Master Thesis

Designing WordPress Themes by Example

Author(s):
Murolo, Alfonso

Publication Date:
2013

Permanent Link:
https://doi.org/10.3929/ethz-a-009980060

Rights / License:
In Copyright - Non-Commercial Use Permitted
Designing WordPress Themes by Example

Master Thesis

Alfonso Murolo
<amurolo@student.ethz.ch>

Prof. Dr. Moira C. Norrie
Dr. Michael Nebeling

Global Information Systems Group
Institute of Information Systems
Department of Computer Science

11th September 2013
Abstract

The number of users and websites in the internet are continuing to show a great growth.\textsuperscript{1} One of the causes of this growth can be found in the idea that more and more individuals concretely need their spot and identity in the World Wide Web. This spot can be of various types: a social network profile, a personal website, a blog. This in turn has caused a massive increase in the numbers of users of website/blog platforms and CMS such as WordPress and Drupal. Thanks to these platforms the number of personal websites and blogs has grown incredibly, making the chase for an effective End User Development technique very important.

At the same time, the progress and growth of these platforms have allowed the creation of many themes; currently themes can be browsed and visualized in general-purpose galleries as examples which have to be entirely applied to the WordPress website. Therefore, the users can currently choose the theme that best satisfies them and can apply it without being able to choose parts of different themes and combine them to create a completely new look and feel, specific to their website.

This thesis, called Designing WordPress Themes by Example, proposes a new metamodel-powered WordPress theme (called X-Theme) as well as a tool to allow end users and developers to build WordPress themes based on the examples they can see on other real WordPress websites running an X-Theme. Thanks to the designing by example tool, called XDE, a user can browse a WordPress website and take the parts of the underlying theme to design their own theme with a simple drag and drop. Exploiting the X-Themes metamodel, the XDE tool supports reuse of layout pieces, Javascript and PHP functionalities between the websites. Finally we investigate the capabilities and results of design powered by XDE in a user study. Our results suggest that test subjects appreciated the approach, finding it efficient and easy. Thanks to XDE, they were able to develop a WordPress theme (which could be completed in some hours coding manually, based on their own estimations) in a matter of minutes using examples they could see running in other websites.

\footnote{http://royal.pingdom.com/2013/01/16/internet-2012-in-numbers/}
Acknowledgements

*In primis*, I would like to thank Prof. Moira Norrie for the great advice and support that she gave to me, as well as for the passion and the commitment she feels to this project and to the students she guides. I would thank Michael Nebeling for his always honest and precise feedback in every moment of my work which was *REALLY* appreciated. I would also thank Prof. Filomena Ferrucci of University of Salerno for the great support she always gave me in every moment of my last academic years.

I would then thank Linda Di Geronimo, my colleague and love, who worked with me at the X-Themes project and who shared with me every moment of the last four years of my life. I would not have come this far without you. Love you! A huge thanks goes to my parents Felice and Alessandra and my sister Antonella, who supported me and gave me the opportunity to start this unbelievable ERASMUS adventure here at ETH Zurich, making a great sacrifice to let me come here.

A big thanks goes to my friends Adriano, Claudio, Daniele, Elia, Fabio, Federico, Mattia, Marco which shared many evenings with me on TeamSpeak, without all of you I could never feel so happy while being so far from home.

I could not forget to thank Giulio Franco for his unbelievable support he provides even at such a big distance such Switzerland-Sweden may be.

I would then thank the friends who have been with me for almost my whole life Francesco, Fabrizio and Vincenzo.

Thanks to the friends who continuously check in and make me and Linda feel their affection and are counting the days until we come back: Denise and Chiara, we will meet again soon!

In the end, thank you little Paolo; i know that even if i cannot see you or hear your voice anymore, you’re always by my side giving me your affection and advice.
# Contents

1 Introduction .................................................. 1  
1.1 Goals .................................................... 2  
1.2 Contributions ............................................. 3  
1.3 Structure of the Thesis ..................................... 4  

2 Background ................................................... 5  
2.1 Component-based Engineering and End-User Development ............. 5  
2.2 Designing by Example ....................................... 6  
2.3 The WordPress Platform ..................................... 8  
  2.3.1 Requirements and Goals ................................ 8  
  2.3.2 Design concepts ....................................... 9  
  2.3.3 Reuse within the WordPress theme development process ........ 12  

3 X-Themes Metamodel ......................................... 15  
3.1 Existing definitions of a theme ................................ 15  
  3.1.1 Wordpress Codex ........................................ 15  
  3.1.2 Professional Wordpress Design and Development ............ 16  
  3.1.3 Wordpress 3 Complete ................................... 17  
3.2 Adopted theme definition ................................... 18  
3.3 X-Themes Metamodel ........................................ 20  
  3.3.1 Components ............................................ 20  
  3.3.2 Layout Components ...................................... 22  
  3.3.3 Theme ................................................. 23  
3.4 Metamodel DOM implementation ................................ 23  
3.5 Converting a WordPress theme to an X-Theme ....................... 24  
  3.5.1 First steps .............................................. 24  
  3.5.2 The conversion process .................................. 25  
  3.5.3 Converting a Component .................................. 26  
  3.5.4 Converting a Layout Component ............................ 28  

4 Designing by Example based on X-Themes ........................ 31  
4.1 Concepts ................................................. 31  
  4.1.1 Use case scenario ...................................... 31  
  4.1.2 Component isolation .................................... 35  
4.2 Design Process ............................................ 35  
4.3 Implementation ............................................ 38  
  4.3.1 Software architecture and subsystems ...................... 38
## Contents

4.3.2 Theme Preparation ............................................. 41  
4.3.3 Reuse algorithm ............................................... 41  
4.3.4 Technical limitations ....................................... 43  

5 Evaluation: User Study 47  
5.1 Planning ........................................................... 47  
5.1.1 Tasks .............................................................. 48  
5.1.2 Questionnaires ............................................... 52  
5.2 Results .............................................................. 52  
5.2.1 Presentation ..................................................... 53  
5.2.2 Discussion ....................................................... 58  

6 Conclusions 59  
6.1 Limitations ........................................................ 59  
6.2 Future Work ....................................................... 60  

Appendices 64  
A User Study 65
List of Figures

2.1 The Loop (blue) printing a post in each iteration (in yellow). .......................... 10
2.2 A sidebar (blue) containing widgets (in yellow). ................................. 11
2.3 A starter Underscores theme. Some common components are already present (title, description, Loop), but are not styled. ................................. 12

3.1 Conceptual view of a WordPress theme. ........................................... 19
3.2 A diagram presenting a broke down structure of the X-Themes metamodel. The reader can see elements that are planned to be migrated between themes (marked in red) and elements that have to be exported as a consequence (marked in blue). ................................. 21
3.3 The XML representation of a Component, which involves logic as well as markup and style rules. .................................................. 21
3.4 The XML declaration of a layout component. The component's id is a, while the generated id is Name.a where Name is the generated X-Theme's identification. .................................................. 23
3.5 An example of the XML definition of a theme. ........................................... 23
3.6 The resulting DOM of a Component node. The metamodel has been mapped into the HTML5 dataset attribute. .................................................. 24
3.7 The current contents of the directory of the converted twentytwelve-nav Component. .................................................. 27

4.1 A mockup of a gallery browsing X-Themes. ........................................... 32
4.2 A detailed view of the browsing of previews of each component. ................. 32
4.3 An X-Theme with highlighted navigation menu. ......................................... 33
4.4 A navigation menu powered by PHP, JavaScript and CSS code. This is the menu the user can see while visiting an X-Theme. The menu entries are the WordPress' pages. .................................................. 34
4.5 The finished result of the reuse operation of the navigational menu. The component been executed in the user's WordPress installation, showing their pages. .................................................. 34
4.6 The first tool created to mix parts of themes. ........................................... 36
4.7 A generated mix with the first prototype. The two themes mixed were TwentyEleven and TwentyTwelve. .................................................. 37
4.8 The deployment diagram of the XDE tool. ........................................... 38
4.9 The two different views of an X-Theme. The first picture shows a normal theme view, while the second picture shows an assisted design view in which the user is hovering the mouse on the left navigation menu. ........ 39
5.1 The mockups the users had to reproduce during the user study. ........... 49
5.2 Detailed specification for the Task A. ........................................... 50
5.3 Source Theme 1 ........................................................................... 51
5.4 Source Theme 2 ........................................................................... 51
5.5 This picture shows the source X-Themes that had to guide the users in
task B, as well as being source for the examples to be reused in task C. .... 51
5.6 Task B specification as provided to the users. ................................. 51
5.7 The task specification car the users have followed to complete task C. .. 52
5.8 The users' self evaluations of the WordPress usage .......................... 53
5.9 The self evaluation (average) of the users' development skills of Word-
Press extensions .............................................................................. 53
5.10 Average time (in minutes) spent for tasks by the users ...................... 54
5.11 The user evaluation of their success level compared to the task requests
and the satisfaction level after doing tasks A, B and C respectively .......... 55
5.12 The users' evaluation of the designing by example approach realized by
the XDE tool, after they completed task C ......................................... 55
5.13 The user estimation of the manual time needed to obtain the same
result as task C by coding manually ................................................... 56
5.14 The users' evaluations of the potential improvement provided by X-TG
and XDE in the Theme development process, the reusability and the time
spent developing a theme ................................................................... 57
6.1 A mockup of a gallery browsing X-Themes ................................. 60
6.2 A detailed view of the browsing of previews of each component ....... 61
Introduction

During the last years, the web has faced an enormous growth in terms of numbers. Both companies and individuals have begun to understand the importance of their presence on the web. We are assisting to a process in which every individual concretely needs his spot and identity in the World Wide Web. Each individual thinks of their identity as their own way: some think about a profile, some think about a personal website, and some think about a blog. This concrete need was indeed satisfied, until now, by social networks, website/blog platforms and CMS such as WordPress, Drupal and so on, causing an enormous growth of personal and company websites throughout the world. This advocates for an improvement of end user development techniques that can be provided to the users of these platforms, allowing them to design the look and feel of their website as completely as possible.

On the other side, web application development and end user development have always been considered a challenge from the research community. Many research teams throughout the world have focused their efforts in creating and exploiting methods, techniques, modelling languages and tools to improve the quality of the development projects or to allow end users to develop their own web application freely and easily. Unfortunately, the end user development techniques and tools supporting the most widely used website/blog creation platforms has been low. These tools could mostly help the end user customize specific pre-defined elements of the appearance of their website. Moreover, there was absolutely no support to provide additional functionalities to their website; in the end, the user could only customize their layout. Finally, an end user could not build their own website with additional functionalities that they could see on other websites.

These kind of tools do not take into account one of the most important ways of supporting design: designing-by-examples. Providing examples during the design process is crucial for the success and the quality of the final product though (Herring et al. [7]). As we can read in [7], we can categorize examples in two different typologies:

1 http://royal.pingdom.com/2013/01/16/internet-2012-in-numbers/
• **Tangible Examples**: Design assisted by concrete, physical, tangible examples, which can be nice to review but hard to reuse and to analyze after a review; for example, the paper shows an interesting scenario in which a designer comes back to his tangible example after a couple of weeks, noticing he has flagged some part of the example, but forgetting why he flagged it. Moreover, this kind of example is quite expensive to store.

• **Intangible Examples**: also known as digital examples, this category involves pictures, mockups, working design prototypes. This strategy is quite interesting because it is concretely influenced by the way the designers store and browse their digital examples. Sharmin et al. [16] have shown that designers tend to identify the example with the real, running artefact in a real website or a gallery of examples. This shows how the designers like to identify the example as the working artefact rather than a similar representation; one of the goals of this thesis is to support examples that are the real, working artefacts, rather than a proxy or a representation.

Recently though, some research has been done to improve designing by example techniques. Kumar et al. [8] [9] have provided an algorithm that can identify pieces of websites and automatically mix them into different layouts, giving the chance to rapidly prototype website design thanks to different examples provided. Although this improvement is remarkable and empowers reuse of layout examples, it only provides the migration of the content from a design example (source) to another (destination), completely ignoring the possibility of dynamic functionality in the source website such as Javascript.

This thesis investigates a different approach that could improve website development by using the designing by example approach, while supporting reuse of different running examples (involving both layout pieces and functionalities) between the websites and that could be also oriented to massively-used website platforms. Moreover, this approach should also support a way to eventually browse examples of existing websites. The approach of the example is somehow already present in these platforms thanks to the theme galleries which show theme examples, but in a limited way: the users can browse various themes which can be applied atomically to their websites. By atomically we mean that unfortunately there is no way to make combinations of parts of these themes to create new themes. Therefore, the reuse provided by this approach is limited to the whole theme and does not extend to its parts. This thesis investigates an approach which can provide reuse at the component level, in order to make compositions of reusable parts of different themes in a visual way.

### 1.1 Goals

WordPress is one of the most widely used platforms in the web by individuals and companies²; this makes WordPress the best candidate as a target platform in which to improve the designing by example approach. More in detail, this work aims to support design operations providing running examples from some WordPress websites and reuse them in other appearance themes for WordPress blogs and websites. Currently, in the WordPress platform, there is no way to combine parts of themes into a new one; therefore, another goal of this thesis is to boost the theme development capabilities to easily provide the opportunity to mix parts of

²http://en.wordpress.com/stats/
different themes using the designing by example approach. Unfortunately, in the WordPress community, there is generally a lack of formal definition of the basic concepts upon which the WordPress platform is based. Moreover, the WordPress knowledge-base entirely lacks a formal definition of the concept of a theme. Finally, this whole design support should be coupled with a dedicated theme generator supporting the designing by example approach by realizing the design interface to support reuse.

Therefore, the goals of this thesis are the following:

1. **Analysis of State of the Art.** Analyze the WordPress platform but also the development practices described in tutorials and documentation.

2. **Definition of Theme Concept and Metamodel.** A clear theme concept and associated metamodel must be defined that would allow developers to define their themes at a higher level of abstraction and support reuse by designing by example.

3. **Design and Development of the XDE Tool to support Designing by Example.** A graphical tool, called XDE, should be developed that allows users to create a web site by selecting and combining components from galleries of existing websites developed using the X-Themes theme generator. It should be possible to adapt these components to integrate into the desired layout. An important factor would be to address technical issues of being able to reuse functionality implemented using JavaScript rather than just assuming static components as has been done in previous projects in the HCI community. [8]

4. **Design and carry out a User Study.** A User Study should be designed and performed in order to assess the efficiency and the validity of the designing by example approach in terms of End User Development effectiveness and efficiency, as well as satisfaction and quality.

