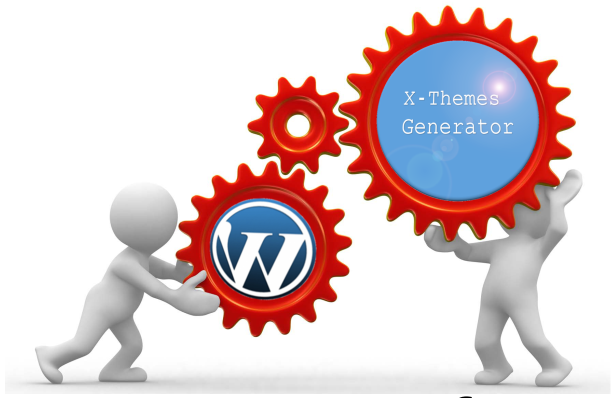
Theme generator for WordPress



Theme generator for WordPress

Master Thesis

Linda Di Geronimo
<ldigeronimo@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

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

globis

Abstract

Platforms such as WordPress are increasingly used by professional Web developers and by casual users with few or no background skills in web development. There are 68 million WordPress websites in this moment on the internet, and currently this platform counts over three hundred million people who are reading WordPress blogs.
Even if the platforms are extended and redefined to be more accessible to a larger number of users, sometimes they are based on models that are not reusable or not well-formed. These platforms use the concept of *theme* to identify or represent the website's look & feel. Though, the concept of a theme is not precisely defined, and does not clearly identify what makes a theme and what does not. For these reasons, it is important to focus on these kinds of tools trying to improve them by defining more formal models.
A lot of theme generators for such platforms now exist on the Web to help users create new themes without having skills in web development and without knowing what a theme precisely is. As studied in this thesis, theme generators have limitations even if these tools can help the users develop a theme that they like. They usally allow users to customize only static components in a theme (such as font color, logo image etc.) and they do not allow the modification of dynamic functionalities (such as Javascripts and PHP scripts) that are the main parts of many website.
The thesis presents *X-TG*, a theme generator for WordPress that allows users to create new themes for WordPress following a well-defined meta-model. This meta-model gives a formal structure to the theme, based on clearly-defined basic components to allow the reuse between different X-Themes. Moreover *X-TG*, unlike other theme generators, allows users to insert and customize more complex elements.
In a task-based user study, we investigated the potential of *X-TG*. The results showed that the users, without having technical skills, were able to develop a WordPress theme following a mockup in a few minutes. Moreover, the participants of the study, were satisfied with the theme generated and appreciated the opportunity to customize it with complex functionalities. The users estimated that coding the same theme manually would have cost hours instead of minutes.

To my little friend Paolo.

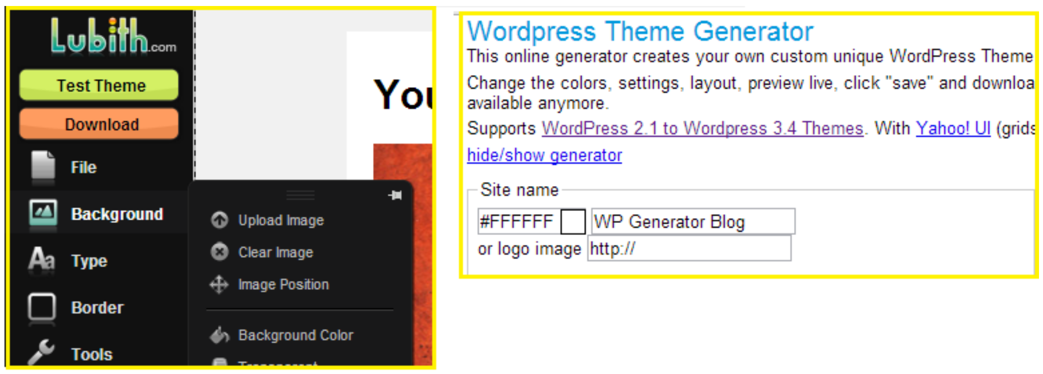
Contents

List of Figures

1 Introduction

Nowadays a large number of people use platforms such as WordPress[1] to create new websites. On the internet there are more than 60 million WordPress websites, and more than 47.2 million users produce 68.7 million new comments each month[2]. These numbers are already big but, despite that, WordPress continues to grow; from 2011 over 100.000 new websites have been created on this platform every day, and this number shows no signs of slowing[3].
WordPress users are quite different between each other. This platform hosts famous websites for equally famous companies: emblematic examples are the blog for: CNN[4], the New York Times[5], SONY[6] and more. However, in addiction to these famous company websites, millions different kinds of blogs exist. These websites are written by thousands of users, and certainly developers are only a small part of them, so it is easy to imagine that they were able to approach the process of development without skills, only using WordPress.
For these reasons, it is important not to underestimate these tools; it is necessary, in fact, to understand the real meaning of these data: the number and the kinds of end-users are changing fast thanks to all these platforms.
Moreover, platforms such as WordPress use the concept of *theme* to represent the look & feel of the websites. It is difficult to find a precise definition of a theme, and WordPress itself does not give a clear identification of what makes a theme and what does not, but simply identifies a theme as a repository of files (PHP, JavaScript, CSS and images)[7] without defining a theme model or a basic structure.
Research communities are now focusing in bringing modelling abstractions and rules to these platforms. For example Leone et al. [4], introduces a multi-level modelling framework which is realized by a WordPress plug-in.

[1]http://en.wordpress.com
[2]http://en.wordpress.com/stats/posting/
[3]http://en.wordpress.com/stats/
[4]http://edition.cnn.com/exchange/blogs/index.html
[5]http://www.nytimes.com/interactive/blogs/directory.html
[6]http://blog.us.playstation.com/
[7]http://codex.wordpress.org/Glossary#Theme



Wordpress Theme Generator
This online generator creates your own custom unique WordPress Theme
Change the colors, settings, layout, preview live, click "save" and downloa
available anymore.
Supports WordPress 2.1 to Wordpress 3.4 Themes. With Yahoo! UI (grids
hide/show generator
Site name
#FFFFFF WP Generator Blog
or logo image http://

Figure 1.1: Screenshot of two theme generators menu (in the left side Lubith in the right side Wordpressthemegenerator). In both generators, the main functionalities are the modification of the color of the elements of a theme.

At the same time, other works focus on model-driven web engineering approaches which, in general, provide graphical formal specification for the complete design process (cf. Ceri et al. [1]). These methodologies can be applied only by people who are used to modelling abstractions; unfortunately, these model-driven approaches are not focused on a web development process on platforms such as WordPress.
Given the huge spread of WordPress, a lot of theme generators can be found. The main issue of theme generators, as studied in this thesis, is that they do not allow the customization of dynamic functionalities (such as JavaScripts) from web applications, also if these pieces are the main part of a website. For these generators the main important pieces of a web application are only the static layout components (such as CSS and HTML); the user is able to change only static properties of a theme (such as the position of the logo, the color or the font of the text, cf. Figure 1.1).
This thesis proposes *X-TG*, a new tool that could allow users to design new websites upon widely-spread Web platforms, while trying to suppor reuse and allow the customization of complex elements of a theme, thanks to a well-formed model.

1.1 Goals

Until today, most of the current theme generators have many limitations: in most cases, they allow the user to change only colors, fonts or background of a theme, and only in some of them it is possibile to personalize more complex parts of a website but with a lot of limitations (cf. Figure 1.1). Moreover, they are not structured by a well-formed model that supports reuse. At the same time, model-driven approaches proposed by the reasearch comunity, are built mainly for developers rather than endusers and they are not focused on platform such as WordPress.
In order to design new themes supporting reuse, we present the X-Themes generator (*X-TG*), a theme generator on WordPress that helps the users to create a new theme. *X-TG* allow users to customize themes; the users can insert and modify complex elements, not only static components (like the font color or the background image) and, at the same time, the genereted theme will follow a meta-model that will permit reuse between X-Themes. However, The

meta-model will be created transparently to the user in order to allow the users to develop themes without having to learn a new modelling language.

The detailed goals of this Masters thesis are as follows:

1. **Analysis of State of the Art.** Analyze and study existing theme generators, trying to classify them for functional and non-functional requirements, in order to investigate and understand the main limitations of these tools.

2. **Definition of theme Concept and Metamodel.** Build a new formal theme concept using a metamodel (called X-Themes) that permits reuse.

3. **Design and Development a theme generator.** Develop a theme generator, *X-TG* generates theme according to the metamodel, taking into account the limitations founded during the study of existing theme generators. The generator will generate WordPress themes following the metamodel, but this model will be generated transparently to the user.

4. **User Study.** Design and carry out user evaluation to investigate the power of *X-TG*. Goals are to identify problems and limitation of the generator and to demonstrate the capacity to improve the development of a theme using the application.

1.2 Contributions

In order to understand the main issues and limitations of current theme generators, the first step for this thesis was to analyze and categorize these tools by functional and non-functional requirements. Based on this analysis, this work presents a new meta-model (called X-Themes) that was designed to support reuse between different themes. Following that, a theme generator (*X-TG*) was developed based on the meta-model, at the same time extending the functionalit of existing theme generators.
Finally, a user study was performed to test the potential of *X-TG*. The study showed that using *X-TG* improves, in terms of time, the creation of a new theme for WordPress. The participants were able to develop a WordPress theme in a few minutes, even if the users had no WordPress knowledge. Moreover, the theme created by the users had complex elements which the participants appreciate.

1.3 Structure of the Thesis

We start with a discussion of related work, and present the background to this work in Chapter 2. In Chapter 3 we give an analysis of the current theme generators for WordPress, showing their limitations. Chapter 4 describes the concept and the usage of the X-Themes generator in detail showing the meta-model, its implementation and architecture. We show the user study in Chapter 5, with its tasks, structure and results also reporting important issues and limitations found during the study. Finally, we give conclusion and future work in Chapter 5.

2 Background

X-TG is tool that helps the users to develop their own themes in WordPress; we achieved this goal by using a metamodel that isolates the components of a theme and at the same time is created to support future reuse. For these reasons, this thesis is related to research studies done in the last years, and in this chapter we will introduce them, trying also to explain some important issues of the research area. First, we will talk about the purpose of model-driven approaches also giving some important examples (cf. 2.1). After, we will focus on the end-user development, its goals and its categorization (cf. 2.2). Finally, we will do a big picture on WordPress (cf. 2.3), also to allow the reader to easily understand the *X-TG* chapter.

2.1 Model-driven approaches

The design development and maintainence of web applications is one of the main issues of the software industry. For this reason model-driven approaches were studied by the research-community trying to improve the web development process.
One of the emblematic examples of the model-driven approaches is WebML (cf. Ceri et al. [1]). WebML is a notation for describing web applications at a high level of abstraction. This model has the goal to help the developers specify complex websites using different dimensions: the data content, the pages of the Web application, the topolgy of the links, the graphic elements for the redering of the page, and customization features. Each dimension is associated with a well-defined graphic notation and an XML syntax. One of the more interesting results given by WebML use in software industry, was that this approach was capable to achieve extensions by companies or single developers. This flexibility permitted many extensions of the model, in order to cover requirements of different kinds of applications (cf. Koch et al. [3]).
In addition to WebML, an extension to this approach is WebRatio. Other studies were done on it, trying to improve the the abstractions of WebML giving birth to WebRatio. This tool uses a formal graphical language for the specification of the data of web applications; in WebRatio the data are specified using the Entity Relationship model and the WebML language

for the functional requirements (cf. Ceri et al. [2]).
These kinds of approaches have some limitations: they are targeted to a small number of end-users. In fact, to study and use these models is necessary to have technical skills. In addition, to correctly use these models, a period of training it is necessary also for technical users. Moreover, the studies of these models were done in the years between 2000 and 2001 when platforms such as WordPress or Drupal were not used as now, so they are not developed to work with these tools. In this way, these models lose a lot of effectiveness.
Different kinds of model-driven approaches were developed in the last years to avoid these limitations. For example Leone et al. [4] give a model for component-based web engineering, that supports composition at various levels: data, schema, application logic and user interface. This approach allows the combination of these components, and it has been realized in WordPress. Moreover, this composition process is realized with a graphical user interface to target non-technical end-users.

2.2 End-user development

For Lieberman et al. [5] the goal in human-computer interaction will become to create applications which are not only ”easy to use” but also ”easy to develop”. In fact, Lieberman et al. [5] say that for most of the people it is now easy to interact with the main interface metaphors but the situation is different when the users have to create new applications, where some technical skills are still needed. Thus, one challenge now is to develop platforms that help non-technical end-users create new applications. End-user development is a research topic that has these goals.
Lieberman et al. [5] categorize the end-user activities in two types:

1. **Parameterization or customization.** Activies that permit users to choose between different alternatives that already exist in the application.

2. **Program creation and modification.** With these activities the users create a program from scratch or modify a software artefact.

End-user development in general, involves the second type of activities described before. In the *Parameterization or Customization*, activities the modifications are strictly standard options of the application itself. More in general, a goal of a platform that allows end-user development, is to allow the user to move between activities of the first type and activities of the second type (cf. Lieberman et al. [5]).
In this regard, platforms were developed to allow the users to create new applications, but in recent years end users demonstrated a keen interest in developing especially the web (cf. Nestler et al. [8]). Recently, many researches have explored this matter such as WebFormulate [9], WebSheets [13] and Click [10].
WebFormulate is a web-based tool, that permits users to create web application while *WebSheets* uses a mix of methaphors: programming-by-example, query-by-example and spreadshet concepts. *Click* is a proof-of-concept prototype for end-user web programming.
All these studies try to find strategies to improve end-user development for the web. In addition to these studies, the research community spends effort on the concepts of component-based design (cf. Lieberman et al. [5]). In this kind of design, the users can combine different components at runtime; such platforms have already divided the system into different components. This design has two main requirements: the modularization has to be clear to the

end-user and, at the same time, the application has to be flexible towards its context and its environments.
The complexity level of these applications can be divided in three types:

1. **Configuring a component.** The easiest way to customize a system is to configure one individual component. The tailoring action can be allowed by special enviroments which can help the user to modify the slot.

2. **Changing the component composition.** In this level of complexity, the users can insert, modify and delete a component; in addition they can also rewrite the components or create new complex ones. In this level, the users have to understand also the interaction between the components.

3. **Design new components.** In this level, the users can design and build new components, but they have to know some basics of programming.

More in general, the goal of component-based design is to allow the users to combine these components trying to reduce the complexity of these operations, so the main issue is to find a good trade-off between complexity and flexibility.

2.3 WordPress

WordPress is a content management system (CMS) tool[1]. A Content management system is software that permits a user to customize and publish personal content [6]. WordPress allows users to create websites or blogs. It is free and open source; many volunteers have written thousands of plugins and themes to extend this CMS tool.
In this Chapter, we will present an overview of WordPress including its basic concepts and the way it works, introducing some important definitions used in this thesis. WordPress has a huge community of developers that works to extend or improve it; to help these developers the tool has a web platform (called WordPress Codex[2]) with all the softwares pecifications. Almost all the information that will be presented in this chapter is based on the definitions stated in this website.

2.3.1 A big picture

As introduced before, WordPress is a platform that allows users to create their blog or websites. The main goals that WordPress pursues are the following[3]:
Easy to use: The software should be is to be easy to use, in this way any kind of user can interact and insert any content they want. One of the peculiarities of WordPress is that, after the installation, the users can already write their first post in the blog in few minutes.
Extensible: WordPress should be extensible to allow developers to create plugins. In fact, over than 20.000 of free and open source plugins exist now on WordPress.
Publish anywhere: The opportunity to insert new content in your blog anywhere, WordPress is supported by many applications for all the devices now in use[4].

[1]http://wordpress.org/
[2]http://codex.wordpress.org/
[3]http://codex.wordpress.org/WordPress_Features
[4]As of

Complete control: WordPress can be installed on a private web server or in the cloud, in this way the users can have complete control of everything that is in their site.

The users can manage their WordPress blog or website by the administration panel of WordPress. With the administration page they can modify the look and feel of the blog, insert new information, add contacts, insert new media and so on. More detail about these functionalities and concepts of WordPress are given in the next sections.

2.3.2 Design Concepts

The WordPress core is a set of files, that belong to the original WordPress software (cf. Williams et al. [12]). The core files will change only when WordPress will be upgraded, so these files do not include the themes of the users. The core is written in PHP, but it also contains CSS, JavaScript, XML, HTML and image files. The core controls anything that happens in WordPress.
In the next sections, we will focus on the major functions contained in the WordPress core including also the major components that WordPress uses.

Hooks

A fundamental concept of WordPress is the hook. As the name suggests, the hooks are functions defined by the user that can be added to the dynamic execution of the platform calls[5].
There are two types of hooks:

1. **Action.** An action is a function that in WordPress core is enqueued at a specific point[6]. The developers can create their own custom action using `add_action (A,B)`[7]. This function will exectute B after A.

2. **Filter.** A filter is a function related to an existing action. Using the function `add_filter(A,B)` means that we are replacing the code in B with the code in A[8].

All the WordPress extensions use these two basic concepts.

Themes

A real definition of Theme is hard to find, we can give here the definition explained by WordPress[9]:

> *"'A theme is a collection of files that work together to produce a graphical interface with an underlying unifying design for a weblog. A theme modifies the way the weblog is displayed, without modifying the underlying software. Essentially, the WordPress theme system is a way to skin your weblog.'"*

[5]http://codex.wordpress.org/Plugin_API/Hooks_2.0.x
[6]http://codex.wordpress.org/Glossary#Action
[7]http://codex.wordpress.org/Function_Reference/add_action
[8]http://codex.wordpress.org/Glossary#Filter
[9]http://codex.wordpress.org/Glossary#Theme

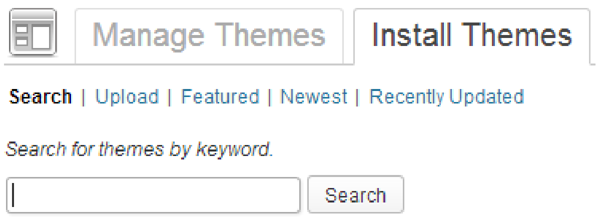

Manage Themes
Install Themes
Search | Upload | Featured | Newest | Recently Updated
Search for themes by keyword.
Search

Figure 2.1: Screenshot of the 'appearance' page of WordPress administration panel. The users can upload their own theme or they can search a new theme in the collection of WordPress.

