X-Themes: Supporting Design-by-Example

Moira C. Norrie, Michael Nebeling, Linda Di Geronimo, and Alfonso Murolo

Department of Computer Science, ETH Zurich
CH-8092 Zurich, Switzerland
{norrie,nebeling,lindad,amurolo}@inf.ethz.ch

Abstract. Design-by-example enables users with little technical knowledge to develop web sites by reusing all or parts of existing sites. In CMS such as WordPress, themes essentially offer example designs for all-or-nothing reuse. We propose an extension to the theme concept that allows web sites to be designed by reusing and combining components of different themes. In contrast to previous research advocating design-by-example, we do not restrict ourselves to static web pages, but also support the reuse of dynamic content including functionality for animations and database access. Our approach is to provide a theme generator that structures the themes that it generates in terms of reusable components which can then be reused in future themes. We present a first prototype tool, called the X-Themes Editor, developed to demonstrate the viability of the approach and investigate requirements and issues. We describe how the X-Themes Editor has been integrated into the WordPress platform as well as discussing the outcomes of these initial investigations.

Keywords: design-by-example, content management system, theme, end-user development, reuse in web design.

1 Introduction

Within the research community, model-driven approaches to web engineering have been promoted as the best way of supporting the systematic development of web sites [1]. Often methods are designed to cater for projects involving large, diverse teams including graphic designers, database architects, programmers and marketing staff.

However, an increasing number of professional as well as personal web sites are being developed by individuals using platforms such as WordPress[1] which allows users to dynamically customise crowdsourced themes. A theme defines a set of templates, stylesheets and media for an entire web site and therefore essentially supports only all-or-nothing reuse.

The idea of this paper is to show how this paradigm of design-by-example could be extended to allow users to design their web sites by reusing and combining features of different themes. For example, they might choose the colour scheme and front page layout of one theme, a slider content component from

[1] http://www.wordpress.org

S. Casteleyn, G. Rossi, and M. Winckler (Eds.): ICWE 2014, LNCS 8541, pp. 480–489, 2014.

a second theme, and the drop-down menu style from a third. As discussed in Sect. 2, this approach has been advocated by researchers in the HCI community [2,3], but typically their experiments and studies focus on the static parts of web sites and do not address the technical challenges of extracting and reusing functionality. A key factor in our research was to support the reuse of all aspects of a web site, including animations, client-side processing and database access. Further, to avoid the need for developers to change their work practices, we wanted to investigate how such an approach could be integrated into the WordPress platform.

Our approach, which is presented in Sect. 3, exploits the concept of a theme generator to produce a collection of so-called X-Themes that conform to a well-structured model designed to support reuse. The special X-Themes generated by our tool are then available to other developers who can drag-and-drop components of existing themes into their newly created themes at design time. We outline the main features of a first prototype designed to demonstrate the viability of the approach in Sect. 4 before discussing the main requirements and challenges to be addressed in future research in Sect. 5.

2 Background

Several model-driven approaches to web site development have been proposed, for example WebML [1], Hera [4] and UWE [5]. While these approaches have received acclaim in the research community, they have had limited impact in the web development community at large where many professionals work with a mix of technical knowledge and design skills and build on modern content management systems (CMS) such as WordPress or Drupal[2]. An indication of the scale of the developer community using WordPress is the estimate that 21.9% of the top 10 million web sites are implemented on WordPress[3]. The size and complexity of these web sites varies enormously, but has certainly gone well beyond the original focus on personal blogging sites and now includes online newspapers, e.g. Metro UK[4], e-commerce sites, e.g. Kuborra[5], and information platforms for communities e.g. SAP.info[6].

As highlighted by a recent survey on modern web development practices [6], many of these developers have no formal computer science training and are either self-employed or working in very small organisations. In these settings, developers typically adopt an *interface-driven* approach, starting from a mockup of the interface and first adding client-side functionality before adding server-side functionality by migrating to a CMS platform.

The WordPress platform was developed using a crowdsourcing model that allows their user community to develop and share both *themes* and *plugins*.

