Towards More Flexible Configuration Systems: Enabling Product Managers to Implement Configuration Logic

Klaus Pilsl and **Martin Enzelsberger** and **Patrick Ecker**[1]

Abstract. Developing a configurator requires a deep understanding of the configurable product. The configuration logic must encompass the way product components may be combined and customized, as well as how the integrity of a configuration can be verified. When products evolve over time, the configurator must be adapted accordingly. Product Managers are intimately familiar with the features and capabilities of a product and drive its development. By enabling them to specify the configuration logic of a product, the time required to introduce new products or respond to changes in existing ones can be reduced significantly. In order to achieve this, an environment must be provided that facilitates the implementation of configuration logic in an efficient and intuitive manner. In this paper we try to identify the key aspects of such an environment and present our experiences in realizing a product configuration system based on our findings.

1 INTRODUCTION

Creating and maintaining a product configurator is usually a complex task [1]. The configurator database report 2014 [2] said, that 14% of the 900 configurators running 2013 disappeared 2014. Keeping a configurator running and up to date is often more time consuming than expected.

For reducing the creation and maintenance expenses for configurators, it is the target to give product managers the tools to create configurators themselves.

This article focuses on the very practical problems product managers face in building product configurators. Further on we identify key elements of an optimal configuration system environment. This leads to new approaches in the way the data is being entered, calculated and presented through the web. Notable findings will be presented in the last section.

2 DRAWBACKS OF COMMON DATA STRUCTURES USED IN CONFIGURATION SYSTEMS

As described in Ecker [3] and Šormaz [4] there are different ways of how the product data can be provided:

1. Relational form
2. Code / macros / scripts
3. Object oriented form
4. Mixture

Due to the penetration of relational database management systems [5] and the software solutions the configurators are built with (i.e. ERP-Systems) most configurator system use that relational data structure.

Each of the above listed data structure has its own problems and drawbacks. The following paragraphs will highlight those.

2.1 Relational form

Relational data structures need to image the product data in tables. These tables follow either a predefined [6] or a user generated schema. It is obvious, that product managers have more flexibility if they can define their own schema, which is necessary for certain kinds of products [7][8], but it also demands a higher skill level from the product managers, which most of them do not have.

But even in predefined schemata the problem remains, that the data of the product need to be squeezed in this predefined form. The challenge here is, to find a way to maximize functionality and readability which are adversary.

Our experience has shown that the majority of the product managers quickly lose sight on the complexity of the data structures they develop.

Problems:

- The product managers are forced to establish an extra documentation layer to keep the system manageable.
- The complexity especially for rule driven visualizations outruns many product managers' capabilities.
- The chronology the system has to calculate the rule needs to be defined by the product manager. This is discovered as a major weak point in terms of error and debugging expenses.

2.2 Code / macros / scripts

Code allows the product managers to transcript even most complex rules. The product manager does not have to follow a predefined schema at all.

Problems:

- The period of vocational adjustment is very high.

[1] IndiValue GmbH, Sarleinsbach, Austria, email: klaus.pilsl@combeenation.com, martin.enzelsberger@combeenation.com, patrick.ecker@combeenation.com

- This informal freedom, however, quickly leads to a lack of lucidity, which makes this form of product data unpopular for product managers who are not software developers.
- It demands a high skill level in software development from the product manager
- Developing a configurator by coding is time consuming and expensive
- Future adapting and enhancements are also expensive and time consuming

2.3 Object oriented form

Just like with the relational form there is a schema, but here they are called classes. The product manager can define an individual classes for every product. The main difference to the relational form is that the rules create dependencies, not the entities of the database.

That gives the product manager the ability to freely transcript the product rules into the system without the need of creating a database structure or a schema fitting his requirements first.

Problems:

- The period of vocational adjustment is moderate.
- This form demands a very sophisticated user interface to guide the user properly.