So for this definition, a theme determines how the blog is displayed while defining a theme like a collection of files. More in detail, a WordPress theme contains not only files that change the look and feel of a websites but also it can contain plug-in, widgets and so on. The users have different ways to choose and select a theme for their blog. As we shown in Figure 2.1, they can select one theme by searching it on WordPress or, in addition, they can upload their own theme.

Posts and Pages

Each entry of a weblog is a post. Generally, every post has at least two information elements: the title and the content. The content is the text written by the users, and it may contain images, links, video and so on (cf. Silver [11]). Posts can contain also categories, tags and comments. The categories and tags are used to organize the posts in a blog and across blogs. The differences between categories and tags are that the first are like topics while the second are more like keywords (cf. Silver [11]). WordPress gives to the users an interface to insert a post (cf. Figure 2.2), also giving many options to customize it.
More in detail, in the WordPress core there is a file called *post.php* that contains functions related to the posting process of WordPress. These functions can be used externally to the core by the developers that want to develop with WordPress. An emblematic example of a post function in the core is:

- `get_post_meta`: that gets metadata of a post[10].

WordPress gives also the chance to write pages (like the "About us" or "Contact" pages). In general, the pages are for content that changes slower than posts[11].
A simple function related to the pages in WordPress is:

- `wp_list_pages`: that displays a list of WordPress pages as links[12].

[10]http://codex.wordpress.org/Function_Reference/get_post_meta
[11]http://codex.wordpress.org/Pages
[12]http://codex.wordpress.org/Template_Tags/wp_list_pages

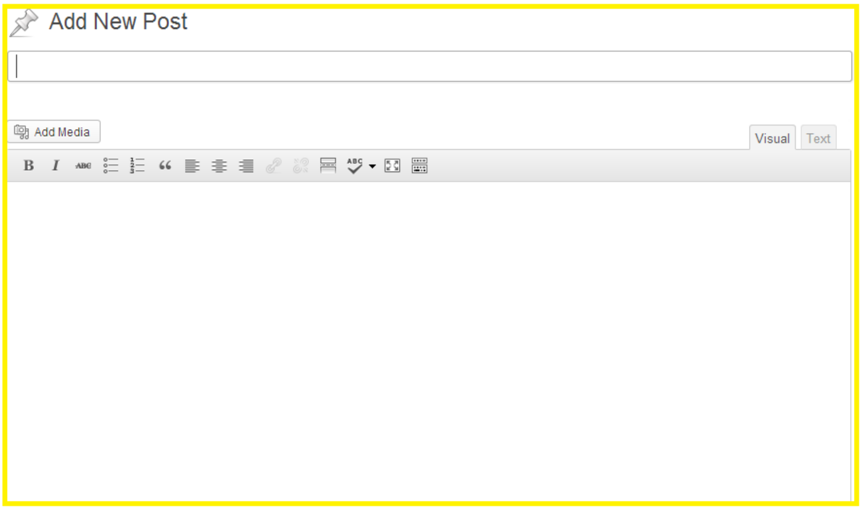


Figure 2.2: Screenshot of the 'new post' page of WordPress administration panel. The user can insert content, images, video and media in general.

Loop, Website title and description

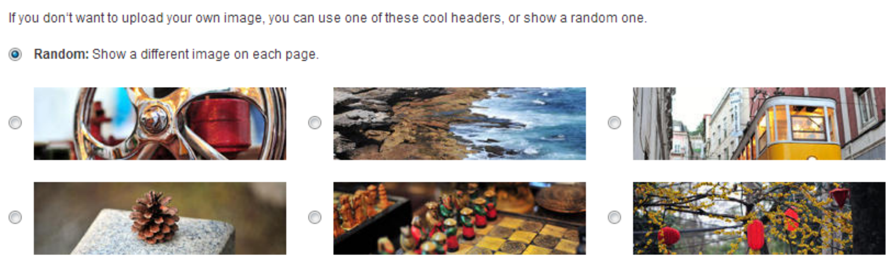
Some elements of a WordPress theme are dynamic. By "dynamic", we mean that they change if their context changes, without modifing something manually. Examples are the loop, the website title and the description.
The loop is PHP code that displays the posts. WordPress processes every post of a blog using the loop[13]. The posts are formatted according to specified criteria within the loop tags; by default the loop displays only the title, the categories and the time stamp of the posts. The loop is a dynamic element because the list of the posts changes if the user deletes, modifies or adds new posts. An example of the loop is the following:

```
<?php if ( have_posts() ) : while ( have_posts() ) : the_post();
endwhile; else:
_e('Sorry, no posts found');
endif; ?>
```

This code prints all the posts written by the users, if there are no posts it shows an error message.
Other dynamic elements are the title and the description of the Website; they are dynamic because, if the users modifies them in the administration panel of WordPress, they change at the same time in the website.
In the WordPress core the main function related to the title and the description of the website is[14]:

- `bloginfo( 'name' )`: displays the name of the website.
- `bloginfo( 'description' )`: displays the description of the website.

[13] http://codex.wordpress.org/The_Loop
[14] http://codex.wordpress.org/Function_Reference/bloginfo



If you don't want to upload your own image, you can use one of these cool headers, or show a random one.

[✓ **Random:** Show a different image on each page.]

Figure 2.3: Screenshot of the ”header” WordPress page in the administration panel. The user can choose one of the image in the list, or in addition, they can decide to show a random one.

Custom header

Custom header is another important element of a WordPress theme. A custom header is a representative image of the blog and it is generally on the top of the theme, in the header section[15].
To use this functionality, the theme has to support custom headers by using the following function of the WordPress core:

- `add_theme_support( 'custom-header' )`[16]

In addition, WordPress allows also the users to have more than one single image, in WordPress can choose randomly the image to display when someone visits the website (cf. Figure 2.3).

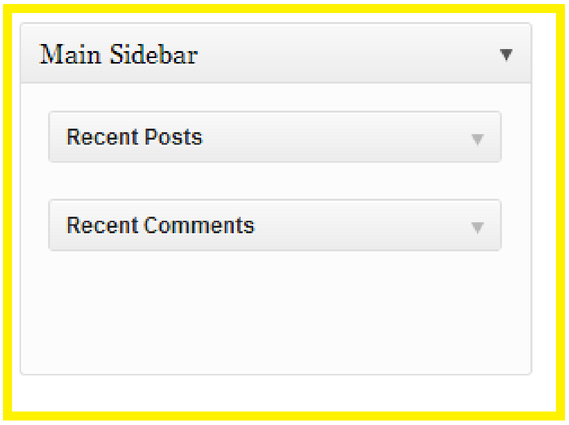
Custom post types

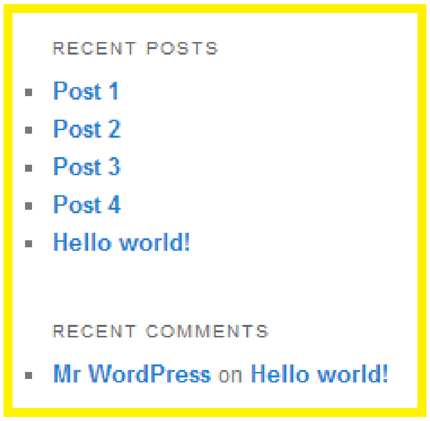
Custom post types are customizable posts that the users can create. The custom post types have a special kind of structure personalized by the user. To use a custom post type it has to be added to WordPress using the *hook* methods and the following function of the WordPress core:

- `register_post_type('name','args')`: register a *name* custom post types[17].

The ”name” parameter is the *name* of the custom post type, while in the *args* parameter the developer can specify different fields of this custom post type.
It is also possible to customize the administration page where the user can insert a new post with a specific custom post type.

[15] http://codex.wordpress.org/Custom_Headers
[16] http://codex.wordpress.org/Function_Reference/add_theme_support
[17] http://codex.wordpress.org/Function_Reference/register_post_type




RECENT POSTS

- Post 1
- Post 2
- Post 3
- Post 4
- Hello world!

RECENT COMMENTS

- Mr WordPress on Hello world!

Figure 2.4: The screenshot of the left side comes from the ”widget” page of WordPress administration panel; it represents the main sidebar of the web sites, inside it there are two widgets. In the left side there is the look and feel of that sidebar in the WordPress website.

Figure 2.5: The screenshot of the left side comes from the ”widget” page of WordPress administration panel; it represents the main sidebar of the web sites, inside it there are two widgets. In the left side there is the look and feel of that sidebar in the WordPress website.

Plugins

Plugins are tools designed to extend WordPress[18]. In fact, one of the main goals of WordPress is to maximize flexibility.
To add a new plugin, the user has to download it (as a zip file) from some resources[19]; after that they can upload the zip archive into WordPress and activate it in the administration panel. There are different kinds of plugins for WordPress, and their goals are pretty heterogeneous. For WordPress this factor is one of the best features of the software.

Sidebar and widget

Sidebars are simply vertical columns in a website (cf. Figure 2.5). Usually sidebars display some information like the main content of the website. Like the custom post types, the sidebar has to be registered with the following function:

- `register_sidebar`[20]

Usually in the sidebar there are also widgets. The widgets are elements that have to be specific functions. Both widgets and sidebars can be easily personalized via the administration panel (cf. Figure 2.5), so it is not necessary for the user to have development skills to modify such parts of a theme.

[18]http://codex.wordpress.org/Plugins
[19]http://wordpress.org/plugins/
[20]http://codex.wordpress.org/Function_Reference/register_sidebar

3 Analysis of Theme Generators

In the last years, many tools were created to help users to develop a WordPress theme. These tools, called theme generators, want to allow non-technical users to create new themes as they desire and in this moment every person can easily find them on the internet. There are different kinds of theme generators and each of them provides different functionalities. In order to develop a WordPress theme generator, we analyzed some of these tools trying to understand their limitations and capabilities. In this Chapter we will first describe the functional and non-functional requirements necessary for a theme generator (cf. Section 3.1). After, we will indicate the theme generators that we studied giving also the reasons why we chose those tools instead of other generators (cf. Section 3.2). Finally, we evaluate these theme generators following the requirements specified before, also giving a classification of them and conclusions of the study (cf. Section 3.3).

3.1 Goals

Common to most theme generators is that they aim to help users create their own themes easily, without having some technical background skills. At the same time, usally a theme generator has to allow the users to modify the theme as they desire also insterting some complex elements.
In this regard, we introduce the main functional requirements that a good theme generator should have. We defined these requirements following the main goals that the common theme generators have or declare to have:

1. **Capacity to customize static element.** The capacity to modify (e.g.: change the color, font and the background) of any theme element.

2. **Capacity to customize the layout.** Capacity to modify the layout of the theme, supporting also the mostly used layouts.

3. **Capacity to customize non-static element.** The users should also insert and customize non-static element (e.g.: functionalities in JavaScript or PHP).

At the same time, a theme generator should have these non-functional requirements:

1. **Usability.** The required technical skills (such as knowledge of HTML, PHP, JavaScript and CSS) should be lower and in addition, the theme generator should be easy to use.
2. **Multidevice.** The theme generator should work well in many devices and screen resolutions.
3. **Reliability.** One of the most important non-functional requirements is that the theme generated by the user should look exactly the same in WordPress.

3.2 Existing theme generators

There are many kinds of theme generators on the internet, with different kinds of requirements and facilities, but, in general, they tend to focus only on the customization of static elements. For this reason we analyzed only some of these tools studying the most used ones.
The theme generators that we evaluated are the following:

1. **WordPressthemegen**[1]
2. **Templatr**[2]
3. **Lubith**[3]
4. **Wpthemegen**[4]
5. **Wpthemegenerator**[5]
6. **Artisteer**[6]

A theme generator analysis was already carried out in the research group in 2011: the analysis took into account the first four theme generators in the list (*WordPressthemegen*, *Templatr*, *Lubith* and *Wpthemegen*). These four theme generators were re-evalueted according to the goals introduced and explained in Section 3.1. In addittion, we decided to analyze two other theme generators. We chose to study **Wpthemegenerator** in order to analyze more complex and non-free theme generators. We studied also **Artisteer** in order to involve in the evaluation two standalone generators (**WordPressthemegen** and **Artisteer**), because this kind of tool is quite rare.
Moreover, we re-evaluated *Templatr*, *Lubith* and *Wpthemegen* because they are quite famous on the internet, and they encompass the most common functionalities of the theme generators

[1]http://wordpressthemegen.com/ accessed at: 04/2013
[2]http://templatr.cc/
[3]http://www.lubith.com/
[4]http://www.yvoschaap.com/wpthemegen/
[5]http://www.wpthemegenerator.com/
[6]http://www.artisteer.com/

which can now be found.
There are many other theme generators available, but it would be impossible and also redundant to study all these tools. For this reason, we tried to study the generators that should represent the most common goals that every WordPress theme generator generally tries to achieve.

3.3 Evaluation

By following the goals that we defined in the Section 3.1, we will now evaluate in detail each theme generator that we showed in Section 3.2. In general, we used each theme generator trying to analyze it as objectively as possible but it is important to note that we will give information that is subjective to the authors. We studied each generator trying to create and customize a theme with these main components and functionalities:

- The website title and description
- The logo of the theme
- A sidebar
- The loop
- A gallery
- A custom header

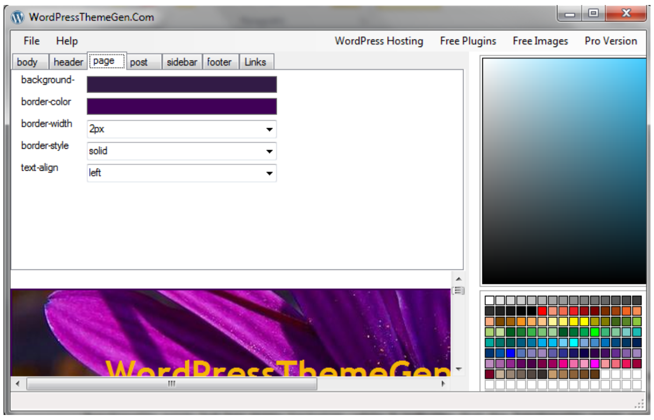
After the creation of a theme with these main components, we tried to export the theme into WordPress also checking how the theme looks on different devices (laptop, smartphone) and screen resolution. We summarize the output of this analisys in Table 3.1. To any generators and requirements we associate a number (from *zero* to *five*) to indicate the evaluation of the functionality: *zero* indicates that the functionality does not exist in the corresponding theme generator, *one* indicates that the functionality exists but has many limitations, while *five* indicates that the functionality exists and it is effective. The *N.D.* (not defined) entry, indicates that it was not possible to judge the generators against the requirement. For example if it was not possible to insert a gallery and a custom header in the theme generator we give a *zero* evaluation for the ”Customize non-static elements” requirement.
As Table 3.1 shows, it is clear that, in general, all the studied theme generators allow the user to customize the basic properties of a theme element. The main difference is how they allow the users to do that. In some of them (*Templatr*, *Wpthemegen*) some technical knowledge is required, while in other generators no background skills are necessary (*Lubith*) giving a high usability. Another important factor, is that only one theme generator (*Wpthemegenerator*) allows the user to manage non-static elements; all the other tools tend to ignore the importance of the dynamic elements of a WordPress theme reducing the potentiality of WordPress itself.
We will now explain in detail each theme generator that we studied following the output of the study resumed in the Table 3.1.

	Functional Requirements			Non-functional requirements		
theme generator	Customize static elements	Customize the Layout	Customize no-static element	Usability	Multi device	Reliability
WordPresstheme	3	0	0	2	0	N.D.
Templatr	4	3	0	3	N.D.	N.D.
Lubith	4	5	0	4	4	5
Wpthemegen	4	2	0	3	N.D.	N.D.
Wpthemegenerator	5	3	2	4	N.D.	N.D.
Artisteer	5	4	0	1	N.D.	1

Table 3.1: Evaluation of the theme generators

3.3.1 WordPressthemegen

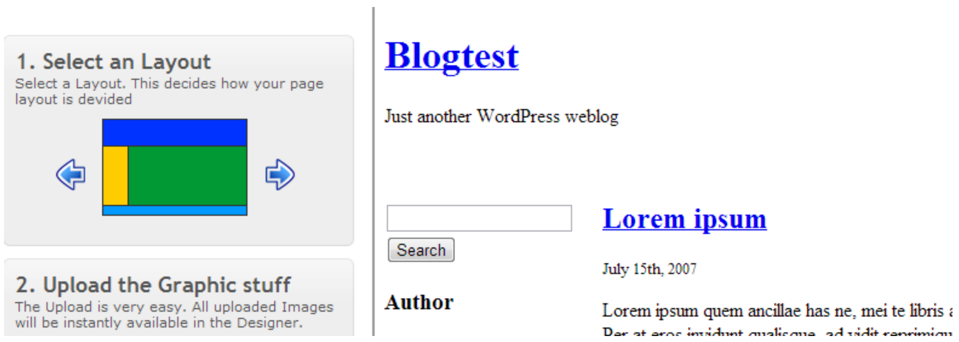
WordPressthemegen[7] (cf. Figure 3.1) is a free standalone theme generator. Like *Artisteer*, we analyzed them because there are only few generators that work as a standalone; it is interesting to understand how these kinds of tools work and in what they are different compared to the web-based generators. A *professional version* of *WordPressthemegen* is also available but, in this thesis we analyzed only the free version.
Table 3.1 shows that with *WordPressthemegen* it is not possible to modify the layout. In this theme generator the user can use only one kind of layout decided by the developers. Moreover, the users cannot manage complex elements (such as sidebar, custom header etc.) they can just modify the font, the background color and other basic properties of static elements, such as the title of the web sites, the logo and content.
Therefore, also non-technical users can develop a theme with this generator, but the tool was not easy to use: during the analysis, the generator crashed many times losing all the progress done. In addition, it was not possible to see if the theme created by the standalone looked the same in WordPress, because the theme generated by the tool does not work on WordPress itself.



background-
border-color
border-width 2px
border-style solid
text-align left

Figure 3.1: Screenshot of the WordPressthemegen tool.

[7] http://wordpressthemegen.com/ accessed at: 04/2013



Select a Layout. This decides how your page layout is devided

The Upload is very easy. All uploaded Images will be instantly available in the Designer.