[2] `http://www.drupal.org`
[3] `http://www.w3techs.com`, accessed 10 April 2014
[4] `http://metro.co.uk/`
[5] `http://www.kuborra.com/`
[6] `http://en.sap.info/`

A theme is a set of PHP templates, CSS stylesheets and media objects that define the presentation, structure, functionality and types of content of a web site. To develop their own web site, a user simply has to select a theme and then start adding their own content. Themes are usually parametrised so that users can easily customise their sites and they can also extend the types of content and functionality by adding plugins.

Themes can be considered as a form of *design-by-example* [7]. While WordPress themes only support all-or-nothing reuse, researchers have investigated approaches which would allow users to design their web sites by freely selecting and combining parts of example web sites from interactive galleries [2,3]. While these studies have demonstrated the benefits of this general approach, they did not address the technical challenges of being able to extract and combine arbitrary elements of modern web pages that are often dynamic rather than static and make heavy use of JavaScript and jQuery[7]. As a result, their methods were only able to deal with the reuse of appearance and content and not functionality.

A number of approaches for developing web applications from reusable components have been proposed. WebComposition [8] is an object-oriented support system for building web applications through hierarchical compositions of reusable application components. While some mashup editors help users to integrate information from distributed sources, others provide infrastructure for building new applications from reusable components. For example, MashArt [9] enables advanced users to create their own applications through the composition of user interface, application and data components. While our approach shares some of the goals and enables similar extraction and reuse techniques, it offers these at the theme level to base web site designs on multiple examples, which is different from mashups that are direct compositions of existing web sites. More recently, extensions to CMS such as WordPress have been proposed to allow web applications to be developed from a component model that supports composition at the data, application and interface levels [10]. The focus of this work differs in that our approach is *interface-driven* rather than *model-driven* and our component model captures concepts of popular web development platforms such as WordPress as well as the new HTML5 and CSS3 web standards together with jQuery that power the underlying themes.

Summarising, the solution proposed by many researchers has been to try and bring discipline into web engineering by requiring developers to first model different aspects of their web sites and only then generate code. But this usually requires that developers abandon popular platforms and learn new modelling skills, tools and possibly even languages. It also makes it more difficult to support rapid prototyping and allow developers to start by adapting existing web site designs developed either by themselves or other developers. Instead of trying to force developers to change their ways of working, we think it is important to instead find ways of better supporting them. This means that not only should we find ways to support the design-by-example paradigm, but this should be done using existing, popular platforms and technologies.

[7] `http://jquery.com`

3 Approach

While the WordPress platform is very flexible and powerful, the basic model behind it is relatively simple and developers are offered a very loose framework in which to work. As a result, it is possible to build advanced web sites but widely varying approaches are used to achieve similar functionality and presentation.

In the rare case that a personal or professional developer finds an existing theme that fully meets their requirements, the process of developing a web site mainly consists of setting some parameters and adding content through the WordPress dashboard. To some degree, they can even extend the functionality of the theme through this interface by selecting and adding various plugins. However, as soon as a developer is faced with the task of adapting or extending a theme, they have to start working at the level of the HTML, CSS and PHP files and learning about the core WordPress model and system operation. Developers often work on a need-to-know basis, learning only enough to solve the particular task at hand. Further, the documentation and tutorials vary a lot in terms of guidelines and solutions offered to developers. It is clear from reading tutorial-style books, e.g. [11,12] as well as online forums[8] that many developers simply copy and paste bits of CSS, HTML and PHP with the hope that it will achieve the desired effects. However, often these attempts to reuse code fail because they are inconsistent with how other parts of the site have been developed.

