Product Configuration in the Wild: Strategies for Conflicting Decisions in Web Configurators

Thomas Thüm[1] and **Sebastian Krieter**[2] and **Ina Schaefer**[1]

Abstract. Customization is omnipresent in our everyday live. There are web configurators to customize cars, trucks, bikes, computers, clothes, furniture, and food. At first glance, customization using configurators appears trivial; we simply select the configuration options that we want. However, in practice, options are usually dependent on each other. Reasons for dependencies are manifold and are typically specific for the particular domain. Dependencies can be simple, such as one option requiring or excluding another option, but also arbitrarily complex, involving numerous options. In this study, we aim to understand how today's web configurators support users in their decision making process. In particular, we are interested in understanding how configurators handle decisions that are in conflict with dependencies. To abstract from different visualizations, we classify the existing strategies of web configurators and discuss advantages and disadvantages of them. While we identified eight strategies, a single configurator typically uses several of those strategies.

1 Introduction

Mass customization is the vision that customized products are produced to a price similar to that with mass production [3, 4, 13, 15]. This vision requires that customers specify their needs explicitly [10]. To this end, there are thousands of product configurators available in the world wide web,[3] which guide customers during the decision making process. Unfortunately, customers do not only need to understand which options are available, but also which combinations of options are valid. For the success of mass customization it is crucial that potential customers are able to easily explore valid combinations and to make compliant decisions. Our personal experience with product configurators suggests that we are not there yet, as configurators often require a considerable mental effort from their users.

In Figure 1, we show an excerpt of a configurator for a Lenovo ThinkPad. The excerpt contains three configuration options, of which each stands for a different display being available for the notebook. However, these three options cannot be chosen freely due to existing dependencies. First, the options are alternatives to each other, meaning that a notebook must have exactly one of those three displays. For that purpose, the three options are arranged in a category called *Display*. Second, the green hint below the last option reveals that this display is only available if option *WWAN* (not shown in the excerpt) is chosen. While the first dependency is enforced by the configurator, the second must be taken care of by the user. We argue that dependencies are one of the main challenges for customers, as they heavily influence the decision making process.

[1] TU Braunschweig, Germany
[2] University of Magdeburg, Germany
[3] `https://www.configurator-database.com/`

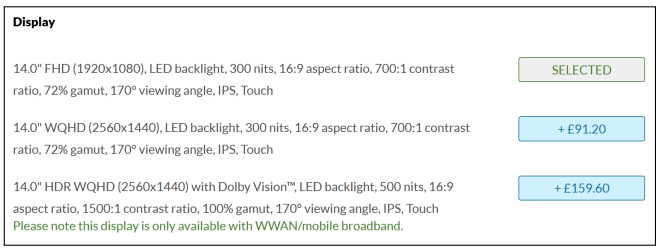

Figure 1. Example dependency in Lenovo's ThinkPad configurator.

While there are several approaches to implement configurators [5, 15], we aim to understand how configurators handle dependencies from a user's point of view. That is, we want to identify strategies to guide the decision process, which are currently applied in real-world configurators. Our insights may be used to uncover gaps between research and practice on product configuration. In particular, we analyze advantages and disadvantages of each strategy, which may support development of more sophisticated strategies in the future.

By studying a corpus of seven web configurators (cf. Section 2), we identified eight strategies to handle dependencies and discuss their advantages and disadvantages (cf. Section 3). We focus on web configurators, because they are freely available and are used by thousands or even millions of customers. Interestingly, of each configurator uses a different subset of all identified strategies. Moreover, configurators with fewer categories seem to be simpler in a sense that they apply fewer strategies. In particular, we even studied a configurator applying all eight strategies.

2 Subject Configurators

As considering all product configurators on the web is certainly not feasible, finding a representative selection of configurators is crucial for our study. Initially, we googled the term "configurator" and inspected the first 100 search results to understand which configurators are used frequently on the web. Almost all entries that we found were configurators for cars, but we wanted to cover more than just the automotive industry. We though of further industries where we experienced mass customization before. Then, we tried to find representative configurators for those industries that are likely to be used by a wide audience. Ultimately, we have chosen configurators from the domain of automotive, mobile computers, smartphone accessories, and clothing.

In our study, we do not distinguish between configurators that are used for customized production opposed to a selection of pre-produced goods (aka. selectors), as this distinction is typically not visible to customers anyway. Even though the number of potential

combinations and length of the delivery period might be good indicators. For instance, in order to speed up delivery, t-shirts are typically available in different sizes and colors, which are often produced before customers make their decision.

