the rated configurations are added to the rating storage. Then, from these configurations and the designs from the previous iteration of the algorithm, we pick a set of configurations as an initial population, the composition of which is described in Section 4.4.1. We then evolve this initial population of designs using a genetic algorithm, as explained in Section 4.4.2. From the results of this evolution, we then select three designs to return, as shown in Section 4.4.3.

4.4.1 *Initial population*

The initial population is the set of designs that is evolved using the genetic algorithm. For this purpose, we select 100 different configurations from a few different sources.

For one, we add all configurations to the initial population that have been given a good rating. We consider a rating to be good if the average of the category ratings is more than 125, i. e. half of the maximum rating of 250.

Then, we add the following groups of configurations, the size of which is randomly chosen:

- 20 to 40 configurations from the pool of designs resulting from the previous iteration of the algorithm
- 5 to 10 configurations with few given ratings, as described in Section 4.3.6
- 5 to 25 completely random configurations, as shown in Section 4.1.1

The remaining population consists of random presets, as explained in Section 4.1.2.

This composition of the initial population has been chosen arbitrarily, and is most likely not optimal. Nevertheless, it results in a wide range of designs, which then are further modified in the next step.

4.4.2 *Design evolution*

To evolve the designs, we use a genetic algorithm as introduced in Section 3.3. Its initial population is selected as described in Section 4.4.1.

As a fitness function, we use the design rating estimation of our rating storage as described in Section 4.3.4. Based on this estimation, we then select 80 quadruples of parents, randomly weighted by fitness. Then, we combine each quadruple into a new configuration, as described in Section 4.1.4.

Afterwards, all of these 80 combined configurations are mutated as shown in Section 4.1.3, and finally replace 80 random individuals from the previous population.

After eight generations, we terminate the genetic algorithm and continue with the next step of selecting configurations form the final generation's population, which is explained in Section 4.4.3.

4.4.3 *Results selection*

The algorithm returns a list of 100 designs, along with their current fitness value (estimated rating). From this list we pick three different configurations:

- The configuration with the best estimated rating
- A random configuration, weighted by estimated rating
- A random configuration, unweighted

4.5 DESIGN PROCESS

This section describes the entire design process from beginning to end. Figure 4.8 contains a diagram showcasing it in full.

The two actors in this design process are a human and the algorithm. The inner workings of the latter were explained in Section 4.4; in this section we instead focus on the bigger picture and how human and algorithm interact.

4.5.1 *Initialization*

To initialize the design process, first an empty rating storage for our feature model is created. This rating storage is used and filled by the algorithm throughout the entire process.

Then, we generate nine random presets, which are shown to the user. The user can then inspect and rate these designs as explained in Section 4.2, according to their subjective impressions and opinions.

The rated designs are then returned to the algorithm, where the ratings are added to the rating storage and the designs are evolved, as explained in Section 4.4.

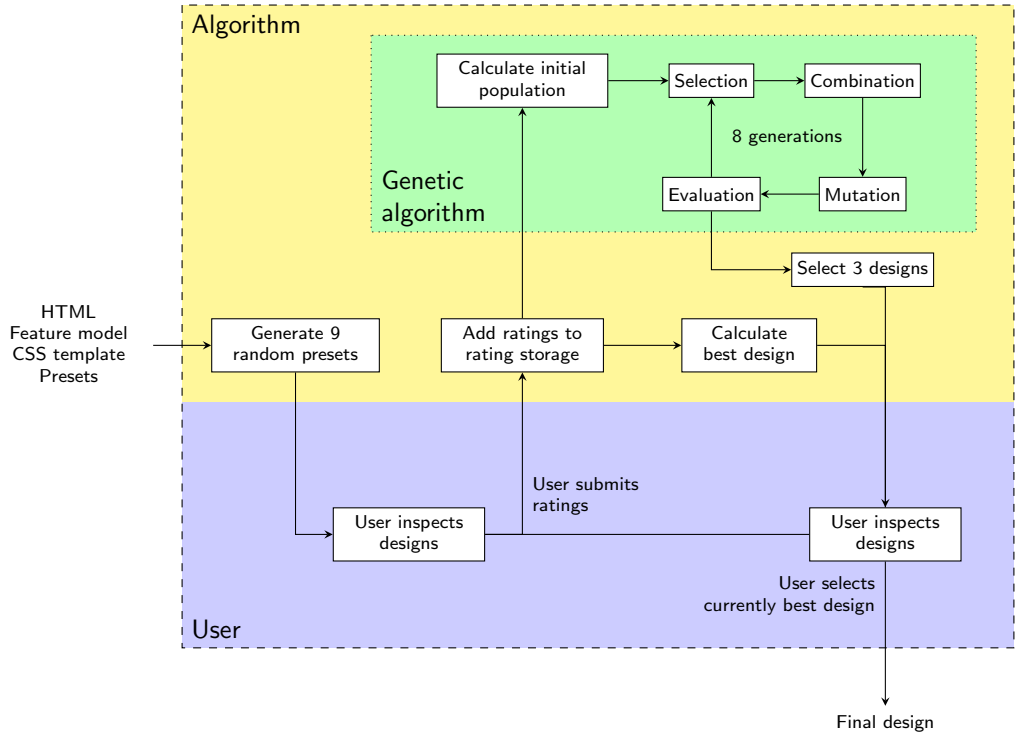


Figure 4.8: An overview of the entire design process

4.5.2 *The loop*

After this initialization phase, the iterative design process begins. We call it "the loop". This section explains how one iteration of this design loop works.

From the previous execution of the algorithm, we get three random designs for the user to rate, as well as the currently best design according to the rating storage as explained in Section 4.3.5.

Both the best design and the three random designs are now shown to the user, who then can decide whether they would like to improve on the currently shown best design by continuing to rate more designs.

If they do, they can inspect the three random designs shown to them, and give ratings to them, according to their personal opinions. These ratings are then again returned to the algorithm, which adds the ratings to the rating storage and returns new designs.

At this point, the cycle repeats and the user is shown a new best design and three new random designs, and can decide again whether they would like to continue rating or if they are satisfied with the current design.

4.5.3 *Finalization*

If the user decides that they like the currently best design shown to them, the design process ends. In this case, the final result of the design process is the currently bast design that was just shown to the user.

5 TEST SCENARIO

To test our implementation, we have created an example website, which we style using a website design framework that we manually built specifically for it. This section showcases this test scenario in its entirety.

Figure 5.1 shows the full website with a basic default design. The website depends on multiple text blocks, and has a navigation bar at the top, as well as a sidebar on the left with a table of contents and an embedded Twitter feed.

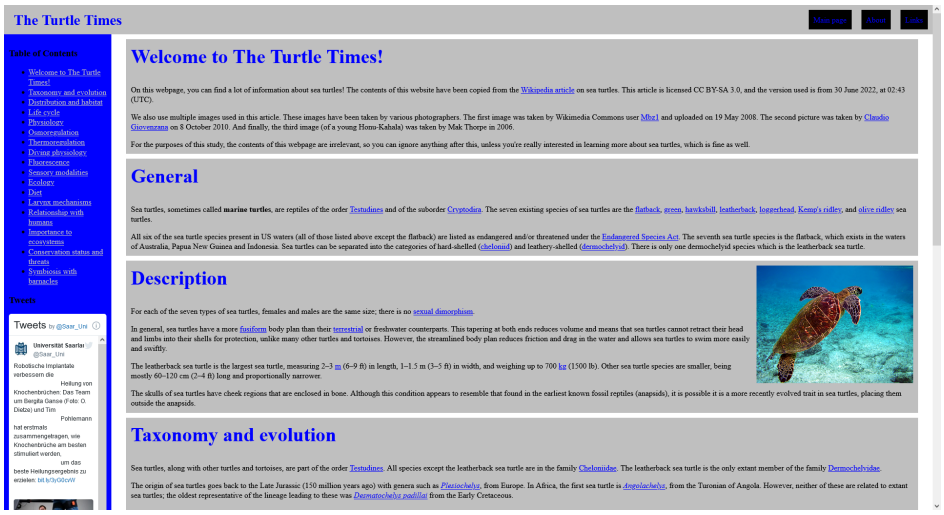


Figure 5.1: Our website test scenario with a basic default design

Figure 5.2 shows the feature model for our website design framework. To make it more manageable, it is split up into four categories: Color scheme, fonts, alignment, and layout. Each of these categories has its own feature model and is explained in the following sections.