### 1.2 Contributions

After investigating the theme development techniques currently applied by the WordPress community, this work provides a new definition of a WordPress theme. This definition allowed us to better represent a theme in the X-Themes metamodel which has been defined to support the theme generation operation and the reuse of its pieces, made possible eventually by the designing by example approach. In addition, this work presents a way to convert normal themes to X-Themes, showing a step-by-step procedure to adapt target themes to the X-Themes metamodel structure. Thanks to this procedure, we have shown that even existing WordPress themes can still be adapted to eventually support the designing by example approach.

To further support users in the design process we present also the XDE tool which allows the user to mix parts from existing X-Themes and which is built as an extension of the X-Themes Theme Generator [5]. Moreover, given the digital nature of web artefacts, the XDE tool will support the user in seeing their examples during their execution, directly into other X-Themes in their browser, allowing them to mix the examples using the drag and drop interaction.

Finally, in a user study we show that this reuse can be achieved easily and with a good level of user satisfaction, even if the tool’s performance can be still improved: the creation of
a theme (which included complex functionalities from other themes) required a stable and small amount of time (around 6 minutes) compared to a time estimation of some hours when coded manually. Thanks to this approach, the users have generally appreciated the capability to mix layout parts and functionalities from different running themes on different WordPress installations without having to look at any low-level artefact.

1.3 Structure of the Thesis

The thesis starts in Chapter 2 where we analyze the related work, the WordPress platform as well as the theme development process in detail. In Chapter 3, we present the definition of a WordPress theme as adopted by the X-Themes project and the X-Theme metamodel that allows the XDE tool to realize the designing by example paradigm. In Chapter 4, we introduce the main concepts and assumptions that we rely on for the realization of the designing by example tool, as well as the architecture and implementation of the tool itself. Chapter 5 presents the methodology, the planning and the results of the user study carried out on both the theme generator and the designing by example tool. Finally, Chapter 6 states the conclusions of the work, the limitations, the threats to this work and the future directions.
2

Background

XDE aims to improve the WordPress platform by giving the users the opportunity to get inspired by examples in other WordPress themes and reuse them as isolated components in the theme they want to create. This can be considered a goal towards end user development based on the reuse of components, which play the role of *examples* for the user, who is playing the role of the *designer* of his own WordPress theme. Therefore, this work is strictly related to other research that has been conducted in the last years and regarded similar topics. This chapter gives background information about the research previously completed by the research community, in order to better clarify the motivations and the differences between the previously researched approaches and the one proposed in this thesis.

2.1 Component-based Engineering and End-User Development

It has been some years that the research community has understood the importance of component-driven software development; soon enough, it has been also clear that improving this kind of component-based software development technique could improve reuse during the development process (Mørch et al. [15], 2004). Moreover, the research community has imagined a future in which the development of software could be powered by this component concept, giving the opportunity directly to the end users to create their own software without going out of their context and writing low-level code (End User Development) [15]. Also Lieberman et al. [14] have conceived this goal of making a software *easy-to-develop* rather than focusing only on the *easy-to-use* aspect. In their perspective of end user development, the users can combine different components at runtime; the more the system is correctly separated into components, the more flexible the resulting application will be. According to Lieberman et al. [14], there are three different complexity levels for the component-based end user development.
1. **Configuring a component**: The easiest way to customize a system is to configure each component. These environments might help the user to customize the behavior of each component.

2. **Changing the component composition**: The user can insert, modify and delete a component; in addition they can also rewrite the components or creating new ones. In this level the users have to understand how the components interact.

3. **Design new components**: The users can design and build new components, but this requires them to know programming languages of some sort.

More generally, the goal of component-based end user development is to allow the users to combine these components while reducing the development effort. Therefore, this goal translates into finding a compromise between complexity in the user perspective and flexibility of the developed software. One of the contributions of this thesis is to give the user a way to design new components they can reuse inside the XDE tool.

Later on, a new kind of end user development environments was conceived: **Software Shaping Workshops**. These environments focused on specific problem domains the users were experts of [3][4]. Costabile et al. [3] propose a scenario in which medics could tailor their own software (regarding medical knowledge or domain). The technical instrument that allowed the medics to achieve this goal was the annotation: the software could be extended in the information it was storing and showing to the various medics who were making a diagnosis. Moreover, Costabile et al. [3] also propose a design tool capable of creating and modifying the interfaces used by the medics for their diagnosis. Thanks to this approach, senior medics could create the diagnosis tools without feeling lost in the low-level details of the software under development; unfortunately, this approach can be applied only for enclosed problem domains.

A harder goal is, instead, to provide support for end user development in a wider and more chaotic environment such as the World Wide Web. Moreover, given the great growth of platforms such as WordPress or Drupal, it has become of primary importance to support end user development in these platforms. To achieve this goal, Leone et al. [13] propose a component-based composition approach that could allow the end user to extend their WordPress platform at various levels: the data level, the schema level, the application logic level and the user interface level. Thanks to the definition of some connectors and mappings, Leone et al. [13] have built a WordPress plugin that realizes their approach, called **Composition**. This could be possible also thanks to another work [12], in which the authors proposed a WordPress meta-plugin called **Relationships**, capable of realizing data level associations starting from an ER model. Thanks to Composition the end user could create compositions and mappings between data (and higher level) entities, making them exploitable in the WordPress platform.

### 2.2 Designing by Example

The use of examples in the design process appears to be very important. The exact relationship between the usage of examples during the design process and the resulting quality of the developed product is still unclear, though many researchers have focused on this matter. Moreover, the impact of the example-aided design is variable depending on the context to which it is applied.
Herring et al. [7] have investigated the application of examples to the following contexts: web design, graphic design and product design. In the context of web design, a study with some professional designers has been carried out. These designers stated that they were mostly getting inspiration from other websites, instead of thinking of the website design from scratch. This statement marked a huge distinction between web design and the other two types of design considered in the work (Herring et al. [7]).

Moreover, the use of examples is improving the quality of design products as show in Lee et al. [11]. Thanks to three experiments, this work proved that the effects of presenting, borrowing and adaptively browsing examples aided the design activity. This was proven thanks to independent raters who preferred design products created exploiting examples to design products created without exploiting examples. Another important point stated by Lee et al. [11] is that users make use of multiple examples when they are creating new designs.

Sharmin et al. [16] focused on early design activities, when designers are striving to search for new ideas and examples for their project. In their work, they show some findings which are relevant from our point of view:

- The designers are concretely interested in the reuse of prior ideas and knowledge
- It is necessary for the designers to search for examples, but a new way (maybe visual) of browsing them would be more efficient than common grid/list navigations

As we have shown, examples are widely used by designers to improve their designs. Therefore, the research community has tried to propose a new approach in web development that could take advantage from design examples and reuse them in the project at hand. This paradigm was called designing by example. In order to give to the end users the opportunity to take inspiration from the examples they could find in the web, Fitzgerald [6] developed in 2009 CopyStyler, a tool that could enable end users with no technical skills to reproduce layout pieces from other websites, realizing in this way a designing by example approach. It was implemented as a Firefox extension, and provided a customized interface with two pages, compared side by side:

- the user’s page, positioned at the right side.
- the example page, positioned at the left side.

In CopyStyler, the user could make a mouse selection in their page and a selection in the example page; then the tool allowed them to reproduce the example styling rules in the selected element in the user’s page. This tool was realized with two IFRAMES, in which the users made their selections. The mixing algorithm analyzes the DOM tree of the selection in the source page; then it grabs various types of style rules, and applies them to the selected element in the user’s page.

A different approach can be found in Bricolage [8], which proposes an automatic algorithm to migrate websites between various example design proposals. Kumar et al. [8] propose an algorithm based on machine learning that splits the source page into components and tries to match these components with those from a destination design website. After every match was found, Bricolage replaces the content of each component with the corresponding one from the source website. In order to keep the style rules of the content during the transfer, it transforms CSS style rules to inline style for each DOM selector.
Although these approaches are supporting the designing by example paradigm, none of them can deal with dynamic components involving JavaScript or server-side code (for example: PHP).

### 2.3 The WordPress Platform

Since this thesis is proposing a designing by example approach built on the WordPress platform, we are now introducing the platform goals and every necessary concept which WordPress is based upon.

#### 2.3.1 Requirements and Goals

The WordPress platforms gives the opportunity to every user in the World Wide Web to become a publisher of information or content. Moreover, the WordPress Codex\(^1\) defines the platform as a light content management system with an extremely customizable core. WordPress provides several extension types that can be implemented to enrich the platform’s capabilities and its look and feel. The major requirements and goals for the platform as stated by the WordPress Codex are the following:

- **Easy to use:** WordPress should be easily understandable and usable by every user in the web, allowing them to publish their content in a fast and effective way.
- **Extendable:** WordPress should be easily extendable thanks to the community support, that is providing all the kind of extensions that a common user would like to have.
- **Fault tolerant:** WordPress should be bullet-proof towards every mistake or fault that may happen during the publishing of the content, automatically saving every modification the user is making on the content.
- **Multi-Device:** WordPress could be used from desktop as well as mobile devices.
- **Metaphor-powered administration panel:** WordPress should be easily manageable, powering the customization of the platform with the metaphor of Drag and Drop.
- **Overridable:** WordPress’ default behaviour may be overridden or altered using the hooks and filters mechanism, powered by the WordPress API, which is based on PHP.
- **Standard compliant:** WordPress should guarantee that every code that it generates is compliant with the W3C\(^2\) standards.
- **Platform:** WordPress should be executed on a xAMP\(^3\) server.

---

\(^1\)http://codex.wordpress.org/

\(^2\)http://www.w3.org/

\(^3\)An Apache web server featuring MySQL and PHP.
2.3.2 Design concepts

WordPress has been conceived to be extensible. Therefore, the platform tries to establish a set of concepts which represent these extensions, in order to provide a categorization as well as the basic tools to realize the platform’s extensions. In this section, we will introduce every concept that is necessary to the reader to better understand the contributions of this thesis.

Hooks

WordPress provides a tool to override or modify the result of every dynamic call that gets executed in the platform itself: the hooks. The name comes from the idea that user defined functions can be attached to the execution of the internal platform calls, like a hook attaches to some objects. Let $\alpha$ be a generic user-defined function, and let $\beta$ be a generic function defined and automatically executed somewhere internally to the WordPress Platform. WordPress defines two types of hooks:

1. Actions: a user that is hooking $\alpha$ to $\beta$ using an action hook, is simply queuing the execution of $\alpha$ after $\beta$. This hook can be invoked by the developer using the `add_action($\beta, \alpha)` function, present in the WordPress API.

2. Filters: a user that is hooking $\alpha$ to $\beta$ using a filter hook, is essentially replacing the code in $\beta$ function with the code in the $\alpha$ function. This hook can be invoked by the developer using the `add_filter($\beta, \alpha)` function, present in the WordPress API.

The WordPress development model was entirely based on these hooks, making them completely necessary to extend the WordPress platform in whichever way the developer wants to.

Posts and Pages

Since WordPress is mainly a blog publishing platform, clearly it contains posts. A post is a collection of formatted text, images, attachments that the user can write in order to publish the content he creates. The administration panel allows the user to create new posts. Generally, posts are dynamically shown in decreasing order of publishing date (this means, from the newest to the oldest).

WordPress gives also the opportunity to write static pages, which are essentially posts which can be browsed separately, generally through a navigational bar or menu.

Themes

For the average WordPress user, this term encompasses many aspects of the WordPress platform:

- the whole look and feel that a visitor of a WordPress website can see in his browser.
- the definition of the layout, styles, images and elements of the website appearance.
- the files that drive the dynamic behavior that is rendering the page to the user.
- the definition of data types that the user might want to add in order to empower his WordPress website.
• the definition of widgets or sidebars that can be placed in the layout.

Unfortunately, the WordPress community does not provide a formal definition which can correctly and completely identify the elements that compose a WordPress theme. This work also presents a contribution in which we analyze the current definitions of a theme and we provide a more formal and structured definition; for more information, see Chapter 3.

Dynamic theme elements

A theme generally contains some dynamic elements, which are changing during time. A trivial example of this kind of content might be the posts, that are continuously published by the user and have to be shown in a specific order on the web page. The posts are generally shown with a component, called The Loop by the wordpress community (cf. Figure 2.1). Essentially, The Loop is a while loop that is displaying the posts’ content and metadata, such as the publishing user, the title, the publishing date, and so on.

![Figure 2.1: The Loop (blue) printing a post in each iteration (in yellow).](image)

There are also dynamic theme elements regarding the blog itself. The WordPress API provides the bloginfo() function, which can dynamically display the blog information, such as the website title or description. For example, calling bloginfo("name") will display the website title.

Custom Types

Thanks to the hook mechanism, WordPress allows the registration of custom data types directly into the platform. A developer might extend the WordPress platform, providing the name of this custom type, the fields that are stored in this data type, and a basic administration layout page that allows the storage of this data type in the WordPress database. The extension procedure of the platform for custom types has become quite standard and documented in books and tutorials. Internally, WordPress stores an instance of a custom type as a post which is related to a collection of key/value pairs. Each key/value pair is composed as <fieldname,fieldvalue>.

In this way, a generic Loop can be reused and extended to display custom types.

Sidebars and widgets

A WordPress sidebar is a column that displays different information from the main content of the page. WordPress supports dynamically editable sidebars from the administration page. A sidebar is defined into the theme code and has to be registered in the WordPress platform. Generally, a sidebar displays static content or widgets (cf. Figure 2.2).
A WordPress widget is a self-contained area that has a specific function\(^4\) (for example: display all the posts of the blog, or all the post categories). Some default widgets are already defined and usable in WordPress. Moreover, a developer can extend the WordPress platform to support more widgets with the functionality he wants. Widgets can be dragged into sidebars from the administration page; in this way, the user can decide which widgets he wants to display and in which sidebar.