Just another WordPress weblog

Lorem ipsum quem ancillae has ne, mei te libris

Figure 3.2: Screenshot of the Templtr home page

3.3.2 Templatr

Templatr[8] (cf. Figure 3.2) is a free web-based tool. It is quite famous; in fact, there are many tutorials on the internet viewed by many people who want to know more informations and tricks of this tool[9]. Table 3.1 shows that the users can customize static elements with this generator; more in detail, with this tool the users can change many proprierties of the main elements of a theme, giving the opportunity to personalize very well the look and feel of the web sites.
In addition, the user can choose twenty different layouts. *Templatr* is easy to use, but in order to customize the fine details of a theme, the users have to know the basic concepts of HTML (e.g. the tag *p*, *div*, *a* etc.).
According to the developer of *Templatr*, the theme created by the tool should work on different devices, but it was not possible to test this functionality because, during the analysis, the theme did not work on WordPress. For the same reason, it was not possible to see if the theme created with the tool will look the same on WordPress.

3.3.3 Lubith

Lubith[10] (cf. Figure 3.3) is a free web-based generator which can create themes for WordPress. Lubith is different from the other theme generators, because of the interaction of the drag and drop used to customize and move each component of the theme. *Lubith* is quite famous, in fact, many articles talk about this tool, and many of these are written by web developers[11]. For these reasons we decided to study this generator.
As the Table 3.1 suggests, with *Lubith* the users can modify many proprerties of each element of the theme. At the same time, the users can have a virtually infinite number of possible layouts, thanks to the use of the drag & drop. With the drag & drop, in fact, the users can move each component in any position of the theme. As with the other generators, it is not possible to customize complex elements, and it is only possible to manage the content, the loop, the sidebar and the title of the theme.
Lubith is easy to use and the users do not need techical knoledge to develop a theme and the theme created by the tool looks exactly the same in WordPress.

[8]http://templatr.cc/
[9]http://www.youtube.com/watch?featurep̄layer_embedded&vh̄aK6UNAFO_s
[10]http://www.lubith.com/
[11]http://www.stunningmesh.com/2013/03/lubith-the-easy-way-to-make-wordpress-themes/

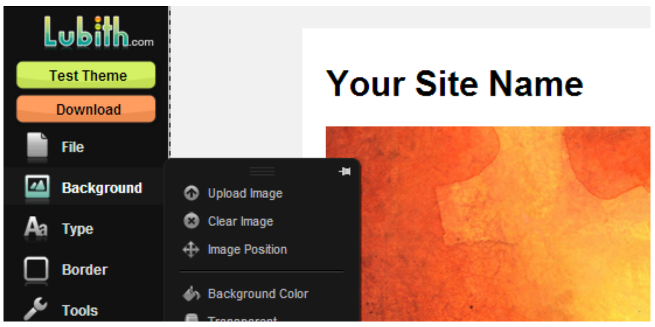


Figure 3.3: Screenshot of the Lubith generator.

3.3.4 Wpthemegen

Wpthemegen[12] (cf. Figure 3.4) is another web-based theme generator. This tool looks like *Templatr*; they have almost the same functionalities, but while *Templatr* has more possible layouts, this is not true in *Wpthemegen* where there are only four possible layouts.
Like *Templatr*, it is not possible to customize more complex elements (cf. Table 3.1), but only static components; in addition it was not possible, during the analysis, to see the theme generated, because the theme archive did not work on WordPress. For the same reason it was not possible to say if the theme will work on different devices and screen resolutions.

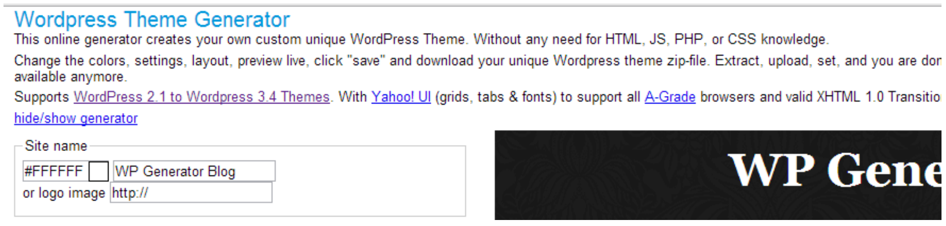
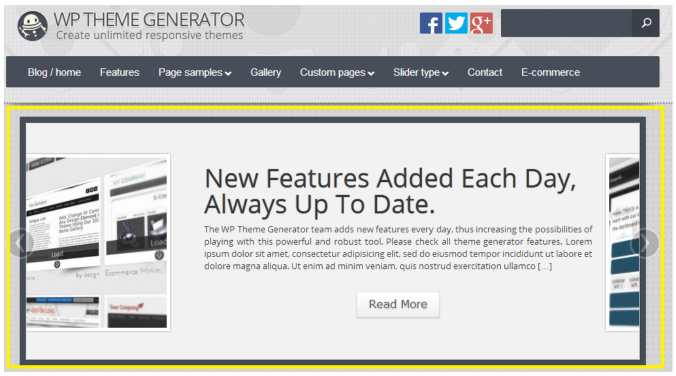

Wordpress Theme Generator
This online generator creates your own custom unique WordPress Theme. Without any need for HTML, JS, PHP, or CSS knowledge.
Change the colors, settings, layout, preview live, click "save" and download your unique Wordpress theme zip-file. Extract, upload, set, and you are dor
available anymore.
Supports WordPress 2.1 to Wordpress 3.4 Themes. With Yahoo! UI (grids, tabs & fonts) to support all A-Grade browsers and valid XHTML 1.0 Transitio
hide/show generator
Site name
#FFFFFF WP Generator Blog
or logo image http://
WP Gene

Figure 3.4: Screenshot of the Wpthemgen page.

3.3.5 Wpthemegenerator

Wpthemegenerator[13] (cf. Figure 3.5) is a non-free web-based theme generator. *Wpthemegenerator* allows the users to choose from different types of themes (eg. dark theme, blue theme etc.), from which the users can personalize the elements.
The tool permits the users to change the properties of each element and they can also decide the layout (from a list of layouts decided by the generator). The difference between this generator and the other is that, for each element, the users can choose different kinds of borders and blur effects. As the Table 3.1 shows, another important difference is that *Wpthemegenerator* also allows the users to customize more complex elements. With this tool it is possible to add and customize a gallery and a slider (an example of a slider is in Figure 3.5). Unfortunately, it is not possible to add other kinds of functionalities. The tool is not difficult to use, but the features are not well organized, so at first it is confusing to use the generator.

[12] http://www.yvoschaap.com/wpthemegen/
[13] http://www.wpthemegenerator.com/



The WP Theme Generator team adds new features every day, thus increasing the possibilities of playing with this powerful and robust tool. Please check all theme generator features. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco [...]

Figure 3.5: Screenshot of Wpthemegenerator website. The element highlighted in the Figure is a slider, with the arrows the user can see different images and information of the website.

It was not possible to try the theme generated because it is necessary to buy the tool to download the theme.

3.3.6 Artisteer

Like *WordPressthemegen*, *Artisteer*[14] (cf. Figure 3.6) is a free standalone tool to create WordPress themes. *Artisteer* uses an interface similar to the Microsoft *Ribbon interface*.
As the Table 3.1 indicates, with the tool the users can modify each element of a theme altering many properties. It is also possible to choose from various layouts, but it is not possible to drag & drop each element like *Lubith*.
The tool is easy to use, but there are severla bugs and so finishing a theme, during the analysis, becomes difficult and frustrating. Moreover, unfortunately the theme exported in WordPress does not look like the theme created in the tool. For the same reason, the theme looks different on other types of device or screen resolutions.

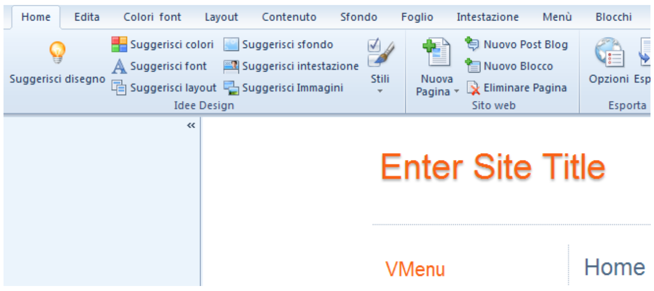


Figure 3.6: Screenshot of Artisteer

3.3.7 Conclusion and Classification

The types of theme generators are different, and each of these tools has different goals and functionalities, but, as we showed in Section 3.3, many of these generators tend to allow the

[14]http://www.artisteer.com/

users only to customize static elements. Moreover, none of them is developed in WordPress, and, because of this, it is not possible to perform some important tasks (e.g. : display the real name of the website during the design of the theme). For this reason, sometimes the theme generated by the tools does not work properly on WordPress most probably because of compatibility problems between various WordPress versions.
The Figure 3.7 shows the classification of the theme generators. We divided the tools into three big categories: the tools where the users can personalize static elements, the tools where it is possible to modify static elements and modify the layout of the theme, and the tools where it is possible to personalize static elements, change the layout and manage more complex elements. This classification was done after the study described in Section 3.3 and summarized in Table 3.1.
We classify a tool in the first category if it only allows static theme elements to be customized. *Wpthemegenerator* is the only theme generator that belongs to the third category; with this tool, the users cannot modify the layout or more complex elements, and can only change the font and background color of the website title or the loop. In addition they cannot choose the layout of the theme, as *Wpthemegenerator* allows only one kind of structure.
In the second category, we have all the other tools (*Lubith*, *Templtr*, *Artisteer* and *Wpthemegen*) with the exception of *WordPressthemegen*. We classify the tools in this category if they allow the users to change properties of an element and the theme layout but at the same time, they do not allow the users to manage more complex elements. The main difference between the tools in this category is the way they allow the users to do these tasks.
Wpthemegenerator is the only tool that belongs to the third category. We classify this tool in this category, because with the generator, it is possible to customize more complex elements such as a gallery.
This categorization helps us develop *X-TG* in order to avoid the main problems of these tools and achieve the main qualities that they have. *X-TG* was developed to be in the third category aiming to allow the users to personalize each static element as they desire but at the same time give the chance to customize more complex elements (e.g.: custom header, sidebars etc.). Like *Lubith*, we wanted to use the the drag & drop interactionto build the theme structure.

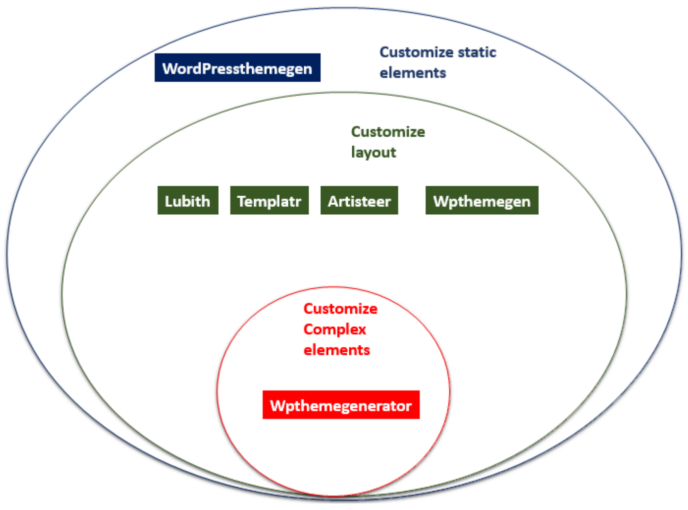


Figure 3.7: WordPress theme generator classification.

4 X-Themes generator

X-TG is a WYSIWYG theme generator for WordPress in WordPress itself; it allows the users to create their own theme also customizing complex elements and properties. The theme generated by *X-TG* follows a well-formed metamodel to permit the reuse and, at the same time, the presence of this structure is completely transparent to the users. In this Chapter we will explain *X-TG*'s main functionalities and its architecture.
In order to help the reader to better understand the tool, we will first describe an use case scenario of *X-TG* in Section 4.1. After, we will give some *X-TG* basic concepts regarding the user interface and the metamodel (cf. Section 4.2). In Section 4.3, we will explain the *X-TG* design process. Finally, in Section 4.4, we will explain the architecture and the implementation of the tool talking also about the trade offs done during the development.

4.1 Use scenarios

In this Section we will do a brief introduction of the *X-TG* interface and its usage, trying to explain the main functionalities and the interactions used by the tool.
The users can access *X-TG*, after the installation, going on their WordPress administration page and click on the ”X-Themes” tab, and then they can simply click on the ”Create your theme” link. The system will show now the *X-TG* interface (cf. Figure 4.1).
In order to help the readers to better understand the tool and its usage, we dived the explain-ation of *X-TG* by its main functionalities. In the Section 4.1.1 we will explain how the users can manage the containers and how they can style the container itself. In Section 4.1.2 we will introduce how the users can reuse their functionalities in the generator and how they can manage them by using *X-TG*.

4.1.1 Usage and style of the containers

Each element in *X-TG* is a container. There are different kinds of containers but we can see a container as a box the users can customize by selecting them. As Figure 4.1 shows, the users

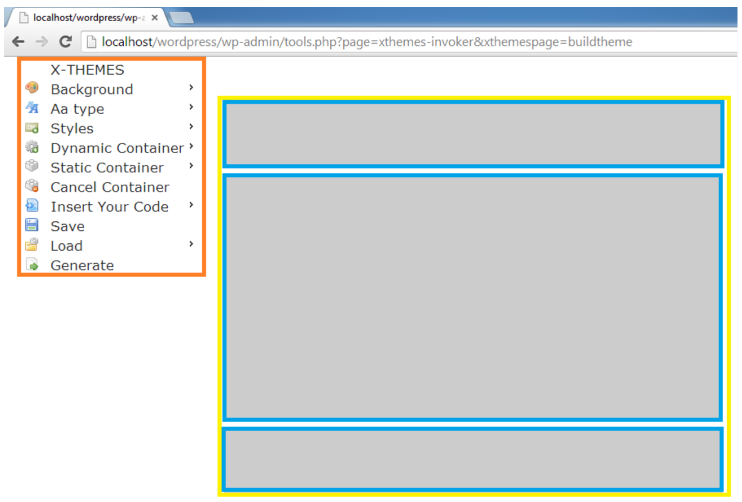


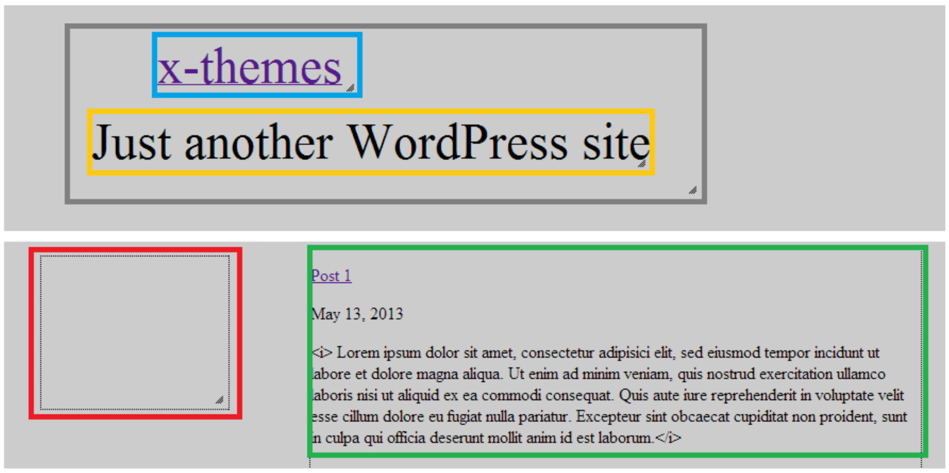
Figure 4.1: Screenshot of X-TG. The menu of the tool is highlighted in orange, while the theme designed by the users is highlighted in yellow. The main containers (header, body and footer) of the theme are highlighted in blue.

can start their design of a theme from the three main containers: the header, the footer and the body. The users can resize these main containers by selecting the border of the box and dragging it.
There are many ways to add a container in *X-TG*: the users can add an empty container by clicking on "Static container" and after on "Add a container" button on the *X-TG* menu, and in this case the system will create an empty box. Another way is also instering dynamic elements (e.g.: website title or the description, the loop and more; to have more information about static and dynamic elements see Section 4.2.2), but in this case the system will create a box with some information: this information depends on the elements the user wants to insert. Figure 4.2 gives some examples of containers.
Independently from the type of the container and the way the users insert it, the system will create this box in the cointainer selected by the user at the moment of the creation.
The users can drag & drop and resize each container in any position of the theme with the exception of the main contaniners. A cointainer can contain other containers (cf. Figure 4.2) and when the users drag & drop the parent container, all the elements inside will be dragged aswell. In the same way, the users can drag & drop containers in other containers.

The users can style each element of the theme. In *X-TG* it is possible to change basic properties of a container (e.g.: background and font color, font size and type) using the appropriate function in the left menu. The *X-TG* also allows the users to insert their own CSS; clicking on the "Insert your CSS rule" the system will show a dialog box where the users can insert their own CSS to the element selected.

4.1.2 Upload your code and generate the theme

In order to allow the users to insert more complex elements in the theme, *X-TG* permits to insert external code. The users can upload their code on the tool and after, they can manage


x-themes

Just another WordPress site

Post 1

May 13, 2013

<i> Lorem ipsum dolor sit amet, consectetur adipisici elit, sed eiusmod tempor incidunt ut abore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco aboris nisi ut aliquid ex ea commodi consequat. Quis aute iure reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint obcaecat cupiditat non proident, sunt n culpa qui officia deserunt mollit anim id est laborum.</i>

Figure 4.2: Screenshot of X-TG tool. In this theme there are some example of containers. The container highlighted in blue is a dynamic container that shows the real name of the user WordPress website. The cointainer highlighted in orange is the website description. These two containers are in an other container highlighted in grey. The box highlighted in red is a static empty container while the element highlighted in green is the loop.

it as well as they manage all the other containers.
In *X-TG* there are two ways to insert the code. The users can upload their code via a .zip package file that has a XML file following our metamodel (for more details about this kind of file see Section 4.2.1), or they can insert different files. As the Figure 4.3 suggests, if the users do not have the .zip package they can choose the second option in the first dialog. After that, the users can insert their code (in PHP and HTML) in the textarea of the second dialog; they can also chose the type of the code they are going to insert (custom header, sidebar or general). Once the users have inserted this information, they can upload their files (JavaScript, PHP, CSS and other) using the appropriate buttons in the third dialog.
The system will upload all the files and insert in the theme all the informations that are necessary for them. The users can, in this way, reuse their own code or simply find code of other developers.