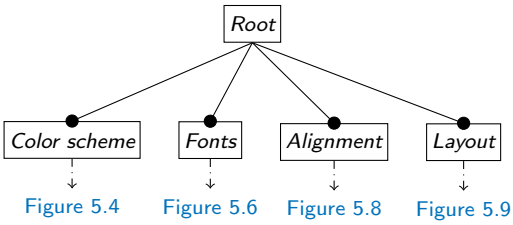


Figure 5.2: The root feature model for our website design framework, with four sub-categories

As explained in Section 4.2, with our approach we give the user the ability to give nuanced ratings in multiple categories. In our test scenario, all four categories can be rated separately or together, depending on the preferences of the user.

Figure 5.3 shows the rating interface of our design tool. The top slider can be used to move all the sliders below if they have not been set specifically, but by itself, it has no effect.

In this case, it is set to 180. The "Color scheme" slider has been set to a rating value of 35, and the "Alignment" slider to the best possible value of 250. By clicking on the revert arrows on the right of these two sliders, it is also possible to reset these sliders to instead be controlled by the large slider up above. The other two sliders are both currently controlled by the large slider at the top and have a value of 180 as well.

Figure 5.3: The interface for rating a website design

Additionally, for each category, we have curated a set of presets. The presets for the entire website then are created by combining all presets of the categories as previously described in Section 4.1.2.

We include a full list of the presets in the appendix (Section A.1). Most presets were selected arbitrarily and based on our personal preferences. For the color scheme presets, we manually collected a selection of color schemes from various websites instead [6, 7].

5.1 COLOR SCHEME

The color scheme determines the colors used by all elements throughout the website. Figure 5.4 shows the feature model for this category.

We only allow for four different colors in our color scheme model. This limited selection of colors is intended to keep the website simple, and make it less likely for too many different colors to conflict with each other. However, occasionally this limitation can cause contrast issues, especially in the navigation bar.

We model colors as three separate numeric features with a range from 0 to 255; one each for red, green, and blue. This is a common way for specifying digital colors. This makes the colors from the configurations easy to transform into CSS values, since they are also based on this RGB color space.

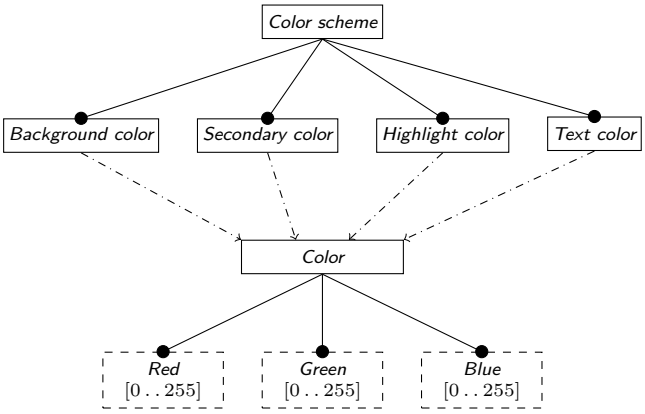


Figure 5.4: The feature model for the website color scheme

In the color scheme feature model, we distinguish between four different colors:

- The *background color* is used for the background of the entire website, as well as for the hover effect of the links in the sidebar.
- The *secondary color* is used for the background color of the article blocks and the navigation bar, as well as the color for the links in the sidebar.
- The *highlight color* is used for the headings and links inside of the article blocks, as well as the background color of the sidebar.
- The *text color* is used for the text inside of the article blocks. Additionally, it is used as the background color of the buttons in the navigation bar and the text color in the sidebar.

Table A.1 lists the color scheme presets. Notably, some of these presets have identical background and secondary colors. As an example, Figure 5.5 shows the website with the color scheme preset "Lavender", which is set to use the following colors:

Background			Secondary			Highlight			Text		
#c0a9bd			#f4f2f3			#94a7ae			#64766a		
R	G	B	R	G	B	R	G	B	R	G	B
192	169	189	244	242	243	148	167	174	100	118	106

5.2 FONTS

The fonts category determines the font family and font size of text on the website. Figure 5.6 displays the feature model for this category.

Font styling is split up into three child features: heading, text, and interface font. The heading font is only used inside of the headings of the articles, the text font used in the rest of the articles, and the navigation font everywhere else.

Figure 5.5: The website with the color scheme preset "Lavender"

The available font families in our website design framework are Arial, Calibri, Consolas, and Times New Roman. These four fonts cover sans-serif, serif, and monospace fonts.

The font size is given by the numeric feature in pixels, and the line spacing is the space between lines. Technically, the line spacing is responsible for the line height, which is calculated by adding font size and line spacing. It would be possible to instead use line height directly as a feature instead of line spacing, however that would require a cross-tree constraint between font size and line height, since the line height cannot be smaller than the font size.

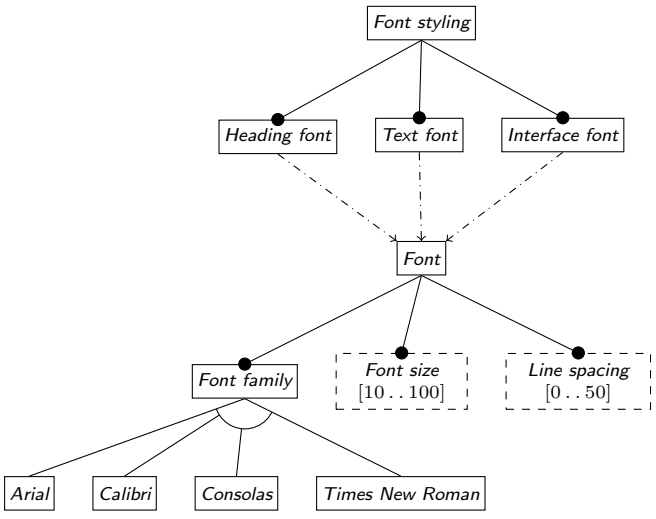


Figure 5.6: The feature model for the fonts used in the website

There are five presets for font styling, which are shown in Table A.2. These presets only include differences between the different font families. Font size and line spacing are the same for all of them: a font size of 16 and a line spacing of 4 pixels for text and navigation fonts, and a font size of 40 and a line spacing of 10 for heading fonts.

Figure 5.7 showcases the font preset "Mixed", which features Calibri for headings, Times New Roman for text, and Consolas for navigation.

Figure 5.7: The website with the font preset "Mixed"

5.3 ALIGNMENT

Figure 5.8 depicts the feature model for the alignment category. It describes how the article blocks look: it decides the alignment of the headings and the text, the location of the images, and the inner margins inside of the blocks.

There are four presets for alignment, which we show in Table A.3.

