Interactive Visualisation of Products in Online Configurators

A Case Study for Variability Modelling Technologies

Marianela Ciolfi Felice[*], Joao Bosco Ferreira Filho[*],
Mathieu Acher[‡*], Arnaud Blouin[†*], Olivier Barais[‡*]
Inria / IRISA[*], University of Rennes 1[‡], INSA Rennes[†]
Rennes, France
firstname.lastname@inria.fr

ABSTRACT

Numerous companies develop interactive environments to assist users in customising sales products through the selection of configuration options. A visual representation of these products is an important factor in terms of user experience. However, an analysis of 100+ existing configurators highlights that not all provide visual representations of configured products. One of the current challenges is the trade-off developers face between either the memory consuming use of pregenerated images of all the combinations of options, or rendering products on the fly, which is non trivial to implement efficiently. We believe that a new approach to associate product configurations to visual representations is needed to compose and render them dynamically. In this paper we present a formal statement of the problem and a model-driven perspective for addressing it as well as our ongoing work and further challenges.

Categories and Subject Descriptors

D.2.13 [**Reusable Software**]: [Domain engineering]; D.2.2 [**Design Tools and Techniques**]: User Interfaces

Keywords

Variability modelling, software product line, configurator, user interface

1. INTRODUCTION

Online shopping is nowadays a consolidated field in which vendors usually allow their customers to customise the products according to their preferences. The lack of physical contact with the goods that they are about to buy, though, is a source of customer defiance about online shopping [12]. In compensation, other ways of feedback must be developed to achieve the expected user experience. The most intuitive strategy to overcome this drawback is to let customers visualise and interact with products. This becomes an important factor in industries where the visual aspect of the product – more than other properties – is crucial, like apparel or automotive. A typical scenario is customers selecting and deselecting a wide range of configuration options of a product. The visual representation of that product is rendered on the fly at each step of the configuration process (see Figure 1). Surprisingly, not all the configurators found in companies websites provide this feature. An empirical study [7] on 100+ sales configurators showed that it is the case for only half of them. Developing a solution for the *interactive visualisation of products in online configurators* is hindered by many non trivial quality criteria (*e.g.* performance, high visual quality of the rendering). In the meantime, configurator developers have to manage a large amount of options and potential configurations/visual representations (*i.e.* hundreds of configuration options with boolean, string, or integer values may be present). As the number of valid configurations increases exponentially with the number of options and the possible values associated to options, storing and retrieving images for each configuration is unrealistic in the general case. So, trade-offs between storing all the images or computing and rendering them on the fly must be defined. The overall challenge is to have a formal approach to associate configurations to visual representations in order to dynamically generate them while the user configures the product.

Despite research interests (*e.g.* configuration, product derivation) related to the problem, the variability and product line community has neither identified it as a potential case study[1] nor developed or evaluated comprehensive solutions. Configuration user interfaces (configurators for short) have been studied from different perspectives (usability, design standards, configuration process, *etc.*), but these works do not address the problem of depicting an associated visual representation [7, 9, 11–14]. Decision or feature-based configuration techniques and environments do not support this visualisation feature as well [1,5,8,10,11]. The visions exposed in [3,4] seek to develop complete configuration environments; a visual rendering mechanism is likely to be part of such configurators.

Besides a preliminary study of 60+ online configurators, we present a model-driven perspective to interactive product representation and discuss some of the associated challenges to be tackled.

Reminder of the paper. In Sect. 2 we define the problem. In Sect. 3 we report on our current progress and we present further research challenges. Sect. 4 concludes the paper.

[1]According to Yin [16], a case study is "an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident". We consider ours as an exploratory case study that will serve as a preliminary research of the presented phenomenon [6] (i.e., interactive visualisation of products in online configurators), in order to formulate new hypotheses and challenge variability and modelling technologies.