**Custom headers**

WordPress also supports dynamically changing image headers. A theme can be developed to support this functionality. If the applied theme supports custom headers, the user can choose a set of header images to display in his website as a heading image. WordPress natively provides the opportunity to crop the header to the size expected by the applied theme, in order to keep the theme layout unchanged. Moreover, WordPress also provides the chance to insert more than one image, giving the opportunity to have a randomly chosen header image at each website view.

**Plugins**

The final WordPress concept we present in this section is the plugin. Plugins are defined by the WordPress Codex as a group of PHP functions that extend WordPress’ functionalities. Generally plugins are stored in zip archives that contain all the PHP files required to properly extend the platform. In order to install a plugin, the user should upload the zip archive and

\(^4\)http://codex.wordpress.org/Glossary#Widget
2.3. THE WORDPRESS PLATFORM

activate it from the administration page. The X-Themes project is completely built on WordPress through the plugin mechanism. Therefore, a user could access all the features provided by the X-Themes Theme Generator (XTG) and the Designing by example tool (XDE) through the WordPress administration interface.

2.3.3 Reuse within the WordPress theme development process

The WordPress theme development process has become pretty standard. Since the platform has spread throughout the web, many books and tutorials have been written in order to share techniques and best practices to develop a theme. Many of these books and guides pursue a practical approach that will show ready-made code examples and explain them to the reader (such as [2]), giving him a better understanding of what he is doing. The example approach is much used in this learning process and gives the chance to better understand all the tools and weapons that a WordPress developer can use to create a complete theme. Unfortunately, since many of the WordPress theme developers follow the mostly referenced guides and books, they tend to reuse the basic theme elements that are repeated throughout all the themes they develop (for example: the WordPress title, The Loop). Therefore, a new project called Underscores\(^5\) (also known as _s) was born, trying to encompass all the best practices into a starter theme, already containing stylesheets and basic (unstyled) theme elements (cf. Figure 2.3). Moreover, this starter theme has been designed to be extended and is better separated in its parts, in order to achieve greater maintainability. The goal of Underscores is to speed up the theme development process making the designers focus on the look and feel of the theme, making them fly over the coding of the theme basic elements, which can be reused throughout almost every theme.

WordPress tries to natively embrace the reuse techniques in order to improve the theme development process. It provides in its API the \texttt{get_template_part($slug,$name)}

\(^5\)http://underscores.me/
function, which performs a PHP \texttt{require()} call that will include the file $slug-$name.php. This call will be executed locally to the theme directory in which \texttt{get_template_part()} is called. This gives the chance to the theme developers to isolate reusable parts of the themes in separate files, which can be included with this function and therefore be reused in other themes.
In order to organize and simplify theme generation and reuse, we have defined a metamodel that can provide a better structure to the themes, while giving the chance to store metadata that will empower the reuse of parts between themes, thanks to the XDE tool. This metamodel is used in the whole X-Themes project, of which this thesis is only part. Unfortunately, this metamodel had to be defined on the concept of theme, which was not completely clear and correctly defined. Therefore, in this chapter we will first introduce the analysis of existing definitions of a theme, as well as clarifying our adopted definition. Then, we will introduce the X-Themes metamodel and how a WordPress theme can be converted to an X-Theme, in order to support designing by example.

3.1 Existing definitions of a theme

In this section, we will first analyze some existing definitions of theme, in order to better understand which weak points these definitions may have and which parts state very important points about the concept of a theme.

3.1.1 Wordpress Codex

According to the WordPress Codex, a theme is defined as follows:

"Fundamentally, the WordPress Theme system is a way to "skin" your weblog. Yet, it is more than just a "skin." Skinning your site implies that only the design is changed. WordPress Themes can provide much more control over the look and presentation of the material on your website. A WordPress Theme is a collection of files that work together to produce a graphical interface with an underlying unifying design for a weblog. These files are called template files. A Theme modifies the way the site is displayed, without modifying the underlying software. Themes may include customized template files, image files (*.jpg,
**Analysis**

This definition aims to simplify the understanding of the theme concept to the people using WordPress, to make them understand what themes are and why they should use them, or even create their own. This definition recalls some design principles [1] that tend to isolate the graphical/presentation layer of an application from the application logic layer, so that the presentation layer could be changed/replaced independently from the rest of the application, without risk a breaking in the functionalities of the system.

Moreover, this definition stresses also the fact that apart from changing some skin, something more effective can be done: a theme can provide more control over the presentation of the website. This part of the definition makes it clear that some further logic control may be defined in the theme, to effectively customize the look and feel of a theme. A second part of the definition clarifies which files have to be edited to customize a theme, jumping towards a practical and concrete definition that ends with a how-to install a theme (not reported here for practical reasons). One of the most important issues about this definition is the lack of links between the concept of a theme and the other concepts of Wordpress: The Loop, plugins, custom post types. While it is true that a theme specifically defines the look and feel of the WordPress website, it is not true that it is exclusively a theme’s responsibility: plugins also have a huge impact on a WordPress website’s look and feel. The most important notes to be highlighted for this definition are:

- + Links a theme to the presentation layer of a web application.
- + Clarifies the independence between the theme presentation layer and the application logic layer.
- + Clarifies that a developer can also define some custom logic to customize the theme itself.
- - Fails to stress the importance of customization and reuse of themes, which has been lately considered very important (cf. Chapter 2).

### 3.1.2 Professional Wordpress Design and Development

This book [18] provides a long and detailed definition of a theme and all of its components, moving through template files, CSS, images and assets, and plugins. Here is a little excerpt from the definition:

"[...] a theme does several things, including structuring your content and providing the personality of your website. This is done through a combination of files and file types. You will notice a mix of PHP files and CSS files in the theme. A good WordPress theme keeps the style, which is CSS, separate from the structure and logic, which make up the PHP files. Although there are always reasons for breaking the rules, striving to keep these separate will improve the maintainability and efficiency of your theme."
Each theme has variations on these files and each themes files are different. […] Plugins […] contribute advanced functionality to a website. Some themes require specific plugins because the functionality is part of the themes personality, or they are needed to achieve a certain purpose in the theme. These plugins may be packaged with theme or may require separate downloads. […]”

Analysis
Just like the WordPress Codex (cf. Section 3.1.1, this definition also clarifies that the goal is to separate the look and feel from the procedural logic that will populate the themes content. This definition is mostly practical and tries to make the reader hold against practical concepts to communicate efficiently the meaning of the separation (styles in CSS, logic in PHP). It is explicitly stressed that one of the most important secondary goals is the maintainability of the theme: this is often neglected, preferring copy-paste approaches that tend to make things work instead of designing them efficiently. Finally, the definition provides an important concept: there might be plugins shipped within the theme, since they might be a core component of the look and feel of the theme itself. This concept is both mission-critical and dangerous: people might choose a theme instead of another because it provides some functionalities that the others do not provide; on the downside, if a theme ships functionalities inside, it means that the functionality is a core part of the websites look and feel, so theme logic and presentation might be tightly coupled. Software Engineering practices underline that tightly coupled layers are dangerous, because changes to the functionality might damage core parts of the themes.

The most important notes to be highlighted for this definition are:

- + Links a theme to the presentation layer of a web application.
- + Clarifies the independence between the theme presentation layer and the application logic layer.
- + Shows that some themes might feature some specific functionalities inside the theme itself.
- - Explicitly authorizes plugins to be shipped inside of a theme because they provide a core component to the look and feel, without underlining that if a plugin is part of the core, it is not a plugin anymore.
- - Although it stresses the separation between logic and presentation, it fails to stress the importance of customization and reuse of themes, which has been lately considered very important (cf. Chapter 2).

3.1.3 Wordpress 3 Complete
This book [17] provides a very short definition of a theme, moving through template files, CSS, images. According to the author of this book, a theme is nothing but a collection of files in a folder: some of them specify only the style (CSS), others specify some logic (PHP), and in the end there are the images. It is almost the same approach as the one provided in [2].
Analysis
For the author of this definition, the only important things are the layout, whether the theme offers customization pages, whether it is widget-ready and whether it is flexible or not. There is no word about logic-layout separation or maintainability, and there are no practices to be followed to create a good and maintainable theme. The whole learning approach provided by this book is similar to copy-paste, showing example ready-made code and explaining it a little.

The most important notes to be highlighted for this definition are:

- + Very basic definition, for improvised developers that just want to make their only theme of their life without planning to change it lately.
- + Logic only appears separated in forms of different file types from the presentation styles.
- - No explicit presence of custom defined theme functionalities which can be part of the look and feel.
- - Fails to stress the importance of customization and reuse of themes, which has been lately considered very important (cf. Chapter 2).

3.2 Adopted theme definition

Every presented definition has something that has to be kept in order to make a concrete and meaningful definition of a theme. A better definition of a generic theme would be the following:

"A theme is a collection of presentation style elements, rules, images, and logic that fetches the applications contents from the systems back end, and provides at least the basic functionalities and layout styles to correctly show the desired content. A theme makes the front end of the application. A theme represents a subset of the presentation component of the application and should be completely interchangeable with other themes that respect the same interface with the other system components. Moreover, a theme can contain theme-specific additional functionalities which result to have a core meaning in the look and feel; these functionalities should be shipped within the theme itself. Moreover, a theme is strictly related to the kind of data that has to be presented; it may contain a structured definition of the data types that are to be shown. Finally, in a correctly designed theme logic and presentation style rules as well as each of the elements that compose the theme should be kept separated in order to achieve the best maintainability."

This definition can also be applied to any application that supports themes, that can be a Content Management System on the market (including WordPress) or a general purpose application, or also an operating system.

An application of this definition to the Wordpress platform might be:
"A WordPress theme is a collection of CSS style rules, HTML structure elements, images, and PHP logic that fetches the website’s contents from the WordPress database using the WordPress API, and provides at least the basic functionalities and layout styles to correctly show the site’s posts or pages. All of these elements (layout sections, sidebars, custom headers) should be kept separate in order to improve reusability of the theme parts. A theme is located on the top of the WordPress API and makes the public front end of the website. It can specify general layouts as well as specific layouts for uniquely identified pages in the website. A theme represents a subset of the presentation layer of the application and should be completely interchangeable with other WordPress themes. Additionally, a theme can contain theme-specific additional functionalities that result to have a core meaning in the look and feel, as well as custom post types that are to be displayed; all of these should be shipped within the theme itself, while being kept separate from the other theme files in order to improve reusability; they should be not present in a plugin. In a correctly designed theme, logic and presentation style rules are kept separated in order to achieve the best maintainability."

Figure 3.1: Conceptual view of a WordPress theme.

This definition identifies the elements that a WordPress theme may contain, as well as stressing the separation between the parts of a theme, which should be kept separated to improve reusability and maintainability. Please also note that this definition does talk about additional functionalities that are part of the theme, but not about WordPress plugins. There is a main reason for that: a plugin is something that should be pluggable in and out without breaking the system in any way; if a functionality is of primary importance to the look and feel of a theme, it must not be a plugin, because it cannot be plugged out. Also referring to the
concepts stated in Chapter 2, a WordPress theme has been broke down into different parts, presented in Figure 3.1.

3.3 X-Themes Metamodel

Now that we have identified the components of a theme, we can present the X-Themes metamodel which tries to give a structured representation of each of the parts of a theme in an XML format. Starting from the theme structure given in Figure 3.1, we developed a new structure that involves more abstract concepts (cf. Figure 3.2). As we can see in the picture, the X-Themes project identified various components: some of them are planned to be reusable between themes; these items are marked in red. Moreover, some elements may require to be exported as a consequence of the exportation of a red item; these elements are generically called dependencies and are marked in blue.

As shown in Figure 3.2 we can identify various types of reusable items: Components, Layout Components, and Custom Types. In this thesis we focused on the first two. Finally, we have the X-Theme, that is a collection of the previous items. Please note that the plugin is not present in the X-Theme diagram; this is because according our theme definition it is not part of a theme. We will now define each of these items separately and more in detail, using a bottom-up approach. We will start from the basic components, going on to define the whole theme in the end.

3.3.1 Components

Before we go on, we have to define what a component is: by this name, we mean generic components that can include layout and style rules and dynamic behavior such as JavaScript or PHP logic. If a component has to be exported, it has to be clear that a functionality is not only specified by look and feel elements but also by logic units. JavaScript functions, CSS style rules and PHP logic might have an important role in the functionality so they have to be exported as well. Each Component is stored in a separate directory in the theme files. Each directory contains:

- every JavaScript, PHP or CSS required file for the component to work.
- the info.xml file, containing the metamodel information (cf. Figure 3.3).

The XML structured format will help us represent this Component in every aspect. The  \(<feature>\) tag will involve several child tags (cf. Figure 3.3):

- Each of the client side elements that are necessary are embedded completely in a  \(<clientlogic>\) tag; inside it, a  \(<name>\) tag will tell us the JavaScript file name.
- The same is done for the PHP logic that needs to be exported, placing it into a  \(<serverlogic>\) tag;
- Also for the style rules we have a tag: the  \(<styles>\) tag.
- The  \(<include>\) tag will define which file has to be included into the theme code in order to execute and show this component.
CHAPTER 3. X-THemes Metamodel

Figure 3.2: A diagram presenting a broken down structure of the X-Themes metamodel. The reader can see elements that are planned to be migrated between themes (marked in red) and elements that have to be exported as a consequence (marked in blue).

Figure 3.3: The XML representation of a Component, which involves logic as well as markup and style rules.
• The `<component_type>` tag will define if this is a generic imported component (with the `undefined` value) or a WordPress custom header or sidebar.

• The `<functionspointer>` tag will define PHP files that have to be included directly in the `functions.php` file of the theme. This involves files such as Sidebar or Custom Header registration calls.

• `<ziptostring>` tag specifies a temporary sub-directory to store this Component into the theme generator.

### 3.3.2 Layout Components

This type of component represents the layout structure of the theme. These components are simply included by the page logic. They are divided into three types: It is a specialization of the generic Component and does not include JavaScript or generic PHP code. It may only include static content (images, markup, text) or dynamic theme elements (cf. Chapter 2), which are handled separately by the XDE tool.

• **Container**: generic block that can contain other Containers.

• **Header**: the website header; it can contain other Containers. Generally exporting a layout component means that its content has to be reused into another theme.