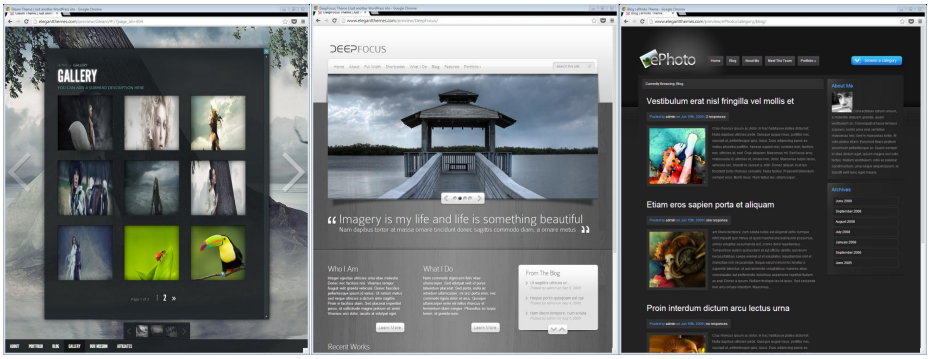

Fig. 1. Examples of portfolio themes from the company Elegant Themes

We want users to be able to create their own themes by selecting and combining parts of different existing themes. For example, Fig. 1 shows three portfolio themes offered by the company Elegant Themes[9]. A user might want to have the general styling of the theme on the left which includes a slider of background images and an animated drop down element in the gallery page, but want to include in the front page the slider component of the theme in the middle and have

[8] For example, `http://www.wpbeginner.com`
[9] `http://www.elegantthemes.com`

a blog post page with the layout shown in the theme on the right. To achieve this, we have developed a graphical theme editor, called the X-Themes Editor, which allows users to select parts of other themes that can be integrated into their own themes using simple drag-and-drop operations.

To support this kind of reuse, it is necessary that themes adhere to a well-defined component model. One way to do this would be to develop a new platform or domain specific language based on a component model but, as stated previously, this is something that we wanted to avoid. Instead, we designed our own metamodel for WordPress themes and then based the X-Themes Editor on this model. This means that any themes generated with our editor are clearly structured in terms of components that can be shared and reused. By offering a theme editor that is integrated into the WordPress platform, developers can already be offered a valuable tool for creating new themes from scratch. Themes generated using the tool are called X-Themes and the collection of X-Themes are made available for reuse in an interactive gallery.

It is important to note that a number of theme generators for WordPress already exist but many of these have serious limitations, especially when it comes to customising functionality. For example, Templatr[10] is a free web-based tool that allows users to customise static elements but they can only select from a fixed set of layouts. Some tools such as Lubith[11] allow users to customise layout via drag and drop, but they usually do not support the customisation of functionality. Another limitation of existing generators is the fact that they are not integrated into WordPress. This means that it is not possible to perform content-related tasks such as displaying the pages or latest posts and comments during the design of the theme and often there can be compatibility problems between WordPress versions. Therefore, offering a theme editor that is integrated into the WordPress platform, and enables not only presentation but also layout and functionality to be customised, is in itself a valuable contribution.

4 X-Themes Editor

Using the possibilities to customise and extend the functionality of the WordPress dashboard, we were able to integrate the X-Themes Editor into the tool options menu offered to developers. When the X-Themes tool is selected, the standard dashboard interface is replaced by the GUI of the X-Themes Editor which, in addition to offering a menu for creating and customising elements of a theme, also provides access to an interactive gallery of previously generated X-Themes. A user can create a theme using a mix of editing operations to define new components and drag-and-drop operations to import selected components from various source themes opened in the X-Themes gallery.

Every theme created in the editor has three main components—Header, Body and Footer— that are visually separated as indicated in Fig. 2. Users can create