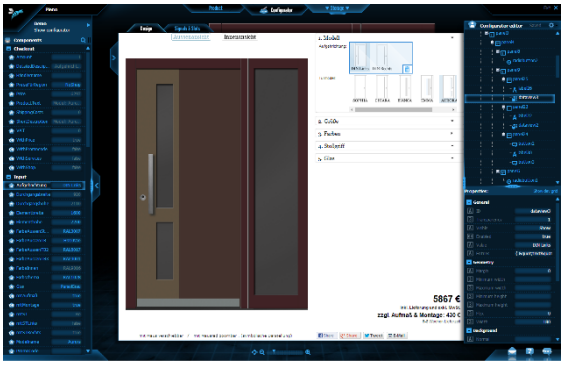
The advantages of this approach are:

- Because of the fact, that the class is individual for each product, the system can create it, while the product manager models the product.
- Classes can also be derived (inherited). So the product manager can easily create variations of the product data.

3 KEY ASPECTS OF THE OPTIMAL ENVIRONMENT

There are several key aspects which define an optimal product configuration environment which support the product manager in building the product configurator.

As a result of our experience and findings we created a configurator management system which fulfills the requirements described in this paper. This system is call Combeenation[9].



The Configurator Management System *Combeenation* from IndiValue GmbH. Showing the user interface designer of a front door configurator.

3.1 Instant feedback

The process of creating product data is usually defined by the following steps:

- Modeling the structure and rules
- Saving and compiling
- Testing

This iteration is continuously repeated, until the product data is finished for publishing. The described process is called progressive evaluation [10].

Progressive evaluation can be very time consuming. Especially testing (opening a test configuration, navigating through the configurator to the point of interest, collecting the test data, closing the test configuration) is very expensive.

If a system would save the data instantly give instant feedback even into every displayed data affected by the changes including an always open test configurator, the time needed for modelling product data would greatly be reduced.

Leitner et al. [11] state that testing and adapting the layout of the configurator interface plays a major role in developing a suitable user interface for configurators. Rapid prototyping processes can be implemented with Combeenation, an innovative configuration environment that supports application development on a graphical level and enables immediate user testing.

3.2 Simple user interface structure

Concerning the user interfaces of the configurator management system many user interfaces are IT oriented. That means the interface is built on the necessities of the system, the data structure and the underlying technology that runs the configurator system.

As described in Ko et. al. [12] most of the product managers are experts in the field of their product, but aren't very practiced in the use of integrated development environments (IDEs) or other user interfaces which are mainly designed for software developers.

The key aspect in this area is to create a user interface for a configurator management system that focuses on the product itself instead of the technical system. The product in its actual visual appearance should always be visible to give instant feedback of the changes made. The components, the properties, the rules and the controls the product manager has to work with needs to be presented in a continuous and integrated way, regardless how these items are used for. This reduces the times needed to jump between screens, menus or pages.

The user interface of a configurator plays a key role for both, consumers and product managers. Leitner et al. [11] identified five key principles for developing user interfaces for configurators which are suitable for both types of users.

- Customize the customization process: Adaption of the user interface depending on the type of customer.
- Provide starting points: Initial design from which the customer can continue the configuration process.
- Support Incremental Refinement: Tradeoff analysis (i.e. product comparison functionalities).

- Exploit prototypes to avoid surprises: Development and Teach the customer: Increasing the user's knowledge about product properties.

Besides these key principles, the arrangement of user interface elements of the configurator as well as the kind of process navigation (i.e. handling, ease of use, guidance through the configuration process) affects the customer's satisfaction with the configurator [13].

3.3 Separation of data and rules

In mass production industries it is common to integrate values (sizes, angles, weights, etc.) into the structure data (i.e. CAD systems). That is okay as long as the products do not change much after they are released to market.

In mass customization [15], however, the product and with it its values change continuously using product configurators. It is also often needed to start configurations of a specific product via different starting values (several presets for the same product).

These requirements can be met by separating the configurable values from the rules and structure data. This way different sets of values can be easily combined with different versions of rules and structure data.

How difficult/easy it is to apply a small change in an established structure with/without this separation will be defined by the viscosity of the system [15], which describes the flexibility of the system.

To supply such a flexible system it is important to make this separation. That also has to be taken care of in the user interface, so that the product manager knows, which parts (values vs. rules) of the product are stored where.