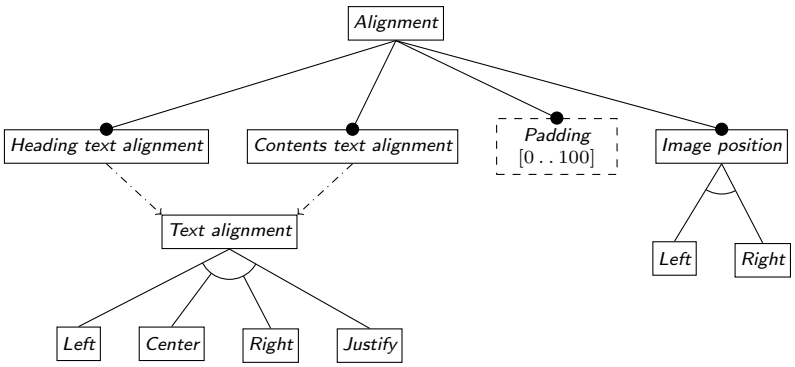


Figure 5.8: The feature model for alignment

5.4 LAYOUT

The final category describes the general layout of the website. We show the feature model for it in Figure 5.9.

There are three main areas where the layout feature model takes effect: The navigation bar, the sidebar, and the article content area. In the latter, the only property of the design is the width of the article blocks; this is expressed as a horizontal outer margin of the entire article block column: the higher the value is, the more distance the articles have to the edges of the content area.

There are five layout presets, which we show in Table A.4.

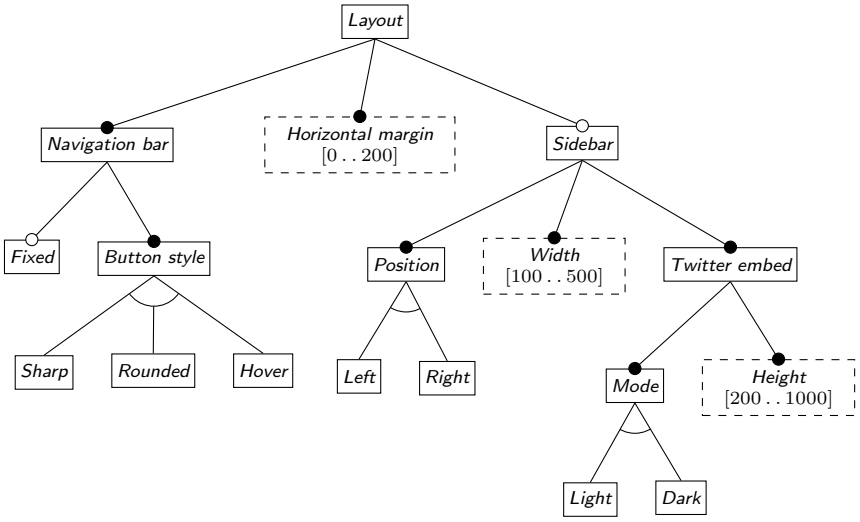


Figure 5.9: The feature model for the website layout

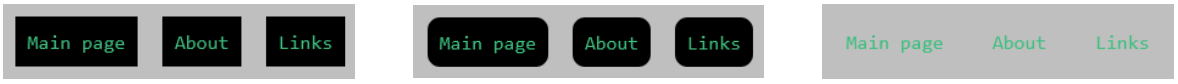
5.4.1 *Navigation bar*

The navigation bar is always at the top of the screen and has two elements: The title of the website, and three navigation buttons linking to other parts of the page. In Figure 5.10, we show an example of a navigation bar.

It has two properties, the first of which is whether it is fixed or not. A fixed navigation bar always stays at the top of the screen, even when the user scrolls down to read more.

Figure 5.10: The navigation bar at the top of the website

The second property is about how the buttons look. There are three types of button styles, shown in Figure 5.11. The *Sharp* and *Rounded* button styles are similar in that they use the text color as the background color, and their difference lies only in their rounded or non-rounded corners. Meanwhile, the *Hover* button style does not have any background by default, it only appears when the user moves their cursor over the button.



(a) Sharp (b) Rounded (c) Hover

Figure 5.11: The three different button styles

5.4.2 *Sidebar*

Finally, there can be a sidebar. If it is enabled, it can either be on the left or the right of the website. It contains a table of contents with a list of links to jump to different sections on the website, and an embed of a Twitter feed.

The width of the sidebar is variable, and so is the height of the Twitter embed. Additionally, the embed can also be either in light or dark mode.

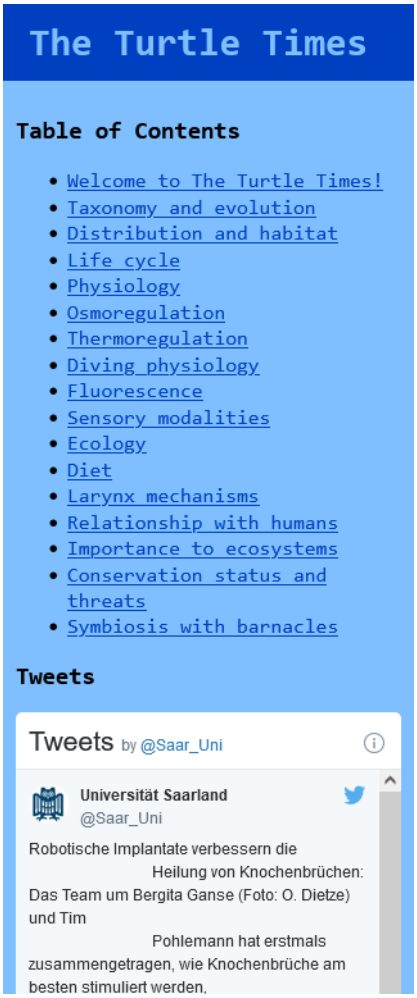


Figure 5.12: The sidebar to the side of the website

6 EVALUATION

In this chapter, we describe how we evaluated our design process and the tool we developed for it. For this, we posed research questions to investigate the relation between the user and the algorithm in our design process. These research questions are explained in Section 6.1.

We then describe the structure of our user study that we conducted in Section 6.2, and its results in Section 6.3, where we summarize the interviews that we conducted and try to answer our research questions based on the findings from these interviews. Finally, we close off the chapter with Section 6.4, where we discuss potential threats to the validity of our evaluation.

6.1 RESEARCH QUESTIONS

With our design process, we aimed to create a tool that can generate good website designs based on the user's subjective preferences. To determine to which extent this goal was met, we want to examine the relationship between the human user and the algorithm. We therefore pose the following research questions.

RQ 1: Does the human feel integrated into the rating-based automated design process?

With this research question, we investigate whether our design process successfully incorporates the human user. In that context, another interesting question is what role the user plays in the design process.

Optimally, the user is well incorporated. In this case, the algorithm is able to pick up the user's preferences and develop them further. As such, the user is also able to incorporate their own creative ideas into the algorithm and take on a more active role in the design process, directing it themselves. On the other hand, if the user is not well incorporated, they might take a more passive role and follow the lead of the designs generated by the algorithm without being able to integrate their creative decisions.

We further divide this rather broad question into two more specific questions *RQ 1.1* and *RQ 1.2*, which we want to answer additionally to this more general question.

RQ 1.1: Does the user feel like the rating-based system adapts to their wishes?

With this research question, we want to find out how well the user's subjective preferences are understood by the algorithm, and whether there are any results the user does not expect.

For one, we want to see how the rating system reacts to the user's ratings and whether the new designs shown to the user lead in the right direction, in their opinion.

Additionally, we want to determine to what extent our rating-based approach results in a good approximation for what the user wants. If it does, it could be a promising solution for extracting subjective criteria for an optimization problem.

RQ 1.2: Does the user feel like the design process hampers or boosts creativity?

With our design process, we want to provide a method for the user to explore as much of the design space as possible. With this research question we want to find out how much the design space is explored, as perceived by the user.