[10] `http://templatr.cc`
[11] `http://www.lubith.com`

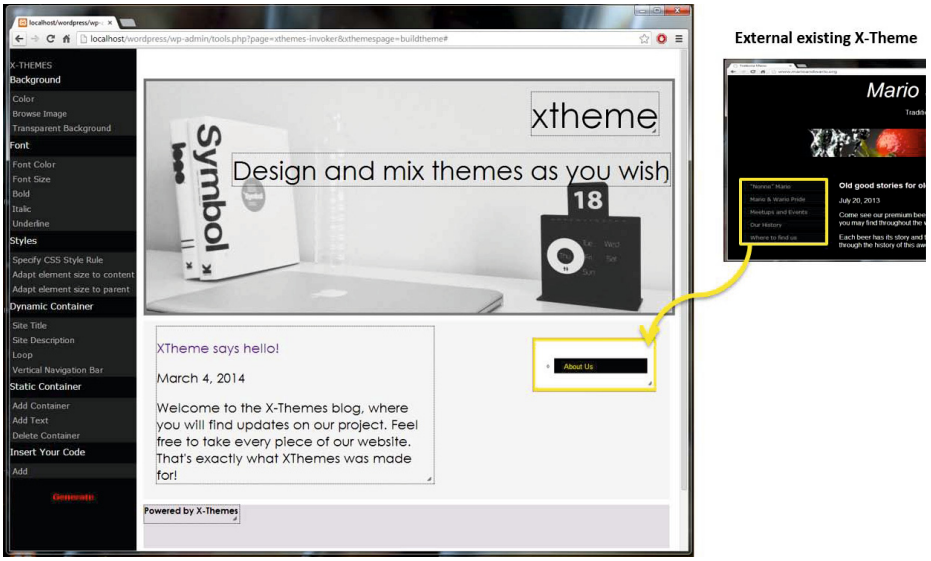

Fig. 2. Dragging a navigation menu from an existing website into the X-Themes editor

different types of containers within each of these main components and each container may itself contain other containers.

As soon as a component from an existing theme is dropped into the new theme, it will be active and any dynamic behaviour experienced. For example, if the user drag-and-drops a dropdown menu into the navigation component as illustrated in Fig. 2, they will immediately be able to try out this functionality in the theme under construction.

There are two main types of container—static and dynamic. A static container has no elements that can change dynamically. A dynamic container is one in which at least one JavaScript or PHP function appears. This could be a function to access a theme parameter or content of the database such as the site's title or application data. A container with any kind of dynamic content will be executed and show the corresponding result already at design time, even when dragged-and-dropped from a different theme.

After a drag-and-drop operation, the user can choose whether or not they want to keep the style of the source theme or have the style of the destination theme applied. For example, a user may wish to keep the information displayed in an imported database query (referred to as a Loop in WordPress) but apply a different design in the new theme. If the user decides to keep the style of the source theme, they can adjust both the format and style later using the general editing functionality of the X-Themes Editor.

When the editing of a theme is complete, the X-Themes Editor generates the set of templates for that theme together with the files defining the components of the X-Theme and associated metadata based on the X-Themes metamodel shown in Fig. 3.

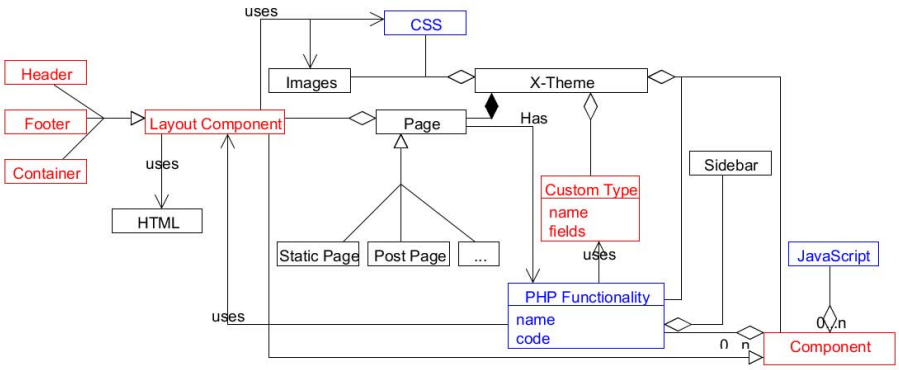

Fig. 3. X-Themes metamodel

Concepts defining basic structural elements of a web page such as header, footer and sidebar are fundamental to the WordPress model as are posts as the primary content type. To extend the model to support other types of content, custom post types can be introduced. Our metamodel introduces the concept of a component shown at the bottom right of Fig. 3. A component can include layout and style rules as well as dynamic behaviour specified as PHP logic or JavaScript. If a component is to be exported and reused, then clearly all code, including CSS rules as well as PHP logic and JavaScript, must also be exported along with the component.

To support the extraction and reuse of components, each component is described in an XML file of the form shown in Fig. 4. All the required PHP, CSS and JavaScript files are stored in a separate folder in the WordPress theme directory, and the DOM of the theme is annotated using HTML5 dataset properties to indicate reusable components based on the metamodel.