The reason why header and footer are separate components is that they are special containers which have to be reused throughout all the pages of a theme. This can also improve the X-Themes metamodel adaptability in a multi-page theme generator. Moreover, in the WordPress theme Header and Footer are natively separate files (respectively `header.php` and `footer.php`), therefore this separation was required for the metamodel.

Each Layout Component is stored in a separate directory in the theme files. Each directory contains:

• a PHP file that contains the component’s code.

• a CSS file that contains the component’s style rules.

• a XML file that contains the metamodel information used to generate the component (info.xml).

If the exported component has nested components, they have to be generated as-is while replacing the inner components with the inclusion of the corresponding Layout Component PHP file.

The XML file follows this structure (cf. Figure 3.4): A `<layoutcomponent>` tag includes all the declaration. This tag has three attributes: `name` which is the generated name of the Layout Component; `type` which determines if the Layout Component is a header, a footer or a generic container; `lcname` which is the original name of the Layout Component, as specified by the user when he uses the X-TG tool.

The component we can see in Figure 3.4 contains a `<dependency>` tag. This tag means that this component requires another file to work correctly; for example, a *background image*. A Layout Component may also contain dynamic theme elements, such as The Loop, the website title or description (cf. Chapter 2).
3.3.3 Theme

We have defined all the basic elements of our metamodel; we can now define a theme in our metamodel as a nested collection of these elements. As we can see in Figure 3.5, all the elements of the generated theme are present in this representation, and the nested structure gives the chance to infer which component is containing another. This definition is used during the theme generation algorithm and is reported here for completeness. In the example, a theme called Name was created. The <theme> tag requires the name attribute. Moreover, we can see another tag, the <page> tag, that was placed in order to be able eventually to support more design outputs for different WordPress pages (cf. Chapter 2).

3.4 Metamodel DOM implementation

In order to speed up and create a better understanding of the examples in the browser during the designing by example scenario, some key aspects of the XML metamodel have been also brought into the DOM. Exploiting HTML5 dataset attributes [10] and classes we have been able to create a metamodel-powered DOM that could graphically assist and make possible the design of the XDE tool.

More in detail, the DOM of an X-Theme has been extended in the following way:

- **Layout Components**: these components have been identified in the DOM and marked with a class called layoutcomponent.

- **Components**: these components have also been identified in the DOM using a class called feature; moreover, the XML metamodel has been mapped into some data attributes (exploiting the dataset HTML5 attribute) as we can see in Table 3.1 (cf.
24

3.5. CONVERTING A WORDPRESS THEME TO AN X-THEME

Figure 3.6: The resulting DOM of a Component node. The metamodel has been mapped into the HTML5 dataset attribute.

<table>
<thead>
<tr>
<th>XML Metamodel Tag</th>
<th>Dataset attribute</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;clientlogic&gt;</td>
<td>data-jsfiles</td>
<td>JavaScript files</td>
</tr>
<tr>
<td>&lt;serverlogic&gt;</td>
<td>data-phpfiles</td>
<td>PHP files</td>
</tr>
<tr>
<td>&lt;styles&gt;</td>
<td>data-cssfiles</td>
<td>CSS styles</td>
</tr>
<tr>
<td>&lt;functionpointer&gt;</td>
<td>data-fps</td>
<td>PHP files registering Sidebars or Custom Headers</td>
</tr>
<tr>
<td>&lt;dependency&gt;</td>
<td>data-otherfiles</td>
<td>Dependencies</td>
</tr>
<tr>
<td>&lt;component_type&gt;</td>
<td>data-comptype</td>
<td>Component type</td>
</tr>
<tr>
<td>&lt;include&gt;</td>
<td>data-includepath</td>
<td>PHP included file</td>
</tr>
</tbody>
</table>

Table 3.1: A table describing the mapping between the XML metamodel and HTML5 Dataset attributes for the XDE tool.

Thanks to this mapping, the XDE JavaScript logic can easily access the information expressed by the metamodel, having at the same time still valid HTML code; this would not be true if we had mixed HTML and XML code.

3.5 Converting a WordPress theme to an X-Theme

In this section we will analyze the process to convert a normal theme to an X-Theme. The only requirement is that the X-Themes plugin should be installed into the WordPress installation that runs an X-Theme. We will now introduce the first step to make the theme support the X-Themes tools, and then we will try to understand how to convert each type of element defined in our metamodel.

3.5.1 First steps

The first thing to do to convert a theme into an X-Theme, is to include the designing by example logic and style auxiliary rules into the theme. To do so, one must open the themes header.php file and add this in the <head> tag:

```php
<?php if($_GET["design"]) {  
    <script type="text/javascript" src="<?php echo plugins_url("xthemes/ui/js/DADAuxiliary.js"); ?>"></script>
```
This code will start the execution of the XDE JavaScript logic into the theme, but only if the user wants to see the theme in an assisted design mode that we will present in detail in the next chapters. Otherwise, the user will see only the theme as it is, without any XDE logic running in the background.

### 3.5.2 The conversion process

Generically, the conversion process from a WordPress theme to an X-Theme involves six major operations (which can be executed in a different order than the following):

- **Identification.** The component to be converted has to be identified in the DOM of the original theme.

- **Wrapping.** Each Component or Layout Component is wrapped into two styled DIV tags which enclose the content of each element. An inner DIV encloses the content, while a wrapper DIV encloses the inner DIV.

- **Comment enclosing.** Some parts of the X-Theme should be enclosed into some sentry comments. This step executes the enclosure. The possible comments the XDE will understand are the following, which enclose different elements:

  ```html
  <!--namesite-->
  <!--/namesite-->
  <!--descrsite-->
  <!--/descrsite-->
  <!--loop-->
  <!--/loop-->
  <!--verticalnav-->
  <!--/verticalnav-->
  <!--feature name="NAME"-->
  <!--/feature-->
  <!--layoutcomponent name="NAME" type="TYPE"-->
  <!--/layoutcomponent name="NAME"-->
  ```

- **File organization.** Each part of an X-Theme is stored in a separate subdirectory, which only contains the file requirements of the part. Therefore, all the parts are kept separate.

- **Inclusion.** Each part is included within the theme with a PHP `include()` call. Nested components are included by parent components in the same way.

- **XML creation.** The XML notation for the component should be created into the part’s subdirectory.
Converting a logic Component is slightly more complicated than converting a Layout Component. In this example, we will try to convert the navigation bar from TwentyTwelve, the latest default theme in WordPress. Therefore we perform the **identification** step; the original code will look as follows in `header.php`:

```php
<nav id="site-navigation" class="main-navigation" role="navigation">
  <h3 class="menu-toggle"><?php _e( 'Menu', 'twentytwelve' ); ?></h3>
  <a class="assistive-text" href="#content" title="<?php esc_attr_e( 'Skip to content', 'twentytwelve' ) ?>"><?php _e( 'Skip to content', 'twentytwelve' ) ?></a>
  <?php wp_nav_menu( array( 'theme_location' => 'primary', 'menu_class' => 'nav-menu' ) ); ?></nav>
```

Now we perform the **file organization** step: let us create a directory called `index` in the theme directory (the component is in the main index page). In this directory, we create a subdirectory with our component name: for this example, we choose `twentytwelve_nav`. In the `index/twentytwelve_nav/` directory, we create two files:

1. The file `twentytwelve_nav.php` will contain the original code (the file has to start with a `<?php` tag, so we close it right after):

   ```php
   <nav id="site-navigation" class="main-navigation" role="navigation">
   <h3 class="menu-toggle"><?php _e( 'Menu', 'twentytwelve' ); ?></h3>
   <a class="assistive-text" href="#content" title="<?php esc_attr_e( 'Skip to content', 'twentytwelve' ) ?>"><?php _e( 'Skip to content', 'twentytwelve' ) ?></a>
   <?php wp_nav_menu( array( 'theme_location' => 'primary', 'menu_class' => 'nav-menu' ) ); ?></nav>
   ```

2. We make an exposed copy of this code by creating a file with the same name, but with the `.code` extension. So we create the `twentytwelve_nav.code` file. This file has to contain the exact content of the corresponding PHP file, without the starting `php` tag:

   ```
   <?php
   <nav id="site-navigation" class="main-navigation" role="navigation">
   <h3 class="menu-toggle"><?php _e( 'Menu', 'twentytwelve' ); ?></h3>
   <a class="assistive-text" href="#content" title="<?php esc_attr_e( 'Skip to content', 'twentytwelve' ) ?>"><?php _e( 'Skip to content', 'twentytwelve' ) ?></a>
   <?php wp_nav_menu( array( 'theme_location' => 'primary', 'menu_class' => 'nav-menu' ) ); ?></nav>
   ```

3. We now put the style rules for this component in a `.css` file. The file can have any name. Let it be `tt_nav.css`. Let us save this file in the `/index/twentytwelve_nav/` directory as well.

4. We finally have to perform the **wrapping** operation on the code in the `header.php`, which originally contained the code navigation bar from TwentyTwelve. The wrap-
ping shall be executed using our X-Themes wrapping technique. The ID we will give to the wrapping div is `xt_nav`:

```html
<div id="wrapper_xt_nav" style="" class="dadwrapper">
  <div id="xt_nav" style="" draggable="true" class="adaptContent selectable tgdadarea draggable feature"
    data-phpfiles="twentytwelve_nav.code"
    data-featurename="twentytwelve_nav" data-cssfiles="tt-nav.css"
    data-comptype="undefined"
    data-includepath="twentytwelve_nav.php"
    data-zippath="twentytwelve_nav" data-feature="yes"
    data-localpath="<?php echo get_template_directory_uri();
    ?>/index/twentytwelve_nav">
    ORIGINAL_CODE
  </div>
</div>
```

This wrapping follows the mapping of the metamodel explained in Section 3.4. The last operation to perform in this step is to give the minimum required CSS rules to the wrapping DIVs, which are the following.

```css
#wrapper_ID {
  float: left;
  border: 0px;
  margin-right: -100%;
  overflow: hidden;
  padding: 0px;
}

#ID{ 
  position: relative;
  margin-top: 0px;
  margin-left: 0px;
}
```

These rules should be inserted into the style of the theme or of the parent Layout Component (if there is one). Every eventual customization to the position of the element should be done on the ID rules. Please note that if there are other PHP, JavaScript or CSS files to be included, they have to be inserted into the Component’s directory as

![Figure 3.7: The current contents of the directory of the converted twentytwelve_nav Component.](image)
well, which is `/index/twentytwelve_nav/` in our example. The directory of the converted navigational bar now looks as we can see in Figure 3.7. Please also note that for each `.php` file there has to be the corresponding exposed `.code` file.

5. We now have to perform the **inclusion** step and change the **ORIGINAL CODE** in the `header.php` file with an inclusion (a PHP `include()` to the newly created PHP file. Also we need to perform the **comment enclosing** add the wrapping comments. The code changes as follows:

```php
<?php
include(get_stylesheet_directory()."/index/twentytwelve_nav/twentytwelve_nav.php");
?>
```

6. Now, in the theme `header.php` file, the CSS style declarations of this component have to be included with the `<link>` tag:

```html
<link rel="stylesheet" type="text/css" href="<?php echo get_template_directory_uri();?>/index/twentytwelve_nav/tt-nav.css"/>
```

7. In the end, we now create an `info.xml` file in the `/index/twentytwelve_nav/` directory; please note that the XML notation should be equivalent to the one specified in step 4. The mapping between the dataset attributes and the XML metamodel notation has been presented in Section 3.4. This realizes the **XML creation** step.

If everything was executed correctly, the theme navigation bar from TwentyTwelve should be reusable thanks to the XDE tool.

### 3.5.4 Converting a Layout Component

The steps to convert a layout element of your theme to a Layout Component are the following:

1. The first step is the **identification** step; let **MARKUP CODE** be the content code we identified and we want to convert to a Layout Component. We have now to perform the **wrapping** operation and wrap the Markup HTML code with two divs. Let ID be the identifier of the component we want to create, we wrap our markup code as follows:
To finish the wrapping operation, we create a new file: style.css that contains the following rules, in order to properly position the newly added DIVs. The style that has to be put in the css file is the same as the one specified in the Section 3.5.3 during the wrapping step.

2. As we said, MARKUP CODE is the code we want to be reusable. If this code contains dynamic elements that have to be re-interpreted such as the website title, or the description, we have to wrap the element with a sentry comment. Therefore we go on with the comment enclosing step. Let us assume that the code we want to wrap is the following, which displays the website description:

```html
<h2 class="site-description"><?php bloginfo( 'description' ); ?></h2>
```

Here we see that there’s the description to be wrapped, so we wrap it with a HTML sentry comment.

```html
<h2 class="site-description"> <!--descrsite--><?php bloginfo( 'description' ); ?><!--/descrsite--></h2>
```

We can now put this code into a theme subdirectory, going for the file organizing step. We assume that the component is in the index page of the theme. We create the subdirectory: index/ID where ID is the identification of the Layout Component. We create a file called container.php that contains the MARKUP CODE showed before, plus its parent div.

Now, container.php contains this code:

```html
<div id="ID" draggable="true" class="selectable adaptContent
draggable layoutcomponent ui-resizable">
<h2 class="site-description"> <!--descrsite--><?php bloginfo( 'description' ); ?><!--/descrsite--></h2>
</div>
```

Now we perform the inclusion step and replace the div containing the MARKUP CODE with a PHP include() directive to:

```html
<?php include(get_stylesheet_directory()."/index/ID/container.php"); ?>
```

In the end, we terminate with the XML creation step and create the info.xml file as shown in Figure 3.4 which should be very easy.
In this chapter, we will present the XDE tool which can provide the designing by example approach to the end users of the WordPress platform. The XDE tool was realized as an extension of the X-TG, which was realized as a WordPress plugin (cf. Chapter 2). In the next sections, we will analyze the major concepts and assumptions that were required for the implementation of the XDE tool, as well as providing implementation details such as the architecture of the tool, the reuse algorithm and some technical limitations that were encountered.

4.1 Concepts

In this section we will present the use case scenario for the XDE tool, and how it relates to the underlying X-TG. Moreover, we will present the underlying assumptions the XDE tool relies upon.

