A Generation Environment for Front-end Layer in E-government Content Management Systems

Vitor Mesaque Alves de Lima, Ricardo Marcondes Marcacini, Marcelo Henrique Pereira Lima
Campus de Três Lagoas
Universidade Federal de Mato Grosso do Sul
Três Lagoas, Brazil
{vitor.lima, ricardo.marcacini, marcelo.lima}@ufms.br

Maria Istela Cagnin, Marcelo Augusto Santos Turine
Faculdade de Computação
Universidade Federal de Mato Grosso do Sul
Campo Grande, Brazil
{istela, turine}@facom.ufms.br

***Abstract*—The internet is a way by which organizations may show its identity, its purposes, its achievements, providing services and information to the public. A Content Management System (CMS) provides an efficient solution for content managing for websites and portals in general. Researches show that the organizations are interested in cost reduction associated to software development, and it is essential to use automated tools and a reuse systematic process. In this paper we present a generation environment for front-end layer in e-government CMS, in context of systematic reuse using Software Product Line (SPL) automated through frameworks, application generators and reuse repository. The generation environment implements automated mechanisms to reduce accessibility problems in generated web applications.**

Keywords—software product line; application generators; framework; reuse repository; e-government; web accessibility

I. INTRODUCTION

The recent developments in Information Technology and Communication (ICTs) have enabled significant improvements in Electronic Government (e-Gov) services, allowing high social participation in government decisions as well as increasing the government transparency and democracy [12]. In this scenario, one of the important online services offered to citizens are the e-Gov portals, which are Web-based applications where different sectors of public administration provide your identity, planning, actions, and outcomes for society [15]. Therefore, the e-Gov portals are key services for promotion of digital democracy and should ensure universal access to information for all citizens without distinction [12].

Although international standards and guidelines for the accessibility of Web-based applications are widely available, such as the WCAG (Web Content Accessibility Guidelines) [16] and e-MAG (Accessibility Model of Electronic Government) [7], many e-Gov portals still have critical issues that limit universal access to the information. For example, a study published in 2010 showed that 98% of the Brazilians e-Gov portals lack basic requirements for Web accessibility [4], thereby hindering access by people with disabilities. These facts have increased the demand for Web accessible environments in the context of e-Gov management systems.

Besides the aspects related to Web accessibility, government organizations also need an agile management process of the information published on the e-Gov portals. In this sense, the use of the Content Management Systems (CMS) has been a useful solution, since they allow the software reuse and the reduction of effort, cost and time of the system management [8] [13] [14]. However, most existing general-purpose CMS, such as Joomla! [1], Drupal [2], PLONE [3], OpenCMS[4], and Typo3[5], also lack the basic requirements of Web accessibility in the front-end layer [2] [10]. Moreover, a study reported by Lima (2013) [10] showed that even the CMS proposed specifically for e-Gov portals do not adequately address the requirements of Web accessibility [11].

Considering the challenges discussed above, an interesting direction of research in the e-government field is Software Product Lines (SPL) for developing accessible Web solutions. According to Pohl et al. (2005) [13], a SPL can be defined as a set of applications developed using platforms and mass customization. In particular, frameworks and application generators can be applied to SPL automation, thereby optimizing the process of instantiation of Web applications. In this context, Carromeu et al. (2010) [3] proposed a SPL architecture for the generation of e-Gov Web applications, and defined a generation environment, which implements the SPL architecture, composed essentially by Titan Framework tool. An advantage of the Titan Framework is the systematic software reuse through a component-based architecture that implements a generation environment for back-end layer. Moreover, the generation of a new instance (Web application) can be automated by a graphical tool, called Titan Architect, which facilitates the parameter settings of the new instance. Some government Web applications have been developed successfully using the Titan Framework, such as Web portal at the Foundation for Research Support of the state of Mato Grosso do Sul – Fundect, the Federal University of Mato Grosso do Sul – UFMS, the National Phone-in System of

[1] Joomla!: http://joomla.org/
[2] Drupal: http://drupal.org/
[3] Plone: http://plone.org/
[4] OpenCMS: http:// opencms.org/
[5] Typo3: http://typo3.org/

978-1-4799-6953-1/14 $31.00 © 2014 IEEE
DOI 10.1109/LAWeb.2014.20

Accusations – DDN 100 for the Special Secretary of Human Rights from the Presidency of the Republic and the Information System of Evaluating Cultural Projects – SIAC (http://minc.ledes.net/) for the Ministry of Culture.

