Vázquez-Ingelmo, A., García-Peñalvo, F. J., & Therón, R. (2018). Domain engineering for generating dashboards to analyze employment and employability in the academic context. In F. J. García-Peñalvo (Ed.), TEEM'18 Proceedings of the Sixth International Conference on Technological Ecosystems for Enhancing Multiculturality (Salamanca, Spain, October 24th-26th, 2018) (pp. 896-901). New York, NY, USA: ACM. doi:10.1145/3284179.3284329

Domain engineering for generating dashboards to analyze employment and employability in the academic context

Andrea Vázquez-Ingelmo
GRIAL Research Group, Department of Computer Science, Research Institute for Educational Sciences. University of Salamanca
Paseo de Canalejas 169, 37008
Salamanca, Spain
andreavazquez@usal.es

Francisco J. García-Peñalvo
GRIAL Research Group, Department of Computer Science, Research Institute for Educational Sciences. University of Salamanca
Paseo de Canalejas 169, 37008
Salamanca, Spain
fgarcia@usal.es

Roberto Therón
GRIAL Research Group, Department of Computer Science, Research Institute for Educational Sciences. University of Salamanca
Paseo de Canalejas 169, 37008
Salamanca, Spain
theron@usal.es

ABSTRACT

Data analysis is a key process to foster knowledge generation regarding particular domains or fields of study. With a strong informative foundation derived from the analysis of collected data, decision-makers can make strategic choices with the aim of obtaining valuable benefits in their specific areas of action. However, given the steady growth of data volumes, data analysis needs to rely on powerful tools to enable knowledge extraction. Dashboards offer a software solution for visually analyzing large volumes of data in order to identify patterns and relations and make decisions according to the presented information. But decision-makers may have different goals and, consequently, different necessities regarding their dashboards. Having a methodology to efficiently generate dashboards taking into account differing needs would add a customization layer to allow particular users to reach their own goals. This approach can be achieved through domain engineering and automatic code generation processes. This paper presents the application of domain engineering within the dashboards' domain through a case study in the context of the Spanish Observatory for University Employment and Employability, in which a set of dashboards can be generated to exploit different perspectives of employment and employability data in the academic context.

CCS CONCEPTS

• **Software and its engineering → Reusability** • **Human-centered computing → Visualization toolkits**

KEYWORDS

Domain engineering; software product lines; information dashboards; information systems; university employment; university employability.

1 INTRODUCTION

Informed decision-making processes have gained relevance over the years given the potential benefits of using data for building strong informative foundations [1]. Data collection is a crucial stage in data-driven [2] activities, but until its analysis, data has no real value. The analysis of the collected data opens up the possibility of generating knowledge from it and, consequently, to improve and obtain benefits from the execution of strategic choices [3]. However, the introduction of information systems to support a great diversity of processes has caused an exponential growth of generated data.

This situation has led to the necessity of powerful tools for managing and analysing large volumes of collected data in order to support and ease knowledge generation processes.

Information dashboards are one of the most commonly used software tools to explore data in an interactive and friendly way [4], providing a solution for visually analysing datasets and identifying relevant factors or patterns at a glance.

There are certain fields of study, like employability, that could take advantage from these tools. This research area has not yet a strong theoretical foundation given the complexity of acquiring influential indicators to evaluate it. In addition, several variables need to be taken into account in order to obtain a wide and complete view of this field: from identifying different skills that individuals could need in their careers to sociodemographic variables [5].

Having technological support to explore employability and employment data (through information dashboards, for example) could ease the recognition of relevant or influential factors within the domain. What is more, the study of these fields in an academic context can help to reach insights about the linkage between university training and the career path of the graduates.

Universities (as they have a key role related to the employability of their students) can benefit from the introduction of information dashboards to conduct well-informed knowledge management [6, 7] and to support decision-making processes. Specifically, policymakers and institutions can improve and promote identified factors that affect the employability and employment of the students, placing these processes in the context of emergent areas like the Academic Analytics [8, 9] or Institutional Intelligence [10, 11].

However, developing information dashboards is not a trivial task. Several requirements can be involved and can vary among the different user profiles that would use the dashboard.