When the users are satisfied of their work, they can press on the button ”Generate” on the left menu and generate the theme. The users can see the result on their website and they are free to change properties of the theme in the WordPress administration panel.

4.2 Concept

X-TG works on WordPress and the users can develop a theme using the interface explained in Section 4.1, but at the same time the theme generated follows a well-formed metamodel to allow the reuse beetween X-Themes. In this chapter we will introduce to some fundamental concepts of *X-TG*: we will first explain the metamodel and its main components; after we will talk about the differences of dynamic and static containers for our tool specification and

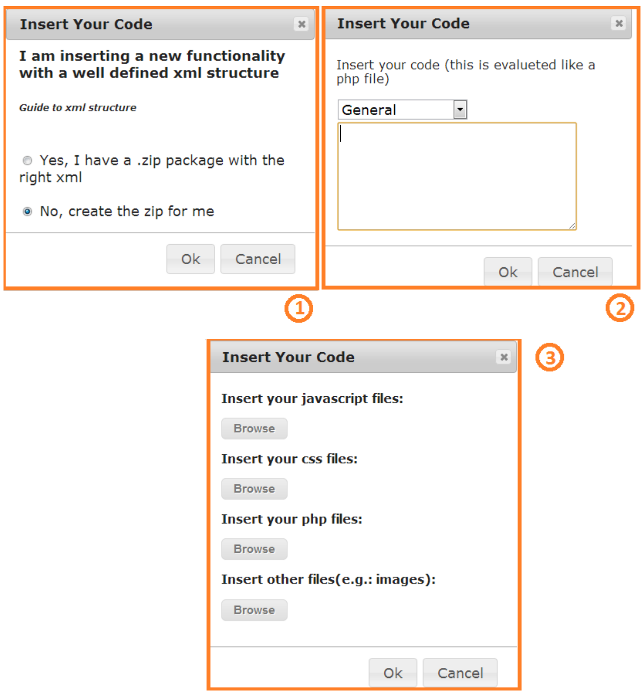

Insert Your Code
I am inserting a new functionality with a well defined xml structure
Guide to xml structure
Yes, I have a .zip package with the right xml
No, create the zip for me
Ok
Cancel
1
Insert Your Code
Insert your code (this is evalueted like a php file)
General
Ok
Cancel
2
Insert Your Code
Insert your javascript files:
Browse
Insert your css files:
Browse
Insert your php files:
Browse
Insert other files(e.g.: images):
Browse
Ok
Cancel
3

Figure 4.3: Screenshots of the ”Insert your code” dialogs.

interface. Finally, we will explain in detail the basic concepts of the reuse of functionalities that *X-TG* has.

4.2.1 X-Themes metamodel

We will explain now the X-Themes metamodel that is followed by the *X-TG* theme generated. We decided to develop this kind of model in order to allow a semplified reuse between themes giving a good and well-formed structure to each element of a WordPress theme using an XML format. Figure 4.4 shows a general structure of a WordPress theme and its main elements and the links betweem them; from that structure we created a new model (cf. Figure 4.5) with the goal of creating a metamodel that helps the reuse between different X-Themes.
An X-Theme has three different types of elements: **Components**, **Layout components**, and **Custom Types**. In this weok we focus only on Components and Layout components. In this Section we will describe in detail each of these elements, and at the end we will define the whole X-Theme.

Components

A component is a theme element that can contain style rules, JavaScript and PHP functionalities. Component is stored in a seperate directory in the theme directory. A component has:

- the JavaScript, PHP and CSS needed from the component;
- the info.xml file with the metamodel information (cf. Figure 4.6).

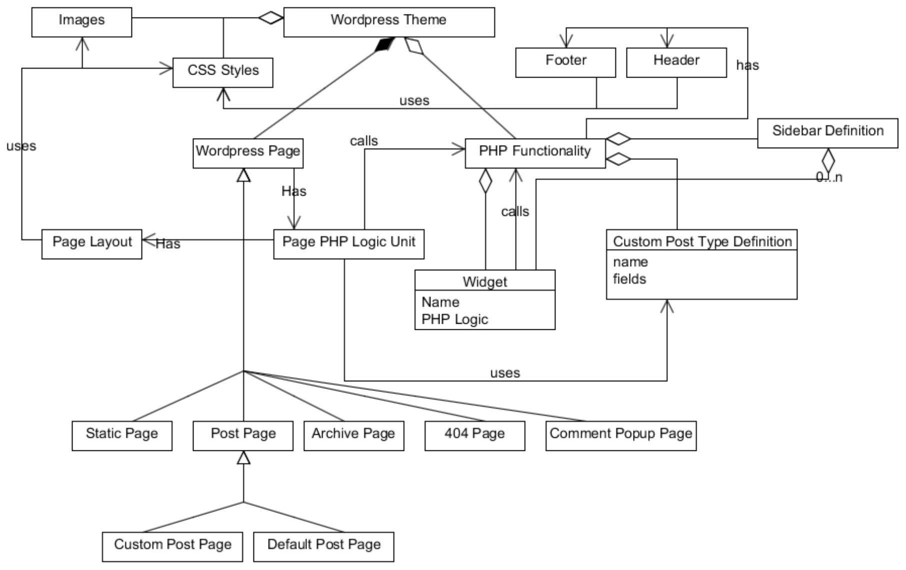


Figure 4.4: WordPress theme structure

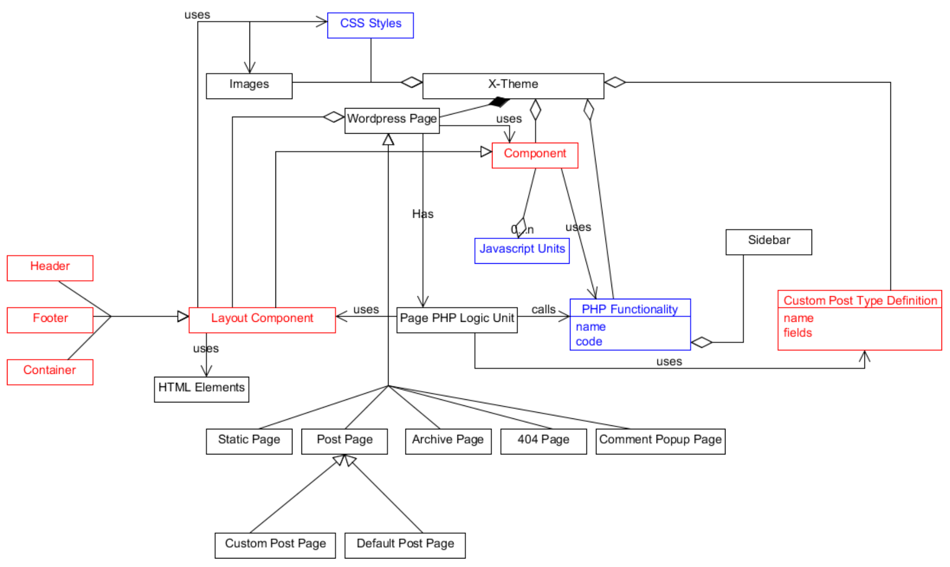
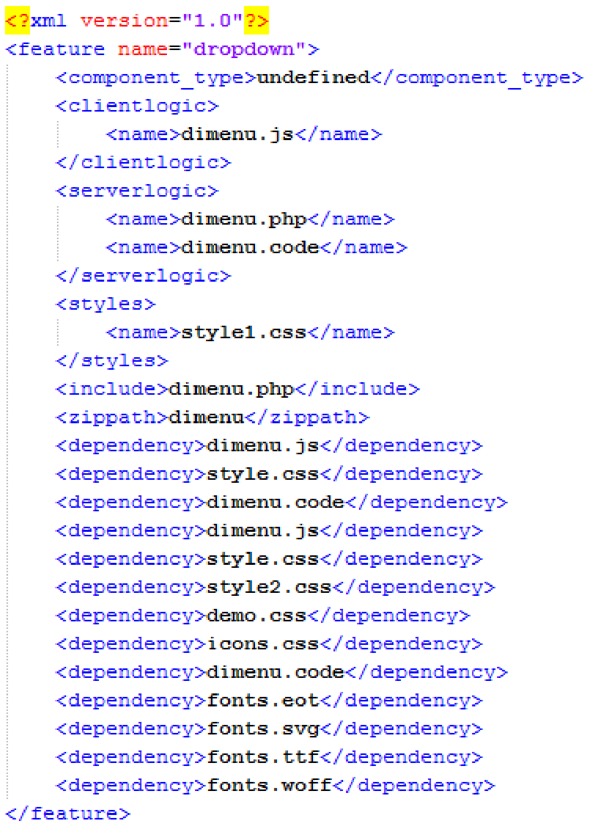


Figure 4.5: The X-Themes metamodel. The entries highlighted in red are the elements that can be reused between X-Themes and the elements highlighted in blue have to be exported as a consequence.

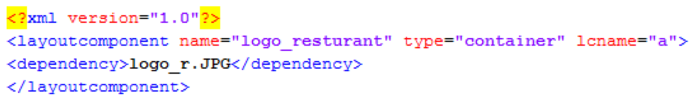


```xml
<?xml version="1.0"?>
<feature name="dropdown">
    <component_type>undefined</component_type>
    <clientlogic>
        <name>dimenu.js</name>
    </clientlogic>
    <serverlogic>
        <name>dimenu.php</name>
        <name>dimenu.code</name>
    </serverlogic>
    <styles>
        <name>style1.css</name>
    </styles>
    <include>dimenu.php</include>
    <zippath>dimenu</zippath>
    <dependency>dimenu.js</dependency>
    <dependency>style.css</dependency>
    <dependency>dimenu.code</dependency>
    <dependency>dimenu.js</dependency>
    <dependency>style.css</dependency>
    <dependency>style2.css</dependency>
    <dependency>demo.css</dependency>
    <dependency>icons.css</dependency>
    <dependency>dimenu.code</dependency>
    <dependency>fonts.eot</dependency>
    <dependency>fonts.svg</dependency>
    <dependency>fonts.ttf</dependency>
    <dependency>fonts.woff</dependency>
</feature>
```

Figure 4.6: Example of info.XML file of a component.

Figure 4.6 shows an example of info.XML file. The tag *<feature>* has many child tags:

- All the client side files are in the tag **<clientlogic>**; The tag *<name>* inside it indicates the JavaScript file name;
- The same happens for the PHP files that are embedded in the**<serverlogic>** tag;
- And the CSS files are in the **<styles>** tag.
- The **<include>** defines the file that has to be included into the theme.
- The **<component_type>** describes the type of the component (custom header, header or general).
- The file that has to be included into the *functions.php* file (like registration call for the sidebar of the theme)is in the **<functionspointer>** tag.
- **<zippath>** tag indicates the temporary directory of the component.
- The component contains a tag *<dependency>* and in the example the component needs a various file to work correctly.



```xml
<?xml version="1.0"?>
<layoutcomponent name="logo_resturant" type="container" lcname="a">
<dependency>logo_r.JPG</dependency>
</layoutcomponent>
```

Figure 4.7: An example of info.XML file for a layout component.

Layout components

The layout components define the structure of the theme. These elements are a specialization of components, because they contain only static elements or dynamic theme elements but they cannot contain JavaScript or user defined PHP code. We can divide layout component in three type:

- **Container**: it is a generic block and it can contain other containers and markup elements;
- **Header**: the header of the website, it can contain containers other containers;
- **Footer**: the footer of the website, like the header it can contain containers;

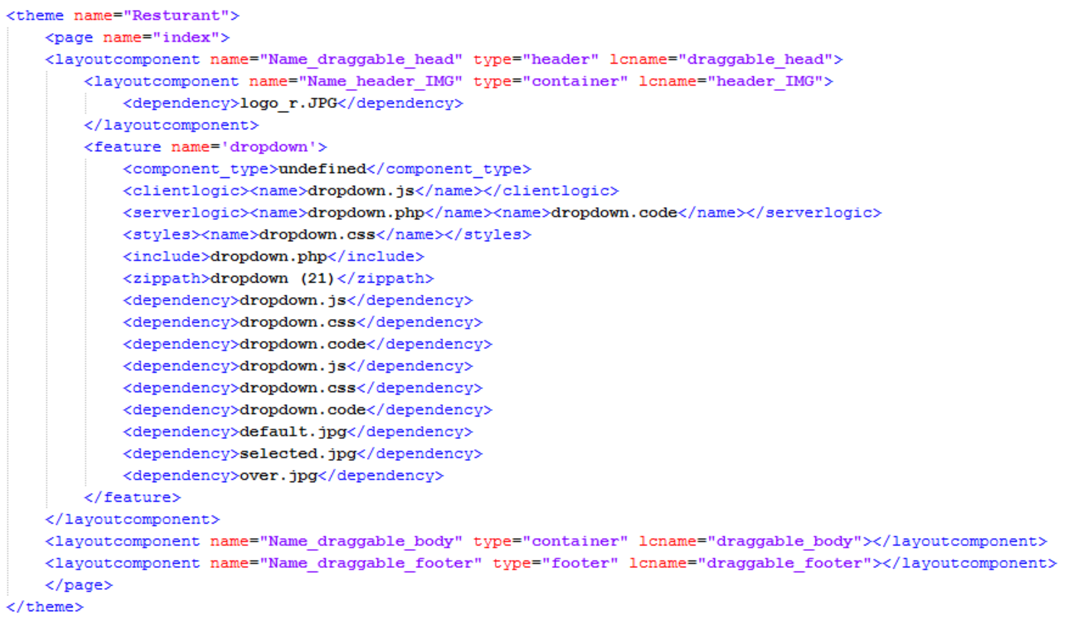
We divided the header and footer from the container because they are special elements; usually they are shared between the pages of a web application. Moreover, in WordPress the header and the footer are in different files. Each layout component of an X-Theme is in a separate directory and for each of them it contains;

- a **PHP** file that has the code of the layout component;
- a **CSS** file that has the style rules of the laytout component;
- a **info.XML** file that has the metamodel information used by *X-TG* to generate the component.

If a component has other components inside itself, *X-TG* will generate the inner elements seperately and the parent element will `include` the children code. Figure 4.7 shows an example of a layout component info.XML file. The tag *<layoutcomponent>* includes the declaration and it has three attributes. The *name* attribute reppresents the name of the layout component. The *type* attribute indicates the type of the layout component (header, footer or container). The attribute *lcname* is the name gived by the users on *X-TG*. The component contains a tag *<dependency>* and in the example the component needs a JPG file to work correctly.

Theme

After we showed all the single elements of an X-Theme, now we describe the strucure of a whole theme. Figure 4.8, shows an example of the XML structure of a theme. In the example, *X-TG* creates a theme with the name *Resturant*; the theme has various compontents inside itself (like a component called *dropdown*). There is also a tag <**page**> that was developed to support eventually the possibility to have different designs for different WordPress pages.



```xml
<theme name="Resturant">
    <page name="index">
    <layoutcomponent name="Name_draggable_head" type="header" lcname="draggable_head">
        <layoutcomponent name="Name_header_IMG" type="container" lcname="header_IMG">
            <dependency>logo_r.JPG</dependency>
        </layoutcomponent>
        <feature name='dropdown'>
            <component_type>undefined</component_type>
            <clientlogic><name>dropdown.js</name></clientlogic>
            <serverlogic><name>dropdown.php</name><name>dropdown.code</name></serverlogic>
            <styles><name>dropdown.css</name></styles>
            <include>dropdown.php</include>
            <zippath>dropdown (21)</zippath>
            <dependency>dropdown.js</dependency>
            <dependency>dropdown.css</dependency>
            <dependency>dropdown.code</dependency>
            <dependency>dropdown.js</dependency>
            <dependency>dropdown.css</dependency>
            <dependency>dropdown.code</dependency>
            <dependency>default.jpg</dependency>
            <dependency>selected.jpg</dependency>
            <dependency>over.jpg</dependency>
        </feature>
    </layoutcomponent>
    <layoutcomponent name="Name_draggable_body" type="container" lcname="draggable_body"></layoutcomponent>
    <layoutcomponent name="Name_draggable_footer" type="footer" lcname="draggable_footer"></layoutcomponent>
    </page>
</theme>
```

Figure 4.8: An example of the XML definition of a theme.

4.2.2 Dynamic and static containers

As we introduced in Section 4.1.1, in *X-TG* from the user interface side we can see every element of the theme as a box, and we dived these boxes in two types: dynamic and static containers. For the *X-TG*'s user interface, a static container is a container that has no elements that can change dynamically, while this is not true for the dynamic container. In Figure 4.5, we can see an example of dynamic and static container: at left we have a container with a background image, while at right we have the container with website description. The website description is a dynamic container because if the user will change the description of the website in the WordPress administration page, this container will change. More in general, we can say that a container is dynamic if it contains PHP functions inside itself; the website description container has a PHP code (hidden to the user) which calls a WordPress core function to read the description of the website. The cointainer on the left in Figure 4.9, is a static container because it does not contain PHP code but only HTML and CSS.