As we aim to provide evidence for our study by means of screenshots, we decided to use the UK version of all configurators. Nevertheless, we have drawn samples in other languages, too. The only differences we identified were options being available only in some countries, whereas the principle strategies have been identical. We accessed all configurators in May 2018, whereupon we experienced that configurators and their behaviors change on a daily basis. That is, dependencies are updated frequently, whereas we did only experience changes in the strategy for one configurator (cf. Section 3.2).

BMW 1 (3 Door) BMW is a German car manufacturer providing individual configurators for 49 different car models.[4] As cars are known to have hundreds of configuration options, we decided to use the configurator for the cheapest car model to find a comparatively small, and thus manageable number of configuration options for our study. There are 42 alternative options for engines and gearboxes, eleven exterior colors, five alloy wheels, three options for upholstery, four interior designs, two packages, and 81 further options with respect to optional equipment.

Toyota AYGO x-play 5 Door Hatchback With about 9 million sold cars a year, Toyota is one of the largest car manufacturers. Toyota offers 19 models, which are available in a total of 84 editions of which each can be configured separately. Just as for BMW, we wanted to select the cheapest model and edition available, namely Toyota AYGO x3 Door Hatchback. However, users cannot even choose anything in the first configuration step for this edition. Thus, we have chosen the cheapest edition for this model for which users can choose between manual and automatic gearshift, namely Toyota AYGO x-play 5 Door Hatchback.[5]

HP Velotechnik Streetmachine GTE Besides cars, many other transportation means exist that can be configured. In particular, bicycles can often be configured by customers. We investigated the configurator of HP Velotechnik, as we already had experience with this configurator prior to this study. Customers can choose one of 14 recumbent bicycles and then start the configurator. We have chosen the model Streetmachine GTE which provides 34 categories of which customers have to chose one option each.[6]

Lenovo ThinkPad X1 Yoga (3rd Gen) ThinkPad X1 is one of the most expensive and powerful convertibles of Lenovo. Before users can configure it on Lenovo's website,[7] they have to choose between three models in black and one model in silver. Of those, we have chosen the cheapest as it seems to provide the most configuration options (i.e., many options could not be downgraded in the more expensive models). The configurator provides 39 configuration options in eleven categories, such as processor, operating system, display, hard drive, and keyboard. Besides those, there are another eleven categories with a single option (i.e., cannot be selected by the user).

Microsoft Surface Book 2 Surface Book 2 is the latest high-end convertible by Microsoft. With 13 configuration options in five categories there is only some rudimentary support for customization. On Microsoft's website there is a link to pre-configured products,[8] but also to different versions of a configurator to which we later refer to as a version with a default configuration[9] and a version without a default configuration.[10]

T-Shirts at Amazon Online shopping is an increasing market and Amazon is one of the most popular web stores. Although not all products can be configured, there is huge amount of products for which at least certain properties, such as color and size, can be specified. Even though there are many products, the configurator technology seems to be rather generic at Amazon. For our study, we have chosen a t-shirt configurator, as t-shirts are a rather common consumer good and typically have two categories, which is necessary to study dependencies. The subject configurator offers 21 colors and 8 sizes, whereas 123 of 168 combinations are valid (i.e., 73.2%).[11]

Smartphone Cases at Ebay Another popular web store is Ebay, which appears to be similar in terms of configuration compared to Amazon. Again, the underlying configurator seems to have the same behavior for very different kinds of products. Nevertheless, we have to decide on one product for our case study, for which we have chosen a configurator for smartphone cases.[12] The configurator supports the selection out of 36 smartphone models and eight colors. Overall, 208 out of 288 principal combinations are valid (i.e., 72.2%).

3 Strategies for Conflicting Decisions

We investigate all seven configurators to explore how they deal with conflicting customer decisions. For that purpose, we randomly selected options and observed whether those decisions had any consequences for the selection of other options. We identified the following eight strategies, whereas we abstract from the visual representation of selected and deselected options.

3.1 Automatic Deselection in Alternatives

A simple but effective strategy exists for alternative options within one category. If an option A is already selected and a user attempts to select an alternative option B in the same category, a configurator may automatically deselect option A. Users are typically not notified about this change.

All configurators that we studied apply this strategy. As an example, we refer again to Figure 1 and the display options of a ThinkPad. If a user selects the second or third display option, the first display option is deselected (i.e., the price difference is shown).