3.4 Separation of product design and UI design environments

We experienced, that product managers working with configurator management tools, which do not separate the user interface designer from the product designer, struggle with data duplication.

Some applications need to be displayed on different devices (desktop, tablets, smartphones), or on different channels (websites, apps, social media channels …), or some of them need to be refurbished to meet new requirements.

In order to create several different user interfaces for the same product, they have to also duplicate the product data. That leads to more data, which greatly increases the workload of the product manager, if this product data needs to be modified.

By separating the product design from the user interface design environment the product manager can address these requirements.

4 IMPLEMENTATION OF A PRODUCT CONFIGURATOR SYSTEM

To meet the needs described we developed a system for creating product configurators. In order to make the use of the system for the product managers as easy as possible, we have chosen to use these standards and technologies:

- HTML5: For maximum acceptance and further developments we have chosen to use HTML5 for presentation and communication. So the system is supported by every modern Browser without the need to install plugins or downloaded software the product manager would have to install first.
- Quick start: To allow new product managers an easy and quick start into the system, we decided to implement the solution as Software-as-a-Service (SaaS) and hosted it in the Cloud (in this case Microsoft Azure [3])
- Another aspect of SaaS is, that the product manager does not need to invest administration and maintenance work to run the system.

4.1 Graphical product representation

Visuals are crucial and the biggest part of the mission. Most users valuate product primarily on its visual appearance. This applies for consumers just as for product designers. We put a lot of effort into providing a flexible system for the visual representation of the product.

Changes in the visual will be applied in real-time. Instead of rasterized graphic formats we use scalable vector graphics (SVG) for a better image quality. The visual editor is interactive and you get what you see (WYSIWYG). After the configuration process is ended by a user each scalable vector graphic can be converted to a PNG or PDF file, making it easy for further processing or printing.

The usage of SVG also allows us the enable custom fonts, gradients, patterns, mask, filters and many other effects. For more information about SVG see [16].

4.2 Quick response times

Our system is built with low response times in mind, since researches have shown that higher latency times have a negative influence on the user acceptance, no matter if that is for consumers or product managers [17].

We experienced that latency times need to stay below 250ms otherwise most users interact with the same UI element again (e.g. clicking a button).

This speed must be achieved with any representation of the changes: simple values, results of complex chained calculations, visuals, and so on.

4.3 Calculation on server and client

To achieve quick response times, it is necessary to split the calculations on the server and client side. The product manager, however, must not be confused. It always must be clear on where the calculation will be done, because there are advantages and disadvantages with either method:

The advantages of server side computations [18]:

- Big data necessary for a calculation does not have to be transmitted to the client
- Product rules are knowhow of the company which needs to be protected. Some product managers don't want to transmit this knowledge to the browser of the client. Our system won't transmit the company's knowledge to the

client at any time. Just the results are transmitted and presented to the client.

- Virtual machines, which are hosted in a datacenter, offers far more computation power than a client device.
- Progress is always saved. If a user catches up later he may continue where he left the configuration (also on other devices).
- It would be possible that more users collaborate on the same product.

Disadvantage of server side computing:

- The system needs to be designed to scale up on demand. This can be done through distributed computing with automatically adding virtual servers if needed. This is especially difficult to implement if the machines work with stateful sessions.

Advantages of client side computing:

- Quick responses are possible
- No network communication needed

Disadvantage of client side computing:

- Only simple calculation should be done, because transmitting big raw data into the client browser can be ineffective

To take a good mix of both advantages it is possible let the system operates on server side calculation for the product design rules and on client side calculation for the UI design rules.

This way calculations of the product itself are done on the server and calculations concerning the user interface (i.e. jumping to a certain page based in input data) are done on the client.

The separation of product design and UI design environments (see 3.4) allows the system to intuitively distinguish between both methods.

4.4 One solution

Most systems on the market cover one part of the mission. The other parts are done by other software tools which needs to be connected via interfaces.

A common thing for instance is, to create a product configurator by using an ERP-software [19] and linking a CAD-software [20] with it to create the visuals.