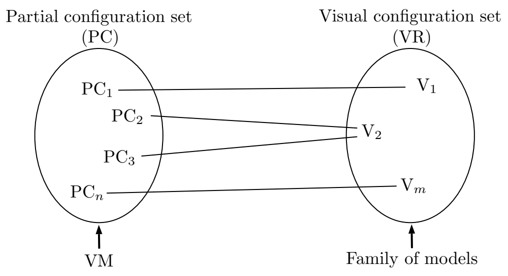


Figure 2: Outline of the problem

2. PROBLEM DESCRIPTION

The configuration of products in an online environment is a complex task. It often involves a multistep process during which the user might change her mind about the choices she made before. In addition, the product usually consists in a considerable amount of customisable parts. Though the set of options can be considered to be fixed, a large list of them might be available for each part. There is also a *valid configuration* notion – given by the constraints and the relationships among these options – that determines which products can be sold.
Figure 1 shows the Marks & Spencer's online shirts configurator. In the first step the user can select the fabric from a list of samples. Then, she can customise the design by picking the type of collar, sleeve, pocket, *etc.* At any moment she can see the price of the product, as well as different zoomed parts of it to appreciate the details. As in this case, when several options in a configurator have an impact on the visual representation of the product, the web configurator developers may face a trade-off. They can store all the possible combinations as images, either in a database or in a file system. Besides storage expenses, this strategy implies programming some logic to match a product configuration to a specific image. When too many combinations are possible this approach is no longer viable. Developers have to support the dynamic composing of the visual representations. Such a composition process can be performed either on the client side, on the server-side, or both. Given this scenario, the challenges are: *i)* the definition of the relation between the product configuration made by the user and its visual representation; *ii)* the correct rendering of the visual representations in a reasonable amount of time.

2.1 Problem Statement

An online configurator contains a set of available options that users can select, unselect, or customise during the configuration process. This group of selected and customised options constitute a *product configuration*. At the end of the process, the configuration is *complete* and it should correspond to a product that the user can buy. In the intermediate steps, though, the configuration can be seen as *partial* since some options are not configured yet. More formally, every available option o can be defined as:

$$o = (name, A)$$

where A is the list of the option's attributes. Each attribute is:

$$a = (name, D, value)$$

where D is the domain of validity for $value$.

For example, the option $Fabric$ could have as attributes the visual design (*e.g.* stripes, flowers, some solid color), the type of fabric (*e.g.* cotton, silk) and an implicit boolean attribute to indicate if it is selected or not. An option with multiple attributes could also be represented by modelling them as options, with a finer granularity. We consider that flexibility should be given to the modeller to take a decision on this matter.

A *partial configuration* (PC_i, as shown in top of Figure 2) does not necessarily include all available configurable options. It can be characterised as a set of *partially instantiated options*. A partially instantiated option has at least one of its attributes defined with a concrete value set; *i.e.* it is an option that might have non defined values for some attributes. Besides, each partial configuration has one visual representation, which is an element in the set of visual representations VR. These can be image files, 3D models, portions of HTML code, *etc.*

The problem has essentially 3 parts: *i)* The modelling of the partial configurations set PC, *ii)* the modelling of the visual representations set VR, and *iii)* the definition of a relation between them as shown in Equation 1.

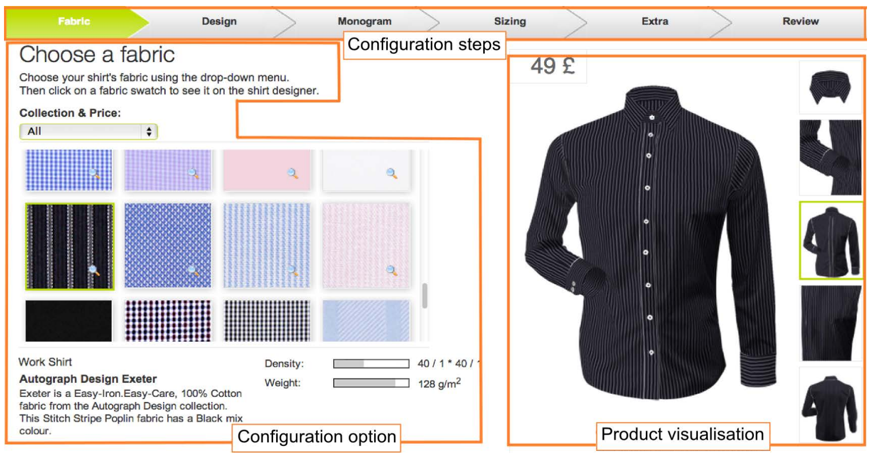
$$g : PC \rightarrow VR \qquad (1)$$

Finally, the visualisation of the products should meet the expected properties listed in Sect. 2.2.