4.1.1 Use case scenario

In this section we will analyze a typical use scenario which involves the designing by example approach as conceived by the XDE tool. As we already said in Chapter 2, it is important for the reusability of the examples that the users may browse them into a theme gallery. This gallery should be browsable by functionality rather than the look and feel of the theme itself. An example scenario might be:

"Mark is creating a WordPress website for his restaurant. He needs a fancy menu which gives the opportunity to browse the pages of his WordPress blog. He opens the X-Theme gallery and searches for X-Themes which provide a page navigation menu. The gallery shows him a list of X-Themes which provide a page
32 4.1. CONCEPTS

Figure 4.1: A mockup of a gallery browsing X-Themes.

Figure 4.2: A detailed view of the browsing of previews of each component.

navigation menu, and he is able to look at the preview of each of the elements present in each of the X-Themes found during the search” (cf. Figure 4.1).

Thanks to this gallery, the user can find examples they can reuse and look at their preview; moreover, they can slide between the complete preview of the X-Theme and each component which is present in the theme, so they can look at it more in detail. We can better see a mockup of the planned interaction as conceived in the scenario in Figure 4.2.

When the users are satisfied of the look and feel they can perceive from the previews, they can open the X-Theme from which they want to take the example from. The scenario then goes on as follows:

Mark finds a suitable navigation menu in one of the found X-Themes. He clicks on the X-Theme and therefore he opens the WordPress website running the corresponding X-Theme. Though, he can see a different perspective of the X-Theme: each reusable component is highlighted as he passes over it with the
Thanks to this view, the user can understand which components can be reused with a drag and drop interaction. At this point, the user needs a spot where he can drop the component into. This spot is realized by the design interface of the X-TG, in which the user can generally design his own X-Theme from scratch. Thanks to the XDE tool, he can drag the component and drop it into his preferred location in the X-Theme he is designing into the X-TG.

Moreover, after he performed the drop operation, he can see the component running into the X-TG (cf. Figures 4.4 and 4.5), realizing a WYSIWYG interface. In addition, considering the example of the WordPress pages, the dropped component should display the pages of the WordPress installation of the user, and not of the source X-Theme. More in general, the component has to be re-executed considering the contents of the user’s WordPress website. Therefore, the user scenario ends in this way:

Mark moves the cursor over the page navigation menu and starts the drag operation. Then he drags the menu inside the X-TG design interface and drops it, releasing the mouse button. Now he can see the menu executed in his WordPress website and contained into the X-TG interface.

Therefore, thanks to a single drag and drop operation, the user is capable of reusing Layout Components or Components (cf. Chapter 3) along with their dynamic behavior (JavaScript, PHP) and style rules. In the case of the reuse of Layout Components, the user might also choose if he wants to keep the style as he looked it into the source X-Theme, or not.

We can provide another example of the reuse operation with Figure 4.4: we can see a menu as seen by the user in the source X-Theme. When he performs the drop operation, he can see the resulting functionality show his own data, as shown by their WordPress installation (cf. Figure 4.5).
4.1. CONCEPTS

Figure 4.4: A navigation menu powered by PHP, JavaScript and CSS code. This is the menu the user can see while visiting an X-Theme. The menu entries are the WordPress’ pages.

Figure 4.5: The finished result of the reuse operation of the navigational menu. The component been executed in the user’s WordPress installation, showing their pages.
4.1.2 Component isolation

One of the most important assumptions we make in the process of reuse in the X-Themes project is that each Component should be self-contained: every necessary rule or code should be specified within the Component itself and no side-effects should spread from a Component to the others. This is a key assumption which can be explained with some trivial examples. First of all, we should make clear that there are two mechanisms that a Component can use to harm the surrounding elements: JavaScript and CSS rules. More in detail, we can provide two examples of side-effects that would damage completely the rendered theme. The first one is a CSS rule:

```css
body { display:none; }
```

This is a trivial code that a Component may enclose and which may easily damage the display of the surrounding components. The second one is a jQuery JavaScript code which can easily destroy the page:

```javascript
$("*").remove();
```

This code would iteratively go in each DOM element of the page, removing it. A Component enclosing this JavaScript code would easily damage the display of surrounding components.

4.2 Design Process

The tool proposed in this thesis was developed concurrently with the X-TG tool [5], which it is based upon. Moreover, the general workflow of the reuse implies that there should be correctly generated X-Themes before the designing by example approach can be applied to some X-Theme Component (cf. Chapter 3). Therefore, we had to put a great effort in creating a workflow between the two tools that could support concurrent development. Since the development started after the X-Themes metamodel (presented in Chapter 3) was formalized, we both had a common base on which we could start realizing a prototype. The first tool which we developed to support the reuse, and which can be considered the precursor of the XDE tool, was a more generic drag and drop tool powered by the X-Themes metamodel, which was capable of mixing Layout Components, CSS style rules, in a non-visual way. This means that the user was not able to directly see what the generated result would have been. Though, in order to be able to mix Layout Components, we first had to split some themes into Layout Components which could be mixed. We chose TwentyEleven and TwentyTwelve, which are the default WordPress themes, since they contain an easily separable layout.

Each Layout Component separated was stored into a repository, which contained the Layout Component’s code, the CSS rules and the XML metamodel file (cf. Chapter 3). We can see a sample interface of the first tool in Figure 4.6. In the upper areas, the user could see previews of the Layout Components they wanted to drag; finally, they could drop it into the lower area, which was divided in three parts: a header area, a body area, and a footer area. It can be easily understood that this first prototype was not WYSIWYG. If the Layout Component which was being dragged could contain more inner elements, the drag and drop interface provided more nested droppable spots in which the user could drag whatever they want. To do so, the X-Themes metamodel previously provided a key attribute called hotspots that stored how many inner components a Layout Component could keep. This attribute was removed later
36

4.2. DESIGN PROCESS

Figure 4.6: The first tool created to mix parts of themes.

on.

Basically, after the user chose what they wanted to mix, they could press a generation button which performed some nested PHP inclusions corresponding to the hotspots in each of the Layout Component code. Moreover, it created a new, bigger CSS style declaration which imported (with a @import rule) the CSS style declarations of every Layout Component mixed together. We named it the mixing engine in a first moment.

We can see a result of a mix created with this prototype in Figure 4.7.

Thanks to the mixes created with this first prototype, we first came in contact with the main issue in mixing parts of different themes: collisions in the CSS style rules. Since the two mixed themes contained DOM elements which were styled through CSS files, it happened that the themes were using the same IDs for many DOM elements. Therefore, the mixed CSS rules were colliding on these IDs: only the last applied rule wins. Though, it is not always true that the last applied rule is the one containing the intended style for the element mix. Therefore, we had to find a way to mix the styles of different parts of themes which could also be contained in a sandbox that did not influence the other components; some way of limit the scope of the CSS rules of a Layout Component to that component only. Thanks to a new HTML5 feature named scoped CSS\(^1\), we made a try mixing TwentyEleven and TwentyTwelve with this technique; unfortunately, the feature is still highly experimental and therefore cannot be guaranteed to work correctly in complex mixes. Unfortunately there was no other way to assume that each of the Component was completely isolated, as specified in Section 4.1.2. Luckily, the X-TG project could reuse the mixing engine to extend it, in order to build the back-end component which now aggregates the pieces generated by the X-TG theme generator.

\(^1\)http://css-tricks.com/saving-the-day-with-scoped-css/
In the meantime, a WYSIWYG tool was developed to generate themes (X-TG); therefore, we could start the development of the reuse algorithm in a WYSIWYG way, which involved the X-TG tool as well. Since the first metamodel element which could be generated with the X-TG was the Layout Component, this was the first element which was targeted as reusable during the development of the XDE tool. Therefore, we had to understand how to bring the code and the style of the Layout Components from a source X-Theme into the X-TG, caring also for the content which had to be shown: the data shown into the reused Layout Component had to change from the one displayed in the source X-Theme to the one displayed into the user’s WordPress installation (cf. Section 4.1). More details on the reuse algorithm are presented later on in this Chapter.

While we were developing the reuse algorithm for the XDE tool, the X-TG developed the capability to introduce functionalities (file by file) into the X-Themes. These functionalities are called Components (cf. Chapter 3) and contain dynamic behavior such as JavaScript or PHP. Therefore, after we finished creating the reuse algorithm for the Layout Components, we could focus on reusing Components (along with their JavaScript or PHP code and style rules). An example of Component could be a navigation menu, which we can see in Figure 4.4. Like Layout Components, we have to display the correct data once the Component has been reused also for the Components. Therefore, we had to develop a reuse algorithm which could also fetch JavaScript, PHP, CSS and other files and re-execute them into the X-TG environment. A result of this workflow can be seen in Figures 4.4 and 4.5: in the first picture we can see the original menu before the reuse operation; after the user had dragged and dropped it into the X-TG interface, it was re-executed and it is correctly showing the user’s contents, which
38

4.3 Implementation

We developed the XDE tool on the top of the X-TG tool which is following the same metamodel (cf. Chapter 3) to generate X-Themes. Both XDE and X-TG have been developed on the top of the WordPress platform (version 3.5.1) as a working WordPress plugin. This means that X-TG and XDE can be installed together with the basic plugin mechanism that WordPress already provides. Therefore, X-TG and XDE have both been built using various technologies: PHP for the server side and JavaScript/jQuery, AJAX, CSS3 and obviously HTML for the client side. Now that we have identified each component in the system and which is its responsibility, we can explain more in detail how the reuse of the examples works.

4.3.1 Software architecture and subsystems

The XDE tool is divided in four major components (cf. Figure 4.8):

- a CSS/JavaScript powered design interface which is loaded by the browser and executed (also called XDE-DI).
- a JavaScript logic component which is loaded by the browser and executed when the X-Theme’s design interface is opened (also called XDE-LT).
- a JavaScript logic component built upon the X-TG (also called XDE-LG).
- a PHP logic module stored in the X-TG plugin (also called XDE-CF for now on).

Figure 4.8: The deployment diagram of the XDE tool.

can be seen in the second picture. More details about the reuse algorithm for the Components are provided later in this Chapter.
We will now describe the responsibility of each of these components separately and how they relate to each other.

**Theme design interface**

This assisted interface (XDE-DI) will exploit the metamodel DOM implementation (cf. Section 3.4) to assist the user in their reuse of examples. The major responsibility of this interface component is to provide an alternative view of the theme in which the user can understand the separation between the components of the page and understand which parts of the theme they are going to eventually reuse. More in details, the user can use their mouse to explore the components in the page, seeing that the pieces they are hovering with their cursor are going to be highlighted (cf. Figure 4.9).
Theme and XTG JavaScript counterparts

These two counterparts handle the data transfer of the components that have to be reused, realizing the communication between the assisted design view that the user is browsing and the X-TG theme design interface. More in detail, the reuse algorithm (which will be presented later) is divided in two major sections: the first section regards bringing all the necessary information from the X-Theme into the drag event, and the second is to use this information inside the X-TG. The two counterparts are the following:

- Theme counterpart (XDE-LT): its responsibility regards handling the drag of the elements when the user starts dragging some part of a theme. It takes all the information and content from the viewed theme and stores it into the drag event. It is loaded by the browser when the theme page is loaded.

- X-TG counterpart (XDE-LG): its responsibility is to handle the drop operation of the component into the X-TG. Depending on the type of the component being dragged, the reuse algorithm will behave differently. It will perform some conflict-detection and handling operation if necessary. It is loaded by the browser when the X-TG is loaded.

Each of these counterparts are developed in JavaScript/jQuery. They communicate between each other thanks to the `dataTransfer` object\(^2\) that allows us to store all the required information to grab the X-Theme parts and to reuse them into the X-TG. Moreover, the X-TG counterpart communicates with another component, the Component Fetcher, that is involved in case the reused piece is a Component (cf. Chapter 3).

Component Fetcher

This component (XDE-CF) has a major impact in the reuse of specific theme parts called Components. These parts enclose also logic elements such as JavaScript and PHP (cf. Chapter 2). This component has the responsibility to receive AJAX requests from the XDE-LG in case of the drag event of a Component, and copies all the required files to make the components work into the theme generator, creating a running clone of the Component in the X-TG interface. Moreover, other than copying the required code to make the Component work, it also dynamically executes it into the X-TG WordPress installation, giving the chance to the user to see the execution of the Component inside his website.

For example: the user finds an attractive example of a navigation menu (cf. Figure 4.4) displaying the pages of the WordPress website. If the user drags this menu into the X-TG, the Component Fetcher will:

1. Grab all the required JavaScript, CSS, PHP files and dependencies required for the Component to work from the source X-Theme.

2. Execute the PHP code into the X-TG installation.

3. Give the result back to the XDE X-TG counterpart.

---

\(^2\)http://www.w3.org/TR/2011/WD-html5-20110113/dnd.html
In this way, the user performing the operation will see the pages of their installation into the X-TG (cf. Figure 4.5); in the example provided, the user’s installation only contains one page called "Sample Page".

4.3.2 Theme Preparation

When it is generated, an X-Theme is automatically supporting the assisted design interface XDE-DI. More in details, in the head section of the theme, the following code is put by the X-TG:

```php
<?php if($_GET['design']){
    <script type="text/javascript" src="<?php echo plugins_url("xthemes/ui/js/DADAuxiliary.js");?>"></script>
    <script type="text/javascript" src="<?php echo plugins_url("xthemes/ui/js/DEDDADTheme.js");?>"></script>
    echo '<link rel="stylesheet" href="'.plugins_url('xthemes/ui/css/dbedesign.css')." type="text/css" /> ";
} ?></php
```

This slice of code represents the insertion of the XDE-DI, accessible using the ?design=true suffix after the URL of the WordPress website. More in detail, it imports the CSS rules assisting the user in highlighting the various theme elements (cf. Figure 4.9) and the JavaScript logic of the XDE-LT component (cf. Section 4.3.1).

4.3.3 Reuse algorithm

We will now detail the process behind the reuse of each type of component presented in our metamodel. The reuse process exploits the `dataTransfer` object as well as the `ondragstart` event. Just after the drag operation starts, every CSS style declared to create the assisted design view XDE-DI is being disabled; the reason will be explained in the next sections. For each of the elements we are dragging away from an X-Theme, the XDE-LT stores into the `dataTransfer` object the following information:

- The ID of the dragged component.
- The string representation of a clone of the dragged DOM subtree.
- The coordinates of the drag event relative to the dragged component.