It is important to take into account all requirements and necessities in order to provide a good user experience to enable knowledge generation.

Software engineering paradigms like software product lines (SPL) [12, 13] provide solutions for managing sets of differing requirements, focusing on the reutilization and composition of base software assets (also known as core assets) to improve scalability and maintainability of particular products that share commonalities.

This paper describes the application of domain engineering to obtain a software product line of dashboards for analyzing university employment and employability data, in the context of the Spanish Observatory for University Employment and Employability (known in Spanish as OEEU, https://oeeu.org/). This organization has the vision to become an information reference for understanding the behavior of the variables related to employment and employability of students from Spanish universities through the recollection of data from the Universities' administrative records and the students themselves [14, 15]. The variety of users that consume information from the Observatory makes the SPL paradigm a potential solution for developing customized information dashboards to explore its data.

The rest of this work is organized as follows. Section 2 describes the methodology followed to develop the dashboard software product line for the Observatory. Section 3 presents the results of the application of the SPL paradigm within the dashboards domain. Section 4 discusses the results obtained, followed by the final section (Section 5), where the conclusions derived from this work are presented.

2 METHODOLOGY

2.1 Domain engineering

The main stage of the SPL paradigm is called domain engineering. In this stage, the domain of the product line to be developed is studied in order to identify the commonalities and variability points of the products that will conform the product family.

By identifying these characteristics, it is possible to model the SPL through a feature diagram [16] which allow the developers to generate the final products through the combination of the different features identified.

During the domain engineering stage, a series of core assets (software assets that will conform the foundation of the product line) are implemented. These core assets are generally configurable, which means that developers will be able to reuse and adapt them given specific requirements of particular users.

For the pilot product line of dashboards presented in this paper, three components have been modeled. These components are three kind of information visualizations that could have different data sources, functionalities or even layout: a scatter diagram, a heat map and a chord diagram, as shown in Figure 1. The generated dashboards need support from the base system of the Observatory to manage user permissions and data retrieval from the persistent storage.

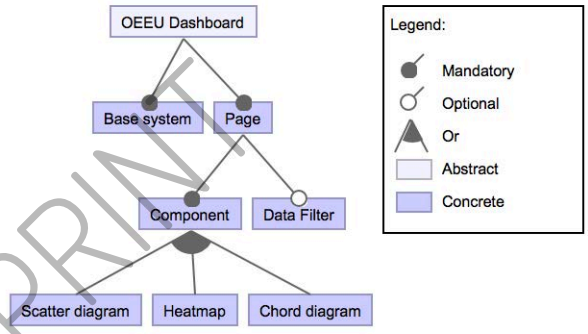

Figure 1. Top-level feature model of the SPL for the Observatory's dashboard product line. The individual features of every individual component have been omitted for simplicity

Once the core assets for these components have been implemented, developers are able to combine and configure them with different parameters to build dashboards pages that fit particular requirements or necessities.

2.2 Code generation

Once the domain engineering phase is done and a set of core assets is available, the application engineering stage starts. In this stage particular products of the line are generated through the combination and configuration of software components.

This process can be automated through code generators fueled by configuration files, obtaining the source code adapted to the specified requirements.

There are several strategies to materialize the variability points previously modeled in the domain engineering phase [17]. One of these strategies is to implement the core assets as a series of code templates [18, 19] that will be filled once the requirements for the product are defined.

In this case, a domain specific language (DSL) has been implemented with XML technology [20] to provide a structured file to the code generator so it can easily extract the features and inject them into the code templates (with Jinja2 [21] as the

template engine). This code generator is implemented in Python, and it is in charge of processing the XML configuration files and inject the functionalities through the code templates logic.

The outcomes of this process are the source files (HTML and JavaScript files) that will be deployed to make them accessible to the final users, conforming the personalized information dashboard. The general workflow followed to generate the source code can be seen in the Figure 2.

