A. Vázquez-Ingelmo, F. J. García-Peñalvo and R. Therón, "Application of domain engineering to generate customized information dashboards," in *Learning and Collaboration Technologies. Design, Development and Technological Innovation. 5th International Conference, LCT 2018, Held as Part of HCI International 2018, Las Vegas, NV, USA, July 15-20, 2018, Proceedings, Part II*Lecture Notes in Computer Science, no. 10925, pp. 518-529, Cham, Switzerland: Springer, 2018. doi: 10.1007/978-3-319-91152-6_40.

Application of domain engineering to generate customized information dashboards

Andrea Vázquez-Ingelmo[1[0000-0002-7284-5593]], Francisco J. García-Peñalvo[1[0000-0001-9987-5584]] and Roberto Therón[1,2[0000-0001-6739-8875]]

[1] GRIAL Research Group, Computer Sciences Department, Research Institute for Educational Sciences, University of Salamanca, Salamanca, Spain
[2]VisUSAL Research Group. University of Salamanca. Salamanca, Spain
{andreavazquez,fgarcia,theron}@usal.es

Abstract. Information dashboards play a key role in analyzing and visualizing data about a specific topic or domain. In essence, these dashboards display information and enable users to reach insights and make informed decisions. However, end users can have several necessities that are different from each other, including the displayed information itself or other design features. For these reasons, a domain engineering approach is proposed in order to produce customized dashboards adapted to singular requirements of every involved user (or group of users) by the parameterization of features, presentation components and data sources, obtaining a software product line of information dashboards. The creation of a product line would increase productivity, maintainability and traceability regarding the evolution of the dashboards' requirements. To validate this approach, a case study of its application in the context of the Spanish Observatory of Employability and University Employment ecosystem is described, where users (Spanish universities and administrators) will control their own dashboards to reach insights about the employability of their graduates.

Keywords: Domain Engineering, SPL, Information Dashboards, Information Systems.

1 Introduction

Data is a vital resource for most activities, whether trivial or complex. Data supports informed decision making, and helps to understand the target domain. Over the last few decades, the amount of data has been continuously growing, making its management increasingly difficult. This tendency has led to a growth in data-driven systems [1], in order to provide efficient and effective data management.

However, data do not generate knowledge themselves; it is necessary to exploit the collected data through its analysis in order to obtain valuable information. This information will help in the decision-making process, forming the basis for knowledge and wisdom generation [2]. In essence, data needs to be processed in order to enable users to reach insights about the domain of study.

Information dashboards play a key role in visualizing the outcomes derived from the data processing through graphical resources, and provide data analysts (and even non-technical users) an instrument to get different perspectives of the results, promoting the establishment of relations among its variables (with the ultimate objective of having a robust information base to make decisions).

But, it is necessary to take into account the continuous evolution of data (and technology). As it has been said before, data are increasing both in quantity and complexity, hence the tools in charge of the data management need to be versatile, scalable and must have the capacity to adapt to new data structures, otherwise the maintenance of the tools will consume a lot of resources. This means that any change in the requirements has to take effect quickly, in order to avoid a slow-down in the decision-making processes, and thus, in the achievement of the established goals.

Therefore, information dashboards need to be flexible, maintainable and scalable in terms of content and design, so that they can evolve fluently along time.

But dashboards not only face the data growth problem; final users can present very different requirements from each other, and it is important to consider all of them because the user experience depends on the achievement of its requirements. However, implementing customized dashboard for each user is not trivial and would consume significant development time.

Study fields like Software Product Lines (SPLs) [3, 4] and Model Driven Development (MDD) [5] provide potentially useful solutions to produce dashboards with the previous characteristics.

Through the exploration of commonalities and variabilities in the requirements of the problem's domain [4], the SPL approach aims to break down the solution in primary features that can be parameterized and composed to create variations adjusted to the target users. Software product lines foster an environment where reutilization and independent component configuration (through variability points) are the key features of its potential benefits. Giving the fact that dashboards can be seen as a combination of visual components with different (and well-defined) functionalities and data sources, the software product line paradigm is suitable for these kind of tools [6].