The automatic deselection avoids further burden on the user for these kinds of simple conflicts, as the conflict is immediately solved. It is also an intuitive strategy, especially if accompanied with a graphical representation that suggests that these options are alternative to

[4] https://www.bmw.co.uk/configurator#/
[5] https://www.toyota.co.uk/new-cars/aygo/build
[6] http://hpvelotechnik.velocom.de/step-1.jsf
[7] https://www3.lenovo.com/gb/en/laptops/thinkpad/x-series/ThinkPad-X1-Yoga-3rd-Gen/p/22TP2TXX13Y
[8] https://www.microsoft.com/en-gb/surface/devices/surface-book-2/
[9] https://www.microsoft.com/en-gb/store/config/surface-book-2/8MCPZJJCC98C/17NG
[10] https://www.microsoft.com/en-gb/store/config/surface-book-2/8MCPZJJCC98C
[11] https://www.amazon.co.uk/gp/product/B00V3IDB3Q
[12] https://www.ebay.co.uk/itm/Sports-Running-Gym-Cycling-Jogging-Armband-Case-Cover-For-Huawei-Mobile-Phones/181809023183

each other (e.g., a drop down box as in Figure 3 or radio buttons as in Figure 5). Nevertheless, the deselection is problematically if the user does not recognize it (e.g., if the deselected option is not on the screen). Furthermore, the deselection itself may result in further conflicts with other decisions.

3.2 Starting with a Default Configuration

Dependencies are typically unsatisfied when no options are selected at all. For example, a notebook needs exactly one kind of display. A very common strategy to avoid invalid states when starting the configuration process is to start with a default configuration. The default configuration satisfies all dependencies and may be changed by the user during the process.

For instance, the first display is already selected when opening Lenovo's configurator (cf. Figure 1) and in BMW's configurator a default engine is selected from the start. Interestingly, Microsoft's configurator can be accessed via different links of which some start with a default configuration and others do not. However, even when starting with a valid default configuration in Microsoft's configurator, any change resets decisions on later categories. As a consequence, changing the display to 15 inch removes all default selections and makes the configuration invalid (cf. Figure 2). It requires decisions on all subsequent categories to finish and to come to a valid configuration again. Note that all decisions of subsequent categories are reseted independent of whether they are actually conflicting or not.[13]

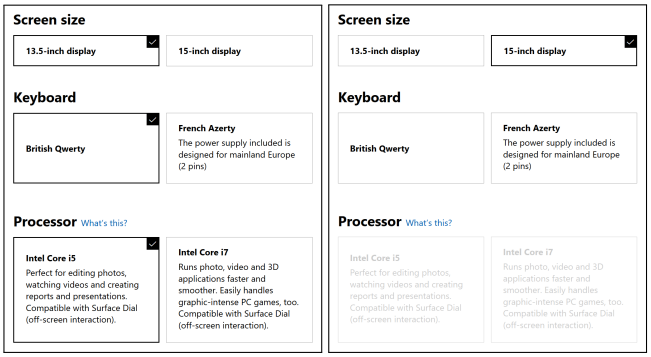

Figure 2. Default configuration and its reset in Microsoft's configurator.

Starting with a valid configuration means that a user can also skip the configuration process completely. It may also help users to avoid making some decisions, if they do not care about all options. They simply have to change options they care about. However, the downside of starting with a default configuration is that users can often not recognize which options they have chosen and which have already been chosen by the default configuration. As a consequence, they may forget to examine some of the decisions and may end up with a product not customized to their needs.

3.3 Hiding Invalid Combinations

Another simple strategy to deal with conflicts is to hide all options that are in conflict with previous decisions. At the start, all options of all categories are visible and no option is selected by default. Once a user makes a selection in one of the categories, all other categories are filtered for compatible options. As a result, no decision of the user leads to a conflicting state.

This strategy is followed by the configurator available to products at the Ebay web store. All models and all colors are shown when nothing is selected. For a given color, between 22 and 33 models are visible, and for a given model, there are between one and all eight colors. In Figure 3, we selected the *Huawei Ascend P7*, for which a case is only available in black, blue, and gray.

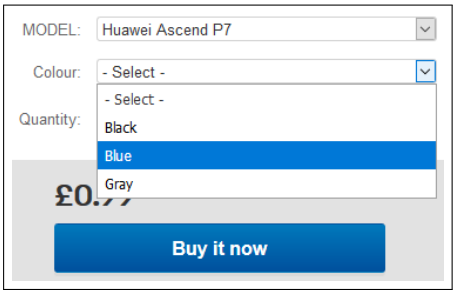

Figure 3. Hiding of invalid combinations in the Ebay configurator.