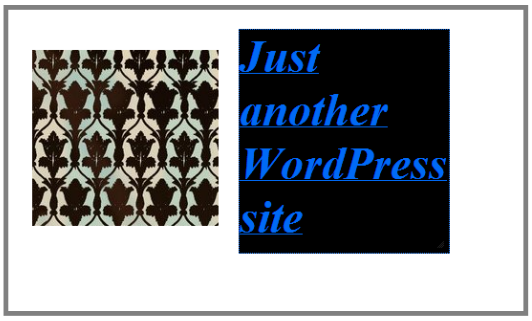
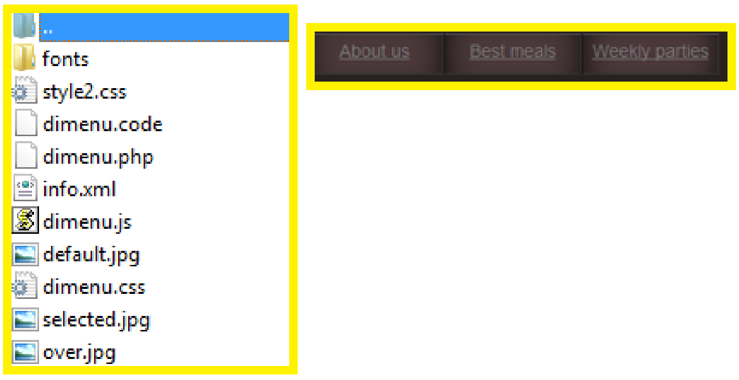


Figure 4.9: Example of a static and a dynamic container



..
fonts
style2.css
dimenu.code
dimenu.php
info.xml
dimenu.js
default.jpg
dimenu.css
selected.jpg
over.jpg

Figure 4.10: Example of a package file. On the left the structure of the zip file, on the right an appareance example of the functionality in X-TG

4.2.3 Reuse of code

As explained in Section 4.1.2, in *X-TG* the users can insert their own code. This code will be inserted in the theme and its output will go in the container selected by the users. The users can choose whether upload a single package file or insert more different files.
The package file must have a XML file inside itself; the Figure 4.10 gives an example of the structure of this package. The .zip file contains various JavaScript, CSS and an info.XML file. The info.XML file looks exactly the same of the XML we showed in the Figure 4.6. If the users does not have the XML file, they can simply insert each file at time, and at the end, the system will create the XML file for them. In this way the users can share between them different functionalities. It is important to note that if the files uploaded by the users do not work well *X-TG* cannot understand it and it is a task of the users know if the code work properly.

4.3 Design Process

In this Section we will discuss about the *X-TG* design process, also indicating some trade offs we done in order to achieve the goals of the thesis. *X-TG* is a theme generator that creates a WordPress theme following a meta-model (cf. Section 4.2.1). Another tool was developed in order to permit reuse between themes, and this tool is called *XDE* (cf. Murolo [7]).
In Figure 4.11 we can see the general design process that we followed during this work. One of the first steps was to analyse the existing theme generators in order to better understand their functionalities and limitations (cf. Section 3). From this evaluation and the analysis of theme definitions (done by Murolo [7]) we worked on the meta-model. From this model we were able to develop two tools *XDE* and *X-TG*.
We decided to develop a well-formed meta-model in order to allow an organized and simplified reuse between themes also thanks to the possibilities to store metadata. During the tools' development process we changed some parts of this meta-model (dashed arrows in Figure 4.11) in order to cover all the WordPress functionalities that we wanted to support and that we did not plan in a first step.

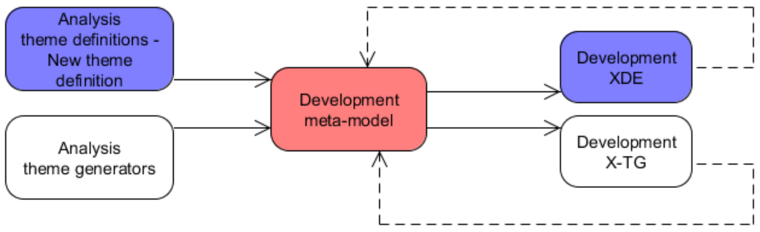


Figure 4.11: General design process of X-TG and XDE. The state colored in blue refers to XDE, the state colored in white refers to X-TG and the state colored in red refers to X-TG and XDE.

In Figure 4.3 we can see the detailed concurrent development process between *X-TG* and *XDE*. After the first implementations steps (such as a first *X-TG* user interface prototype) we started to support new elements (e.g.: the loop, web site title and description) in *X-TG*. When we added an element, we tested it by genereting the theme; when the theme generated by *X-TG* did not look the same as the theme developed in the user interface (cf. Figure 4.1) or, more in general, it did not work properly, we debbuged the tool and modified it. In the case that the element added worked properly, Murolo [7] started to support the drag & drop of that element as well as testing it; in the meantime we started to add a new element in *X-TG*.

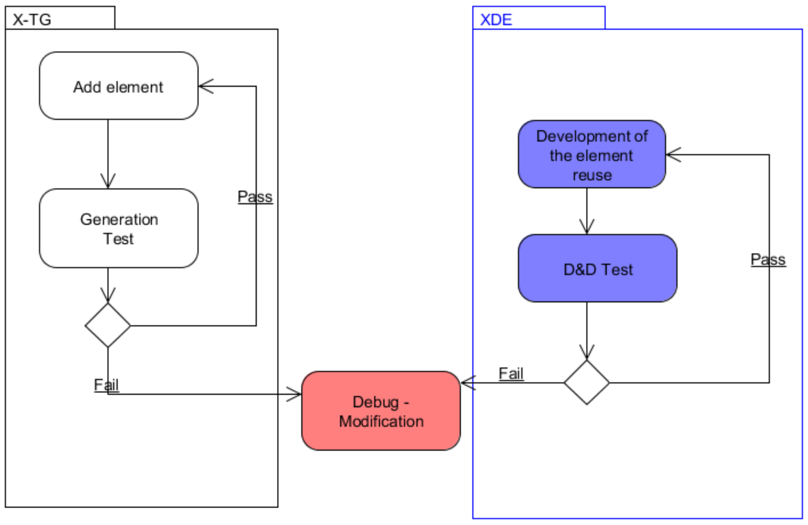


Figure 4.12: Concurrent development process of X-TG and XDE. The state colored in blue refers to XDE, the state colored in white refers to X-TG and the state colored in red refers to X-TG and X-DE.

Figure 4.13 represents the development process that is strictly related to *X-TG*. At first, we started implementing the user interface using JQueryUI, but we decided to not use this JavaScript plugin in order to satisfy our goals (see Section 4.5.1). For this reason, we started

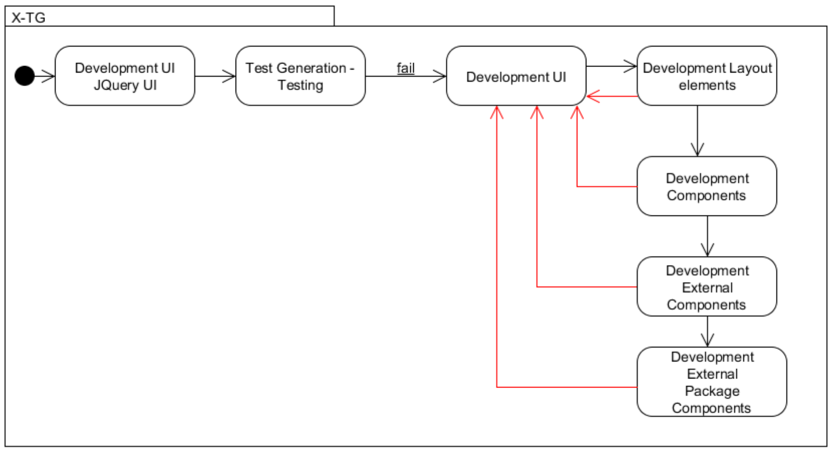

X-TG
Development UI JQuery UI
Test Generation - Testing
fail
Development UI
Development Layout elements
Development Components
Development External Components
Development External Package Components

Figure 4.13: X-TG development process

to develop our own implementation for the drag & drop and for the user interface more in general. Than, we started to implement at first the layout elements (cf. Section 4.2.1) then the component elements. We started with the layout elements, because these are easier than components and in this way we tested soon the X-TG capabilities. For each element we added to *X-TG*, the development process follows the structure we given in Figure 4.5.
After the layout components, we decided to allow users to insert their own code (component elements). In this way the users can customize more complex elements, and they can insert easily their own functionalities without changing them. Finally, we wanted to help the users to insert their code giving the chance to upload a single archive file or more files. With this archive file, the implementation details are hidden to users; they can share these packages increasing the number of functionalities in the theme, and users can do this without having any technical knowledge. As Figure 4.13 shows (red arrows), adding a new element involved the modification of the user interface in order to allow users to exploit these new functionalities.

4.4 Architecture and X-Theme Generation Process

In this Section we will introduce to the *X-TG* architecture, trying to explain the responsibilities of each node and subsystem as well as explaining our x-theme generation process.
Figure 4.14 shows the deployment diagram of *X-TG*. *X-TG* works on two separates nodes which collaborate: the browser node and the server node. The browser node contains two main components: the user inferface logic that manages the interaction between the users and the system and the metamodel generation logic that ”transforms” the theme developed by the users in an X-Theme. The server node contains three components: the component separation logic, the filter logic and the XML theme generation component. These three components work together to create a working WordPress X-Theme.
We will describe now in detail, each subsystem of *X-TG* trying also to indicate the interaction between them.

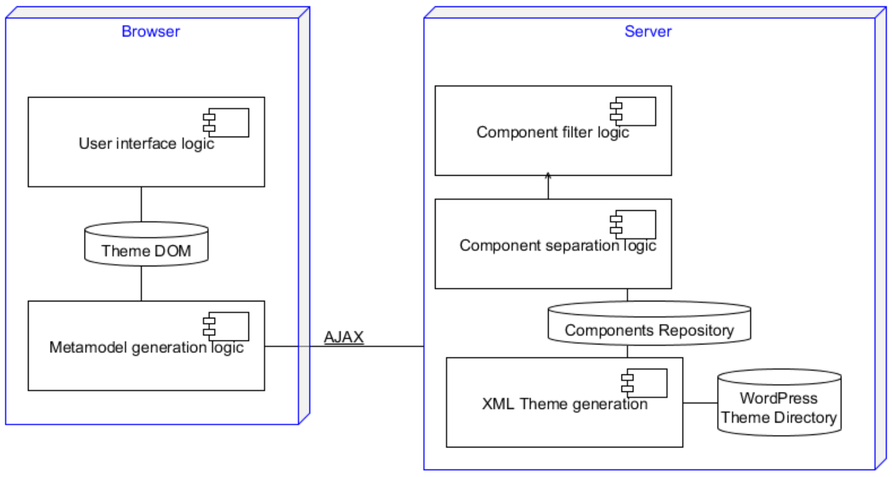


Figure 4.14: Deployment diagram of X-TG.

4.4.1 Browser node

The browser node contains the user interface of *X-TG* and its logic. The *user interface logic* has the responsibilities to allow the users to develop their own theme, it has to keep track of the modification done by the users on the theme and it has, also, to display correctly the interface itself. The user interface logic contains all the necessary code to the drag & drop the container; it also contains the functionalities which allow the users to change the properties of the container itself. While the users develop their theme, they are modifing the DOM of the page and its properties. As the Figure 4.14 shows, the user interface logic has access to the theme DOM data and when they change some CSS rules to a container the user interface logic insert a style inline rule to that box.
We have an example of this process in Figure 4.15. The user creates a container called ”parent” in *X-TG*, inside this container the user inserts two other containers: a ”logo_container” and the website description container. The ”logo_container” is an empty container with a background image while the website description is simply the description of the website.
A simplified reppresentation of this container DOM is a tree with the root as the ”parent” container that has two children the ”logo_container” and the website description. This DOM will have the inline style rules for each node, for example, the ”logo_container” node of the DOM will contain the rule `background-image:logo.png`.
When the users are satisfied of their work, they can click on the button ”Generate” asking the system to generate the theme. The ”user interface logic” receive this message from the user and comunicates with the *metamodel generation logic* to start the generation of the X-Theme giving to this component the DOM of the theme.

Only at this moment the *metamodel generation logic* starts its work. Like the interface component, also the *metamodel generation logic* can access the theme DOM (cf. Figure 4.14).
The *metamodel generation logic* has the responsibility to ”trasforms” the theme created by the users in an X-Themes without chaging anything of the theme look & feel and

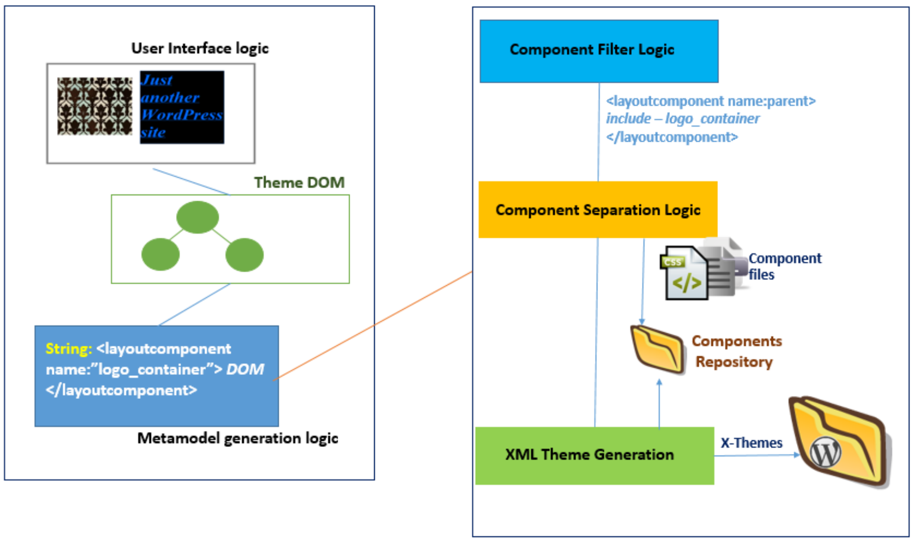

User Interface logic
Just another WordPress site
Theme DOM
String: <layoutcomponent name:"logo_container"> DOM </layoutcomponent>
Metamodel generation logic
Component Filter Logic
<layoutcomponent name:parent> include – logo_container </layoutcomponent>
Component Separation Logic
Component files
Components Repository
XML Theme Generation
X-Themes

Figure 4.15: Example of X-Themes generation.

functionalities. This component parses the DOM of the theme with a recursive procedure: for each type of element it stores different information in a string with an XML format. In addition, the ”metamodel generation logic component” creates another CSS rules that corresponds to the style inline rules of each of the theme containers. Finally, the subssystem sends this information to the server via an AJAX request. Following the simplified example in Figure 4.15, the *metamodel generation logic* creates a string with a XML format for the ”logo_container”. Inside this tag, it inserts the DOM of the container, and gives this structure to the server node.

4.4.2 Server Node

The *component separation logic* receives the strings formatted by the *metamodel generation logic* component. Its job is to parse these information given by the browser and create the corresponding file XML for each element of the theme with the structure we defined in Section 4.2.1. At the same time this component has to write the CSS files with the inline styles rules given by the *metamodel generation component*. These CSS files are related to the corresponding component and they are created to make the X-Theme maintainable and reusable: using inline style rules is not a good practice in the web development.
The *component separation logic comunicates* with the *component filter logic* that understands and parses the different types of theme elements. It trasforms the dynamic elements (cf. Section 4.2.2) in the corresponding dynamic code. For an example a dynamic element of a theme is the website title. The component filter logic understands that this component is dynamic and overwrites it with the corresponding PHP code: in this case the function `bloginfo( 'name' )` already explained in Section 4.4.2. In the case of nested layout components (cf. Section 4.2.1) or components, the filter replaces the component code with a `include` PHP function.
At the end of this filtering process, the component separation logic writes the separeted directory of each element in the component repository. Following the example in Figure

4.15, we can see that the *component filter logic* inserts in the "parent" code file the code of its layout component child :"logo_container" using the `include` PHP function. Using these information, the *component separation logic* creates all the necessary files for each component and it stores this data in the components repository.

The *XML theme generation component* can now use the information in the components repository and creates the theme files in the WordPress theme directory. This component has also to create the XML file of the theme (cf. Figure 4.8). The theme created in this way is a X-Themes and it follows the metamodel described in Section 4.2.1.

4.5 Implementation

In this Chapter we saw already the main functionalities of *X-TG*, its basic concepts and its X-Theme generation process. In this Section we will now focus on the main implementation aspects of *X-TG* also trying to explain the trade offs we faced during the development process.
In Section 4.5.1 we will say how we developed the user interface (cf. Figure 4.1), and in detail, the usage and the drag & drop of the containers. We will also present another important aspect of *X-TG* in Section 4.5.2. *X-TG* allows the users to upload their own code in two different ways, giving also the chance to share *packaging of theme functionalities* between developers.

4.5.1 Drag and Drop, Usage of the container

As already explained in Section 4.1, *X-TG* allows the users to drag & drop each element in the theme. The users can also *group* these elements (cf. Figure 4.2) and then they can drag & drop all the these boxes together.
The implementation of this interaction was one of the part that requested the main effort during the development process. The first try was to develop this functionality using the API of the jQuery UI[1] library. JQuery UI provides, among many things, some JavaScript functions to perform some user interface interactions. The function:

```
$(function() {
    $( "#drag" ).draggable();
  });
```

makes the DOM element with the `#drag` id a draggabble element, this means that the users can drag this element around the webpage. Similary to the `draggable` function, we have the droppable function:

```
$(function() {
    $( "#drop" ).droppable();
  });
```

that makes the element `drop` a droppable element, it means that this droppable element is a target for the draggable elements. At the same time we needed that the users could also resize each element, but not outside their parent so we used the following JQuery function:

[1]http://jqueryui.com/

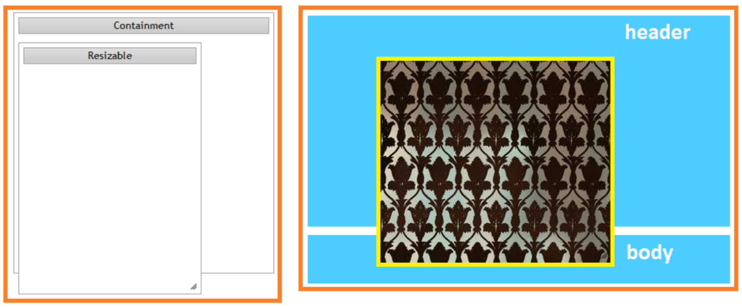


Figure 4.16: Bug of the JQuery UI resizable function. On the left the example screenshot of the JQuery UI official page. The ”resizable” element in theory, should not be resizable out the ”container” element, but this is not true as the image shows. On the right the consequence of this bug in X-TG. The box with a brown background image goes out of the header element, giving a wrog appareance to the theme.

```
$(function() {
    $( "#child" ).resizable({
      containment: "#parent"
    });
  });
```

in this way the users cannot resize an element going out of its parent: the `child` element should not be resizable out of its `parent`. Unfortunately, this jQuery UI function has some problems, this is proved by the same JQuery UI website[2]. The Figure 4.16 shows this bug: the ”resizable” element goes out of its container and this bug gives bad consequences in *X-TG*. This bug happens every time we define an element both resizable and draggable.
We did not find a ”work around” to avoid this problem, and we had to do a trade-off: use jQuery UI giving a bad interaction to the user or develop from zero the drag & drop functionalities. We chose the second option because, in our opinion, giving a bad user interface may create several problems during the theme development process, and it goes in the opposite direction of the *X-TG* goal.
For these reasons we started to develop our own implementation of the drag & drop interaction managing only the ”relative” position of the elements in the interface. This implementation also allows the user to group different elements and to drag & drop together. This was done in the following way:

- The user drags an element
- The user drops the element into another element
- The system catches the cordinate position of the drop target area
- The system indetifies the target element
- The system adds to the target innerHTML the HTML of the element dropped

[2]http://jqueryui.com/resizable/#constrain-area acceded on 08/2013

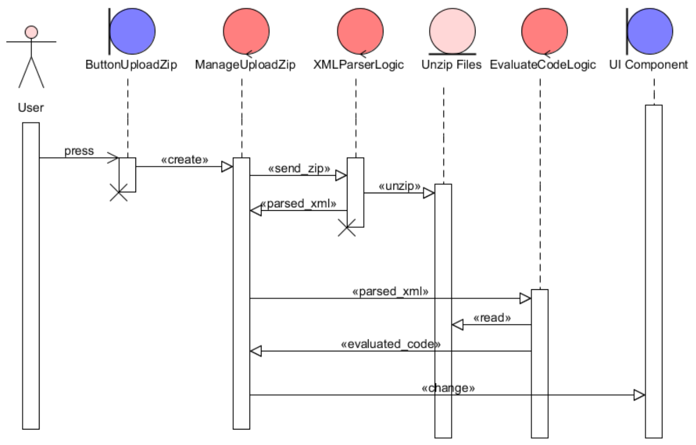


Figure 4.17: Sequence diagram for the uploading external code via zip file.