However, embracing the practices from the SPL paradigm could not be enough in some contexts. A software product line can have a lot of variability points and features, and the configuration process (to obtain specific products from the family) may still be a time-consuming task. That is why the combination of Model Driven Development and Software Product Lines paradigms [7] can be very powerful in terms of development time and effort.

With regard to all these considerations, a proposal to automate the generation of customized dashboards (in the context of the Observatory of Employability and Employment, described in Section 2) is made through the design of a domain specific language (DSL) to control all the features and parameters of the dashboard product

family. The designed DSL fuels a source code generator that injects the specific configuration on the SPL core assets, in order to create new members of the product line.

The rest of this paper is organized as follows: Section 2 describes the problems faced by the OEEU regarding the visualization of its variables and the personalization of its products for each user, followed by Section 3, in which the materials and methods used to implement the SPL are listed. Section 4 summarizes the results obtained by the application of the SPL approach to the OEEU ecosystem, followed by section 5, where these results are discussed. Finally, section 6 offers the conclusions derived from this work and future research lines.

2 The Observatory of Employability and Employment (OEEU)

The Observatory of Employability and University Employment (also known as its Spanish acronym: OEEU, http://oeeu.org) is an organization formed by a group of researchers and technicians with the vision of becoming an information reference for understanding and exploiting knowledge about employment and employability of students from Spanish universities [8, 9].

All of the Observatory activities (and products) are backed up by data, so it can be treated as a data-driven organization. Given its vision and mission, the OEEU needs tools and technical support for an appropriate data management. In addition, they also require methods to extract knowledge from all the data gathered (more than 500 variables from approximately 19500 students' responses to the OEEU questionnaires and from the information provided by Spanish universities).

The Observatory's studies can help universities to improve their policies and metrics regarding their students' employability, as well as their skills level and satisfaction with the educational methodologies, among other factors. This analysis of both organizational and students' data frames the Observatory in the scope of fields like Academic Analytics [10] and knowledge management in the university scope [11, 12].

The mentioned Observatory's technical support is composed by a series of components (with well-defined responsibilities like collecting, storing, analyzing, disseminating or visualizing the domain's data [13, 14]) that conform a data-driven technological ecosystem. This ecosystem is based on the collaboration of its components through data flows to achieve the Observatory's main goals.

Once the data has been collected and analyzed, the Observatory faces the task of presenting the study results to the ecosystem's users (i.e. administrators, Spanish universities and general users). However, this task is not trivial, as the users have different requirements and permission levels to access the study data.

The path taken in previous study editions was to make a generic dashboard for every user, changing only the data source (and consequently, the data being presented) to match the users' permission level. Although this solution worked fine, it was poorly scalable and maintainable along time, because data, design and functional requirements change among study editions (having to implement a new generic dashboard to meet

the new requirements). Also, personalization of the dashboards in terms of functionality and design was not affordable.

For these reasons an approach based on domain engineering fits the OEEU necessities, as it will allow developers to generate customized dashboards.

3 Materials and Methods

The methodology used to aboard the customized dashboard generation is based on the SPL and MDD paradigms, as introduced in the first section. This methodology allows the Observatory to specify and combine the particular dashboards' features to subsequently generate and deploy their source code.

3.1 Base system

The base system in which the dashboard product line is integrated is based on Django (a Python web framework, https://www.djangoproject.com/) giving support for user and views management. This base system also provides support for a GraphQL API [15], in order to decouple the different ecosystem's components from the data sources.

These two elements (the base system and the GraphQL API) are common to all members of the SPL: users need to authenticate in order to access their customized dashboard page and they will consume data from the OEEU's GraphQL API.

3.2 GraphQL API

The GraphQL paradigm gives the chance to create flexible data flows between the dashboard components and the backend (i.e. the OEEU database) while maintaining high levels of decoupling.

One of its most beneficial features is the use of a single endpoint [16-18], so all the data requests are made through queries (embedded in the HTTP requests as payload) instead of specific HTTP requests to specific endpoints. The queries can be easily parameterized, which is a valuable feature because it allows their contents to be defined through the configuration files depending on the users' data requirements.