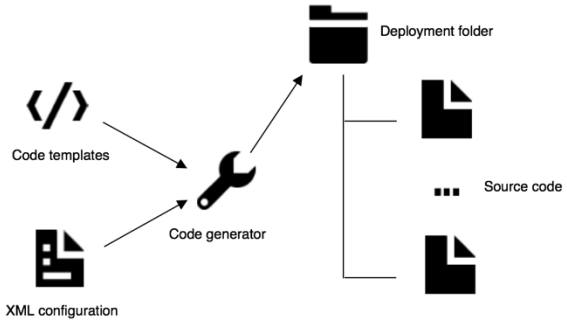

Figure 2. Code generation workflow.

2.3 Interoperability

The backbone of an information visualization is the data showed. One of the main concerns of this approach applied to the dashboards domain is the interoperability of the data sources that will fuel the information visualizations.

Having good interoperability levels means that any modification on the data sources would not imply several or critical changes in the visualizations' code.

To retrieve the data to be presented from the Observatory's bank of knowledge, a GraphQL API [22] has been implemented in order to decouple the presentation components from the persistent storage.

GraphQL provides a flexible query language to build requests that can be parameterized. What is more, GraphQL allows to specify the fields that will be retrieved, saving bandwidth by including only the necessary data in the responses [23, 24].

The API calls are executed by specific GraphQL connectors implemented to add an abstraction layer and to be able to modify (if necessary) the data requests without affecting the actual components or even change the data sources.

3 RESULTS

The following section presents the results derived from the application of the SPL paradigm to create a software product line for the Observatory in order to generate customized dashboards. The results are presented through three perspectives: the results obtained regarding the customization of functionalities, layout and data sources.

3.1 Results regarding functional personalization

One of the main goals of this approach is to provide an automated method to manage the generation of products with different functionalities.

Having the possibility to change the components' functional features gives freedom to the users to choose and obtain components that fit best their requirements.

For example, a potential user could need a scatter diagram to explore three variables at the same time through the X and Y axes and the radius of the points being represented, in order to study potential patterns or relations among them.

On the other hand, another particular user could only need to explore two variables at the same time, because the addition of a third variable could aggravate the analysis process.

By only changing the configuration of a scatter diagram component it is possible to achieve (through the specific core asset that supports this component) two different versions of the diagram to fit these particular requirements (an example is showed in the Figure 3).

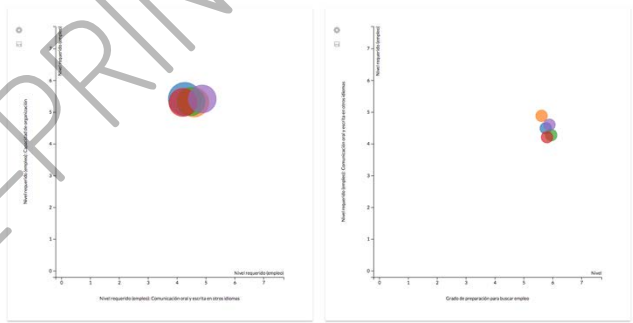

Figure 3. Comparison of two scatter diagram components. On the left, the radius of the elements within the diagram represents the value of a variable. On the right, the radius is not used to represent any variable and every point has a default radius.

There are other kind of features that can be configured, like a set of controls or filters available to explore data with more or less detail or freedom given the final user necessities.

All these variations are introduced through the templates previously implemented; functionalities are injected or ignored on the core assets to produce components with different features from the same template.

3.2 Results regarding layout personalization

This approach also allows the customization of the dashboards pages' layout. A generated dashboard page will be composed by a series of selected components previously configured to fit particular requirements. Once this task is done, these components need to be placed on the final dashboard page.

From an abstract point of view, a dashboard page is composed by containers (rows or columns) that will hold the different components or graphical resources.