For instance, the generated designs could be limiting their decisions and not explore parts of the design space the user would have wanted to explore. On the other hand, it could point the user into a new direction that they had not considered otherwise and give new ideas. Furthermore, we want to examine whether the amount of detail that the user can express with only their ratings suffices for the user to bring their own creative ideas into the design process, and not only rely on the algorithm to generate interesting designs.

RQ 2: Are the design decisions made by the algorithm comprehensible to the user?

With this question, we want to find out how approachable and usable our design tool is for the user.

If the user understands intuitively how the algorithm works, they could be able to employ a strategy for rating designs more effectively to lead to quicker results. Additionally, an easy to understand algorithm would likely lead to a higher acceptance and better identification with the final design.

On the other hand, if the algorithm is not easy to understand, it functions more as a "black box", where the user does not fully understand how their result was created. Also, the usage of the algorithm might be less effective if ratings are given without a deeper understanding of how the design process functions.

6.2 STUDY DESIGN

For the evaluation of our design process, we conducted a study, which was structured as follows. We showcase the general structure of the study in Figure 6.2.

First, we would briefly show our test scenario website and the design framework to the user. Afterwards, we would introduce the concept of ratings and the four different categories that can be rated, as well as the structure of the design process. Following this introduction and throughout the rest of the study, the participant had the opportunity to ask questions about the design tool or the study.

Then, the participant would get to design a website using our process until they were either satisfied or a time limit of roughly 20 minutes was reached. During this, the participant would think aloud about what they were doing on screen. For instance, they would explain why they rate a design the way they do, and what they like and do not like about it.

Finally, as a conclusion, we conducted a semi-structured interview, with a number of prepared questions which have been listed in Figure 6.1. They aim to answer the research questions posed in Section 6.1, which are fairly broad and cannot be answered by themselves easily. From the answers for the interview questions, we can then highlight specific aspects of the respective research questions. We also additionally asked two questions not related to the research questions to capture their general thoughts on the design process and to close off the interview with an open discussion about the design tool as a whole.

1. *Quality of the final design*
   a) On a scale from 1 to 10, how satisfied are you with the design? (RQ 1.1)
   b) Is there anything you would like to change about it? If so, what? (RQ 1.1)
2. *Feeling of the process*
   a) Do you think the design process was fast or slow? (RQ 1)
   b) How mentally exhausting was the design process using the tool? (RQ 1)
   c) What role do you think you played in the design process? (RQ 1)
   d) Are you directing the design process or does the tool "lead" you? (RQ 1)
3. *Creativity*
   a) How diverse were the designs that the tool showed to you? (RQ 1.2)
   b) Were there any designs that stood out to you? (RQ 1.2)
   c) Did the tool hinder you from doing something? If yes, what was it? (RQ 1.2)
   d) Do you think you could have created a better design in the same amount of time, without the tool? (RQ 1.2)
4. *Rating system*
   a) How big do you think was the impact of a single rating? (RQ 1.1, RQ 2)
   b) How well do you think the tool reacted to your ratings? (RQ 1.1, RQ 2)
   c) Do you think the designs shown to you made sense, considering which ratings you gave? (RQ 2)
   d) Why do you think the process arrived exactly at this final design? (RQ 2)
5. *Applicability*
   a) Would you use this tool for designing websites? Why or why not?
   b) Where do you see potential for this approach, and what are the limitations?

Figure 6.1: The interview questions for the semi-structured interview at the end of the study

We ran the study with five participants with varying degrees of experience in website design. Three participants reported to have some degree of prior experience in website design while the other two did not.

After running a pilot study, we decided to limit the time for the participant to use the tool to roughly 20 minutes, after which we would then cancel the design process and use the currently best design as their final design.

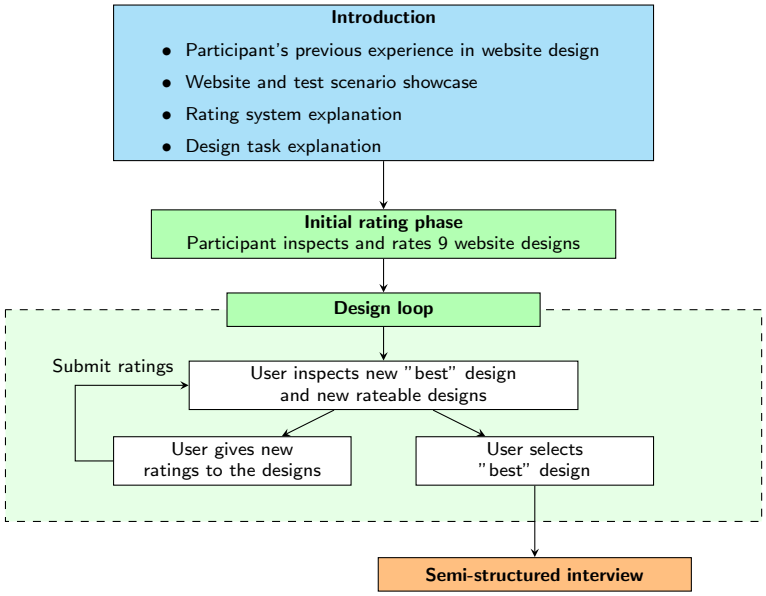


Figure 6.2: The general structure of our user study

6.3 RESULTS

In this chapter, we summarize the participants' running of the design process, and their responses to the interview questions at the end of the study. A full overview of these responses is included in the appendix (Section A.2).

In Figure 6.3, we show the final designs of the five participants. We include how long their design process took and how often they submitted a new set of ratings (i. e., the number of iterations). Additionally, the participants gave their design a score in interview question 1a, which we also show along with each participant's design.

To answer our research questions, we now take a look at the responses from the study participants to our interview questions. In the following section, we annotate each answered interview question with a reference to the question in Figure 6.1.

RQ 1: Does the human feel integrated into the rating-based automated design process?

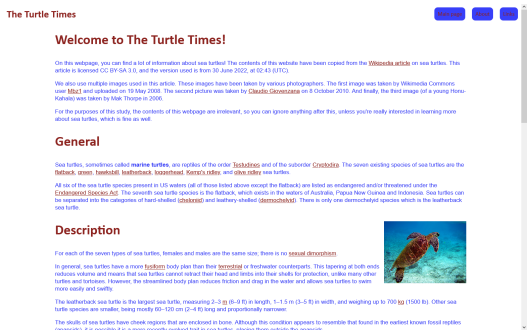
To answer this question, we asked the study participants about the perceived speed and exhaustion of using the design tool, as well as which role they think they played during the design process.

All but one participant considered the design process as a whole to be quick [2a]. However, two participants also noted that the end result is not a finished product but rather can act as a demo or inspiration. The participant who did not consider the process to be fast noted that they got stuck because the algorithm did not go in the right direction, which caused the design process to take longer.