The GraphQL API is restricted so users can only access to their own dataset (i.e. their own students' data or national statistics).

3.3 Feature Model and DSL

A domain specific language (DSL) is proposed to manage the requirements of the Observatory users regarding the dashboard elements in a centralized and high-level way, being the basis for the automatic source code generator.

After analyzing the similarities and differences between the users' requirements and the elements of the information dashboards (domain analysis phase), a set of rules and relationships were drawn to specify the possible configurations of the OEEU dashboard screens. One of the artifacts obtained at the end of this domain engineering phase is

called feature model [19]. A feature model is a tree-structured diagram where its leaf nodes represent primitive features that can be composed to obtain other complex features in which the product line is factorized.

In the case of the Observatory, the complete dashboard (i.e. the dashboard with all the possible features) has access to three different screens so far, each one with its set of potential configurations. For example, one of this screens (the "general overview" screen) has been designed to hold a tool that allow the exploration of all the OEEU collected variables through a scatter chart.

The Fig. 1 shows a portion of the feature model that defines the General Overview screen elements and functionalities. This diagram has been simplified (detailed features have been omitted) in order to make it clearer.

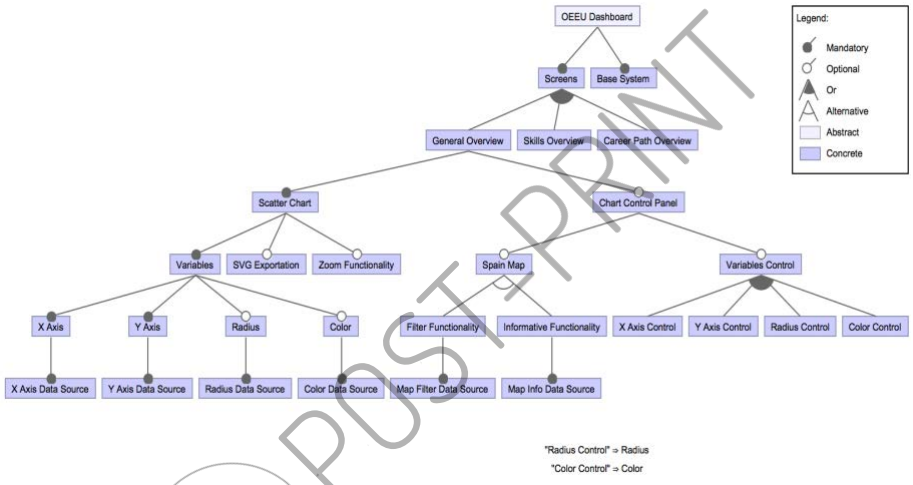

Fig. 1. Portion of the feature model diagram for the OEEU's dashboard product line.

The textual DSL was designed following the feature model (with the purpose of controlling all the parameters and rules specified by this model) and it has been implemented making use of XML [20] technology, with its syntactic rules defined through a XML schema (also referred to as XML Schema Definition, XSD [21]). The XML schema ensures that the configuration files of specific products of the family are valid and consistent with the DSL (and, consequently, with the feature model).

XML technology provides a readable and easy-to-parse way to specify the SPL features, which is beneficial both for the domain engineers (configuration files are close to the domain) and the source code generator (efficient parsing and straightforward extraction of properties). In addition, the XML language has a hierarchical (or tree) structure, as well as feature diagrams. This shared property simplifies the mapping of feature sets to XML schemas.

3.4 Dashboard Code Generation and Deployment

Once the previous resources (the feature model and the domain specific language) were obtained, every screen of the OEEU's dashboard product line was refactored into base code templates and optional features following the structure and restrictions of the DSL.

The source code generator is written in Python, and makes use of Jinja2 templates and macros [22] to inject the code logic of the selected features from configuration files. These templates also follow the syntax of the custom XML schema to enable the injection of the specific features. The Jinja2 template engine allows the definition of variables and macros, as well as the use of conditional and loop statements, which make the templates more readable and their logic easier to code.