Depending on the type of the element dragged away from the X-Theme, some more data is being stored inside the `dataTransfer` object. When all the necessary data is stored inside the `dataTransfer` object, the XDE-DI CSS rules are re-enabled, to assist the user in dragging further components. We will now investigate which information are transferred in every case, and how these information are used into the XDE-LG counterpart.

**Layout Components**

In the case of Layout Components, an additional operation is done just before the component (that we will call A) gets cloned; all the style rules which apply to a DOM element in the
4.3. IMPLEMENTATION

subtree of $A$ are transformed to an inline style declaration in the style attribute of the corresponding element (this technique has also been used by Kumar et al. [8]). Then the node gets cloned and its string representation is stored in the dataTransfer object. For this reason, before the CSS rules are applied, the XDE-DI CSS rules should be disabled; we will re-enable them at the end of the drag operation. Moreover, other information are stored, such as eventual background images.

The entire dataTransfer object is bringing the information to the XDE-LG counterpart, which breaks down the object forming a parameters array with all the fields set in it. The cloned representation of the DOM subtree of $A$ (along with provided inline style) gets filtered using regular expression, searching for dynamic theme elements (cf. Chapter 2) in the cloned Layout Component. The filter searches for wrapping comments such as:

```
<!--namesite--> <!--/namesite-->
<!--descrsite--> <!--/descrsite-->
<!--loop--> <!--/loop-->
<!--verticalnav--> <!--/verticalnav-->
```

Each of these HTML-style comment enclose some dynamic theme element, such as the name of the site, the description or The Loop. Thanks to this filter, each of these dynamic theme elements get overwritten with the corresponding theme element from the WordPress installation where the X-TG tool is installed into. After that, the filtered string representation gets pasted into the X-TG interface (where the user has dropped it) also thanks to the coordinates of the original drag event stored in the dataTransfer object.

Components

In the case of Components (which involve JavaScript and PHP logic) much more information has to be stored. Referring to Section 3.4 for all the attributes present in a Component, the ondragstart event will store in the dataTransfer object also the following fields from the metamodel DOM implementation (cf. Section 3.4):

- the data-featurename field.
- the data-comptype field.
- the data-jsfiles field.
- the data-phpfiles field.
- the data-cssfiles field.
- the data-otherfiles field.
- the dynamically generated data-localpath field which contains the URL to the Component’s HTTP accessible directory in the original X-Theme.
- the data-comptype field.
- the data-fps field.
- the data-includopath field.
• the data-zippath field.

The next step involves the drop event in the X-TG interface, where the XDE-LG counterpart begins its actions. We will now present the steps more in detail:

1. It begins the load of the Component’s CSS files into the X-TG interface (if present), in order to correctly show the component after it will be rendered in the X-TG interface;
2. Builds an AJAX request for the XDE-CF transferring the content of the data-jsfiles, data-cssfiles, data-phpfiles, data-otherfiles, data-zippath, data-fps, data-includepath, data-localpath attributes of the Component;
3. Starts the AJAX request.

The reason why an AJAX request is built and sent to the XDE-CF is that the components may involve WordPress related content that has to be shown, and therefore it has to be executed in the WordPress installation, to show the correct data (e.g. Figures 4.4 and 4.5).

The XDE-CF receives this data and creates a temporary directory in a cache repository with the name specified in the zippath parameter. Then it builds the source fully-qualified URLs to fetch every required file (JavaScript, PHP, CSS or general dependency). Please note that it would be impossible to fetch the PHP code pointing to .php files; that’s why we need to expose the PHP code using .code files which have the same content as the corresponding .php file.

We build the path concatenating the localpath attribute and each filename present in the includepath, jsfiles, phpfiles, cssfiles, fps or otherfiles attributes. Every file is then saved in the zippath directory, where the X-TG can use it to generate an X-Theme. The XDE-CF then uses the data-comptype field to generate the function pointers we defined in our metamodel to register the Component in case it is a WordPress sidebar or custom header. At this point, the code presented in the cached copy of includepath (involving also all the inclusions to eventual other PHP files in the Component) gets evaluated with the PHP eval() code, and its result goes back as a response of the AJAX request. Some <script> tags referring to the fully-qualified fetched files specified in the jsfiles attribute are concatenated to the output. Only now the XDE-LG receives the Component’s evaluated output and prints it into the dropped container. Therefore, the result is that we can see the working Component inside the X-TG interface, evaluated in the context of the user’s WordPress installation. An instance of this operation is given in Figure 4.4: the example in a X-Theme the user is looking at is a navigation bar; they can drag the example into the X-TG interface. After the evaluation is done, the user can see the result, as shown in Figure 4.5.

4.3.4 Technical limitations

In this section we will present some technical limitations encountered during the development of this reuse algorithm. More in detail, we will explain some open issues with CSS inference algorithm we use for the Layout Components and the performance threats to the Component reuse procedure.
44 4.3. IMPLEMENTATION

CSS style inferencing issues

As we said in the previous sections, before a Layout Component (named A) gets cloned in the ondragstart event, we have to navigate all the DOM elements in A’s DOM subtree and verify if there are any CSS rules applied to each element; if yes, the style rules from the CSS have to be applied as an inline style. After that, A can be cloned and transferred from XDE-LT to XDE-LG through the dataTransfer attribute. Unfortunately, we had to make a trade-off between getting the computed style\(^3\) of the element and having to write a CSS inference engine from scratch. The reason why the computed style was not applicable is that it was storing a huge amount of information into the inline rules, slowing the algorithm and making the re-styling in the X-TG very hard. Therefore, we used the following JavaScript functions to implement the trade-off:

```javascript
function css(a){
    var o = {};
    var rules = window.getMatchedCSSRules(a.get(0));
    for(var r in rules) {
        o = $.extend(o, css2json(rules[r].style),
                    css2json(a.attr('style')));
    }
    return o;
}

function css2json(css){
    var s = {};
    if(!css) return s;
    if(css instanceof CSSStyleDeclaration) {
        for(var i in css) {
            if((css[i]).toLowerCase) {
                s[(css[i]).toLowerCase()] = (css[i]);
            }
        }
    } else if(typeof css == "string") {
        css = css.split("; ");
        for (var i in css) {
            var l = css[i].split(":
            s[l[0].toLowerCase()] = (l[1]);
        }
    }
    return s;
}
```

This functions are applied to each element in A’s subtree and thanks to the window.getMatchedCSSRules() call, we can exploit the browser’s functions to understand which CSS rules are applied to each DOM node in A’s subtree and transform them as inline style rules. Unfortunately, with this technique there is a major issue: a CSS rule that has page-wide effects (such as a color rule on the body tag) cannot be converted to inline style because the getMatchedCSSRules() function fails to identify such rule as one applied to an element of the page (other than body). Therefore, the assumption that Component provokes no side-effects becomes even more important.

\(^3\)http://www.w3.org/2003/01/dom2-javadoc/org/w3c/dom/css/ViewCSS.html
Performance threats

The Component reuse algorithm involves several file caching from remote websites; this might be really slow with increasing complexity of the Components. More in detail, the XDE-CF performs a `file_get_contents()` call for each file that has to be fetched from the source Component. The performance of the call depends on the load and efficiency of the two web servers involved: the one hosting the X-TG, the other hosting the reused example. A better implementation that could be considered might be the caching of a single archive containing all the files instead of making multiple connections, one for each required file.
In this Chapter we will present the planning and the results obtained from an user study conducted on both X-TG and XDE tools. The user study was done on both the tools to save time and verify the difference perceived by the users between the reuse concept provided by the Component separation and the designing by example approach. Therefore, the participants joining the study had to use the tools and apply reuse techniques and then had to fill a questionnaire on the Google Drive platform. Thanks to this questionnaire we have collected opinions and data about the tools. The results show a good acceptance of the designing by example approach, which allowed the users to easily create a mix of two themes feeling efficient at the same time, even if they had no knowledge of the WordPress platform. They neither had knowledge of the underlying metamodel, which is completely transparent to the user. The designing by example work flow can also be considered more error-proof than the one proposed by X-TG: the time variance of the task which involved XDE is sensibly lower compared to the ones which involved only X-TG. For the same reason, the XDE tool can be considered less struck by the learning approach while can be considered as easy as the other approaches since it requires the same amount of time (or sometimes even less).

5.1 Planning

The user study was designed on the top of the X-TG tool. Moreover, in order to better evaluate the quality and the efficiency of the approaches proposed to the users, we have defined three tasks which had to be carried out by the participants. Each task involved the creation of a WordPress theme which contained at least the following elements:

- A header with Website title and description.
- A background image.
- A Javascript/PHP custom functionality with custom CSS.
• A footer (with text and/or pictures).
• The Loop.

Each of these elements had to be styled correctly and placed following some mockups which had to be reproduced (cf. Figure 5.1). Therefore we can say that the three tasks were conceived to be similar in complexity as they contained these specific elements. Therefore, we wanted to verify the difference between the time needed to complete each task while varying the goal or the approach used to create the themes. Thus, we designed the three tasks depending on the goal to pursue or on the approach which the users had to apply. We can distinguish two different goals:

• Create a new theme from scratch;
• Mix parts from two different themes.

Moreover, we can distinguish two proposed approaches:

• Designing a theme with no examples to reuse (only the X-TG tool involved).
• Designing a theme reusing other examples from other themes (the XDE tool involved as well).

We asked 12 participants to execute three tasks which will be detailed in the next sections. The task order was presented in all the possible permutations to avoid any learning biases or different biases coming from the difference between the two approaches. The number of the participants was determined to have at least a pair of tests for each possible permutation of the task order. We recruited the participants between the members of a research group in computer science at the ETH Zurich, and also computer science students. During each task, the evaluator was noting the starting and the ending time of each task, as well as notes such as user mistakes. After each task was completed, the users could see the resulting X-Theme running into the WordPress installation in which they generated the theme, as well as using the Components they inserted into the theme itself.

<table>
<thead>
<tr>
<th>Task</th>
<th>Goal</th>
<th>Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Create a new theme from scratch</td>
<td>Pure design (X-TG only)</td>
</tr>
<tr>
<td>B</td>
<td>Mix parts of different themes</td>
<td>Pure design (X-TG only)</td>
</tr>
<tr>
<td>C</td>
<td>Mix parts of different themes</td>
<td>Designing by examples (X-TG + XDE)</td>
</tr>
</tbody>
</table>

Table 5.1: A table describing the different goals and approaches between the three planned tasks.

### 5.1.1 Tasks

For our user study, we defined three tasks:

1. Task A: the users had to reproduce a theme for a restaurant website. This theme was originally created using the X-TG.
2. Task B: the users had to look at two X-Themes and try to mix parts of these X-Themes re-creating them in the X-TG.

3. Task C: the users had to perform the exact same mix of task B, but with the XDE tool.

According to the categorization we have defined before, we can distinguish the three tasks by the goal they have and the approach they use. We present the different goals and approaches for each planned task in Table 5.1. Therefore we planned these three tasks to verify the difference perceived by the users in pursuing different goals of development, as well as using different approaches (designing using X-TG only or the combination of X-TG and XDE), in order to verify if they would have felt more productive using a designing by example approach and if they would have considered it useful and effective, reducing the time needed for them to design a theme which was inspired from the design of other themes they wanted to mix. We will now present the details for each of the tasks.

Task A

In this task the users had to create a theme from scratch following a mockup (cf. Figure 5.2), so without using the designing by example approach. Therefore, no use of the XDE tool was necessary. The X-TG was the only tool that had to be used in this task, placing dynamic theme elements, functionalities and styling the content. In Figure 5.2 we can see the detailed structure of the X-Theme which had to be created. The users had to reproduce the website using the same exact specification presented in this thesis. More in details, they had to:

- Create containers (with background colors and images).
- Insert website title and description in the header.
- Reuse a Component provided in a zip archive, which can be inserted through the X-TG and that is powered by our XML metamodel representation.
5.1. PLANNING

Figure 5.2: Detailed specification for the Task A.

- Insert The Loop in the body of the theme.
- Create a static footer.
- Style the elements (exact positioning was not strictly required).

Task B

In this task, the users had to look at two X-Themes (cf. Figure 5.5) and to reproduce a mix of these two themes to design a third theme, with parts of the first source theme, and parts of the second source theme. This task was made to make the user try to make combinations of two separate themes without the XDE tool (and so without the designing by example approach) evaluating their results and comfort to take examples without being able of reusing them with the drag and drop interaction. We can look more in detail to the specification of the task as presented to the users in Figure 5.6. The header of the resulting theme had to be inspired to Theme 1, while the body and the footer had to be inspired to Theme 2. The dropdown menu present to the left of the website was provided as a Component archive, which can be inserted through the X-TG and that is powered by our XML metamodel representation.

Task C

In this task, the users had to look at two X-Themes (cf. Figure 5.5) and to reproduce a mix of these two themes to create a third theme. This task was conceived to make the user try to make combinations of two separate themes using the XDE tool (and so exploiting completely the designing by example approach) evaluating their results and comfort to take examples while being able of reusing them with the drag and drop interaction.

We can look more in detail to the specification of the task as presented to the users in Figure 5.7. The header of the resulting theme had to be entirely taken with a drag and drop from Theme 1, while the body and the footer had to be taken from Theme 2. The dropdown menu
Figure 5.3: Source Theme 1

Figure 5.4: Source Theme 2

Figure 5.5: This picture shows the source X-Themes that had to guide the users in task B, as well as being source for the examples to be reused in task C.

Figure 5.6: Task B specification as provided to the users.
5.2. RESULTS

Figure 5.7: The task specification car the users have followed to complete task C.

located at the left of the website could obviously be dragged and dropped as well from Theme 2 into the X-TG.

5.1.2 Questionnaires

There were 5 different questionnaires provided to the users. In the first questionnaire we asked about the user’s background skills and WordPress knowledge. After the first questionnaire ended, the user had to fill the questionnaire of each task after they have completed it, depending on the order of the tasks they were given. In each task questionnaire, we asked the level of satisfaction using the tool, some usability questions relative to the X-TG tool (for tasks A,B) and for the XDE tool (for task C) and if the final result was exactly the one expected while designing it. Finally, after every task was completed, we proposed one last questionnaire in which we asked some final considerations, such as which approach they liked most or how much they thought the proposed approaches could empower the WordPress theme development for end users. The reader can see the complete questionnaires referring to Appendix A.