In this paper we introduce the Titan Frontend, a generation environment for accessible front-end layer in E-government Content Management Systems. Our generation environment extends the Titan Framework (generation environment for the back-end layer) and provides a template engine for accessible e-Gov portals. To support the Titan Frontend, we also proposed a repository for software reuse called Titan Extensions, which manages software components and accessible Web templates for creating new CMSs. The accessibility of Web templates generated for the front-end layer is automatically evaluated by using the ASES (Accessibility Simulator and Evaluator for Sites) model, an evaluator and simulator of the Web accessibility for Brazilian e-Gov portals [1]. For this purpose, the proposed Titan Frontend presents a public Web service, called WebASES, which enables online evaluation of the Web accessibility based on eMAG guidelines, thereby controlling the accessibility of the templates inserted in the repository. To demonstrate the effectiveness of Titan Frontend, we present an evaluation using six e-Gov portals of Brazilian city halls. We carried out a comparative analysis of the current e-Gov portals and new portals generated by the Titan Frontend and the results showed that the front-end layer generated by our SPL process is superior from the perspective of Web accessibility.

The remainder of this paper is organized as follows. Section II presents the details of the generation environment proposed in this paper. Section III presents an evaluation of Titan Frontend as well as discussion of the results. Finally, Section IV presents the conclusions and directions for future work.

II. Generation Environment

The generation environment of the SPL for e-Gov accessible CMSs is divided into two generation layers (back-end and front-end), as shown in Fig. 1. Thus, both layers use application generators and frameworks to automate the instantiation process. The back-end layer is first instantiated, and after it the second front-end layer is instantiated. In practice, the instantiation of the front-end layer results in a public area of an e-Gov portal, on the other hand, the back-end layer is a CMS, where the information is entered, stored and retrieved for display on the front-end layer.

The back-end generation layer, proposed by Carromeu et al. (2010), is composed by two tools: the Titan framework and the Titan Architect application generator. The Architect generates the XMLs (eXtensible Markup Language) and SQLs (Structured Query Language), files necessary to instantiate the Titan Framework, and makes configurations of new instances in graphical form. Thus, the e-Gov portals are developed from a common generic architecture implemented by Titan and a set of reusable artifacts. Therefore, each Web application is an instance of Titan Framework that takes as input XMLs files and transforms it in a CMS at runtime. Its architecture has grey-box, flexible, and extendable characteristics, as well as provides diverse native features such as an authentication system and security log, revision control and authorship.

In turn, the front-end generation layer, which we introduced in this paper, is composed by the Titan FrontEnd application generator and Zend framework[6], such that the Titan FrontEnd automatically generates code for an instance of Zend framework. The Titan Extensions in turn, act on both layers of the environment.

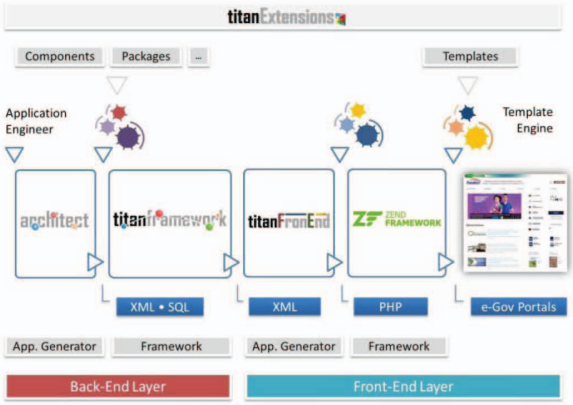

Fig. 1. SPL Environment Generation for e-Gov portals accessible.

A. Titan FrontEnd Tool

Application generators are software systems that transform specifications into an application. Through the use of generators, the application engineer inserts only information about "what" should be done, and the tool decides "how" the information must be transformed into source code [5]. There are two possibilities to build an application generator: constructing a compiler or a composer. Building a compiler means creating a lexical, syntactic and semantic analyzer for a language. Build a composer means creating a software project, derive a set of templates from that project, create a mapping between the specification and these templates, and then use a tool to generate artifacts based on the specifications and templates [17].