With this approach, the code templates represent the core functionalities of every component, and the defined macros represent the variation points [3] of the SPL.

Therefore, the domain assets of the dashboard product line are the code templates that hold the logic of the data visualization components and the structure/design of the dashboard's screens.

The information visualization components are implemented in D3 [23] (a JavaScript library for data visualizations), following Mike Bostock's reusable pattern for charts and building interfaces for each kind of visualization to make them easily configurable and reusable. Every primary dashboard component is encapsulated in a (Jinja2) template, and its internal functionalities are encapsulated in (Jinja2) macros, making it possible to change the behavior of the components through the XML configuration.

This methodology allows the generation of the HTML, CSS and JS files required to run the dashboard.

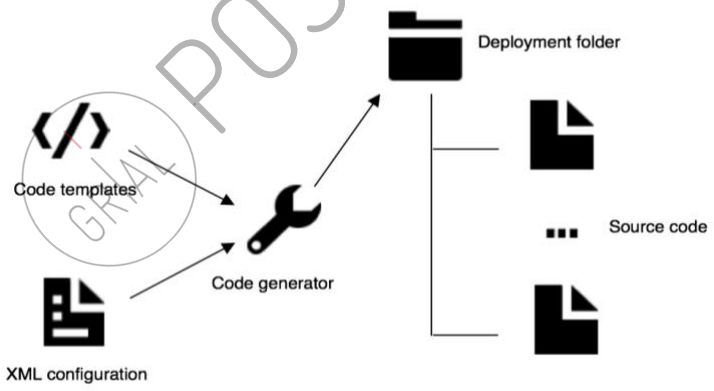

Fig. 2. Outline of the code generator inputs and outputs.

As shown in the Fig. 2, the Python source generator receives as inputs the code templates and the XML configuration file for the target dashboard. The script is responsible for rendering the code templates with the selected features given by the XML specification. While the render process is done, the files are deployed to be served by the OEEU's ecosystem.

4 Results

Applying the software product line paradigm along with automatic code generation has provided valuable results regarding development time, maintainability and scalability of the OEEU's dashboards.

The refactoring of functionalities in a set of macros and templates makes the evolution of the requirements smooth, as new templates can be coded whenever new functional requirements are introduced (without affecting old functionalities). Not only global OEEU's requirements can evolve fluently, also changes in individual users' requirements can take effect quickly without entailing significant development time.

All dashboard components can be modified in terms of functionality, layout and source for data consumption (following the feature model's rules and constraints).

For example, one of the dashboard's optional components is a Spain map, which can have filter functionalities (to filter the data presented on the main visualization of the dashboard screen) or informative functionalities (to show information about variables regarding the different autonomous communities of Spain).

To change the functionality of this component, the specific XML configuration file has to be updated to contain an "Information" (Fig. 3) or "Filter" (Fig. 5) attribute under a "Spain Map" parent node (which represents the base component), depending on the necessities of the user. The resulting (JavaScript) file after the code generation process will only contain the specified functionalities.

```
<ChartControlPanel>
    <SpainMap>
        <Functionalities>
            <Information>
                <Description>Estudiantes residentes en cada comunidad</Description>
                <DataSource>
                    <code>CCAA</code>
                    <metric_code>oneLevelGroupedCount</metric_code>
                </DataSource>
            </Information>
        </Functionalities>
    </SpainMap>
```


Fig. 3. Specification of the "informative functionality" for the OEEU's SPL Map component.

The Fig. 4 illustrates the result of the XML configuration proposed in the Fig. 3. This configuration adds an informative functionality to the component, making it possible to obtain information about the autonomous communities when the user hovers the mouse over the elements of the map.