Through the implemented DSL it is possible to specify the page layout once the components have been configured by referencing them within the containers that will hold them. There is an example of this syntax on the Figure 4.

```
<Page page_id="1">
    <Components>
        <Component>
            <ScatterDiagram component_id="ScatterDiagram_1"...>
        </Component>
        <Component>
            <Heatmap component_id="HeatMap_1"...>
        </Component>
        <Component>
            <ChordDiagram component_id="Chord_1"...>
        </Component>
    </Components>
    <Layout>
        <RowGroup>
            <Row width="100%" height="100%">
                <ColumnGroup>
                    <Column width="100%" height="100%">
                        <Component ref="Chord_1"/>
                    </Column>
                    <Column width="100%" height="100%">
                        <Component ref="ScatterDiagram_1"/>
                    </Column>
                </ColumnGroup>
            </Row>
            <Row width="100%" height="100%">
                <Component ref="HeatMap_1"/>
            </Row>
        </RowGroup>
    </Layout>
</Page>
```

Figure 4. Example of a configuration for a dashboard page in which the layout is specified in terms of rows and columns

The configuration specified in the Figure 4 yields the final dashboard page presented in the Figure 5.

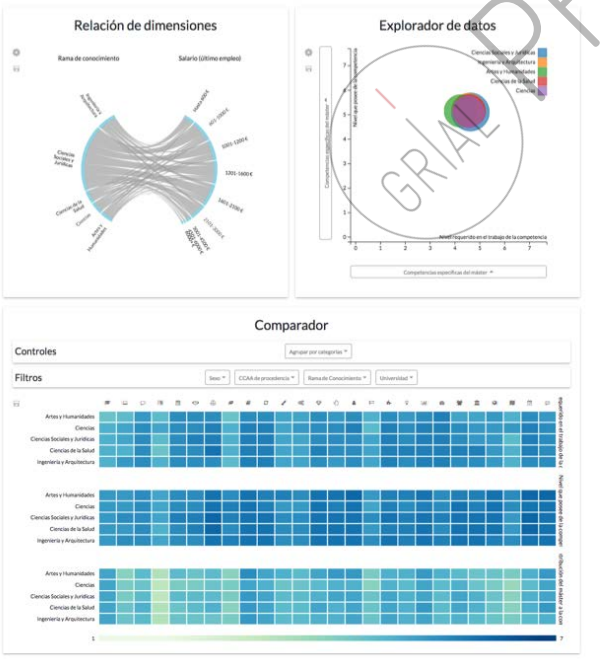

Figure 5. Example of a generated dashboard page given a specific configuration (contents in Spanish)

Thanks to the DSL it is possible to arrange the elements that will conform the dashboard page in terms of rows and columns, allowing developers to create and test different pages' layouts by only specifying them in the configuration files.

3.3 Results regarding data sources personalization

As it has been aforementioned, data sources are vital for dashboards to fulfil their role. There could be users that prioritize specific variables or specific information over another, and it is important to take these data requirements into account, as having too much information on an information dashboard could deteriorate the user experience.

This approach considers data sources as a part of a dashboard's configuration, making also the information presented itself or the information available to show (through a series of interactive controls) a customizable element.

For example, a particular user could require a heat map to have an overview of a series of skills required in the career path of the students (Figure 6).

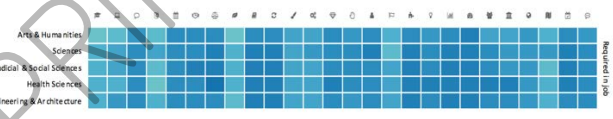

Figure 6. Example of a heat map with the goal of showing the most required skills in the career path of the students

However, another user could require a heat map to study which are the most popular methods to search a job.

Starting from the same component (a heat map visualization), developers only need to specify through the DSL which data sources will be necessary to consume information from (an example of this syntax showed in the Figure 7). In this case, it is important to specify the Observatory's GraphQL API endpoints and parameters necessary to retrieve the requested data. As it will be discussed, the variety and homogenization of data sources is a challenge for this approach.

```
<Heatmap component_id="HeatMap_1">
    <Title>Level of skills</Title>
    <Dimensions>
        <Dimension dimension_id="1">
            <DataResource>
                <DataSource>
                    <label>Required in job</label>
                    <code>C1EMP</code>
                    <metric_code>oneLevelGroupedAverage</metric_code>
                    <endpoint>stats2017</endpoint>
                </DataSource>
            </DataResource>
        </Dimension>
        <Dimension dimension_id="2">
            <DataResource>
                <DataSource>
```

Figure 7. Example of the data source specification in a heat map visualization through the implemented DSL

By solely changing the data resources it is possible to achieve two visualizations adapted to the requested information.