With this implementation we avoided the problems that jQuery UI has and at the same time we develped a better user interface interaction, giving the chance to group elements, drag & drop them and without risking to have a bad theme structure.

4.5.2 Upload your code

Another fundamental aspect of *X-TG* is the opportunity to upload external code in the tool. During the development process, we decided to allow the users to upload their own code in two ways as already introduced in Section 4.1.2. The users can decide if insert a package file or different files (JavaScript, PHP, CSS and more). As explained in Section 4.2.3 the archive has to contain a XML file that describes the structure of the code; this XML follows the structure we gabe in our metamodel (cf. Section)4.2.1. In this way the users can share betweem them different theme functionalities without knowing the detail of the functionalities themselves.
When the users upload this zip file, *X-TG* understands the XML and it places the files in the theme. More in detail, the Figure 4.17 shows the process to add external functionality via this archive.
The Figure 4.17 represents a sequence diagram for the uploading external code via zip archive. The user clicks on the upload button sending the zip file. This boundary creates the *ManageUploadZip* control that sends the zip file to the other control *XmlParserLogic*. The control *XmlParserLogic* unzips the zip archive and via an AJAX request sends back the parsed XML that consists of a string with the contents of all tags. The *ManageUploadZip* receives this string and it sends it to another control *EvaluateCodeLogic* that has the responsibilities to evaluate the code in the zip file using the parsed XML information. Then the evaluated code is sent back to *ManageUploadZip* that using this output, changes the component apperance in the user interface.
If the user does not have a zip archive with the XML, they can insert more files. This process is similar to the process we saw and described before (cf. Figure 4.17), the difference is that when *X-TG* evaluates the code, it also creates the corresponding XML file. In this way, the

next time, it will be possible to upload directly the zip archive, and it will work as well as it works with different files.

5 Evaluation: User Study

The evaluation of *X-TG* has been carried out with a task-based user study in order to investigate the potential of *X-TG* and also its limitations. We planned three different tasks (task A, B and C), which involve the users to create three different kinds of X-Themes with *X-TG* and *XDE* (cf. Murolo [7]). Unfortunately, it was not possible to compare the *X-TG* to other theme generators, because these tools do not allow customization of complex elements while it is possibile in *X-TG*. Moreover, it was also not possible to make the participants develop the themes for the tasks coding manually for two basic reasons: create manually a theme, following a specific mockup, may take several hours of coding also for people who know WordPress pretty well; a second reason is that not all the partecipants knew WordPress, and for these people, coding manually a WordPress theme might have taken also days of learning and developing. We decided to involve also participants who did not know WordPress in order to understand how much *X-TG* allows the non-techinical users to develop a theme.
We analyzed *X-TG* and *XDE* tools together, for practical reasons and also to compare the designing by example approach against our theme generator. One of the task we planned is more oriented to *XDE* (task C) while the other two tasks regard completely *X-TG* (task A and task B). For completeness we will describe the specifications and descriptions of all the three tasks. The results show that the users, on average, developed a WordPress theme in less than seven minutes, and the users were able to do that without knowing WordPress development knowledge. In addition, the users were satisfied of their results and of the main *X-TG* functionalities estimating that coding the same theme without *X-TG* would take hours. The results show also that the users, on average, made, every twenty minutes, around 1 error during the study for task A (cf. Section 5.1.2) and less than 0,2 errors for task B (cf. Section 5.1.2), even if they used *X-TG* for the first time.
In this Chapter we will first describe the design of the study and the specification of each task (cf. Section 5.1). Finally, we will focus on the results regarding *X-TG* in the Section 5.2.

5.1 Planning

In this Chapter we will first describe the study design introducing the decisions made before the study itself (cf. Section 5.1.1). After we will analyze each of the three tasks, showing the corresponding mockups and indicating the main functionalities that the users has to do to perform the tasks (cf. Section 5.1.2). At the end of Section 5.1.2, we will do some considerations about the tasks, like the differences between them.

5.1.1 Study Design

In order to evaluate our tools *X-TG* and *XDE* (cf. Murolo [7]), we planned a task-based user study with twelve participants.
We chose this number of participants in order to cover all the order permutation of the three tasks we planned: the order of the tasks will be different between the users, the first two participants will do first the task A, then the task B and finally the task C. The next two participants will do the tasks in this order: A, C and B and so on. We planned this kind of order between the tasks in order to avoid any learning biases. In addition, during the study the participants will be isolated to avoid any bias.
Moreover, for practical reasons, the users performed the task and filled the questionnarie (powered by Google Drive[1]) on a specific computer that has all the links and the files necessary to complete the tasks.

5.1.2 Tasks specification

As we already introduced in Section 5.1.1, we planned three tasks, and the order of these tasks will change between the participants. Figure 5.1 describes the structure of the study.
At start, the participants filled a questionnaire inserting their personal information and background skills regarding also their WordPress knwoledge. After this first questionnaire, the participant started the task. Before each task, the evaluator explained all the information they need to perform their work also showing the mockups. The mockups-theme are X-Theme and they were developed by using *X-TG*. Moreover, at start and at the end of each task the evaluator noted the starting time and the time of completion respectively. When the user finished a task, they filled a questionnaire regarding that specific task. While the participants develop the themes, the evaluator noted also the mistakes done by the users. This procedure was repeated for all the three tasks. Finally, when the user completed all the three tasks, they filled a questionnaire with last considerations about the study. We will now describe each of these tasks, indicating what kind of theme they had to re-create and the main *X-TG* functionalities that they had to use during the study.

Task A

In the task A the users had to recreate the theme in Figure 5.1. As we can see in the picture, the users had to insert in the theme a logo, the website title and description, the loop and a simple text in the footer.
The users can recreate this theme using only *X-TG*. The mockup of the theme has basic

[1]https://drive.google.com/#recent

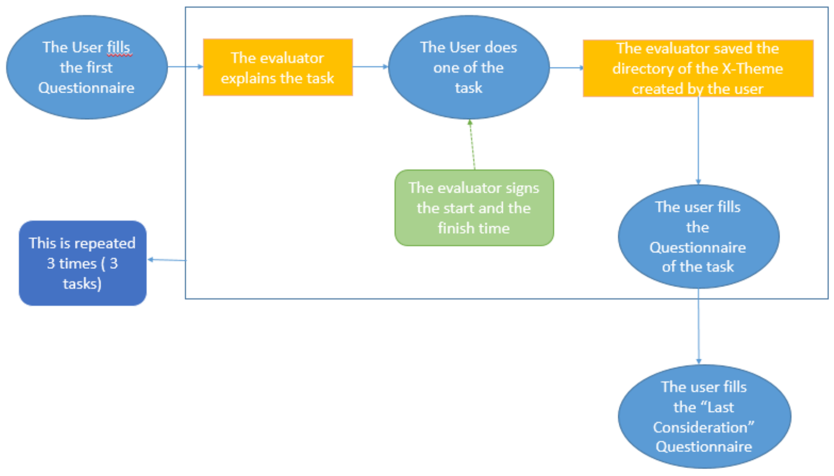
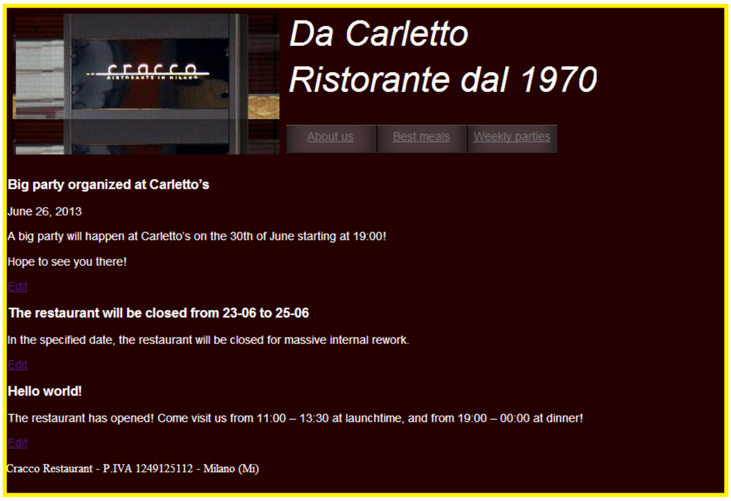

The User fills the first Questionnaire
The evaluator explains the task
The User does one of the task
The evaluator saved the directory of the X-Theme created by the user
The evaluator signs the start and the finish time
The user fills the Questionnaire of the task
This is repeated 3 times (3 tasks)
The user fills the "Last Consideration" Questionnaire

Figure 5.1: Structure of the user study.

features, but at the same time, to complete the task the users had to use the following *X-TG* functionalities:

- **Upload external code via package.** The users had to upload the *dimenu* functionality in order to insert the brown menu with the WordPress pages (cf. Figure 5.2).
- **Change the style of an element.** In order to recreate the mockup the users had to modify the background color of the components and also the font color.
- **Adding website title and description.** The users had to insert the three basic dynamic elements of the theme: the loop, the website title and description.
- **Add background images.** As the Figure 5.2 shows the theme to recreate has a little logo in the header element, the users had to modify the background color image of the containers to reproduce this theme's look & feel.
- **Drag and drop container.** To complete the task, the users had to understand the main aspect of X-TG: the drag & drop of the containers in the theme.



Da Carletto
Ristorante dal 1970

Big party organized at Carletto's

June 26, 2013

A big party will happen at Carletto's on the 30th of June starting at 19:00!

Hope to see you there!

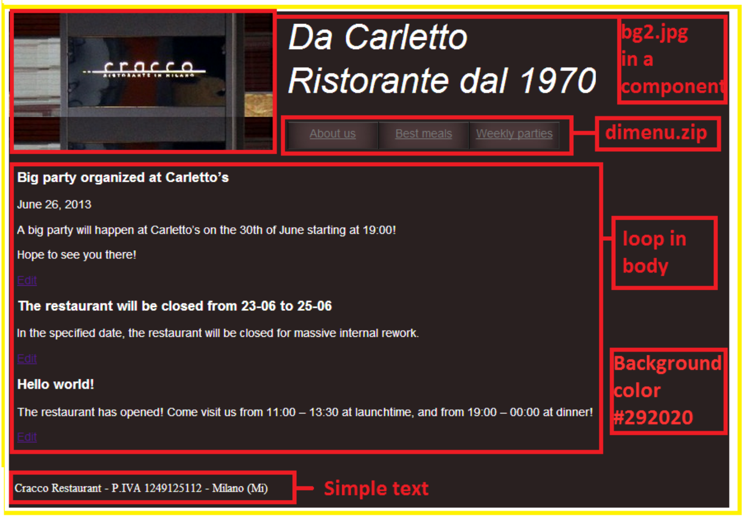
The restaurant will be closed from 23-06 to 25-06

In the specified date, the restaurant will be closed for massive internal rework.

Hello world!

The restaurant has opened! Come visit us from 11:00 – 13:30 at launchtime, and from 19:00 – 00:00 at dinner!

Cracco Restaurant - P.IVA 1249125112 - Milano (Mi)



Da Carletto
Ristorante dal 1970

Big party organized at Carletto's

June 26, 2013

A big party will happen at Carletto's on the 30th of June starting at 19:00!

Hope to see you there!

The restaurant will be closed from 23-06 to 25-06

In the specified date, the restaurant will be closed for massive internal rework.

Hello world!

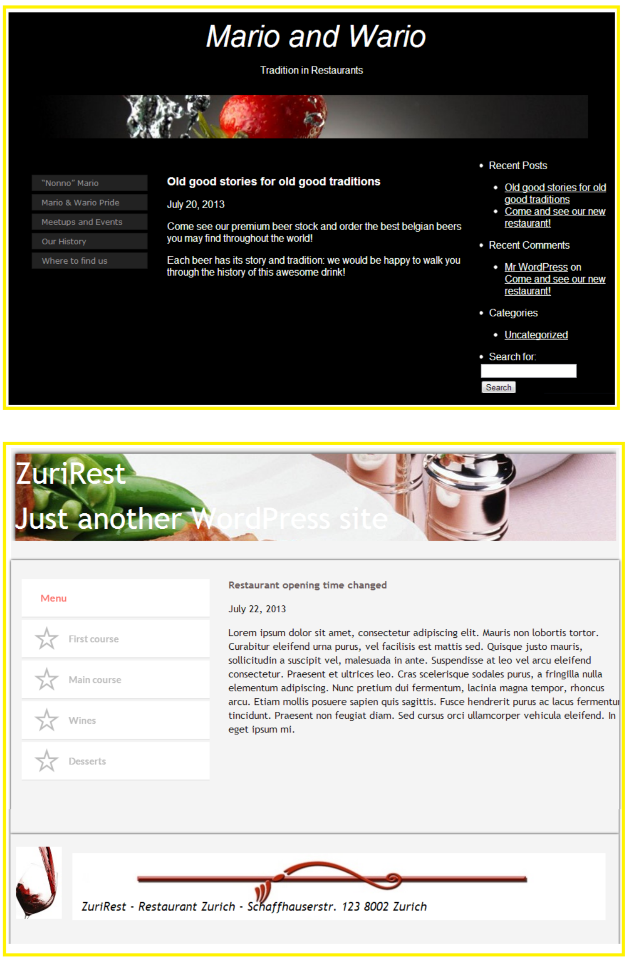
The restaurant has opened! Come visit us from 11:00 – 13:30 at launchtime, and from 19:00 – 00:00 at dinner!

Cracco Restaurant - P.IVA 1249125112 - Milano (Mi)

Figure 5.2: At the top the screenshot of the task A theme. At the bottom the specification of the theme.

Task B

In the task B the users had to mix part of two extisting themes using only *X-TG*. In Figure 5.3 we can see the two starting themes and their specifications. The users had to reproduce the theme in Figure 5.4. Like the Task A, the theme to reproduce is easy, but to complete the task the users had to use the following functionalities of *X-TG*:


Mario and Wario

Tradition in Restaurants

Old good stories for old good traditions

July 20, 2013

Come see our premium beer stock and order the best belgian beers you may find throughout the world!

Each beer has its story and tradition: we would be happy to walk you through the history of this awesome drink!

- Recent Posts
  - Old good stories for old good traditions
  - Come and see our new restaurant!
- Recent Comments
  - Mr WordPress on Come and see our new restaurant!
- Categories
  - Uncategorized
- Search for:

ZuriRest
Just another WordPress site

Restaurant opening time changed

July 22, 2013

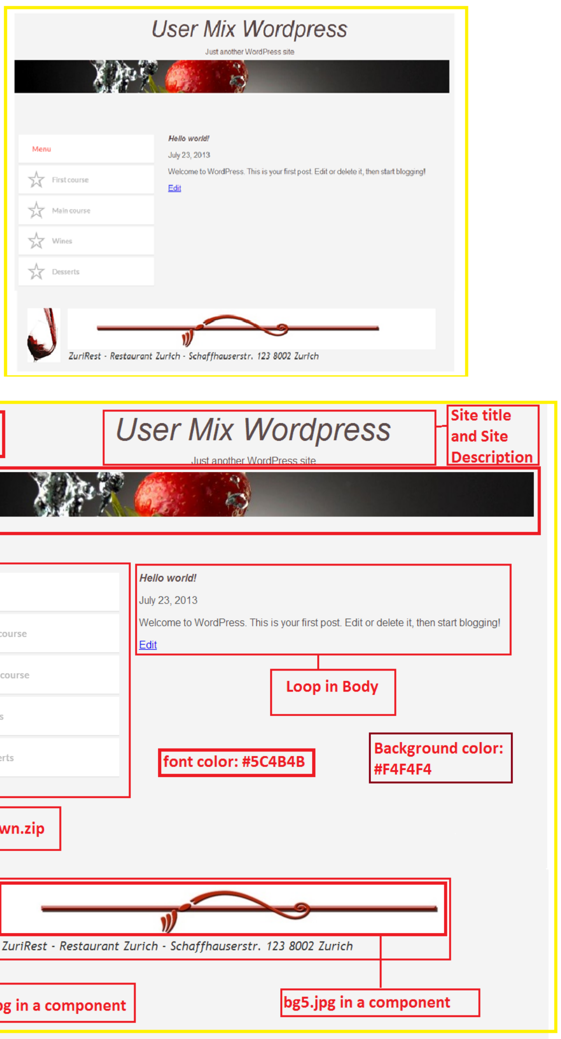
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris non lobortis tortor. Curabitur eleifend urna purus, vel facilisis est mattis sed. Quisque justo mauris, sollicitudin a suscipit vel, malesuada in ante. Suspendisse at leo vel arcu eleifend consectetur. Praesent et ultrices leo. Cras scelerisque sodales purus, a fringilla nulla elementum adipiscing. Nunc pretium dui fermentum, lacinia magna tempor, rhoncus arcu. Etiam mollis posuere sapien quis sagittis. Fusce hendrerit purus ac lacus fermentu tincidunt. Praesent non feugiat diam. Sed cursus orci ullamcorper vehicula eleifend. In eget ipsum mi.