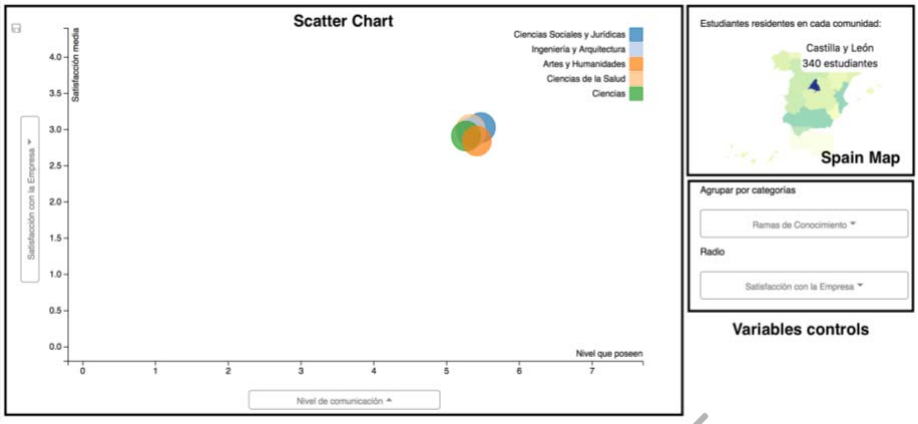

Fig. 4. General overview screen with a map component with informative functionalities (contents in Spanish). The main screen's components are marked within the black rectangles.

On the other hand, the addition of filter functionalities to the map component share the same steps. In this case, the XML configuration must contain the "Filter" attribute inside the "Spain Map" definition (Fig. 5).

```
<ChartControlPanel>
    <SpainMap>
        <Functionalities>
            <Filter>
                <Description>Filtro por comunidad</Description>
                <DataSource>
                    <code>CCAA</code>
                    <metric_code>oneLevelGroupedCount</metric_code>
                </DataSource>
            </Filter>
        </Functionalities>
    </SpainMap>
```

Fig. 5. Specification of the "filter functionality" for the OEEU's SPL Map component.

The result for this configuration is shown in the Fig. 6. This functionality allows users to filter the data being visualized by different variables.

Because the "map component" is an optional feature in the SPL (as described in the Fig. 1), it is also possible to omit it and leave more space for other visualization controls preferred by the user.

This example illustrates the configuration steps for every feature, obtaining a versatile product line of dashboards. Any variability point defined in the feature model can be controlled and parameterized through XML configuration files.

Every user (or set of users) has its own dashboard configuration, so they are able to adapt their screens to meet their own particular needs.

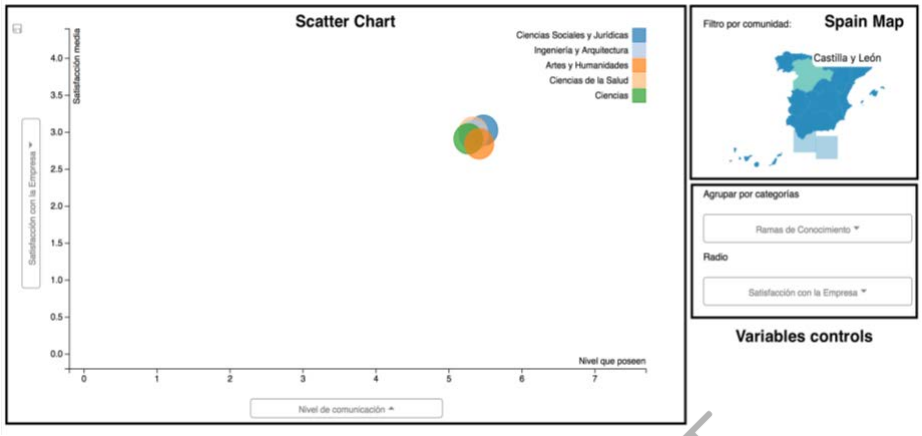

Fig. 6. General overview screen with a map component with filter functionalities (contents in Spanish). The main screen's components are marked within the black rectangles.

5 Discussion

The application of the SPL paradigm to the OEEU's ecosystem (more specifically to the ecosystem's presentation components [14]) has brought significant benefits and solutions regarding different issues the Observatory faced.

First, the study of the domain (domain engineering) to obtain the commonalities and variabilities among the dashboards' features has provided a feature model that will increase the traceability between the formal specification of the SPL and its final implementation. This is an important benefit because the feature model drives the development process instead of acting as mere documentation artifact (one of the basis of the MDA paradigm [5]). The active use of this feature model for code generation grants its constant inspection, providing a valuable and up-to-date documentation resource for the SPL evolution.