Only one participant found the design process exhausting [2b]. They mentioned that it was difficult to decide which ratings to give to get what they wanted, if none of the designs shown on screen appealed to them. They explained that they would need to keep track of the prior ratings they gave to continue rating new designs, noting "I would not like to do

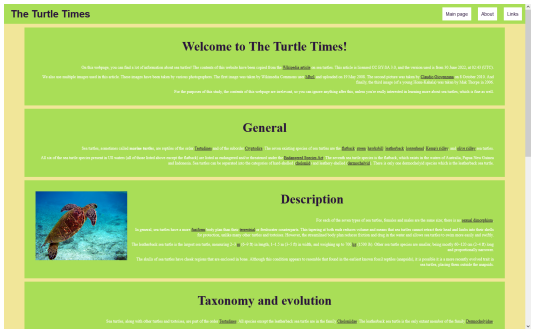
Participant A

Time	24 min
Iterations	3
Score	9/10



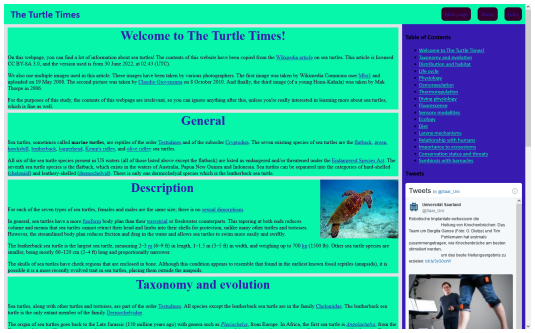
Participant B

Time	23 min
Iterations	7
Score	5/10



Participant C

Time	15 min
Iterations	5
Score	7/10



Participant D

Time	18 min
Iterations	4
Score	7-8/10

Participant E

Time	17 min
Iterations	4
Score	8/10

Figure 6.3: Overview over the participants' designs

that on a daily basis". The other participants stated that in their opinion, it did not take much effort to rate the designs.

All participants felt like they played a more passive role in the design process [2c, 2d]. One participant remarked that they could not really influence the algorithm in a way they would have liked. Another participant said that they felt like the designing was done by the algorithm, and they had no active part in it.

As such, the user of our design tool does not fully feel integrated into the design process. They act more in a passive capacity following the lead of the algorithm, and not like a designer taking creative decisions themselves. Nevertheless, the design process does not take big effort to use and leads to results relatively quickly.

RQ 1.1: Does the user feel like the rating-based system adapts to their wishes?

Figure 6.3 showcases the participants' final designs, as well as the score given to them at the end of the interviews [1a]. The average score is 7.3/10, but nobody was entirely happy with the design.

One participant rated their design with a score of 9/10 and only would have liked to tweak the color scheme slightly. While this was also the case for all the other participants, they also wanted to make other changes as well [1b]. The person who gave their design the lowest score of 5/10 noted that they would have preferred a design that was shown earlier over the final result.

During the rating process, the participants' priorities were diverse. While all participants paid a lot of attention to the color scheme, another major factor was the existence of the sidebar. One participant also wanted the navigation bar to be fixed to the top when scrolling, something that is not immediately visible when inspecting the design, and they paid close attention to it when rating their designs. This fixed navigation bar was then also present in their final design.

The impact of giving a single rating was high according the the participants [4a], however they also noted that they felt like the impact was not as large in the beginning and giving a single rating was more impactful later. Additionally, three participants noted that giving extreme ratings felt especially impactful, whereas moderate ratings did not appear to have as big of an impact.

Sometimes however, the design process produced unexpected results [4b]. One participant explained that they didn't feel like the algorithm reacted well to their ratings since multiple aspects of designs that they liked were mixed, resulting in a worse design. They especially did not get a color scheme in a direction they liked after the initial rating phase, and did not know how to correct the algorithm and steer it into the right direction.

Another participant said that the algorithm reacted well to the given ratings, especially if the rating sliders were put especially high. The other three participants were unsure and noted that some designs would surprise them while others matched their expectations.

On the whole, the algorithm is capable to produce designs the user is satisfied with, to a certain extent. However, the design process did not act as a tool for designing websites by itself, but rather as a tool for exploring ideas in regard to website design. Additionally, the users did not feel like nuanced ratings affected the algorithm to the same extent as extreme

ratings did. As such, the ability of the design process to adapt to the user's wishes varies greatly depending on the situation.

RQ 1.2: Does the user feel like the design process hampers or boosts creativity?

The participants felt like the designs that were shown to them were diverse and different from each other [3a]. However, two participants also noted that while the color scheme made a large difference, the layout was very similar for most designs. Another participant stated that the first nine initial designs were different but became less diverse over time, while yet another participant thought the first nine designs were quite similar to each other.

One participant remembered multiple stand-out designs during the design process that they liked certain aspects of [3b]. The other participants could not think of any designs they saw and especially liked. However, two of them remembered designs that they did not like because of the color scheme.

Two participants remarked that they missed some features during the design process [3c]. For one, they wanted to be able to tweak some little things about the designs while rating them. Another suggestion was that they would like to keep certain parts of the design and prevent it from being changed, or being able to merge the existing design with the new design resulting from the given ratings. Additionally, three participants voiced their displeasure with the fact that the centered text alignment for the text block headings was offset for the text block with the turtle picture, which was something that they were unable to fix using the design tool.

This shows that there were issues with the test scenario. For one, there were some minor issues that could not be fixed by the user of the tool, for instance the colors in the top navigation bar were often causing the text to be unreadable, or the fact that the centered text alignment did not play well with the pictures. Also, the layout itself did not differ much across different designs, so the largest difference according to the participants was the color scheme, and sometimes also the fonts.

All but one participant did not think that they would be able to create a better design manually in a shorter amount of time [3d]. One of them said that doing it manually would be faster if they already knew before how the website was supposed to look, however during the rating process they often changed their mind on what they wanted.

As such, by limiting the user to giving only ratings, they are unable to actively bring in their own ideas if they find a design that they like but would like to change one small aspect of it. Instead, they would need to risk their current design by giving more ratings, but the shown designs often did not feature the aspects they would like to change.

Therefore, on one hand our design process allows for a coarse easy exploration of the design space by the user with the variety of designs shown. However on the other hand, it does not allow the user to incorporate their own creative ideas directly into the design or adjust small aspects of it.

RQ 2: Are the design decisions made by the algorithm comprehensible to the user?

Three participants said that the designs shown to them during the design process made sense [4c]. One participant said that they recognized familiar aspects of designs they have seen before in new designs shown to them later.

Three participants could not find an explanation for why they arrived at their final design [4d]. One participant however said that they found parts that they liked and rated well previously in their final design. A different participant said that they reached the final design by picking parts of the design that they liked in the initial rating phase, and then based their subsequent ratings on those to get the final design.

Three participants noticed that giving extreme ratings influences the algorithm more than giving medium ratings [4a, 4b]. One participant used this to their advantage to get the design aspects they wanted.

It appears that for the users, in its current form the way the algorithm functions is not immediately obvious. It functions more as a "black box" that the user puts ratings in and gets new designs out. Also, the fact that moderate ratings appeared to have had a smaller, but extreme ratings a larger impact poses the question if the ability to give these moderate ratings at all is necessary.

As one participant put it, one needs to know how the algorithm works internally to be able to use it effectively. While it's possible to learn how the algorithm works by using the tool a lot, it did not succeed in making the automatic design decisions easy to understand.

Further Discussion

In addition to our research questions, we were also interested in our participants' opinions about the tool. We asked them about whether they would consider using it or a similar tool for designing a website, and what they think of its applicability in general.

Four participants said that they could imagine using a similar design tool for designing a website, especially for inspiration or ideas for creating a website, or as a prototype or template [5a]. One said that they were unsure, stating that usually a website is very special and personal, but for inspiration the design tool would be very good. Another participant said that for real designers, especially perfectionists, using the tool would be less fun, but that it might be a good helper for non-designers. In a similar fashon, a different participant said that they wished that there were more possibilities and more fine-grained control.

We also asked the participants where they saw potential and where they saw limitations with our approach [5b]. One participant saw potential in this approach for developers who aren't skilled in design themselves, while another noted that a design tool like this would be useful in research, where often the expertise or time is not there to create a good website design, but they would still profit from one. Conversely, two participants mentioned that the tool had the issue that it provided a very limited amount of design options. Another participant also said that designs created by machines may have a bad image.

Furthermore, during the design task, we had some interesting observations. For one, all participants of our study usually used all four sliders to rate their designs, instead of giving all four categories the same rating. Also, often the "best design" shown to the participant

would only change very marginally or not at all, which was sometimes confusing and frustrating for the participants.