The `component_type` can be custom header, sidebar or undefined in the case of a generic component. For header and sidebar components, a `functions` element is included to specify dependencies so it is known which files have to be included in the *functions.php* file of a WordPress theme for that component to be used.

A `clientlogic` element specifies JavaScript code that needs to be exported with the component, while `serverlogic` does the same for PHP logic. Any required style files are specified in the *styles* element. The `include` element specifies the file to be included in a page of the theme to execute and show the component. Various forms of dependencies to other code files are specified in the dependency elements.

A `layout_component` is a special type of component representing the layout structure of the theme and can be a header, a footer or a container. Any of these three types of component can include containers, allowing a fully nested container structure. We distinguish layout components that contain only static content from those that include JavaScript or PHP code.

```
l version="1.0"?>
ture name="dropdown">
omponent_type>undefined</component_type>
lientlogic>
  <name>dropdown.js</name>
clientlogic>
erverlogic>
<name>dropdown.php</name>
<name>dropdown.code</name>
serverlogic>
tyles>
<name>style1.css</name>
styles>
nclude>dropdown.php</include>
ippath>dropdown</zippath>
ependency>dropdown.js</dependency>
ependency>style1.css</dependency>
ependency>dropdown.code</dependency>
ependency>dropdown.js</dependency>
ependency>style1.css</dependency>
ependency>common.css</dependency>
ependency>demo.css</dependency>
ependency>icons.css</dependency>
ependency>dropdown.code</dependency>
ependency>icommon.eot</dependency>
ependency>icommon.svg</dependency>
ependency>icommon.ttf</dependency>
ependency>icommon.woff</dependency>
ature>
```

Fig. 4. Component XML Sample

Each component is allocated a separate directory where its code, resources and XML component files are stored. This includes all PHP required to reuse the component as well as generated CSS files. In the final step of theme generation, an XML representation of the theme defining the component structure is created.

When a user opens a theme in the X-Themes gallery, reusable components are highlighted as the cursor moves over them. If the user drags a component, such as a navigation menu made in PHP/JavaScript and CSS, the X-Themes Editor accesses the metamodel information of that component and acquires references to linked resources which are then cloned for use in the editor. The component is then executed in the context of the X-Themes Editor running on the user's own WordPress installation. The ability for users to view their own content at design time is only one of the advantages of implementing the X-Themes Editor as a WordPress plugin. Another is the fact that it provides an easy means of deploying the tool to existing developer communities and hence raising awareness and chances of acceptance.

5 Discussion

While the X-Themes Editor represents an important first step in showing how design-by-example could be fully supported in CMS platforms such as WordPress, there are a number of issues that need to be addressed in future research.

X-Themes Metamodel. The metamodel that we have worked with so far is based on the WordPress core model and therefore WordPress specific. Further, it focusses more on system and implementation features than general asbtract concepts. One of the next steps will be to generate a conceptual metamodel that could be applied to more than one CMS. Specifically, we are currently investigating how our approach could be applied in Drupal and the metamodel generalised.

Beyond the Front Page. Although the themes that we currently generate are not single page, like many WordPress themes, they very much focus on the

front page which is often seen as defining the main features of a web site in terms of the header, footer, navigation and layout as well as presentation. We need to extend the X-Themes Editor to cater for web sites with more complex structures with variable page layout and content.

Transforming Themes to X-Themes. One of the main ideas behind our approach was to provide users with a graphical theme editor that would in itself be a valuable tool and hence something that developers would want to use independent of the goals of this project. In this way, we could address the cold start problem of generating a collection of X-Themes available for reuse. At the same time, we do think it important that we offer support for transforming existing WordPress themes into X-Themes. This is a topic for future research where we will investigate the extent to which this could be automated.

Interactive Galleries. The current way of searching for WordPress themes relies on descriptions and keywords provided by developers. We want to investigate alternative and complementary ways of supporting search within interactive galleries. This would include search-by-example as well as ways of automatically classifying themes based on a variety of factors.