The Titan FrontEnd tool is available at http://lives.ufms.br/titanfrontend under the Creative Commons license - (by-nd). The tool can be classified as an application generator, composer, and wizard, able to receive specifications and transform them into software [5]. The Titan FrontEnd composes the environment generation SPL for automated creation and mass customization of CMSs. The tool receives specifications (XML files) provided by a given instance (generated by the back-end generation layer) and generates the source code, at runtime, to the instantiation of the front-end layer. In order to minimize the complexity of the generated code and leverage existing solutions, we chose to use the Zend Web application framework. In summary, the application generator Titan FrontEnd, based on the specifications, automatically generates the source code of the classes that

[6] Zend Framework: http://framework.zend.com

implement the MVC (Model-View-Controller) [9] pattern of the Zend application. Thus, in the generation process, each section of the Titan Framework (back-end layer) was mapped on a Zend module. Fig. 2 shows the resulting directory tree after generation. This automation approach enables a coding process more agile and less susceptible to human error, since the generators can produce safer code than traditional programming methods [5] [6].

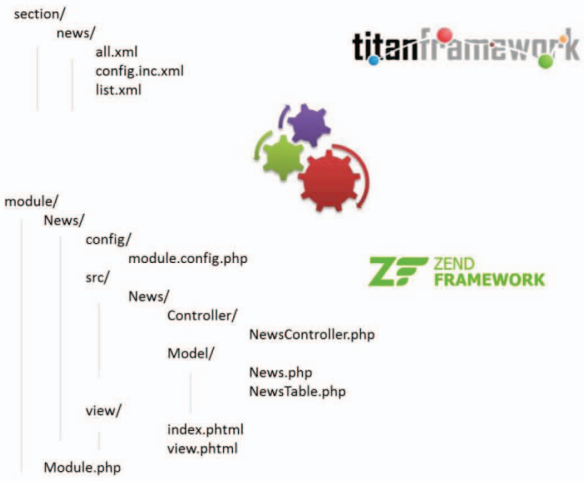

Fig. 2. Directory tree after generation.

The Titan FrontEnd has a template engine that renders the application layout from provided templates. In the generation process, the application engineer can choose predefined layout templates or import it by Titan Extensions repository. Thus, the Web application graphical interface is rendered based on the template layout chosen. Through a well-defined interface, templates can be built and made available on Titan Extensions for reuse, so that the application engineer can change the layout of the pages without the need for coding. Fig. 3 shows an overview of the Titan FrontEnd architecture.

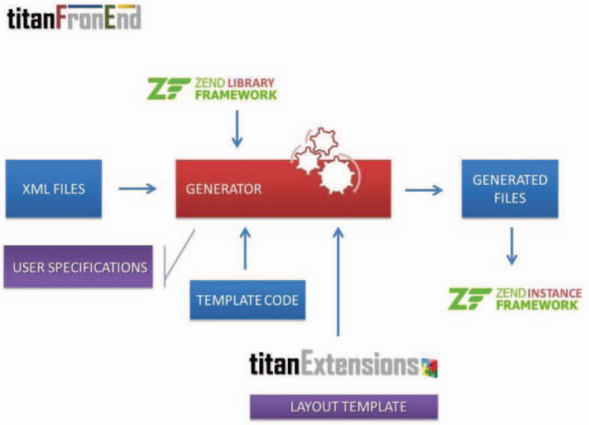

Fig. 3. Titan FrontEnd Architecture.

The generating process is performed in eight steps, as illustrated in Fig. 4, as well as detailed below: i) Zend Configuration: define settings for instantiation of the Zend Framework; ii) Titan Sections: choose the sections that will be generated; iii) Titan Field Sections: choose which attributes/data sections will be generated for the actions list and visualization; iv) Layout Settings: select the layout options; v) Navigation: pagination settings and definition of the content of each section; vi) Advanced Custom: presets positioning of interface elements; vii) Generator: presents a summary of important information relating to the generation and confirmation to the generation process; viii) Finish: presents log of operations performed during the generation process are shown. At the end of the generation process the application engineer can preview the generated portal.