Summary

In conclusion, the design process appears to work as a tool to explore the design space, but it does not result in a design that the user wants to use without any changes. Most participants would have liked the ability to tweak the shown designs manually to improve or fix small aspects of the design and to incorporate their own ideas.

As such, it is not possible for a designer to fulfill their creative vision using our design tool. They act more as a passive judge selecting between the designs presented by the algorithm.

Additionally, it is not easy to understand how the algorithm works internally. However, by strategically giving extreme ratings and focusing on specific properties of the design, one can still influence the algorithm a lot.

Nevertheless, our design process is a suitable tool for exploring the design space in a coarse manner. It is relatively easy to use, and produces results relatively quickly. These results could then act as a starting point for further manual tweaks and improvements.

6.4 THREATS TO VALIDITY

In this section, we discuss the threats to the internal and external validity of our evaluation. While internal factors are potential issues with our implementation and evaluation, external factors threaten the generalizability of our findings.

Internal validity

After the study was conducted, a few issues in the design issues were discovered. For one, the implementation of numeric storages caused them to have little influence on the rating estimation and, subsequently, the fitness function of the genetic algorithm. However, we have tested our design process with this issue multiple times and it did not have a large influence on the outcome.

Additionally, it was possible for the mutation of some numeric features to cause their values to exceed their assigned intervals in some cases. Notably, this did not affect the numeric storages for any of the color features. As such, to our knowledge this bug had no effect on the study results.

Finally, our study was designed in a very open manner, encouraging an open discussion with the study participants about the design process. However, this led to the fact that our evaluation does not include many quantifiable results, and thus may not be entirely reliable. Nevertheless, this way of conducting the study led to some interesting observations that might not have been found otherwise, for example requests for features that the participants would have liked to better incorporate their ideas.

External validity

We settled on a set of parameters to use for the genetic algorithm. For instance, the number of generations is set at eight, and the size of the population at 100 individuals. We did not investigate in detail how and to what extent the genetic algorithm could be further improved by tweaking these parameters. Nevertheless, these parameters still worked well for generating new designs in our own attempts of using our in-development design tool.

Similarly, the genetic operators for our genetic algorithm are defined in a rather simple manner. It is conceivable that by refining them further, the genetic algorithm could be able to produce even better results. For example, a possible improvement could be to weight the parents' properties by their estimated rating in the crossover operation. Still, in our testing, these straightforward definitions of the genetic operators as showcased in this thesis produced a wide range of different designs and, to a certain extent, were able to adapt to the user's wishes.

The study only had a sample size of five participants, who all were computer scientists. As such it is not generalizable to the general user, especially since the choices made by the participants were very subjective and vary wildly from person to person. However, we still tried to cover a larger range of different experience levels, ranging from no website design experience at all, to multiple years of professional website design experience.

Additionally, we only used one test scenario, which showed multiple issues during the study, such as the misalignment of center-aligned text headings if an image is present, or the colors of the buttons in the navigation bar for some color schemes. Other scenarios with different kinds of websites were not tested.

However, the goal of this thesis was to examine the facets of our design automation approach. Investigating differences between different settings of our algorithm, different kinds of users, or different design tasks was out of scope for our work. In the following chapter about future work, we expand on ideas we have for further exploration in this area of research but could not include in this thesis.

7 FUTURE WORK

In this chapter, we outline possible improvements to our approach. Also, while working on this thesis, more ideas for exploration in future research emerged, which we will present here as well.

Design process enhancements

While our rating storage based approach has shown success, there are still some issues with it. For one, it considers every feature individually and does not recognize correlations between different properties. In our test scenario, this happens with colors, for instance. Each color is defined as three distinct properties (red, green, and blue). Changing only one of these properties changes how the color visually looks and thus influences the ratings the user gives to the other two properties as well. How these dependencies between features can be modeled with our rating-based approach could be explored further in the future.

Another possibility for improving our design process would be to try incorporating objective guidelines. For instance, in our evaluation, we found that there often were difficulties with the contrast between different colors. This contrast could be calculated automatically and considered by the algorithm, e. g. by implementing a bias against designs with low contrast between colors.

In this context, it is also conceivable to use a different approach than a genetic algorithm altogether to generate designs to show to the user. For instance, a more involved machine learning algorithm like a neural network could potentially produce better results with less randomness. Another possibility would be to gather data through many users rating many different designs. This data could then be used by an algorithm as prior cumulative knowledge about subjective preferences.

The human element

Another area open to further exploration is how the user can be integrated better into an automated design process and incorporate their own ideas. For improving our approach, we have multiple different ideas.

In our user study, participants requested to still be able to slightly tweak a website design manually. In future works, one could experiment with giving users different amounts of control over the designs themselves. One possible angle on this issue is that users first want to get a broad idea, which they then tweak. As such, it could make sense to split the design process into multiple phases. Another option would be to allow the user to lock specific properties that they don't want to change anymore.

Another interesting facet of our design tool was the decision of how much nuance the user can express with their ratings. In our design tool, we had four categories that the user could rate separately. On one hand, it might be more comfortable or faster for the user to

only have a binary choice of whether they like a design or not. On the other hand, more precise ratings might give more information to the algorithm. In the extreme, one could imagine a rating system where every single style property can be rated by itself.

Additionally, sometimes users want to revert their changes and change back to an earlier design and take a different path. As such, it could be interesting to show a "timeline" to the user where they can browse to previous designs. This notion of a timeline was also used by Todi, Weir, and Oulasvirta [24]. Potentially, a timeline could also be non-linear, i. e., have multiple branches.

Further generalization

Our approach as described in this thesis cannot yet be generalized over all website designs since there is still some manual tasks that depend on the specific design domain. For one, it currently relies on a manual selection of design presets. This process could be automated, for example based on objective knowledge about the design space, or by using designs created by previous users of the design tool.

Additionally, with our approach, it is necessary to create the feature model for the possible website designs manually. Instead, one could imagine a system where it is instead generated, for example based on a given HTML website. Doing so presents several challenges in the selection of features. This website design feature model could even be subject to the design process itself, meaning the feature model would be different across designs. This would increase the diversity of available designs significantly, but also would be difficult to implement with our current approach.

The approach of generating a website design could also be extended to generate the website structure as well. Instead of a CSS file being generated for a given HTML file, the HTML file itself would then also be generated, only based on the page contents.

Finally, it should be possible to apply our approach to configurable systems other than website designs. While we did ignore cross-tree constraints in this thesis, accounting for them in our approach could allow for the creation of personalized configurations for any configurable system. A major challenge in doing so would be to visualize each configuration in a way such that the user can inspect and rate it quickly.

8 CONCLUSION

In this thesis, we explored a method for automating the process of designing a website while incorporating immediate user feedback. By generalizing the concept of website design and modeling it as a configurable system, our approach requires little prior hardcoded domain knowledge for the algorithmic part of the process.

In the design loop, we generate random designs, which then are shown to the user who can rate them. These ratings are then recorded, allowing us to estimate the user's preferences on designs that they have not seen yet by extrapolating from their prior ratings. We employ a genetic algorithm using this estimation as a fitness function to develop new designs to show to the user.

To evaluate our approach, we created a website design framework as a test scenario. Our goal was to examine how the human user interacts with the automated design process. For this, we conducted a user study where the participants could try our design tool and provide feedback. We included a semi-structured interview with each participant, using a questionnaire that we designed to be able to answer our research questions.

The study showed that our design process can produce satisfactory website designs quickly and in few iterations. The general workflow of rating the designs was easy to understand and well-received by the participants. However, it still has some limitations. Users feel that the final result could be further improved and wish that they had more influence on the design process. As such, it feels more passive and gives ideas and inspiration, but does not fully allow for human creativity to take part.