2.2 Expected Properties

We have conducted a preliminary study of 60+ online configurators providing visual representations of products to get an overview of the current practices. We have used an excerpt of 100+ configurators described in [7] and several others randomly chosen. We have focused our study on the strategies used to render the products and the properties that the configurators present w.r.t. the visualisation of products. As a result, we identified six properties that would enhance configurators' usability and users' experience:

- *High visual quality:* when possible, the visual representation of the product should be realistic and aesthetic. If the product is composed of independent configurable parts, the model should be as seamless as possible to give the feeling of a whole. Depending on the product's nature, this could imply the use of illumination methods, scaling, textures, *etc.*
- *Automated and comprehensive synchronisation:* when users configure an option having an impact on the graphical representation of the product (a *visual option*) they expect this change to be reflected graphically and immediately. Forcing users to request a representation update, *e.g.* clicking on an update button, goes against this requirement. Therefore, the visual representation's update method has to be both comprehensive (it should include all relevant visual options) and automatic (no user action required to update representations) when moving from a PC_i to a PC_j and consequently from a V_i to a V_j (see Figure 2).
- *Coherence and stability:* valid configurations should be reflected in their corresponding coherent visual representations. Conflictual selected options should not lead to an incoherent visualisation. Also, at any time the visualisation must be compatible to at least one product that the costumer can buy when finishing the configuration process. This requirement implies that: 1) Every valid PC_i must have its associated element V_i in the set of visual representations (see Figure 2). This can be guaranteed by defining the relation g in the equation 1 (Sect. 2.1) as a total function. 2) Each V_i in the visual representations set must correspond to at least one element in the partial configurations set. This can be provided by making g a surjective function.
- *Interactiveness:* users should be able to interact with the visual representation in order to rotate it, to select and edit the



Choose your shirt's fabric using the drop-down menu.
Then click on a fabric swatch to see it on the shirt designer.

Work Shirt

Autograph Design Exeter

Exeter is a Easy-Iron.Easy-Care, 100% Cotton fabric from the Autograph Design collection. This Stitch Stripe Poplin fabric has a Black mix colour.

Figure 1: `http://www.marksandspencer-madetomeasure.com` configurator (fabric Exeter selected, May 23. 2013)

elements to configure. Moreover, the configurator should provide valuable graphical information in the visual representation itself, *i.e.* limiting the use of other means like textual descriptions or visual representations of options.

- *Performance:* it is crucial to achieve low loading and response time, to let the user compare different combinations quickly. This requirement notably depends on how efficiently is defined and implemented the function g.

3. ONGOING WORK AND CHALLENGES

In this section we describe a variability approach to tackle the presented problem and then we document our progress in this direction. Finally we list some further challenges that we detected that might bring new research questions and interesting subproblems.

3.1 Variability Modelling Perspective

A typical model-based variability approach would reflect the 3 parts of the problem mentioned in Sect. 2.1, and it would consist of *i)* a variability model (VM), *ii)* a family of models, and *iii)* a mapping between the two. The variability stemming from the set of configuration options is captured in a VM that could be a feature model (FM) or a decision model. Using an FM, each configuration option is represented by a feature. Additionally, some features must be added to provide structure to the tree. The (partial) configurations mentioned in Sect. 2.1 can be seen as a set of *(partial) feature configurations* over the FM. A feature configuration is said to be partial or incomplete when the set of selected and deselected features is a subset of the FM's features. VR of Figure 2 can be captured in a *family of models* (*e.g.* characterised by a metamodel).

The crucial challenge is to provide modellers with the means to express the *mapping* between the VM and the family of models, corresponding to function g in the equation 1 (Sect. 2.1). Specifying the set of (partial) configurations and models corresponding to visual representations *by extension* is tempting, but it is not a scalable approach because of the exponential nature of the problem. Already with 300 boolean configuration options, approximately 10^{90} configurations exist.