Data-Intensive Web Sites. Data-intensive web sites require the integration of custom post types to manage application data. In previous work within our research group, a tool was developed that generates a WordPress plugin with custom post types based on an entity-relationship data model defined by a developer [13]. We have now started to investigate an alternative approach that would extend the design-by-example approach to automatically derive data schemas and custom post type definitions based on example data content.

6 Conclusion

We have shown how the support for design-by-example offered by modern CMS platforms such as WordPress could go well beyond the current all-or-nothing approach of sharing entire themes. Users should be able to selectively reuse and combine parts of existing themes, including dynamic components that define functionality. Our work therefore goes beyond previous research on design-by-example paradigms in web development [2,3] which were limited to static views on web sites.

We note however that, while the X-Themes Editor we have developed is sufficient to demonstrate the potential of the approach, as outlined in Sect. 5, there are a number of open issues that would need to be addressed in future research in order for the method to be deployed in practice. Further, the tool would need to advance beyond the prototype stage to support a full palette of editing capabilities expected of a state-of-the-art web design tool.

Acknowledgements. We acknowledge the support of the Swiss National Science Foundation who financially supported this research under project FZFSP0_147257.

References

1. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Applications. Morgan Kaufmann Publishers Inc. (2002)
2. Hartmann, B., Wu, L., Collins, K., Klemmer, S.R.: Programming by a Sample: Rapidly Creating Web Applications with d.mix. In: Proc. 20th ACM User Interface Software and Technology Symposium, UIST (2007)
3. Lee, B., Srivastava, S., Kumar, R., Brafman, R., Klemmer, S.: Designing with Interactive Example Galleries. In: Proc. Conf. on Human Factors in Computings Systems, CHI (2010)
4. Houben, G., Barna, P., Frasincar, F., Vdovjak, R.: Hera: Development of Semantic Web Information Systems. In: Cueva Lovelle, J.M., Rodríguez, B.M.G., Gayo, J.E.L., Ruiz, M.d.P.P., Aguilar, L.J. (eds.) ICWE 2003. LNCS, vol. 2722, pp. 529–538. Springer, Heidelberg (2003)
5. Hennicker, R., Koch, N.: A UML-based methodology for hypermedia design. In: Evans, A., Caskurlu, B., Selic, B. (eds.) UML 2000. LNCS, vol. 1939, pp. 410–424. Springer, Heidelberg (2000)
6. Norrie, M.C., Geronimo, L.D., Murolo, A., Nebeling, M.: The Forgotten Many? A Survey of Modern Web Development Practices. In: Casteleyn, S., Rossi, G., Winckler, M. (eds.) ICWE 2014. LNCS, vol. 8541, pp. 285–302. Springer, Heidelberg (2014)
7. Herring, S., Chang, C., Krantzler, J., Bailey, B.: Getting Inspired! Understanding How and Why Examples are Used in Creative Design Practice. In: Proc. Conf. on Human Factors in Computings Systems, CHI (2009)
8. Gellersen, H.W., Wicke, R., Gaedke, M.: WebComposition: An Object-Oriented Support System for the Web Engineering Lifecycle. Computer Networks 29(8-13) (1997)
9. Yu, J., Benatallah, B., Saint-Paul, R., Casati, F., Daniel, F., Matera, M.: A Framework for Rapid Integration of Presentation Components. In: Proc. 16th Intl. World Wide Web Conference, WWW (2007)
10. Leone, S., de Spindler, A., Norrie, M.C., McLeod, D.: Integrating Component-Based Web Engineering into Content Management Systems. In: Daniel, F., Dolog, P., Li, Q. (eds.) ICWE 2013. LNCS, vol. 7977, pp. 37–51. Springer, Heidelberg (2013)
11. Blakeley-Silver, T.: WordPress Theme Design. Packt Publishing (2008)
12. Casabona, J.: Building WordPress Themes from Scratch (2012)
13. Leone, S., de Spindler, A., Norrie, M.C.: A Meta-plugin for Bespoke Data Management in WordPress. In: Wang, X.S., Cruz, I., Delis, A., Huang, G. (eds.) WISE 2012. LNCS, vol. 7651, pp. 580–593. Springer, Heidelberg (2012)