ZuriRest - Restaurant Zurich - Schaffhauserstr. 123 8002 Zurich

Figure 5.3: Screenshot of the two existing themes.

- **Upload external code via package.** The users had to upload the *dropdown* functionality in order to insert the vertical menu (cf. Figure 5.4).
- **Change the style of an element.** Like in Task A, in order to follow the mockup the users had to change the style of some theme elements.
- **Adding website title, description and custom header.** Also here it is necessary insert the website title and description. In addition, the users had to insert also the custom header in the theme.

- **Add backgrounds images.** Like in task A, the users had to change some backgrounds images.
- **Drag and drop container.** The users had to drag & drop the containers of the theme in order to complete the task.



Hello world!
July 23, 2013
Welcome to WordPress. This is your first post. Edit or delete it, then start blogging!
Edit

ZuriRest - Restaurant Zurich - Schaffhauserstr. 123 8002 Zurich

Figure 5.4: At the top the screenshot of the mixed theme. At the bottom the specification of the same theme.

Task C

In this task, the users had to mix part of existing theme using *X-TG* and *XDE*. The two existing themes are the same of the task B, and the we can see them in Figure 5.3. Moreover, also the theme they had to reproduce is the same of the task B, the difference is that now the users are allowed to do that with both the tools. Figure 5.5 shows the theme that the users had to reproduce and it also indicates which part belong to which theme.

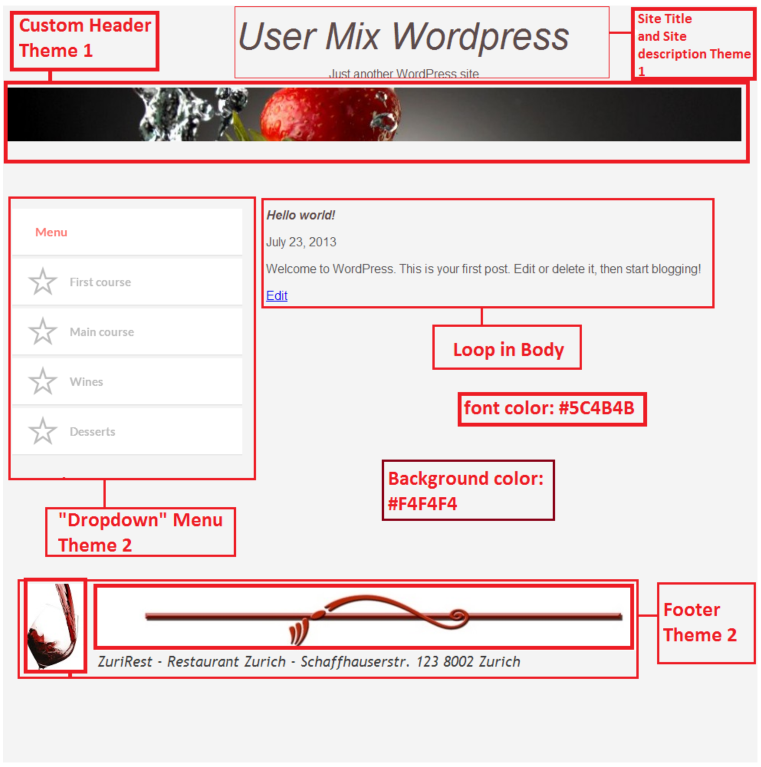

Custom Header Theme 1
User Mix Wordpress
Just another WordPress site
Site Title and Site description Theme 1
Menu
First course
Main course
Wines
Desserts
"Dropdown" Menu Theme 2
Hello world!
July 23, 2013
Welcome to WordPress. This is your first post. Edit or delete it, then start blogging!
Edit
Loop in Body
font color: #5C4B4B
Background color: #F4F4F4
ZuriRest - Restaurant Zurich - Schaffhauserstr. 123 8002 Zurich
Footer Theme 2

Figure 5.5: Task C specification.

The users, in all the three tasks, have to create a theme, following a specific mockup. The differences between these tasks, are the way how the users perform the theme. In task A the users recreate the theme using only *X-TG* following a ”static mockup” on a paper; we planned the task A in order to understand the capabilities of users to develop a theme from scratch using *X-TG*. In task B and task C the users have to develop a new theme that is based on existing ones. In addition, during the task B the users can see the existing themes but they cannot use the *XDE* tool, they can only use the theme generator functionalities. This task was planned in oder to evaluate the *X-TG* capabilities to mix parts of existing themes. In task C, the users can drag & drop each element from that existing themes as well as use the *X-TG* functionalities. Task C was studied in order to understand the capabilities of *XDE* combining with *X-TG* instead of using only *X-TG* like task B.
The three tasks have a similirar complexity; for this reason, we expected similar time to complete the tasks. Moreover, we were mostly interested to the satisfaction of the people towards their result, and their capacity to develop a WordPress theme, without knowing the implementation detail of the theme itself.

5.2 Results

During the evaluation the users performed three tasks where the task A and task B were oriented only on *X-TG*, while the task C (cf. Section 5.1.2) involved also *XDE*. We will now focus only on the results that are strictly related to *X-TG* (task A and B), the other informations (task C) are showed in [7]. In this section we will first present the main results of the user study (cf. Section 5.2.1), after we will do some considerations about the results themselves (cf. Section 5.2.2).

5.2.1 Presentation

The study involed twelve participants: two of them are female while the other ten are male. The age of the users goes from twentyone to thirtyeight years old. At start of the evaluation, the participants filled a questionnaire with their personal information and their background skills. From this questionnaire we noticed that the users had a low-medium knowledge of web development and design user interface skills: with a scale from 1 to 7 (where with 1 we meant novice and 7 we meant expert) they filled, on average, a 4,5 score for the web development knowledge and 3,5 for the user interface skills. In the same questionnaire we asked if they used a theme generator before, but no one of the users had never used these kinds of tools.
As we can see in Figure 5.6, the 20% of the participants never used WordPress before; the 33% percent of the users use WordPress more than once a month while the remaining 45% uses WordPress more rarely.
While Figure 5.6 shows how much the participants use WordPress in general, the Figure 5.7 shows the users' knowledge of WordPress development, and more in detail, if they know how to develop a custom type, a theme and a plugin by their own. As the Figure represents, the users had a low knowledge of the WordPress development; in a scale from 1 to 7 (where 1 indicates novice and 7 indicates a expert) they, on average, inserted a number from one to three.

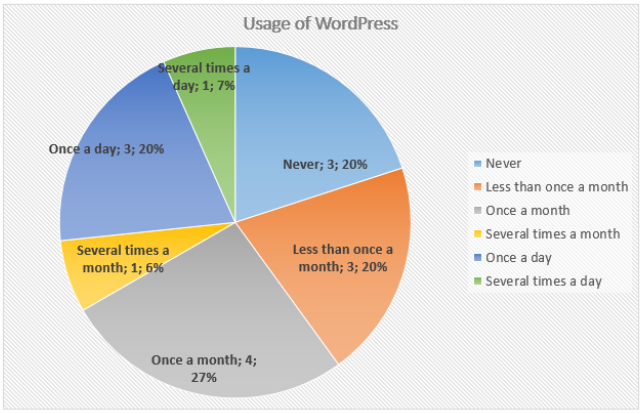


Figure 5.6: Usage of WordPress between the participants.

After these considerations about the participants background skills and personal information, we can now focus on the tasks results. Figure 5.8 shows the time on average to complete the task A and the task B. Moreover, the Figure 5.8 also shows the standard deviation as error bars. Figure 5.8 indicates that the users, on average, perfomed the task A in less than seven

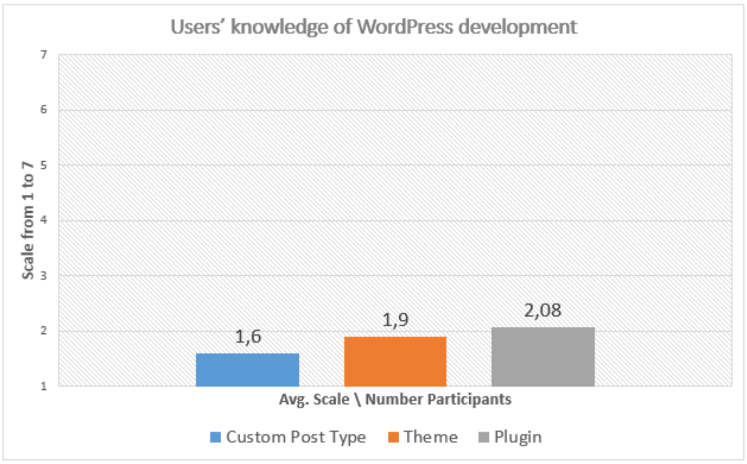


Figure 5.7: WordPress development background skills of the participants

minutes. In addition, all the participants were able to develop a WordPress theme (for both the tasks) in a time between four minutes to eleven minutes, despite their low level of WordPress development skills.

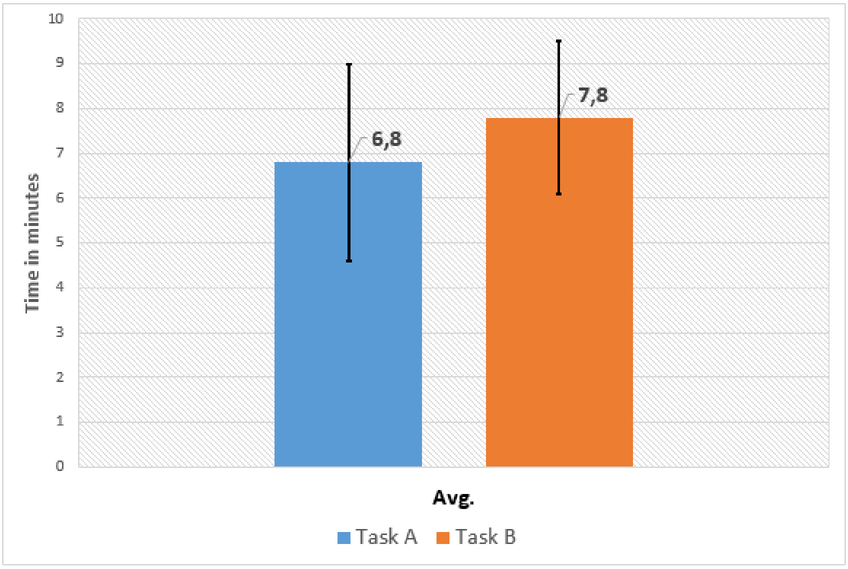


Figure 5.8: Time to complete the task A and B.

Figure 5.9 shows the error / minute ratio for the task A and B. The users made, on average, less than 0,05 errors each minute to complete the task A; this means that they made around one error every twenty minutes. Moreover, the users committed, on average, only 0,01 errors each minute during the task B.
We asked the users if they felt that they succefully completed the tasks and if they were satisfied with the results. Figure 5.10 shows the median value of the answers. Following the

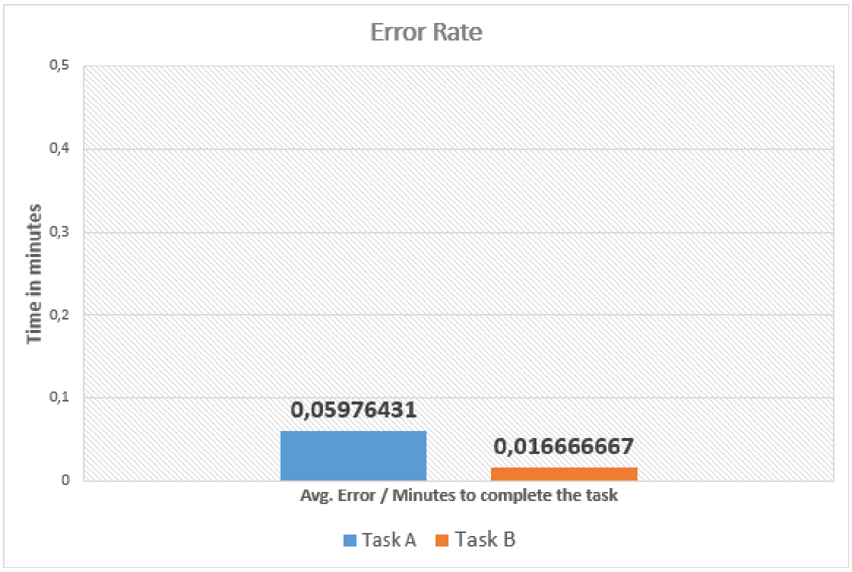


Figure 5.9: Error - minute ratio on average for task A and B

Figure 5.8 and figure 5.10, we can say that the users completed the task in few minutes and, in addition, they were satisfied of the resulting themes.

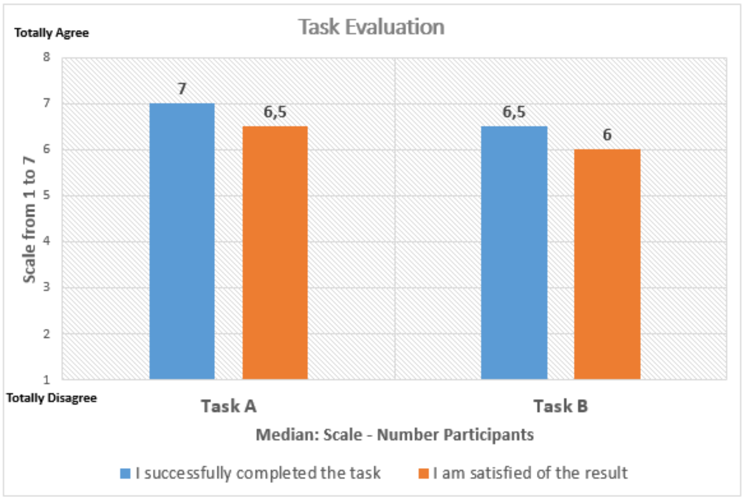


Figure 5.10: Task evaluation.

We also asked the participants how much the main *X-TG* functionalities were easy to understand, easy to use, efficent, useful and effective (cf. Figure 5.11). They seem to appreciate these functionalities, as we can see in Figure 5.11 for task A and in Figure 5.12 for task B.
As we already introduced in Section 5, it was not possible to compare *X-TG* against coding manually the theme for practical reasons; however, we asked the users if coding the same theme directly would be better and easier (cf. Figure 5.13). As Figure 5.13 shows, the users think that coding manually the same theme would not be easier. At the same time, they are

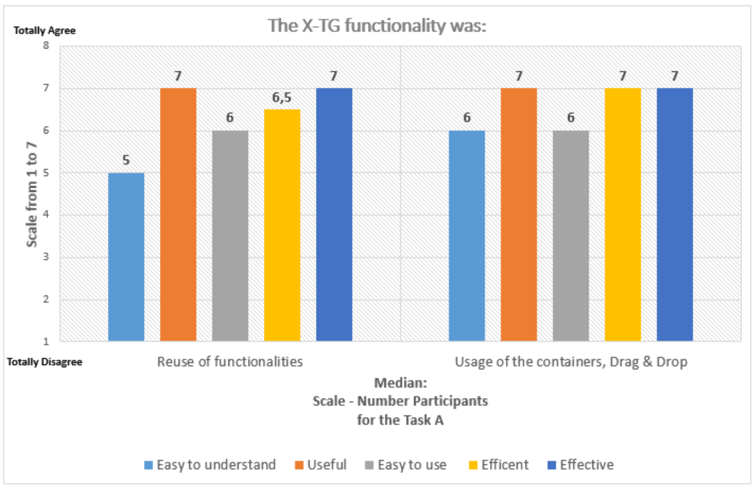


Figure 5.11: Evaluation of the main X-TG functionalities for task A.

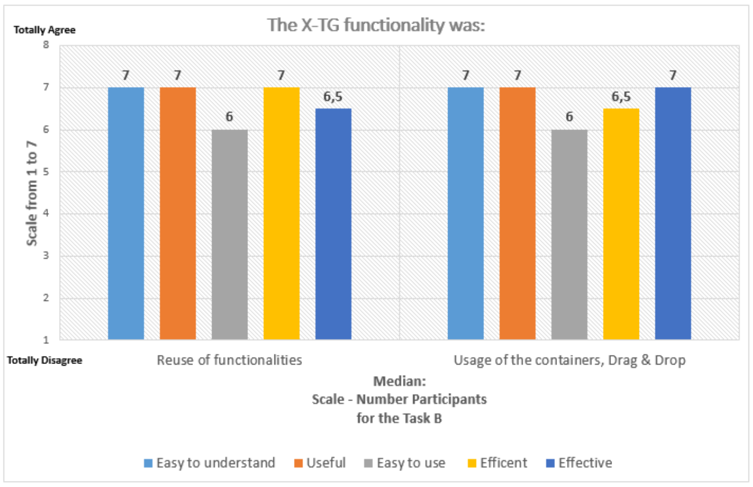


Figure 5.12: Evaluation of the main X-TG functionalities for task B.

not sure if the result of the theme would be better (the value is near to the middle of the scale). We asked the users how much time they would require to perform the task coding manually the theme. Figure 5.14 shows that more than seven people would have needed more than two hours to develop the same theme for both the tasks.