That leads to this constellations: Both systems hold their own product data. Many parts of these 2 data sets need to be redundant on both systems. And there is a third set of data: the interface itself hold data, too.

Product managers struggle with the big amount of human resources needed to keep these data sets up to date.

It is less maintenance works if all the modules needed for product configuration are handled by one system [18] with one data set.

All the modules of this single system run on this one set of data, and any module of that system can directly and without conversion access the data needed to fulfil its function. Combeenation provides such a system.

5 CONCLUSION

We have highlighted the problems and drawbacks of current systems, which product managers use if they want to build a product configurator. Further on we emphasized key aspects which are needed for such a solution, including a responsive and easy to understand user interface and a strict separation of data and rules.

With these findings in mind we built the configurator management system Combeenation [9], which addresses all those problems and provide an all-in-one solution. This solution is trimmed to rapidity, easy to use, flexible and is highly scalable. First client projects are currently implemented with Combeenation and all usability and performance issues in these projects are monitored to provide further data for potential improvements.

REFERENCES

[1] T.A. Rogoll and F.T. Piller, *Konfigurationssysteme für Mass Customization und Variantenproduktion*, ThinkConsult. 2003.

[2] Blazek, P., Partl, M., Streichsbier, C. (2014): *Configurator Database Report 2014*, Vienna

[3] P. Ecker, *Sicherheitsaspekte bei der Entwicklung einer Software-as-a-Service-Lösung mit Windows Azure*, University of Applied Sciences Upper Austria, Hagenberg, 2012.

[4] D.n. Šormaz, *Distributed Agent-Based Integrative Model For Mass Customization Product Development*, Ohio University, Department of Industrial and Systems Engineering, 2010.

[5] C.M. Ricardo, *Databases Illuminated*, Jones & Bartlett Publ., 2011.

[6] *ISS+* from *MoveIT GmbH*, *Product Builder* in *Microsoft Dynamics NAV*

[7] Front Door Configurator from *TOPIC GmbH*, Austria

[8] Window Configurator from *IFN Internorm AG*, Austria

[9] www.combeenation.com

[10] T. Green and A. Blackwell, *Cognitive Dimensions of Information Artefacts: a tutorial*, 1998.

[11] G. Leitner, A. Felfernig, P. Blazek, F. Reinfrank, G. Ninaus, *User Interfaces for Configuration Environments*. In: A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen (eds.), Knowledge-based Configuration: From research to business cases, 2014.

[12] J.R. Ko, R. Abraham, L. Beckwith, A. Blackwell, M. Burnett, S. Wiedenbeck, *The state of the art in end-user software engineering*, ACM Computing Surveys, 1-44, 2011.

[13] P. Blazek and K. Pilsl, *The Impact of the Arrangement of User Interface Elements on Customer Satisfaction in the Configuration Process*, In: T. D. Brunoe et al. (eds.), Proceedings of the 7th World Conference on Masss Customization, Personalization, and Co-Creation (MCPC 2014), Aalbor, Denmark, 2014.

[14] F.T. Piller, *Mass Customization - Ein wettbewerbsstrategisches Konzept im Informationszeitalter*, Deutscher Universitäts-Verlag, 2006.

[15] T. Green and M. Petre, *Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework* Journal of Visual Languages & Computing, 131-174, 1996.

[16] J. Ferraiolo, *Scalable Vector Graphics (SVG) 1.0 Specification*, http://www.w3.org/TR/2001/REC-SVG-20010904/REC-SVG-20010904.pdf, September 2001

[17] F.H. Nah, *A study on tolerable waiting time: how long are Web users willing to wait?* College of Business Administration, University of Nebraska-Lincoln, 2004.

[18] M. Enzelsberger, *Entwicklung von Hive, einer domänenspezifischen Sprache zur Spezifikation von Produktlogik durch Nicht-Programmierer*, University of Applied Sciences Upper Austria, Hagenberg, 2012.

[19] *Internet Pricing and Configurator* from SAP AG (*SAP IPC*)

[20] *Inventor* from Autodesk, *SolidWorks* from Dassault Systems, etc.