For this reason, we propose a compositional approach. Every visual representation (element V_i in VR) is constituted by a set of visual elements, denoted V_E. Following a model-driven approach, visual elements can be model elements or models. In this way, configurator developers would only define a mapping between a *set of partially instantiaded features* (features in which at least one of the attributes has a concrete $value$ set), denoted F, and the visual elements they affect. Ideally, the amount of mappings that developers must specify is much smaller than the amount of partial configurations that the function g requires (Equation 1, Sect. 2.1). Considering the reformulated concepts (see bottom of Figure 2), the mapping that the developer must define is a relation R:

$$R : (F, V_E) \rightarrow VR$$

An additional challenge is faced here, since the enough expressiveness must be provided to her to define concrete mappings. Developers could specify, for instance, that the activation (resp. deactivation) of a specific feature triggers the addition (resp. the removal) of a visual element. Then, given any partial feature configuration, the corresponding set of involved visual elements affected by it can be automatically determined on the fly, based on the features of this configuration. Finally, these visual elements must be combined to generate the concrete visual representation. Many transformational/compositional techniques and languages (*e.g.* action languages) can be considered to realise the derivation.

3.2 Current Status

We have used the CVL language[2] and developed a CVL derivation engine to realise our approach. CVL is a generic language for modelling variability in models expressed in any Domain-Modelling Specific Language (DSL) based on MetaObject Facility[3] (MOF). Roughly, it comprises four models: the base model described in a DSML; the variability abstraction model (VAM) defining variability in terms of variation points and attributes; a resolution model that describes values of variation points and attributes; and the variability realisation model (VRM) that defines how to transform the variation points to create a new model.. In our case, the initial visual representation (instance of the DSML used to represent the set VR) is the base model. The semantics of an FM can be easily captured in a VAM. A default feature configuration is modelled in the initial resolution model. A VRM is properly defined to link elements in the VAM with elements in the base model. The CVL derivation engine is executed to materialise the actual resolved model, according to a feature configuration cod-

[2]`http://www.omgwiki.org/variability/`

[3]`http://www.omg.org/spec/MOF/`

ified in the resolution model. Consecutive executions of the derivation algorithm can be performed whenever the feature configuration changes, resulting in new derived models.

We have modelled a simplification of the shirts configurator referred in Sect. 1 with different VRMs, using different styles for specifying variability (*i.e.* in a positive and negative way [15]). This will allow us to compare them in terms of convenience for the developer, as well as efficiency when executing a product derivation. We are considering to supplement this analysis by running an experiment with users (developers) to establish which mechanism is more intuitive and leads to simpler solutions. Another research direction effort is to develop concrete rendering aspects; we have been only working at the modelling level so far. Additionally, we are preparing a set of CVL examples to create a repository available to the variability and product line communities. These examples will provide a base to compare different approaches and can be used in this specific case study or others.

3.3 Further Research Challenges

The natural next step in the research path implies deciding how to address the presented problem. At a modelling level the objective is to correctly handle product visual representation models without considering configurators' implementation. Another option is to solve the problem at the technological level, closer to code details and web technologies. This implies bringing into scene non functional properties related to them, like client-side versus server-side processing and their consequences in quality attributes such as scalability and performance. Also, a fusion of these approaches can be applied. In addition, in a later stage some complex aspects could be added to the problem, *e.g.* the undo capability and its impact in both models and implementations. This is particularly challenging when combined with the technical approach, since some techniques are more suitable than others to perform this kind of tasks.

Further research in this context can be summarised as follows:

- *RQ1:* How to realise web product renderers with variability technologies? More precisely:
  - *RQ1.1:* Which variability mechanism (*e.g.* positive or negative variability [15]) is more intuitive for web developers and leads to simpler solutions?
  - *RQ1.2:* Which visualisation properties, listed in Sect. 2.2, can variability technologies guarantee?
  - *RQ1.3:* How to fill the gap between the modelling level of the problem and the implementation level? That is, how to compose from models coherent visual representations? Defining interactive and graphical fragments of web applications has been already performed (see, e.g., [2]) and could be considered for this purpose.
- *RQ2:* Can variability approach improve the current practices? This question can be divided into the following:
  - *RQ2.1:* How do current practices solve this problem?
  - *RQ2.2:* What can the variability community learn from current practices and in return, what can be the contributions of the variability community to it?