Secondly, the refactoring of the dashboard's source code into templates and macros makes the code more organized and even clearer for the developers. This does not simply mean that the readability is improved, but also simplifies the code maintainability and code documentation tasks.

However, the most important benefits come from the capacity to vary the dashboards' design and functionalities without requiring significant development time. This has valuable implications regarding knowledge extraction from data, because every user (universities, in the case of the OEEU's context) can have a dashboard that fits their needs in short term, being able to explore up-to-date data and take well-informed decisions regarding the presented results faster.

Some users could only need a few components and data sources to focus their attention in their own priorities (as a deluge of information might cause confusion and even a slow-down in the knowledge extraction process). But other users could request a complete dashboard to be able to explore every aspect of their datasets.

The approach proposed makes that product personalization possible and fosters the data exploitation by focusing the attention on the final users of the dashboards, while consuming less development time.

This solution also implies an improvement in terms of scalability and maintainability. If any new component or functionality not contemplated in the original requirements is needed or modified, it only implies an extension (or modification) of the feature model (without affecting the other features) and the implementation (or modification) of the assets responsible of the new elements of the product line. As it has been said before, keeping track of the feature model's versions allow developers to follow the SPL evolution.

Regarding the concrete implementation of the OEEU's dashboard product family, GraphQL has proved to be a nice option to connect the dashboard components to the backend, creating flexible data flows between them. Its query-based approach makes the data requests more maintainable for this application than other approaches like REST [15, 24].

On the other hand, the use of code templates to implement the core assets makes the solution applicable for other programming languages, as the content of the templates is easily modifiable and the templating language is independent to the final source code language.

In addition, this dashboard product line provides a potential framework to perform A/B testing [25-27] and to study which product configurations give better support to reach insights about a particular topic.

To sum up, the application of the SPL paradigm to information dashboards allows quick creation and testing of dashboards to meet the final users' necessities, and it enables them to achieve their main goal: to take informed decisions.

6 Conclusions

The SPL approach has been applied to address the OEEU issues regarding the customization and maintainability of their information dashboards. The process of building new specific products from the SPL has been automated through a code generator that takes as inputs code templates and XML configurations (based on a custom DSL designed for the OEEU's dashboard requirements).

The dashboard product line has been validated through its application in the OEEU's ecosystem, reducing development time and increasing maintainability regarding the implementation of customized dashboards for each OEEU user.

Future research lines might involve the automatic linkage between the feature model and the DSL, or the application of the obtained framework for A/B testing and other evaluations regarding human-computer interaction.

Acknowledgments. The research leading to these results has received funding from "la Caixa" Foundation. This work has been partially funded by the Spanish Government Ministry of Economy and Competitiveness throughout the DEFINES project (Ref. TIN2016-80172-R).