5.2 Results

We will now present the results of the user study as well as a discussion of the results themselves, underlining weak and strong points of the approach and further investigations that might be conducted in the future.
5.2.1 Presentation

Of the 12 participants, 10 were male and 2 were female with an average age of 30 years. Their general background knowledge of WordPress was low: they evaluated their average knowledge of WordPress development, estimating a level of 2 on a scale from 1 to 7, where 1 meant Novice and 7 meant Expert. Moreover, regarding their personal usage of the WordPress platform, we can see the reported usages in Figure 5.8. More in detail, we can see that most of the participants declared that they did not use WordPress very much; though, most of them already knew and used the platform, even if they do not use it regularly. Only 1 participant declared that they used WordPress once a day, while 3 of them declared that they use it several times a month.

![Figure 5.8: The users' self evaluations of the WordPress usage.](image)

![Figure 5.9: The self evaluation (average) of the users' development skills of WordPress extensions](image)
The 12 participants rated themselves also in their web development and user interface design skills, with an average grade of 4.5 and 3.5 respectively, on a scale from 1 to 7 in which 1 meant *Novice* and 7 meant *Expert*. Finally, the users have presented a more detailed self evaluation of their skills regarding the development of WordPress plugins, themes and custom post types, in a scale from 1 (*Novice*) to 7 (*Expert*), which we present in Figure 5.9; the scale is still the same of the previous questions. The questions regarded the development of the possible WordPress extensions, which we also defined in Chapter 2.

The results show that the users were not experts in developing WordPress extensions, giving an average self evaluation between 1.5 and 2 in each category (please still note that the scale goes from a minimum of 1 to a maximum of 7, in which the higher is better). Comparing this value to the declared self evaluations for the previous questions, we can say that the users had an average level of web development skills which did not regard WordPress, in which they considered themselves quite unexperienced. Moreover, they rated themselves even at a slightly lower level regarding interface design capabilities.

![Figure 5.10: Average time (in minutes) spent for tasks by the users.](image)

**Tasks**

The users have executed the three tasks presented in Section 5.1. Each task done by the users was measured by the evaluator noting time and the number of mistakes/errors they did. The average time spent for each task is reported in Figure 5.10; another interesting information is the standard deviation of the times of the three tasks. The standard deviation can be seen on Figure 5.10 as the black error bar on each task’s column. The average time is measured in minutes. Regarding task A, the average time spent is 6 minutes and 50 seconds with a standard deviation of 2 minutes and 12 seconds. Regarding task B instead, the average time spent was 7 minutes and 35 seconds, while the standard deviation is 1 minute and 42 seconds (lower than task A). Finally, the time required to complete the task was slightly lower in task C (6 minutes and 18 seconds); also the standard deviation of the required time was sensibly lower in task C (1 minute 6 seconds) which involved the *designing by example* approach and the XDE tool.
As results show, the task A and B had a much higher standard deviation. On the opposite side, the standard deviation of the time for task C was quite low. We can see the results in Figure 5.10.

Figure 5.11: The user evaluation of their success level compared to the task requests and the satisfaction level after doing tasks A, B and C respectively.

We have also asked the users to rate their level of satisfaction after the usage of the X-TG/XDE tools after each task, which were executed in every possible order. We show the results in Figure 5.11, which reports the medians for the task in a scale from 1 to 7. In the proposed scale, 1 means Totally Disagree while 7 means Totally Agree, therefore a higher value is better. Their success level (compared to the tasks requirements) and the satisfaction level is high using the X-TG tool (tasks A - B) in which they developed a theme from scratch and tried to mix two themes without the XDE tool; in task C, where they could exploit the designing by example approach, their satisfaction level is high too.

Figure 5.12: The users’ evaluation of the designing by example approach realized by the XDE tool, after they completed task C.

We have also asked the users to rate their level of satisfaction after the usage of the X-TG/XDE tools after each task, which were executed in every possible order. We show the results in Figure 5.11, which reports the medians for the task in a scale from 1 to 7. In the proposed scale, 1 means Totally Disagree while 7 means Totally Agree, therefore a higher value is better. Their success level (compared to the tasks requirements) and the satisfaction level is high using the X-TG tool (tasks A - B) in which they developed a theme from scratch and tried to mix two themes without the XDE tool; in task C, where they could exploit the designing by example approach, their satisfaction level is high too.
More in detail, we can see that the median of the success level evaluated by the users was 7, 6.5 and 7 for tasks A, B and C respectively. Their satisfaction level instead has been evaluated 6.5 for task A, 6 for task B, and 6.5 for task C. Comparing these results to the task times presented in Figure 5.10, there might be a correlation between the task times and the satisfaction level as perceived by the users after using the tool.

According to their opinion, the reuse of the functionalities from other themes with the drag and drop has many positive aspects. After they completed the task C, we asked to rate all these aspects in a scale from 1 (Totally Disagree) to 7 (Totally Agree). We can see the results in Figure 5.12, in which we show the medians of the evaluations of various aspects.

More in details we asked them if the designing by example approach powered by the drag and drop interaction was: easy to understand, easy to use, efficient, effective, and useful. The medians (cf. Figure 5.12) show the evaluation 7 for each of these fields, according to the user responses. This result confirms what we shown with the previous data regarding task times and the evaluation of success and satisfaction.

Figure 5.13: The user estimation of the manual time needed to obtain the same result as task C by coding manually.

This is confirmed also by another self evaluation the users have made, in which we asked to estimate the time they would have needed to obtain the same result of task C by coding manually. In Figure 5.13 we show the estimation done by the users. To better understand this data we have to recall what we shown in Figure 5.9, in which the users declared to have a low knowledge about the WordPress extensions.

Therefore, some of the users were not able to make an estimation of the time they would have needed to create the same mix of themes by coding manually the theme for WordPress: 3 of them declared that they did not know how much time it would have taken to code the same theme manually. The remaining users have answered the question in the following way:

- 1 of them estimated that it would take them between 30 minutes and 1 hour.
- 1 of them estimated that it would take them between one hour and two hours.
- 3 of them estimated that it would take them between two and three hours.
- 4 of them estimated that it would take them more than three hours.
If we compare these results to the time needed to complete the task C that we show in Figure 5.10, we can see that it took them an average of less than 6 minutes and 30 seconds to complete this task with the combination of XDE and X-TG, with a standard deviation of 1 minute and 6 seconds. This stands in line with the feeling of efficiency and satisfaction that the users have declared to feel after using the tool, shown in Figure 5.11 and in Figure 5.12.

Figure 5.14: The users’ evaluations of the potential improvement provided by X-TG and XDE in the Theme development process, the reusability and the time spent developing a theme.

Finally, we show the results of the last questionnaire, in which we asked the users to make some final considerations and evaluations.

More in details, 10 of the 12 users involved declared to prefer the approach presented in task C, which involved the XDE tool presented in this thesis. We also asked the users to evaluate the future improvement that this approach can provide to the development of themes, the reusability and the time needed to develop a theme; their answers were provided on a scale from 1 (Totally Disagree) to 7 (Totally Agree); therefore a higher value is better. We show the results of this question in Figure 5.14, where the users identified in this technique a great improvement for each of these aspects. We show in the chart the medians of the answers to each of the questions. The median answer to the potential improvement that the designing by example approach could give to the theme development is 6. Even better, the median of the improvement that this approach can provide, according to the users, is rated 7. Finally, the median of the improvement of the required time to develop a theme as estimated by the users is 6.5. All of these values stand in line with the general feeling of satisfaction and efficiency that the users declared to feel after the usage of the XDE tool in task C, which can be seen in Figure 5.11 and Figure 5.12.
5.2.2 Discussion

Overall, the users had an average skill in web development and a slightly lower skill in interface design. Although none of these skills were actually necessary to perform the tasks in the user study, their skill level had probably a big impact on the hypothetical questions we asked them to reply in case of an estimation of the time needed by coding manually the theme reproduced in the task.

Moreover, they had a quite low skill level in WordPress development, which might have influenced the results as well. Since the task involved the creation of themes, only a few of them concretely knew the required work that has to be done to create a WordPress theme. Another user study with more experienced people would be interesting, in order to improve the estimation of the efficiency proposed by the X-TG and the XDE tools.

In addition, the satisfaction after using the tool might be related to the background skills of the user; more professional and expert users might feel blocked and inefficient using a different tool that is providing a WYSIWYG (What You See Is What You Get) approach.

Since the users involved in the user study were not experts of WordPress’ internals, it would be interesting to perform further investigation in the satisfaction level as perceived by more expert users after working with the XDE tool. Moreover, they might also feel more efficient and effective coding manually instead of using the XDE tool; therefore, also the evaluation of the effectiveness and efficiency might be different from the perspective of more expert users.

As results show, the task A and B had a much higher standard deviation, probably because the user had to do more manual operation such as element styling or positioning, which became faster with the learning effect or might be cause of mistakes which required more time to be repaired. On the opposite side, the standard deviation of the time for task C was quite low, since the operation involved the reuse with drag and drop and just the re-styling of few text colors for getting the same result as task B, which generally took more time to be done without the XDE tool. The users’ experience might also influence the time required to do the tasks (and therefore its standard deviation); the experience of the users would then influence the error rate and the time needed to recover from a mistake. Moreover, also the time needed to understand the tasks might be different.

Finally, the users had to work with a prototype which was not immune to bugs. Even though the number of encountered bugs during the XDE workflow in task C was low (1 bug found which happened only once) there have been also some unexpected issues into the workflow related to the X-TG tool, which might have influenced the times presented in Figure 5.10. It would be interesting to investigate the difference in time needed (and also the corresponding standard deviation) by the users after every issue has been fixed.
Conclusions

In this thesis we presented XDE - a tool which brings the designing by example approach on the WordPress platform (one of the most spread CMS platforms on the web). We built XDE on the top of the X-TG theme generator, with which makes the X-Themes project. In order to be capable of reusing components between these two tools, we had to share a common base. Therefore, we analyzed the current definitions of a WordPress theme and then we provided a new definition which underlined the importance of reuse and maintainability of the theme itself and of its components. Moreover, we defined the X-Themes metamodel based on the concepts we could identify inside a WordPress theme. The metamodel contains three different types of elements, of which we planned and implemented the reuse of two of them: Components and Layout Components. Thanks to these concepts this thesis presented the XDE tool that implemented a different reuse algorithm for each of these two elements, detailing the tool’s architecture and how the tool’s subsystems behave during the reuse process. In an user study conducted together with the X-TG tool, we found that the designing by example approach was the most appreciated by the users, who could mix parts of two running X-Themes quickly and easily (about 6 minutes).

6.1 Limitations

We should make an important point concerning the reuse algorithm that we proposed for the Component elements of an X-Theme: since the PHP code has to be reused as well, it has to be exposed through text files, in order to be able to fetch it from the XDE Component Fetcher that is located into the WordPress plugin (cf. Chapter 4). This remarks an important issue of the reuse process, that is the security level of the Components which can be reused. Since the PHP code is exposed to the public downloading, malicious users could be capable of some sort of security breaches by looking at the code contained into the Components. This issue was not investigated in this thesis, since it was not one of the goals specified. Moreover, we have to recall the technical limitations presented in Section 4.3, which apply to the reuse algorithm as well:
6.2. Future Work

Since the XDE tool was implemented as a *prototype* to investigate the designing by example approach applied to more complex components (involving JavaScript and PHP), there are still many issues that have to be investigated in order to improve and push this approach:

- Since there might be several security issues with the reuse algorithm, a way to protect the transferred code of the examples should be developed; in alternative, a different algorithm or workflow might be investigated to avoid this issue.

- A new CSS inference engine should be done in order to solve the issues presented in the previous section and in Section 4.3.

- The reuse of Components might be improved by making only one connection to the source of the example to download it as an archive (which is already supported by X-TG) and then unpack it in the WordPress installation executing the reuse.

- A new, queue-based reuse engine might be created in order to allow users to reuse multiple examples at a time; in the current prototype, the users had to reuse one Component and wait that it was fetched completely by the XDE-CF, which required some time; this could also allow them to perform more operation in parallel.

Figure 6.1: A mockup of a gallery browsing X-Themes.
• Although XDE already provides an assisted design interface, it confuses the user in the context of dark themes (it applies black borders to show the elements which can be reused). Moreover, it also applies borders to the Components, making them slightly move during the selection of the examples.

• A new user study should be conducted with more expert people in order to better estimate the effectiveness of the approach, as detailed in Chapter 5. This user study might also help to verify if some bugs introduced bias into the estimations provided by the users.

• An example gallery should be produced (as Sharmin et al. [16] suggest in their work) in order to better browse the examples available to the users; exploiting the X-Themes metamodel, we conceived a brand new way of browsing themes called browsing by examples. Thanks to the definition of a theme (cf. Chapter 3), which is composed by nested XML elements of Layout Components and Components, an X-Theme theme definition can be used to understand which functionalities are contained in an X-Theme. Assuming that we could create different previews of each Component built into an X-Theme, we could be capable of creating a system in which a user can see different types of X-Themes, categorized by "Components", which could be presented one by one to the user as a preview before he actually gets to the XDE design interface of the corresponding X-Theme trying to reuse the examples. We can see an example in Figures 6.1 and 6.2. To do this, the XML Component definition might be extended to support some assistive-categorization (tags) which can be coupled with the preview gallery. More details of a scenario related to this functionality was provided in Section 4.1

• A survey in the WordPress community will be developed in order to collect more data about the development process of a theme and to understand how the WordPress theme developers approach the process in terms of reuse and maintainability.
6.2. FUTURE WORK


User Study
X-Themes User Study Questionnaire

Welcome to the X-Themes User Study Questionnaire!
This user study will help us evaluate the X-Themes approach and tool.
Thank you for your cooperation!

*Campo obbligatorio

1. Please insert the Test ID the team has given you: *
   If you don't have one, just ask the project team.

Personal Information and Background Skills

2. Sex *
   Contrassegna solo un ovale.
   - Male
   - Female

3. Age *

4. Background skills *
   Contrassegna solo un ovale per riga.

<table>
<thead>
<tr>
<th>Novice - 1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7 - Expert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wordpress Usage</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wordpress Development</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Web Development</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Interface Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Wordpress Experience

5. How often do you use Wordpress? *
   Contrassegna solo un ovale per riga.

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Never</th>
<th>Less than once a month</th>
<th>Once a month</th>
<th>Several times a month</th>
<th>Once a day</th>
<th>Several times a day</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6. **How much do you know about WordPress development, its internals and how to extend it?** *

Contrassegna solo un ovale per riga.