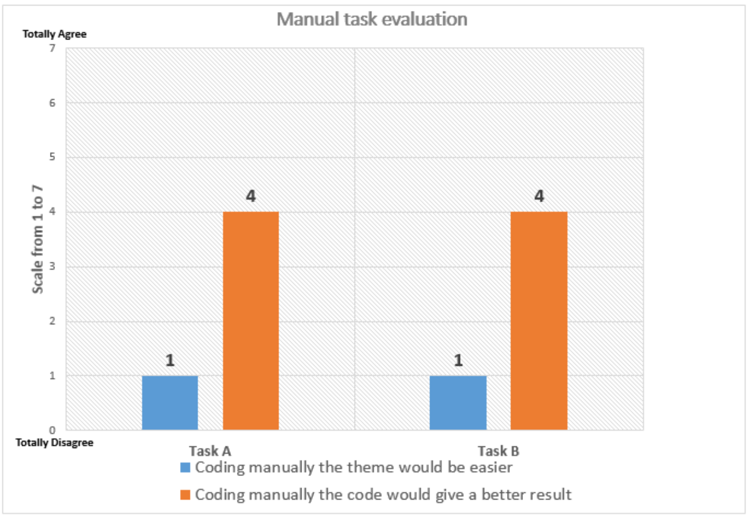


Figure 5.13: Evaluation of doing the theme manually.

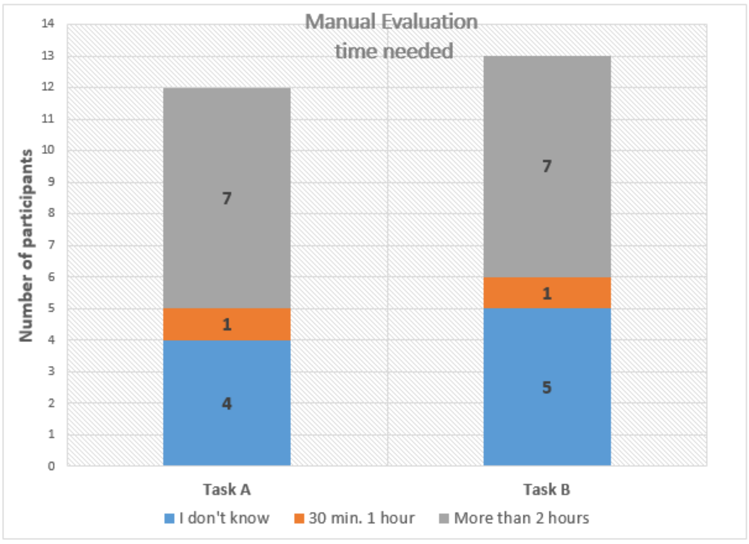


Figure 5.14: Coding manually time evaluation.

5.2.2 Discussion

After showing the main results of the user study (cf. Section 5.2.1) we can now do some consideration about the result itself. Figure 5.8 shows the time, on average, to complete the task A and B. The results show that the the users perfomed the task A in more time than task B. In addition figure 5.9 shows the error / minute ratio for task A and B. As we already introduced, the users made, on average, around 1 error every twenty minutes during the development of the task A theme. The situation is quite different for task B, where the user made less errors than in task A.
We think that these differences between task A and B might be related to bugs that we unfortunately found only during the study. Two participants during the task A created two container with an ID that started with a number; the generator created correctly the corresponding CSS rules, but at the same time, according to CSS, a rule cannot start with a number, so the themes generated in this situation looked diffent from the theme developed by the user. This might have influenced the results, and it is necessary to perform another user study without this problem to ensure if this bias is strictly related to the bug.
Figure 5.11 and Figure 5.12 show what the users think about the *X-TG* main functionalities for task A and task B. They seem to appreaciate the main functions of *X-TG* with the exception of the ”reuse functionalities” in task A. The median value of this functionality for the ”easy to understand” question, is 5 on a scale from 1 to 7 (where for 1 we meant that the functionality was difficult to understand, while for 7 we meant that the functionality was easy to understand). We think that this might be because of the disposition of the *X-TG* menu buttons; during the study the users asked many times how they could insert the task A menu (called ”dimenu”), because they could not find how to do it. In addition, it is possible that this result is given by a learning bias. It will be interesting to understand how to organize better this functions in order to improve the ”reuse of functionalities” part of *X-TG* and also to understand if this organization is the cause of this issue.
Figure 5.14 shows how much time the participants would spend to reproduce the same theme coding manually. Seven users said that for both task A and B, that they would spend more than 2 hours to perform the same tasks; 4 users, for task A, and 5 users, for task B, said that they did not know how much time they would need to develop the theme without *X-TG*. In our interpretation, this may be because, they do not have enough WordPress skills to give an objective answer. The users think that coding manually the same theme would not be easier (cf. Figure 5.13) but they are not sure if the theme whould be better. For us, one reason might be that they do not have enough techinical WordPress skills to understand how a WordPress theme will look if it is coded manually. Another possibility is also that coding manually gives more control to the developer compared to using a theme generator.
Moreover, we had an outlier result for the same questions (cf. Figure 5.14) regarding the time to develop the themes manually. Only one person said that he could do the same theme in thirty minutes or one hour (for both the task A and B), even if, he indicated low (novice level: one) WordPress development knowledge.

6 Conclusions

X-TG is a theme generator for WordPress in WordPress itself; it helps the users to develop their own WordPress theme. The theme generated by *X-TG* follows a well-formed metamodel called X-Themes that was developed to support a possible future reuse between the themes. The creation of the X-Theme is completely transparent to the users, in this way the they do not have to understand and learn a new metamodel; thus it is not necessary to have techinical skills to develop a theme using *X-TG*.
X-TG was developed after an analysis and categorization of the current most used theme generators. This analysis gives us the chance to understand the good aspects of these tools but, specially, their limitations. The main theme generators tend to focus only on the customization of static elements (such as the logo, the font color or background color) while a good generator has to allow the users also to manage complex WordPress elements (the loop, custom header, sidebar and so on). The categorization of the theme generators helps us to develop a tool that had the highest possible number of functionalities to improve the development of a theme.
After the development of *X-TG*, we carried out an user study in order to understand its potential and limitations. The results suggest that the users were quickly able to develop a theme, and they developed some themes without having technical WordPress skills. Moreover, the results show that the users were satisfied of the theme results, and in most of the cases, they estimated that coding the same theme manually would cost hours instead of minutes. Moreover, the participants of the study, were satisfied with the theme generated and appreciated the opportunity to customize it with complex functionalities.

6.1 Future Work

X-TG is still a prototype; for this reason there are several things to improve and investigate. The user study showed some *X-TG* techinical limitations: many participants asked for a better layout support during the drag & drop of the containers, such as the possibilities to align the elements in a grid. Some users, wanted to follow the exactly same positioning of the mockup elements, and for this reason, they had some problems using *X-TG* without a grid support. It would be intersting to develop this kind of design, in order to improve the usability of the tool, its effectiveness and also the quality of the theme generated should be better.

The participants also had some issues with the button names which, for them, are not intuitive, in particular for the differences of dynamic and static elements. In addition, *X-TG* does not allow the users to customize text that is not in dynamic containers. It will be necessary to avoid to these problems, allowing a better disposition of the menu and its functionalities, also helping the users to customize and insert text in the theme, giving an intuitive and more friendly user interface.

Moreover, it was not possible, for the user study, to compare *X-TG* against another theme generators, or against coding manually because the main theme generators studied in this thesis, do not provide the *X-TG* main functionalities, so comparing our tool with these generators would have been not fair and interesting. Moreover, develop manually a theme may take several hours of coding. Therefore, it was not possible to performe this kind of study for practical reasons. Performing an user study that compares developing a theme from scratch for WordPress instead of creating the theme with *X-TG* will be one of the main future work of this study. Finally, this new user study will involve also WordPress developers, in order to cover more kinds of users and measured the effectiveness of *X-TG* for technical and non-technical users.

Bibliography

[1] S. Ceri, P. Fraternali, and A. Bongio. Web modeling language (webml): a modeling language for designing web sites. *Computer Networks*, 33(1):137–157, 2000.

[2] S. Ceri, P. Fratemali, and S. C. P. Fraternali. Architectural issues and solutions in the development of data-intensive web applications. In *In Proc. of CIDR03, Asilomar*. Citeseer, 2003.

[3] N. Koch, A. Knapp, G. Zhang, and H. Baumeister. *Uml-based Web Engineering An Approach Based on Standards*. Springer London, 2008.

[4] S. Leone, A. de Spindler, M. C. Norrie, and D. McLeod. Integrating component-based web engineering into content management systems. In *Web Engineering*, pages 37–51. Springer, 2013.

[5] H. Lieberman, F. Paternò, and V. Wulf. *End user development*, volume 9. Springer, 2006.

[6] A. Mauthe and P. Thomas. System and data integration in cms. *Professional Content Management Systems: Handling Digital Media Assets*, pages 225–244, 2005.

[7] A. Murolo. Designing wordpress themes by example. Master's thesis, 2013, 2013.

[8] T. Nestler, A. Namoun, and A. Schill. End-user development of service-based interactive web applications at the presentation layer. In *Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems*, pages 197–206. ACM, 2011.

[9] J. Rode and M. B. Rosson. Programming at runtime: requirements and paradigms for nonprogrammer web application development. In *hcc*, pages 23–30, 2003.

[10] J. Rode, Y. Bhardwaj, M. A. Pérez-Quiñones, M. B. Rosson, and J. Howarth. As easy as click: End-user web engineering. In *Web Engineering*, pages 478–488. Springer, 2005.

[11] A. H. Silver. *WordPress 3 complete*. Packt Publishing, 2011.

[12] B. Williams, D. Damstra, and H. Stern. *Professional WordPress: Design and Development*. John Wiley & Sons, 2012.

[13] D. Wolber, Y. Su, and Y. T. Chiang. Designing dynamic web pages and persistence in the wysiwyg interface. In *Proceedings of the 7th international conference on Intelligent user interfaces*, pages 228–229. ACM, 2002.

A User Study

A.1 Questionnaire

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.

	Novice - 1	2	3	4	5	6	7 - Expert
Wordpress Usage	◯	◯	◯	◯	◯	◯	◯
Wordpress Development	◯	◯	◯	◯	◯	◯	◯
Web Development	◯	◯	◯	◯	◯	◯	◯
User Interface Design	◯	◯	◯	◯	◯	◯	◯

Wordpress Experience

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

	Never	Less than once a month	Once a month	Several times a month	Once a day	Several times a day
Frequency	◯	◯	◯	◯	◯	◯

6. **How much do you know about WordPress development, its internals and how to extend it?** *
   Contrassegna solo un ovale per riga.

	Novice - 1	2	3	4	5	6	Expert - 7
Custom Type Development	◯	◯	◯	◯	◯	◯	◯
Theme Development	◯	◯	◯	◯	◯	◯	◯
Plugin Development	◯	◯	◯	◯	◯	◯	◯

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.

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

10. **What are the factors that contribute to the quality of a Theme Generator?**
Contrassegna solo un ovale per riga.

	Not important 1	2	3	4	5	6	Very Important 7
Capacity to customize the design (e.g. modify font, color, background)	◯	◯	◯	◯	◯	◯	◯
Capacity to modify the layout	◯	◯	◯	◯	◯	◯	◯
Capacity to reuse parts between themes	◯	◯	◯	◯	◯	◯	◯
Capacity to adapt the created theme to many devices (smartphones and different kind of screen) and in different browsers	◯	◯	◯	◯	◯	◯	◯

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.

	1	2	3	4	5	6	7	
Not Satisfied	◯	◯	◯	◯	◯	◯	◯	Totally Satisfied

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 :

- http://localhost/food/wp-admin/tools.php?page=xthemes-invoker&xthemespage=buildtheme

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.

	1	2	3	4	5	6	7	
Totally Disagree								Totally Agree

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

Contrassegna solo un ovale per riga.

	Totally Disagree 1	2	3	4	5	6	Totally Agree 7
Easy to understand							
Useful							
Easy to use							
Efficient							
Effective							

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

Contrassegna solo un ovale per riga.

	Totally Disagree 1	2	3	4	5	6	Totally Agree 7
Easy to understand							
Useful							
Easy to use							
Efficient							
Effective							

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

Contrassegna solo un ovale per riga.

	Totally Disagree 1	2	3	4	5	6	Totally Agree 7
Easy to understand							
Useful							
Easy to use							
Efficient							
Effective							

9. **The usage of theme elements was:** *
e.g. Title, Description, Loop
Contrassegna solo un ovale per riga.

	Totally Disagree 1	2	3	4	5	6	Totally Agree 7
Easy to understand							
Useful							
Easy to use							
Efficient							
Effective							

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

	I don't know	30 minutes - 1 hour	1 hour - 2 hours	2 hours - 3 hours	More than 3 hours
Time needed					

11. **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

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.

	1	2	3	4	5	6	7	
Totally Disagree								Totally Agree

13. **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

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.

	1	2	3	4	5	6	7	
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/thirdrestaurant/

Go on the following link to use the XThemes Generator:

- http://localhost/usermix/wp-admin/tools.php?page=xthemes-invoker&xthemespage=buildtheme

Use X-Themes Generator in that page.

To see the final result of your Theme go on:
- 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.

	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.

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

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

	Totally Disagree 1	2	3	4	5	6	Totally Agree 7
Easy to understand	◯	◯	◯	◯	◯	◯	◯
Useful	◯	◯	◯	◯	◯	◯	◯
Easy to use	◯	◯	◯	◯	◯	◯	◯
Efficient	◯	◯	◯	◯	◯	◯	◯
Effective	◯	◯	◯	◯	◯	◯	◯

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

	Totally Disagree 1	2	3	4	5	6	Totally Agree 7
Easy to understand	◯	◯	◯	◯	◯	◯	◯
Useful	◯	◯	◯	◯	◯	◯	◯
Easy to use	◯	◯	◯	◯	◯	◯	◯
Efficient	◯	◯	◯	◯	◯	◯	◯
Effective	◯	◯	◯	◯	◯	◯	◯

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

Contrassegna solo un ovale per riga.

	Totally Disagree 1	2	3	4	5	6	Totally Agree 7
Easy to understand							
Useful							
Easy to use							
Efficient							
Effective							

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

e.g. Title, Description, Loop

Contrassegna solo un ovale per riga.

	Totally Disagree 1	2	3	4	5	6	Totally Agree 7
Easy to understand							
Useful							
Easy to use							
Efficient							
Effective							

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

Contrassegna solo un ovale per riga.

	I don't know	30 minutes - 1 hour	1 hour - 2 hours	2 hours - 3 hours	More than 3 hours
Time needed					

11. **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

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.

	1	2	3	4	5	6	7	
Totally Disagree								Totally Agree

13. **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

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.

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

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/thirdrestaurant/?design=true

Go on the following link to use the XThemes Generator:

- http://localhost/usermix2/wp-admin/tools.php?page=xthemes-invoker&xthemespage=buildtheme

Use X-Themes Generator in that page.

To see the final result of your Theme go on:

- 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.

	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.

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

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

	Totally Disagree 1	2	3	4	5	6	Totally Agree 7
Easy to understand	⬭	⬭	⬭	⬭	⬭	⬭	⬭
Useful	⬭	⬭	⬭	⬭	⬭	⬭	⬭
Easy to use	⬭	⬭	⬭	⬭	⬭	⬭	⬭
Efficient	⬭	⬭	⬭	⬭	⬭	⬭	⬭
Effective	⬭	⬭	⬭	⬭	⬭	⬭	⬭

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

	Totally Disagree 1	2	3	4	5	6	Totally Agree 7
Easy to understand	⬭	⬭	⬭	⬭	⬭	⬭	⬭
Useful	⬭	⬭	⬭	⬭	⬭	⬭	⬭
Easy to use	⬭	⬭	⬭	⬭	⬭	⬭	⬭
Efficient	⬭	⬭	⬭	⬭	⬭	⬭	⬭
Effective	⬭	⬭	⬭	⬭	⬭	⬭	⬭

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

Contrassegna solo un ovale per riga.

	Totally Disagree 1	2	3	4	5	6	Totally Agree 7
Easy to understand							
Useful							
Easy to use							
Efficient							
Effective							

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

Contrassegna solo un ovale per riga.

	Totally Disagree 1	2	3	4	5	6	Totally Agree 7
Easy to understand							
Useful							
Easy to use							
Efficient							
Effective							

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

e.g. Title, Description, Loop

Contrassegna solo un ovale per riga.

	Totally Disagree 1	2	3	4	5	6	Totally Agree 7
Easy to understand							
Useful							
Easy to use							
Efficient							
Effective							

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

Contrassegna solo un ovale per riga.

	I don't know	30 minutes - 1 hour	1 hour - 2 hours	2 hours - 3 hours	More than 3 hours
Time needed					

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. **Wich 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.

	Totally Disagree 1	2	3	4	5	6	Totally Agree 7
Easy							
Effective							
Efficient							

4. **The possibility to drag and drop functionality from other themes is:** *
   Contrassegna solo un ovale per riga.

	Totally Disagree 1	2	3	4	5	6	Totally Agree 7
Easy							
Effective							
Efficient							

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

	1	2	3	4	5	6	7	
Totally Disagree								Totally Agree

6. **X-Themes Generator and Design By Example can improve reusability** *
Contrassegna solo un ovale.

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

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

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

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

....

....

....

....

....

Acknowledgements

At first I want to thank prof. *Moira Norrie*, that followed me and Alfonso during all this beautiful new experience and work. She was always interested in our opinions (also bad ones) and every meeting was productive but also funny, thanks to her. With prof. Norrie, I want to really really really thank dr. *Michael Nebeling*, who followed us during this period. He helped us many times, giving professional and friendly advices, like professor Norrie, he always belived in us, giving thousands of compliments making us feel embarassed the most of the time.
From Italy prof. *Filomena Ferrucci* always was near to us, for burocratical and personal problems, she was always there to help us. We decided to come here in Zurich thanks to her advice and her support. Everytime we send to her absurdly long e-mails asking for thousands of problems, and she just answers everytime trying to help and encourage us. Just thank you.

Unfortunately my whole family cannot understand these aknowledgements, because of the language, but it is not a problem, because there are no languages good enough to say what I really want to express. So *Mamma*, *Enzo* (and *nikla*) just thank you for learning how to use Google Hangout to see us once a week, just thank you for believing in me more than I do. *Daddy*, *Monica*, *Carolina* and *Alessia*, paradoxically we were nearer in this period, and we met each other often, and I was so happy of this, but as you already now the kilometers do not make any difference, because your love reaches me in any case.
Finally, I want to thank, my little friend Paolo. I miss you.
We started this together, and we will end this together.