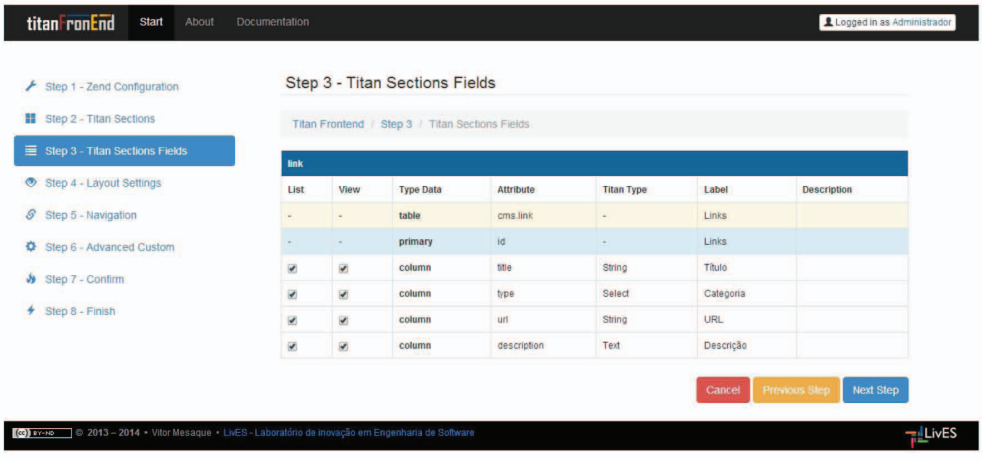

Fig. 4. Step 3: choose which attributes/data sections will be generated for the actions list and visualization.

At each step of the generation process, the checks and validations related to specification generation are performed to provide a safe process. The generation process automates the deployment of some navigational mechanisms such as paging, auto-contrast, resizable font, navigation menu, breadcrumbs, and retrieval system, among others.

B. Accessibility Control Mechanisms

The solution proposed in this paper for the accessibility control in e-Gov portals combines automated mechanisms and the latest Web technologies such as HTML5 (HyperText Markup Language) and CSS3 (Cascading Style Sheets), with the aim to achieve satisfactory levels of accessibility and providing customizable and flexible layouts. The new versions of HTML and CSS provide more semantic and accessible pages and provide more interactivity without the need of installing plugins and loss of performance [15].

The initial code generated by Titan FrontEnd based on the predefined layouts by the generator is accessible and has no error that hinders access to content by people with disabilities. However, the use of custom layout templates may result in insertion of accessibility errors in the code. In this sense, a mechanism for automatic verification of assets was implemented in the Titan Extensions reuse repository to ensure that the available assets in the repository are accessible. Was developed the application called WebASES (available in http://lives.net.br/webases/), which is a Web version of the ASES tool. The WebASES implements a Web service that allows automatic validation of layout templates available in the repository. Thus, whenever a template is added to the repository, its accessibility is evaluated automatically.

III. EVALUATION

In order to evaluate the proposed generation environment, we present an evaluation of six e-Gov portals of Brazilian city halls. Initially, we evaluated the current web accessibility of each front-end layer of the e-Gov portals considering the e-MAG guidelines. The WebASES tool was used to assess the accessibility of the portals. Then, we carried out a simulation for automatic generation of the front-end layer by using the proposed Titan FrontEnd and the templates provided by Titan Extensions. For the evaluation we chose the news section of each portal to perform the comparison. The accessibility errors were categorized according to the priority levels of the e-MAG (Priority 1, Priority 2 and Priority 3).

Below we present the results of the comparative analysis between the current and new portals generated by Titan FrontEnd. In all results, the portals generated by Titan FrontEnd had no accessibility errors classified as Priority 1 (that are points that the creators of Web content should fully satisfy), as shown in Table I. According to the results, all the portals evaluated had accessibility errors of Priority 1, especially the portal of the Belo Horizonte city, with 431 accessibility errors of Priority 1.

TABLE I. COMPARISON OF ACCESSIBILITY ERRORS BETWEEN CURRENT E-GOV PORTALS AND E-GOV PORTALS GENERATED BY TITAN FRONTEND.

Priority Level	Aracaju	Bauru	Belo Horizonte	Campo Grande	Niteroi	São Luis
Accessibility errors of the current e-Gov portals						
Priority 1	83	192	431	76	47	62
Priority 2	17	9	43	12	2	5
Priority 3	0	0	14	0	0	0
Accessibility errors of the e-Gov portals generated by Titan Frontend						
Priority 1	0	0	0	0	0	0
Priority 2	0	0	0	0	0	0
Priority 3	0	0	0	0	0	0

Fig. 5 presents the current e-Gov portal and the new e-Gov portal generated by the Titan Frontend for the Aracaju city, based on developed layout template. It is important to note that the automatically generated front-end layer of the e-Gov portal is similar to the original, without the web accessibility errors.

Fig. 5. Current e-Gov portal and new e-Gov portal generated by the Titan Frontend for the Aracaju city.