References

1. Patil, D., Mason, H.: Data Driven. " O'Reilly Media, Inc." (2015)
2. Zeleny, M.: Management support systems: Towards integrated knowledge management. Human systems management 7, 59-70 (1987)
3. Gomaa, H.: Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures. Addison Wesley Longman Publishing Co., Inc. (2004)
4. Pohl, K., Böckle, G.n., Linden, F.J.v.d.: Software Product Line Engineering: Foundations, Principles and Techniques. Springer-Verlag New York, Inc. (2005)
5. Kleppe, A.G., Warmer, J., Bast, W., Explained, M.: The model driven architecture: practice and promise. Addison-Wesley Longman Publishing Co., Inc., Boston, MA (2003)
6. Freeman, G., Batory, D., Lavender, G.: Lifting transformational models of product lines: A case study. In: International Conference on Theory and Practice of Model Transformations, pp. 16-30. Springer, (2008)
7. Perovich, D., Rossel, P.O., Bastarrica, M.C.: Feature model to product architectures: Applying MDE to Software Product Lines. In: Software Architecture, 2009 & European Conference on Software Architecture. WICSA/ECSA 2009. Joint Working IEEE/IFIP Conference on, pp. 201-210. IEEE, (2009)
8. Michavila, F., Martín-González, M., Martínez, J.M., García-Peñalvo, F.J., Cruz-Benito, J.: Analyzing the employability and employment factors of graduate students in Spain: The OEEU Information System. In: Proceedings of the 3rd International Conference on Technological Ecosystems for Enhancing Multiculturality, pp. 277-283. ACM, (2015)
9. Michavila, F., Martínez, J.M., Martín-González, M., García-Peñalvo, F.J., Cruz-Benito, J.: Barómetro de Empleabilidad y Empleo de los Universitarios en España, 2015 (Primer informe de resultados). (2016)
10. Bichsel, J.: Analytics in higher education: Benefits, barriers, progress, and recommendations. EDUCAUSE Center for Applied Research (2012)
11. Fidalgo-Blanco, Á., Sein-Echaluce, M.L., García-Peñalvo, F.J.: Knowledge Spirals in Higher Education Teaching Innovation. International Journal of Knowledge Management 10, 16-37 (2014)
12. Fidalgo-Blanco, Á., Sein-Echaluce, M.L., García-Peñalvo, F.J.: Epistemological and ontological spirals: From individual experience in educational innovation to the organisational knowledge in the university sector. Program: Electronic library and information systems 49, 266-288 (2015)
13. García-Holgado, A., Cruz-Benito, J., García-Peñalvo, F.J.: Analysis of knowledge management experiences in spanish public administration. In: Proceedings of the 3rd International Conference on Technological Ecosystems for Enhancing Multiculturality, pp. 189-193. ACM, (2015)
14. Vázquez-Ingelmo, A., Cruz-Benito, J., García-Peñalvo, F., Martín-González, M.: Scaffolding the OEEU's Data-Driven Ecosystem to Analyze the Employability of Spanish Graduates. In: García-Peñalvo, F. (ed.) Global Implications of Emerging Technology Trends. IGI Global (2018)
15. Vázquez-Ingelmo, A., Cruz-Benito, J., García-Peñalvo, F.J.: Improving the OEEU's data-driven technological ecosystem's interoperability with GraphQL. Proceedings of the 5th

International Conference on Technological Ecosystems for Enhancing Multiculturality, pp. 1-8. ACM, Cádiz, Spain (2017)

16. https://facebook.github.io/graphql/

17. Faassen, M.: GraphQL and REST, Secret Weblog. (2015)

18. Pandya, D.: GraphQL Concepts Visualized. pp. Web log post (2016)

19. Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S.: Feature-oriented domain analysis (FODA) feasibility study. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst (1990)

20. Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E., Yergeau, F.: Extensible markup language (XML). World Wide Web Journal 2, 27-66 (1997)

21. Fallside, D.C.: XML schema part 0: Primer. W3C, April 2000 (2000)

22. Ronacher, A.: Jinja2 Documentation. Welcome to Jinja2—Jinja2 Documentation (2.8-dev) (2008)

23. Bostock, M., Ogievetsky, V., Heer, J.: D^3 data-driven documents. IEEE transactions on visualization and computer graphics 17, 2301-2309 (2011)

24. Fielding, R.T., Taylor, R.N.: Principled design of the modern Web architecture. ACM Transactions on Internet Technology (TOIT) 2, 115-150 (2002)

25. Kakas, A.C.: A/B Testing. (2017)

26. Cruz-Benito, J., Vázquez-Ingelmo, A., Sánchez-Prieto, J.C., Therón, R., García-Peñalvo, F.J., Martín-González, M.: Enabling adaptability in web forms based on user characteristics detection through A/B testing and machine learning. IEEE Access (2017)

27. Cruz-Benito, J., Therón, R., García-Peñalvo, F.J., Sánchez-Prieto, J.C., Vázquez-Ingelmo, A., Martín-González, M., Martínez, J.M.: Improving success/completion ratio in large surveys: a proposal based on usability and engagement. In: International Conference on Learning and Collaboration Technologies, pp. 352-370. Springer, (2017)