<table>
<thead>
<tr>
<th>Novice - 1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Expert - 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom Type Development</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Theme Development</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Plugin Development</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7. **Did you gain your personal knowledge from books or guides?**

If you never used Wordpress, skip this question

Seleziona tutte le voci applicabili.

- [ ] Tutorials/Guides
- [ ] Books
- [ ] Wordpress Codex Documentation
- [ ] Help Communities (Stackoverflow, Wordpress Exchange..)
- [ ] Altro: .................................................................

**Theme Generator Experience**

A Theme Generator is a visual editing tool for your Wordpress themes. In a Theme Generator, you can generally choose the style details for your theme.

8. **Which of these Theme Generators did you use in your life?** *

Seleziona tutte le voci applicabili.

- [ ] I never used a Themes Generator
- [ ] WordPress ThemeGen
- [ ] Templatr
- [ ] www.yvoschaap.com/wpthemegen/ (WP Theme Generator)
- [ ] Wpthemegenerator.com
- [ ] Artisteer
- [ ] Lulbith
- [ ] Altro: .................................................................

9. **Do you think that a Theme Generator can help the user to develop a theme?** *

Contrassegna solo un ovale.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally Agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
10. **What are the factors that contribute to the quality of a Theme Generator?**

*Contrassegna solo un ovale per riga.*

<table>
<thead>
<tr>
<th>Factor</th>
<th>Not important</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Very Important</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity to customize the design (e.g. modify font, color, background)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capacity to modify the layout</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capacity to reuse parts between themes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capacity to adapt the created theme to many devices (smartphones and different kinds of screen) and in different browsers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

11. **Are you satisfied with existing Theme Generators?**

*If you never used one of the Themes Generators above skip this question.*

*Contrassegna solo un ovale.*

<table>
<thead>
<tr>
<th>Satisfaction Level</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not Satisfied</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally Satisfied</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
X-Themes User Study Questionnaire: Task A

Part of the X-Themes User Study.

TASK A

*Campo obbligatorio

1. Please insert the Test ID the team has given you: *
   If you don't have one, just ask the project team.

Task A: Create a new Theme with X-Theme Generator

In this task you will reproduce a Theme using X-Themes Generator.

Go on the following link to perform the task:

You have to use the X-Themes Generator in the page.

To see the final result of your Theme go on:
- http://localhost/food/

When you complete the task, answer to the other questions.

2. I feel that I successfully completed the task *
   Contrassegna solo un ovale.

   1 2 3 4 5 6 7
   Totally Disagree  □ □ □ □ □ □ □ Totally Agree

3. I am satisfied with the result *
   Contrassegna solo un ovale.

   1 2 3 4 5 6 7
   Totally Disagree  □ □ □ □ □ □ □ Totally Agree

4. The Theme Generator interface is well organized and functions are easy to find *
   Contrassegna solo un ovale.

   1 2 3 4 5 6 7
   Totally Disagree  □ □ □ □ □ □ □ Totally Agree
5. **I felt comfortable using the tool**

Contrassegna solo un ovale.

<table>
<thead>
<tr>
<th>Totally Disagree</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>Totally Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6. **The support for Drag & Drop of elements was**

Contrassegna solo un ovale per riga.

<table>
<thead>
<tr>
<th>Totally Disagree</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>Totally Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to understand</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Useful</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Easy to use</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Efficient</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Effective</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7. **The tools for styling an element were**

Contrassegna solo un ovale per riga.

<table>
<thead>
<tr>
<th>Totally Disagree</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>Totally Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to understand</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Useful</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Easy to use</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Efficient</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Effective</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8. **Importing functionalities that already exist was**

Contrassegna solo un ovale per riga.

<table>
<thead>
<tr>
<th>Totally Disagree</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>Totally Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to understand</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Useful</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Easy to use</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Efficient</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Effective</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
9. **The usage of theme elements was:** * 
   e.g. Title, Description, Loop 
   *Contrassegna solo un ovale per riga.* 

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to understand</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Useful</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Easy to use</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Efficient</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Effective</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

10. **How much time would you need to do the task coding manually?** * 
    *Contrassegna solo un ovale per riga.* 

<table>
<thead>
<tr>
<th>Time needed</th>
<th>I don't know</th>
<th>30 minutes - 1 hour</th>
<th>1 hour - 2 hours</th>
<th>2 hours - 3 hours</th>
<th>More than 3 hours</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

11. **I think that doing this task coding manually for Wordpress would be easier** * 
    *Contrassegna solo un ovale.* 

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

12. **I think that doing this task coding manually for Wordpress would be better** * 
    Where better means that you can provide a more precise result compared to the original mockup 
    *Contrassegna solo un ovale.* 

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

13. **I think that doing this task with an other Theme Generator would be easier** 
    *Contrassegna solo un ovale.* 

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
14. I think that doing this task with an other Theme Generator would be better
Where better means that you can provide a more precise result compared to the original mockup
Contrassegna solo un ovale.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Totally Disagree   Totally Agree
X-Themes User Study Questionnaire: Task B

Part of the X-Themes User Study.

**TASK B**

*Campo obbligatorio*

1. **Please insert the Test ID the team has given you:** *
   If you don't have one, just ask the project team.

---

**Task B: Mix parts of existing Web Sites with only X-Themes Generator**

In this task you will mix parts (following a Mockup) of existing Web Sites with only X-Themes Generator.

Open the following links:
- [http://localhost/wordpress/](http://localhost/wordpress/)
- [http://localhost/thirdrestaurant/](http://localhost/thirdrestaurant/)

Go on the following link to use the XThemes Generator:

Use X-Themes Generator in that page.

To see the final result of your Theme go on:
- [http://localhost/usermix/](http://localhost/usermix/)

When you complete the task, answer to the other question.

2. **I feel that I successfully completed the task** *

   Contrassegna solo un ovale.

   ![Rating Scale](https://docs.google.com/forms/d/1i0H5Qmch5i2pNL0GUePyJm4ShYiWprMCwMqSUWRE/edit)

3. **I am satisfied with the result** *

   Contrassegna solo un ovale.

   ![Rating Scale](https://docs.google.com/forms/d/1i0H5Qmch5i2pNL0GUePyJm4ShYiWprMCwMqSUWRE/edit)
4. The Theme Generator interface is well organized and functions are easy to find *
Contrassegna solo un ovale.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally Agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5. I felt comfortable using the tool *
Contrassegna solo un ovale.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally Agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6. The support for Drag & Drop of elements was *
Contrassegna solo un ovale per riga.

<table>
<thead>
<tr>
<th>Easy to understand</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally Agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Useful</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally Agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Easy to use</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally Agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Efficient</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally Agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Effective</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally Agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7. The tools for styling an element were *
Contrassegna solo un ovale per riga.

<table>
<thead>
<tr>
<th>Easy to understand</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally Agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Useful</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally Agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Easy to use</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally Agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Efficient</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally Agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Effective</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally Agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
8. **Importing functionalities that already exist was** *Contrassegna solo un ovale per riga.*

<table>
<thead>
<tr>
<th></th>
<th>Totally Disagree</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Totally Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to understand</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Useful</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Easy to use</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Efficient</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Effective</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

9. **The usage of theme elements was:** *Contrassegna solo un ovale per riga.*

<table>
<thead>
<tr>
<th></th>
<th>Totally Disagree</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Totally Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to understand</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Useful</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Easy to use</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Efficient</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Effective</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

10. **How much time would you need to do the task coding manually?** *

<table>
<thead>
<tr>
<th>Time needed</th>
<th>I don't know</th>
<th>30 minutes - 1 hour</th>
<th>1 hour - 2 hours</th>
<th>2 hours - 3 hours</th>
<th>More than 3 hours</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

11. **I think that doing this task coding manually for Wordpress would be easier** *Contrassegna solo un ovale.*

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally Agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

12. **I think that doing this task coding manually for Wordpress would be better** *Where better means that you can provide a more precise result compared to the original mockup Contrassegna solo un ovale.*

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally Agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
13. **I think that doing this task with an other Theme Generator would be easier**

Contrassegna solo un ovale.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally Agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

14. **I think that doing this task with an other Theme Generator would be better**

Where better means that you can provide a more precise result compared to the original mockup

Contrassegna solo un ovale.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally Agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
X-Themes User Study Questionnaire: Task C

Part of the X-Themes User Study.

**TASK C**

*Campo obbligatorio*

1. **Please insert the Test ID the team has given you:** * 
   If you don’t have one, just ask the project team.

---

**Task C: Mix parts of existing Web Sites with X-Theme Generator and Design By Example**

In this task you will mix parts (following a Mockup) of existing Web Sites with only X-Themes Generator.

Open the following links:
- [http://localhost/wordpress/?design=true](http://localhost/wordpress/?design=true)
- [http://localhost/thirdrestaurant/?design=true](http://localhost/thirdrestaurant/?design=true)

Go on the following link to use the XThemes Generator:

Use X-Themes Generator in that page.

To see the final result of your Theme go on:
- [http://localhost/usermix2/](http://localhost/usermix2/)

When you complete the task, answer to the other question.

2. **I feel that I successfully completed the task** *
   
   Contrassegna solo un ovale.

   ![Scale](https://docs.google.com/forms/d/1t8lA5qKmhkJCPQhWSF7sr8Pf6PgqOiiGKRplwX7CZCE/edit)

3. **I am satisfied with the result** *
   
   Contrassegna solo un ovale.

   ![Scale](https://docs.google.com/forms/d/1t8lA5qKmhkJCPQhWSF7sr8Pf6PgqOiiGKRplwX7CZCE/edit)
4. **The Theme Generator interface is well organized and functions are easy to find**
   *Contrassegna solo un ovale.*

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5. **I felt comfortable using the tool**
   *Contrassegna solo un ovale.*

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6. **The support for Drag & Drop of elements was**
   *Contrassegna solo un ovale per riga.*

<table>
<thead>
<tr>
<th>Easy to understand</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Totally Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usefulness</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Easy to use</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Efficiency</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Effectiveness</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7. **The reuse of components from other Themes with Drag & Drop was**
   *Contrassegna solo un ovale per riga.*

<table>
<thead>
<tr>
<th>Easy to understand</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Totally Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usefulness</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Easy to use</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Efficiency</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Effectiveness</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
8. **The tools for styling an element were** *

*Contrassegna solo un ovale per riga.*

<table>
<thead>
<tr>
<th></th>
<th>Totally Disagree</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Totally Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to understand</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Useful</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Easy to use</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Efficient</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Effective</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

9. **Importing functionalities that already exist was** *

*Contrassegna solo un ovale per riga.*

<table>
<thead>
<tr>
<th></th>
<th>Totally Disagree</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Totally Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to understand</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Useful</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Easy to use</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Efficient</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Effective</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

10. **The usage of theme elements was:** *

*Contrassegna solo un ovale per riga.*

<table>
<thead>
<tr>
<th></th>
<th>Totally Disagree</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Totally Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to understand</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Useful</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Easy to use</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Efficient</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Effective</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

11. **How much time would you need to do the task coding manually?** *

*Contrassegna solo un ovale per riga.*

<table>
<thead>
<tr>
<th></th>
<th>I don't know</th>
<th>30 minutes - 1 hour</th>
<th>1 hour - 2 hours</th>
<th>2 hours - 3 hours</th>
<th>More than 3 hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time needed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
12. I think that doing this task coding manually for Wordpress would be easier

Contrassegna solo un ovale.

1 2 3 4 5 6 7

Totally Disagree  ☐ ☐ ☐ ☐ ☐ ☐ ☐  Totally Agree

13. I think that doing this task coding manually for Wordpress would be better *

Where better means that you can provide a more precise result compared to the original mockup

Contrassegna solo un ovale.

1 2 3 4 5 6 7

Totally Disagree  ☐ ☐ ☐ ☐ ☐ ☐ ☐  Totally Agree

14. I think that doing this task with an other Theme Generator would be easier

Contrassegna solo un ovale.

1 2 3 4 5 6 7

Totally Disagree  ☐ ☐ ☐ ☐ ☐ ☐ ☐  Totally Agree

15. I think that doing this task with an other Theme Generator would be better

Where better means that you can provide a more precise result compared to the original mockup

Contrassegna solo un ovale.

1 2 3 4 5 6 7

Totally Disagree  ☐ ☐ ☐ ☐ ☐ ☐ ☐  Totally Agree
X-Themes User Study Questionnaire

Final part of the XThemes User Study Questionnaire

*Campo obbligatorio

1. **Please insert the Test ID the team has given you:** *
   
   If you don't have one, just ask the project team.

---

**Last Considerations**

Let's do some considerations on the tasks done and Wordpress in general

2. **Which kind of approach do you prefer overall?** *

   *Contrassegna solo un ovale.*

   - A: Create new theme from scratch with X-Themes Generator
   - B: Mix part of existing themes with X-Themes Generator
   - C: Mix part of existing themes with X-Themes Generator and Design By Example

3. **The possibility to import existing code was:** *

   *Contrassegna solo un ovale per riga.*

<table>
<thead>
<tr>
<th>Totally Disagree</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Totally Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Effective</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Efficient</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4. **The possibility to drag and drop functionality from other themes is:** *

   *Contrassegna solo un ovale per riga.*

<table>
<thead>
<tr>
<th>Totally Disagree</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Totally Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Effective</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Efficient</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5. **X-Themes Generator and Design By Example can improve the development of Themes** *

   *Contrassegna solo un ovale.*

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>Totally Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

https://docs.google.com/forms/d/12dBlXk7j_p_cMRqKnMbaawXU1orauDrQcLZKj5w9-Zk7w/edit
6. **X-Themes Generator and Design By Example can improve reusability**

Contrassegna solo un ovale.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally Agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7. **Using X-Themes Generator and Design By Example reduce the time spent to create a theme**

Contrassegna solo un ovale.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally Agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8. **What do you think is missing in X-Themes Generator and Design By Example?**

- ...
- ...
- ...
- ...
- ...
- ...
- ...