IV. CONCLUSIONS

In this paper we presented the Titan Frontend, a generation environment that automates the SPL in domain of e-Gov accessible portals. The tools related to Titan Frontend have been successfully applied in development of new e-Gov portals and have shown that the efforts and resources expended on SPL instantiation process are systematically reused. The Titan FrontEnd presents an automated solution that can significantly reduce the costs associated with the development of front-end layer of e-Gov portals and provide a systematic and reliable process of generation. The reuse and quality of artifacts can be maximized in SPL instantiation with the support of Titan Extension reuse repository. The WebASES tool, in turn, automates the validation process of layout templates to ensure that they are accessible, reducing barriers to accessibility in generated e-Gov portals.

The environment generation and the tools have been constantly validated. Since it was created, some new WebApps were developed and others are currently being developed, such as the Web portal at the Regional Chemistry Council of Mato Grosso do Sul - CRQ-XX (http://crq.lives.net.br) – Ministry of Labor, Web portal at the Observatory of Human Resources in Mato Grosso do Sul (http://observarh.ufms.br/) – Ministry of Health and Web portal at the Week of Education (http://lives.ufms.br/semanadeeducacao) – Ministry of Education.

The generation environment has been constantly developed and case studies in other domains have been carried out. Moreover, new implementations, for instance for a mobile platform, show its considerable flexibility. Research are being conducted in order to identify and map new navigational features that can be automatically generated and incorporated into the Titan FrontEnd instances as well as the exploration of new technologies to improve the generation of web accessible e-Gov portals.

REFERENCES

[1] ASES. Accessibility Simulator and Evaluator for Sites, http://www.governoeletronico.gov.br/acoes-e-projetos/e-MAG/ases-avaliador-e-simulador-de-acessibilidade-sitios, 2012. (In Portuguese)

[2] Burzagli, L., Gabbanini, F., Natalini, M., Palchetti, E., Agostini, A. Using Web Content Management Systems for Accessibility: The experience of a Research Institute Portal. Lecture Notes in Computer Science, v. 5105 LNCS, p. 454-461, 2008.

[3] Carromeu, C., Paiva, D.M.B., Cagnin, M.I., Rubinsztejn, H.K.S., Turine, M.A.S., Breitman, K. Component-Based Architecture for e-Gov Web Systems Development. Engineering of Computer Based Systems (ECBS), 17th IEEE International Conference and Workshops, p.379-385, 2010.

[4] CGI.br e NIC.br. Dimensions and characteristics of the Brazilian web: a study of gov.br, http://www.cgi.br/publicacoes/pesquisas/govbr/cgibrnicbr-censoweb-govbr-2010.pdf, 2010.

[5] Cleaveland, J. C. Program Generators with XML and Java. Prentice Hall, 2001.

[6] Czarnecki, K. e Eisenercker, U. W. Generative Programming. Addison-Wesley, 2002.

[7] E-MAG. e-MAG: Accessibility Model for Electronic Government, http://www.governoeletronico.gov.br/acoes-e-projetos/e-MAG, 2011. (In Portuguese)

[8] Frakes, W. B. e Kang, K. Software Reuse Research: Status and Future. IEEE Transactions on Software Engineering, v. 31, n.7, p. 529-536, 2005.

[9] Gamma, E., Helm, R., Johnson, R., Vlissides, J. Design Patterns – Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.

[10] Lima, V. M. A. Software Product Line for e-Gov Portals Accessible. Master's Thesis, Federal University of Mato Grosso do Sul, 2013. (In Portuguese)

[11] López, J. M., Pascual, A., Menduiña, C., Granollers, T. Methodology for Identifying and Solving Accessibility Related Issues in Web Content Management System Environments. Proceedings of the International Cross-Disciplinary Conference on Web Accessibility. Anais...New York, NY, USA: ACM, 2012.

[12] PINHO, J. A. G. Investigating e-government portals of states in Brazil: plenty of technology, little democracy. Public Administration Journal. Rio de Janeiro - RJ. v.42, n. 3, p. 471-93, 2008. (In Portuguese)

[13] Pohl, K., Bockle, G., Linden, F. Software Product Line Engineering: Foundations, Principles, and Techniques. Springer, 2005.

[14] Pressman, R. S. Software Engineering: A Practitioner's Approach. 6th ed. New York, USA: McGraw Hill, 2005.

[15] W3C Brasil. World Wide Web Consortium Escritório Brasil, 2013.

[16] WCAG 2.0. Web Content Accessibility Guidelines (WCAG) 2.0, http://www.w3.org/TR/2008/REC-WCAG20-20081211/, 2012.

[17] Weiss, D. M.; Lai, C. T. R. Software Product-Line Engineering. Addison-Wesley, 1999.