We see multiple areas for future research. For one, our algorithmic approach could be developed further. For example, it could use prior domain knowledge or collect ratings of many users to gather large-scale subjective criteria to find better designs even faster. At the same time, our method still requires some manual tasks, like the description of the design space as a feature model, which could also be an area of interest in future research.

Also, there is further room for exploration in how the human user and its creativity can be integrated better into a common design loop. One aspect of this is the amount of control the user has over their ratings and the designs in general.

In conclusion, finding a balance between automated assistance and fine-grained control by the human designer remains a challenge. In this thesis, we showcased a simple method which shows promising results, opening the path to future exploration.

APPENDIX

A.1 TEST SCENARIO PRESETS

In this section, we present the presets which we manually selected for our test scenario website design shown in Chapter 5.

Preset	Background	Secondary	Highlight	Text
Lavender	#c0a9bd	#f4f2f3	#94a7ae	#64766a
Dawn	#fbe0c3	#ffbb98	#7d8e95	#344648
Cherry	#c3cbd6	#748b6f	#b03643	#2a403d
Hazelnut	#ffd5af	#e59a59	#c86820	#712e1e
Peach	#f6f4e8	#e59560	#6a8e41	#1d3124
Radish	#f8efea	#ded369	#e0475b	#192f01
Blueberry	#f2ebe5	#647295	#8f294e	#2b262d
Citrus Blue	#fae6b1	#b3dee5	#ffa101	#31525b
Pale Red	#d69f3a	#f8d4ba	#c34f5a	#541412
Pumpkin	#f7f4ef	#feaa00	#788402	#342628
Whale	#361999	#361999	#78fff1	#ffffff
Deep Teal	#162b32	#162b32	#ff4838	#ffffff
Buttercup	#ffffff	#f1b814	#bd1e51	#490b3d
Mirage	#9daaf2	#1a2238	#ff6a3d	#f4db7d
Neptune	#000000	#12151f	#371bb1	#05f4b7
Pacific	#ffffff	#1fc58e	#fae62d	#191414
Dystopia	#be2f29	#ecaf44	#1a2c42	#0c1115
Waves	#00abe1	#00abe1	#ffffff	#161f6d
Ink Tint	#f7f7f7	#f7f7f7	#7da2a9	#000000
Mint	#ffffff	#ffffff	#8da242	#000000

Table A.1: Color scheme presets

Preset	Headings	Text	Navigation
Modern	Calibri	Calibri	Calibri
Magazine	Times New Roman	Calibri	Calibri
Oldschool	Times New Roman	Times New Roman	Times New Roman
Console	Consolas	Consolas	Consolas
Mixed	Calibri	Times New Roman	Consolas

Table A.2: The font families of the font presets

Preset	Heading alignment	Text alignment	Padding	Image position
Newspaper	Centered	Justify	20	Right
Slim	Left	Justify	10	Right
Flag	Right	Right	50	Left
Middle	Centered	Centered	20	Right

Table A.3: The alignment presets

Preset	Navigation bar		Horizontal margin	Sidebar		Twitter embed	
	Fixed	Button style		Position	Width	Mode	Height
Extra wide	Yes	Hover	0	Left	300	Light	400
Wide	No	Rounded	20	Right	250	Dark	300
Normal	No	Sharp	50	Left	200	Dark	500
Slim	Yes	Rounded	100	Disabled			
Extra slim	No	Sharp	200	Disabled			

Table A.4: Website layout presets

A.2 STUDY RESULTS

This is a summary of the answers to the interview questions in Figure 6.1, which we discuss in Section 6.3.

1. *Quality of the final design*
   a) On a scale from 1 to 10, how satisfied are you with the design?
      Participant A: 9/10
      Participant B: 5/10
      Participant C: 7/10
      Participant D: 7-8/10
      Participant E: 8/10
   b) Is there anything you would like to change about it? If so, what?
      Participant A: Nicer shade of "secondary" blue color
      Participant B: Squeeze text into the middle; change color scheme, similar to how it was in a prior design
      Participant C: Increase font size; change background color
      Participant D: Color scheme: colors too clashing, green too prominent
      Participant E: Color scheme: light mode would be better for reading a lot of information
2. *Feeling of the process*
   a) Do you think the design process was fast or slow?
      Participant A: "Rather quick"
      Participant B: "Can be faster if you know what the algorithm does, if the algorithm goes in the right direction", also the user can get stuck
      Participant C: "Fast", but you don't necessarily get what you want, but it can help for a fast demo design
      Participant D: "Fast to get a general overview", apart from some troubles with the color scheme
      Participant E: Maybe slower than manually choosing options, but in general "quite fast"

b) How mentally exhausting was the design process using the tool?

- Participant A: Not very exhausting
- Participant B: "I would not like to do that on a daily basis"; have to keep track of prior ratings
- Participant C: "Not much"; "Sometimes, it's a bit to think about what do I have to rate in the three designs to get what I want, because none of them are fitting to my preferred color scheme"
- Participant D: Did not take much effort, "quite easy"
- Participant E: Not much effort

c) What role do you think you played in the design process?

- Participant A: Neutral
- Participant B: Passive
- Participant C: Passive
- Participant D: Passive
- Participant E: Passive

d) Are you directing the design process or does the tool "lead" you?

- Participant A: On one hand, tried to direct the algorithm by picking specific ratings, on the other hand there are just four sliders to influence the tool
- Participant B: "Cannot really influence the algorithm's decisions in a way I'd like"
- Participant C: Design itself was given by the tool, no active part in designing
- Participant D: "More of a user"; "Just opinions, not 'does this make sense"
- Participant E: "I was just looking at designs"

3. *Creativity*

a) How diverse were the designs that the tool showed to you?

- Participant A: There was a lot of diversity
- Participant B: Color schemes were diverse, layout not so much, but "diverse enough"
- Participant C: The basic layout was very similar, the rest (mainly color scheme, and font) made a difference
- Participant D: At the beginning, they were different, but later they were less diverse
- Participant E: Designs were similar at the beginning, this made it a bit hard to tell the difference

b) Were there any designs that stood out to you?

Participant A: No, maybe the very first one that was used in the introduction as an example

Participant B: "Yes, at the beginning, the light designs were the most neutral and pleasing, but then I surprisingly liked the more blueish designs, as well as the salmon-red one which looked quite neutral"

Participant C: No, "there was no design where I was like, that is very good"

Participant D: Did not like the aggressive orange designs

Participant E: Pink-green color combination was not great

c) Did the tool hinder you from doing something? If yes, what was it?

Participant B: Yes: Ability to fix small things while rating; want to keep certain parts of the design such that future ratings don't influence the design too much. Or maybe: what if I'm shown a comparison between new and old design and which I would like to keep (maybe merging them)

Participant C: Yes, sometimes wanted to tweak a few little things about some of the designs

Participant D: No

Participant E: No

d) Do you think you could have created a better design in the same amount of time, without the tool?

Participant A: No, only if I knew before how the website was supposed to look. During the rating I also changed my mind on what I want

Participant C: No

Participant D: No

Participant E: Yes, maybe

4. *Rating system*

a) How big do you think was the impact of a single rating?

Participant A: It was impactful, and it became more impactful over time

Participant C: Impact is especially high if extreme ratings are given (very high or very low); unsure about impact of ratings in the middle

Participant D: Depends on how much you slide the bar, in the end it had a direction; in the beginning impact was low, afterwards impact was noticeable

Participant E: In the beginning, I thought it had a low impact, then later I thought the impact was bigger

b) How well do you think the tool reacted to your ratings?