4. CONCLUSION

In this paper we challenge the variability and product line community to address a realistic and non trivial problem: how to effectively produce visual representations of products in online configurators when customers interactively (de-)select configuration options? Techniques, languages, and tools are needed to assist developers in associating configurations with visual representations in order to generate them dynamically. Based on a preliminary analysis on 60+ online configurators, we formulated the problem and identified expected properties when developing a solution for the interactive drawing of products.

To improve our understanding, we want to go further than our preliminary study. From an implementation perspective, we plan to conduct an empirical and qualitative evaluation to detect good and bad patterns, practical difficulties encountered by developers as well as underlying technologies of existing web configurators. From a usability perspective, we plan to run experiments with end users of web configurators. It will help us to better characterize the problem itself and the current practice. Meanwhile we are developing and investigating the use of model-based product line techniques and tools.

We believe this case study can be used to refute, validate, question, or compare the adequacy of variability modelling technologies. We hope practitioners will benefit from the ongoing research results around this case study.

Acknowledgements. We thank Gilles Perrouin for his helpful comments on an earlier draft of this paper. This work was developed in the VaryMDE project, a bilateral collaboration between the Triskell team at INRIA and the Thales Research & Technology.

5. REFERENCES

[1] E. K. Abbasi, A. Hubaux, and P. Heymans. A toolset for feature-based configuration workflows. In *Proc. of SPLC'11*, pages 65–69, 2011.

[2] O. Beaudoux, M. Clavreul, A. Blouin, M. Yang, O. Barais, and J.-M. Jezequel. Specifying and running rich graphical components with loa. In *Proc of EICS'12*, pages 169–178. ACM, 2012.

[3] Q. Boucher, E. Abbasi, A. Hubaux, G. Perrouin, M. Acher, and P. Heymans. Towards more reliable configurators: A re-engineering perspective. In *Proc. of PLEASE'12*, 2012.

[4] Q. Boucher, G. Perrouin, and P. Heymans. Deriving configuration interfaces from feature models: A vision paper. In *Proc. of VAMOS'12*, 2012.

[5] D. Dhungana, D. Seichter, G. Botterweck, R. Rabiser, P. Grünbacher, D. Benavides, and J. Galindo. Integrating heterogeneous variability modeling approaches with invar. In *Proc. of VAMOS'13*, page 8, 2013.

[6] S. Easterbrook, J. Singer, M. A. Storey, and D. Damian. Selecting empirical methods for software engineering research. *Guide to Advanced Empirical Software Engineering*, page 285–311, 2007.

[7] E. Khalil Abbasi, A. Hubaux, M. Acher, Q. Boucher, and P. Heymans. The Anatomy of a Sales Configurator: An Empirical Study of 111 Cases. In *Proc. of CAiSE'13*, 2013.

[8] A. Nöhrer and A. Egyed. Optimizing user guidance during decision-making. In *Proc. of SPLC'11*, pages 25–34, 2011.

[9] A. Pleuss, B. Hauptmann, D. Dhungana, and G. Botterweck. User interface engineering for software product lines: the dilemma between automation and usability. In *Proc. of EICS'12*, pages 25–34, 2012.

[10] A. Pleuss, R. Rabiser, and G. Botterweck. Visualization techniques for application in interactive product configuration. In *Proc. of MAPLE/SCALE'11*, 2011.

[11] R. Rabiser, P. Grünbacher, and M. Lehofer. A qualitative study on user guidance capabilities in product configuration tools. In *Proc. of ASE'12*, pages 110–119, 2012.

[12] T. Rogoll and F. Piller. Product configuration from the customer's perspective: A comparison of configuration systems in the apparel industry. *Proc. of PETO '04*, 2004.

[13] C. Streichsbier, P. Blazek, F. Faltin, and W. Fruhwirt. Are de facto standards a useful guide for designing human-computer interaction processes. the case of user interface design for web-based B2C product configurators. *Proc. of HICSS '09*, pages 1–9, 2009.

[14] A. Trentin, E. Perin, and C. Forza. Sales configurator capabilities to prevent product variety from backfiring. *Workshop on Configuration (ConfWS)*, 2012.

[15] M. Voelter and I. Groher. Product line implementation using aspect-oriented and model-driven software development. In *Proc. of SPLC'07*, SPLC '07, pages 233–242, Washington, DC, USA, 2007. IEEE Computer Society.

[16] R. K. Yin. *Case Study Research: Design and Methods*. Sager, 2002.