4 DISCUSSION

Applying the software product line approach to the automatic generation of dashboards has led to promising results regarding the management of customization within this domain.

Software interfaces require both the study of the domain in which they will be framed and the study of the target users that will end up using the products. Moreover, interfaces that give support to decision-makers present additional difficulties given the particular requirements and factors that could affect the experience and, consequently, the visual data analysis and decision-making processes.

The main focus must be on providing powerful tools that are not only aesthetic and functional but also helpful for the users. However, there could be several user profiles, and the necessities of one user could differ totally from the necessities of another. Developing a specific dashboard for every user profile constitutes a solution, but it is inviable if the number of user profiles is significant within the particular problem to be addressed. In addition, the requirements of every user could evolve along time and that is also another issue that affects maintainability and consumes resources.

The software product line paradigm helps to manage different and dynamic requirements by providing a theoretical framework for implementing modular, configurable and reusable software components (named core assets of the product line) that can be composed to create final and complete software products. The results derived from this work have proved that this approach can decrease the efforts made during the development processes and improve maintainability and the evolution of the products. The most time-consuming tasks are carried out during the domain engineering phase, when the base core assets have to be designed and implemented to be reusable and configurable. Once this phase is done, the creation of particular products is straightforward.

The main challenges of applying the SPL to the dashboards' domain involve different matters. First, usability has to be a priority. As it has been mentioned before, the finally generated dashboards need to be helpful, and that involves having good levels of usability in order to provide valuable user experiences. However, the automatic generation of user interfaces is still a tough process that require semi-automatic or even manual design processes [25, 26]. Further research will involve usability tests to study the perceived usability levels of the automatically generated user interfaces.

On the other hand, the particular domain of the dashboards makes the data sources a vital element for the product line. There could be different data sources, with different data formats or even different ways or protocols to retrieve the information. It is important to take into account the heterogeneity of the sources involved to decouple the logic of the software components from the information that they will finally hold, in order to avoid critical changes on the components if the data sources are modified [27] in some sort of manner.

Developing a framework to efficiently generate flexible and customizable information dashboards could give a strong foundation to create powerful tools with the main goal of helping decision-makers to take well-informed decisions in order to obtain benefits from them.

5 CONCLUSIONS

In summary, the software product line paradigm has been applied to the Spanish Observatory for University Employment and Employability's system in order to provide an automatic method for generating customized dashboards to analyse the organization's data regarding university employment and employability, given a set of particular requirements.

Having a method for managing differing necessities benefits both developers and target users, increasing maintainability and efficiency in development processes and allowing fine-grained customization in the final products, respectively.

In this particular case, the creation of visualization tools for exploring data about university employment and employability could support policy-makers and institutions to identify factors that affect the students' capacity to obtain a job in order to improve the linkage between higher education and employment.

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] S Christian Albright, Wayne Winston, and Christopher Zappe. 2010. *Data analysis and decision making*. Cengage Learning.
[2] Dj Patil and Hilary Mason. 2015. *Data Driven. "* O'Reilly Media, Inc.".
[3] Ramesh Sharda, Dursun Delen, and Efraim Turban. 2013. *Business intelligence: a managerial perspective on analytics*. Prentice Hall Press.
[4] Stephen Few. 2006. Information dashboard design.
[5] Ronald W Mcquaid and Colin Lindsay. 2005. The concept of employability. *Urban studies* 42, 2, 197-219.
[6] Á. Fidalgo-Blanco, M. L. Sein-Echaluce, and F. J. García-Peñalvo. 2014. Knowledge spirals in higher education teaching innovation. *International Journal of Knowledge Management* 10, 4, 16-37. DOI:http://dx.doi.org/10.4018/ijkm.2014100102.
[7] Á. Fidalgo-Blanco, M. L. Sein-Echaluce, and F. J. García-Peñalvo. 2015. 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, 3, 266-288. DOI:http://dx.doi.org/10.1108/PROG-06-2014-0033.
[8] J. P. Campbell, P. B. Deblois, and D. G. Oblinger. 2007. Academic Analytics. A new tool for a new era. *Educause Review* 42, 4, 40-42,44,46,48,50,52,54,56-57.
[9] Paul Baepler and Cynthia James Murdoch. 2010. Academic analytics and data mining in higher education. *International Journal for the Scholarship of Teaching and Learning* 4, 2, 17.
[10] Oficina De Cooperación Universitaria. 2013. Libro Blanco Inteligencia Institucional en Universidades OCU (Oficina de Cooperación Universitaria), Madrid, España.
[11] Francisco J García Peñalvo. 2015. Inteligencia Institucional para la Mejora de los Procesos de Enseñanza-Aprendizaje.
[12] Paul Clements and Linda Northrop. 2002. *Software product lines*. Addison-Wesley.
[13] Hassan Gomaa. 2004. *Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures*. Addison Wesley Longman Publishing Co., Inc.