Participant A: Well, especially if the sliders were put especially high

Participant B: Unsure, was surprised by the design changing in the first place

Participant C: Sometimes, outcome was surprising but at other times it would be as expected

Participant D: Not very well because it mixed certain things I liked in the beginning that resulted in a worse design; initial voting was good, afterwards there was nothing in this direction of the color scheme

Participant E: It was quite good, but the text changed once and that was unexpected

c) Do you think the designs shown to you made sense, considering which ratings you gave?

Participant A: No

Participant B: No

Participant C: Yes

Participant D: Yes, there were some familiarities

Participant E: Yes, at the beginning the designs covered different kinds of designs, later designs were for choosing more specific designs

d) Why do you think the process arrived exactly at this final design?

Participant A: Not sure

Participant B: "Some implementation probably led to this"; factors are unclear

Participant C: Things that were rated as "good" are present in the final design

Participant D: Picked parts of the designs that I liked at the beginning, and then based the ratings on those

5. *Applicability*

a) Would you use this tool for designing websites? Why or why not?

Participant A: Unsure, websites are usually very special/personal, but for inspiration it is really good

Participant B: Yes; similar tools already exist, but not for whole websites

Participant C: Yes, if it had more possibilities, like more fine-grained ratings

Participant D: Yes, definitely; as a non-expert for a start to have some prototype/template/ideas

Participant E: Yes

b) Where do you see potential for this approach, and what are the limitations?

Participant A: Good for getting first ideas on how to make the website look; there are only a few things that you can change, there are a lot but just a fixed number of options

Participant B: Would be better to have more control; for real designers (especially perfectionists), it's less fun, but for non-designers it might be a good helper

Participant C: Good for developers who aren't too good at design themselves, many developers have this problem; limited by just a few basic designs

Participant D: In research, you don't want to spend too much time on designing websites; we don't have the expertise to build a good website but we still want to profit from having one

Participant E: Small companies could maybe use something like this; something made by a machine might have a bad image as opposed to something human-made

A.3 DESIGN TOOL SCREENSHOTS

This section includes some screenshots of the design tool that we developed for this thesis.

Figure A.1: The screen for rating the nine initially selected presets

Figure A.2: The screen during the design loop. The design at the bottom is the currently best design, and the designs at the top can be rated seperately to influence it. The button on the right continues the rating process, while the button on the left selects the currently best design and terminates the design loop.

Figure A.3: The design inspector that can be brought up by clicking on a design

BIBLIOGRAPHY

[1] Mathieu Acher, Philippe Collet, Philippe Lahire, and Robert France. "Composing Feature Models." In: *International Conference on Software Language Engineering*. Springer. 2009, pp. 62–81.

[2] Sven Apel, Don Batory, Christian Kästner, and Gunter Saake. *Feature-Oriented Software Product Lines: Concepts and Implementation*. Springer, 2013.

[3] David Beasley, David R Bull, and Ralph Robert Martin. "An Overview of Genetic Algorithms: Part 1, Fundamentals." In: *University computing* 15.2 (1993), pp. 56–69.

[4] Alexandra Melike Brintrup, Jeremy Ramsden, and Ashutosh Tiwari. "A Review on Design Optimisation and Exploration with Interactive Evolutionary Computation." In: *Applications of Soft Computing* (2006), pp. 111–120.

[5] Sung-Bae Cho. "Towards Creative Evolutionary Systems with Interactive Genetic Algorithm." In: *Applied Intelligence* 16.2 (2002), pp. 129–138.

[6] Alex Clem. *30 Refreshing Color Palette Ideas for Your Website*. Website. Shutterstock Blog. URL: https://www.shutterstock.com/blog/color-palettes-for-websites.

[7] Carrie Cousins. *50 Best Website Color Schemes of 2022*. Website. Design Shack. URL: https://designshack.net/articles/trends/best-website-color-schemes/.

[8] Anna Maria Feit, Mathieu Nancel, Maximilian John, Andreas Karrenbauer, Daryl Weir, and Antti Oulasvirta. "AZERTY amélioré: Computational Design on a National Scale." In: *Communications of the ACM* 64.2 (2021), pp. 48–58.

[9] Krzysztof Gajos and Daniel S Weld. "SUPPLE: Automatically Generating User Interfaces." In: *Proceedings of the 9th international conference on Intelligent user interfaces*. 2004, pp. 93–100.

[10] John H Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT press, 1992.

[11] Gregory Hornby, Al Globus, Derek Linden, and Jason Lohn. "Automated Antenna Design with Evolutionary Algorithms." In: *Space 2006*. 2006, p. 7242.

[12] Markku Laine, Ai Nakajima, Niraj Dayama, and Antti Oulasvirta. "Layout as a Service (LaaS): A Service Platform for Self-Optimizing Web Layouts." In: *International Conference on Web Engineering*. Springer. 2020, pp. 19–26.

[13] Melanie Mitchell. "Genetic Algorithms: An Overview." In: *Complex.* Vol. 1. 1995, pp. 31–39.

[14] N Monmarché, G Nocent, M Slimane, G Venturini, and P Santini. "Imagine: A Tool for Generating HTML Style Sheets with an Interactive Genetic Algorithm based on Genes Frequencies." In: *International Conference on Systems, Man, and Cybernetics*. Vol. 3. IEEE. 1999, pp. 640–645.

[15] Antoine Oliver, Nicolas Monmarché, and Gilles Venturini. "Interactive Design of Web Sites with a Genetic Algorithm." In: *ICWI*. 2002, pp. 355–362.

[16] Antti Oulasvirta, Niraj Ramesh Dayama, Morteza Shiripour, Maximilian John, and Andreas Karrenbauer. "Combinatorial Optimization of Graphical User Interface Designs." In: *Proceedings of the IEEE* 108.3 (2020), pp. 434–464.

[17] Quentin Plazar, Mathieu Acher, Gilles Perrouin, Xavier Devroey, and Maxime Cordy. "Uniform Sampling of SAT Solutions for Configurable Systems: Are We There Yet?" In: *IEEE Conference on Software Testing, Validation and Verification (ICST)*. IEEE. 2019, pp. 240–251.

[18] Juan C Quiroz, Sushil J Louis, Anil Shankar, and Sergiu M Dascalu. "Interactive Genetic Algorithms for User Interface Design." In: *IEEE Congress on Evolutionary Computation*. IEEE. 2007, pp. 1366–1373.

[19] Gábor Renner and Anikó Ekárt. "Genetic Algorithms in Computer Aided Design." In: *Computer-aided design* 35.8 (2003), pp. 709–726.

[20] Juan Romero and Penousal Machado. *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music*. Springer Science & Business Media, 2008.

[21] Reimar Schröter, Thomas Thüm, Norbert Siegmund, and Gunter Saake. "Automated Analysis of Dependent Feature Models." In: *Proceedings of the Seventh International Workshop on Variability Modelling of Software-intensive Systems*. 2013, pp. 1–5.

[22] Karl Sims. "Artificial Evolution for Computer Graphics." In: *Proceedings of the conference on Computer graphics and interactive techniques*. 1991, pp. 319–328.

[23] Davy Sorn and Sunisa Rimcharoen. "Web Page Template Design Using Interactive Genetic Algorithm." In: *International computer science and engineering conference (ICSEC)*. IEEE. 2013, pp. 201–206.

[24] Kashyap Todi, Daryl Weir, and Antti Oulasvirta. "Sketchplore: Sketch and Explore with a Layout Optimiser." In: *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*. 2016, pp. 543–555.

[25] World Wide Web Consortium (W3C). *Cascading Style Sheets – home page*. Website. URL: https://www.w3.org/Style/CSS/.

[26] Web Hypertext Application Technology Working Group (WHATWG). *HTML Standard*. Website. URL: https://html.spec.whatwg.org/.