[14] F. Michavila, Jorge M. Martínez, Martín Martín-González, F. J. García-Peñalvo, and Juan Cruz-Benito. 2016. *Barómetro de empleabilidad y empleo de los universitarios en España, 2015 (Primer informe de resultados).* Observatorio de Empleabilidad y Empleo Universitarios, Madrid.

[15] F. Michavila, J. M. Martínez, M. Martín-González, F. J. García-Peñalvo, J. Cruz-Benito, and A. Vázquez-Ingelmo. 2018. *Barómetro de empleabilidad y empleo universitarios. Edición Máster 2017.* Observatorio de Empleabilidad y Empleo Universitarios, Madrid, España.

[16] Kyo C Kang, Sholom G Cohen, James A Hess, William E Novak, and A Spencer Peterson. 1990. *Feature-oriented domain analysis (FODA) feasibility study.* Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.

[17] Critina Gacek and Michalis Anastasopoules. 2001. Implementing product line variabilities. In *ACM SIGSOFT Software Engineering Notes* ACM, 109-117.

[18] Soheila Bashardoust Tajali, Jean-Pierre Corriveau, and Wei Shi. 2013. A Template-Based Approach to Modeling Variability. In *Proceedings of the International Conference on Software Engineering Research and Practice (SERP)* The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 1.

[19] Timo Greifenberg, Klaus Müller, Alexander Roth, Bernhard Rumpe, Christoph Schulze, and Andreas Wortmann. 2016. Modeling variability in template-based code generators for product line engineering. *arXiv preprint arXiv:1606.02903.*

[20] Tim Bray, Jean Paoli, C Michael Sperberg-Mcqueen, Eve Maler, and François Yergeau. 1997. Extensible markup language (XML). *World Wide Web Journal* 2, 4, 27-66.

[21] Armin Ronacher. 2008. Jinja2 Documentation Welcome to Jinja2—Jinja2 Documentation (2.8-dev).

[22] Facebook. 2016. GraphQL.

[23] Andrea Vázquez-Ingelmo, Juan Cruz-Benito, and Francisco J García-Peñalvo. 2017. Improving the OEEU's data-driven technological ecosystem's interoperability with GraphQL. In *Proceedings of the Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality* (Cádiz, Spain, October 18-20, 2017 2017). ACM, 3145437, 1-8. DOI:http://dx.doi.org/10.1145/3144826.3145437.

[24] Dhaivat Pandya. 2016. GraphQL Concepts Visualized, Web log post.

[25] Andreas Pleuss, Stefan Wollny, and Goetz Botterweck. 2013. Model-driven development and evolution of customized user interfaces. In *Proceedings of the 5th ACM SIGCHI symposium on Engineering interactive computing systems* ACM, 13-22.

[26] Andreas Pleuss, Benedikt Hauptmann, Markus Keunecke, and Goetz Botterweck. 2012. A case study on variability in user interfaces. In *Proceedings of the 16th International Software Product Line Conference-Volume 1* ACM, 6-10.

[27] Shaohua Wang, Iman Keivanloo, and Ying Zou. 2014. How do developers react to restful api evolution? In *International Conference on Service-Oriented Computing* Springer, 245-259.