The advantage of hiding invalid combinations is that users are not overwhelmed by a large number of options that are actually not available for their prior decisions. Also, there is no need to solve any conflicts and users only need to decide on the remaining valid options. Nevertheless, there are also major drawbacks. This strategy seems to be rather incompatible with a default configuration and, thus, it will not be possible to simply continue and skip decisions. Requiring users to make every decision is fine if there are only a few categories, but can be infeasible for large configuration spaces. Nevertheless, we found such a combination in the configurator by Toyota: a default car color is selected when entering the configurator for which black wheels are filtered out. Hence, users will only find out about black wheels if they change the car color first. Furthermore, this strategy might be unintuitive to users who do not expect such a filtering. Indeed, it took us a while to find out that the lists in the Ebay configurator are changed depending on other selections. Finally, it seems that users should start with selecting the most relevant criteria for them, because otherwise they will simply not notice that there are other, potentially better options for later selected categories.

3.4 Alternatives of Compound Options

Most product configurators have categories with alternative options (i.e., options of which exactly one or at most one option can be chosen). If there are dependencies between these categories, one strategy is to build pairs of the corresponding options that are wanted. Those alternative pairs are then grouped under one compound category. Pairs not allowed by dependencies are simply omitted.

For instance, in the ThinkPad configurator, dependencies between the categories *Microsoft Office* and *Adobe Acrobat* are handled this way. Office is available as *365 Home*, *365 Personal*, *Home and Student*, *Home and Business*, and *Professional*. However, what is available does also depend on whether *Acrobat* is selected: *Professional* requires *Acrobat* and *Acrobat* requires *Home and Business* or *Professional*. Instead of presenting six options for *Office* and two options for *Acrobat* with two dependencies, there are seven compound options without dependencies, as illustrated in Figure 4.

[13] When we tested the configurator in April 2018, the configuration in subsequent categories was even re-assigned automatically. This behavior was even worse from a user's point of view as decisions on all subsequent categories are not just requested again, but even overridden without notice. It is likely that this behavior was replaced as being too unintuitive or even questionable from a legal perspective.

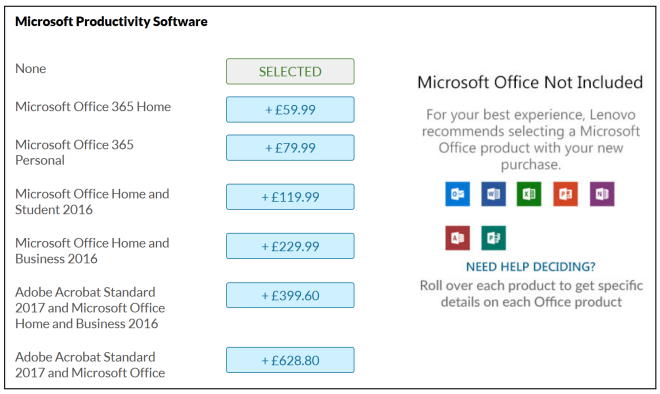

Figure 4. Compound options in Lenovo's ThinkPad configurator.

BMW uses compound options for dependencies between 23 engines and three gearbox options. Out of 69 combinatorial combinations, only 42 are considered valid. In particular, four engines are not available with manual gearbox and engines are either available with *Automatic Gearbox* or with *Sport Automatic Transmission* (i.e., $4 + (23 - 4) * 2 = 42$). A possible reason for compound options besides eliminating constraints is that the choice of engine and gearbox affects the acceleration, consumption, CO_2 emission, and pricing, which are properties shown next to each option (cf. Figure 5).

Figure 5. Compound options with non-functional properties for a BMW.

Compound options can essentially be used to get rid of all dependencies, which results in an enumeration of all products. Microsoft gives customers both options, a configurator and a simple selection among pre-configured products as shown in Figure 6. Interestingly, our comparison of available products revealed that the keyboard is always pre-configured to be *British Qwerty* and else only one products is missing opposed to the product configurator (i.e., *Intel Core i5* with *128GB* storage). That is, ten configuration options in four categories have so many constraints that they only result in eight different products for customers, instead of 32 theoretical combinations.

Compound options are well-suited if the combined options have many dependencies and customers can directly explore the few available combinations. In particular, it would not be feasible to present all 123 combinations of colors and sizes in the Amazon configurator (cf. Figure 10). However, compound options also typically lead to the problem that one original option is represented by several compound options, which increases the effort for customers when comparing

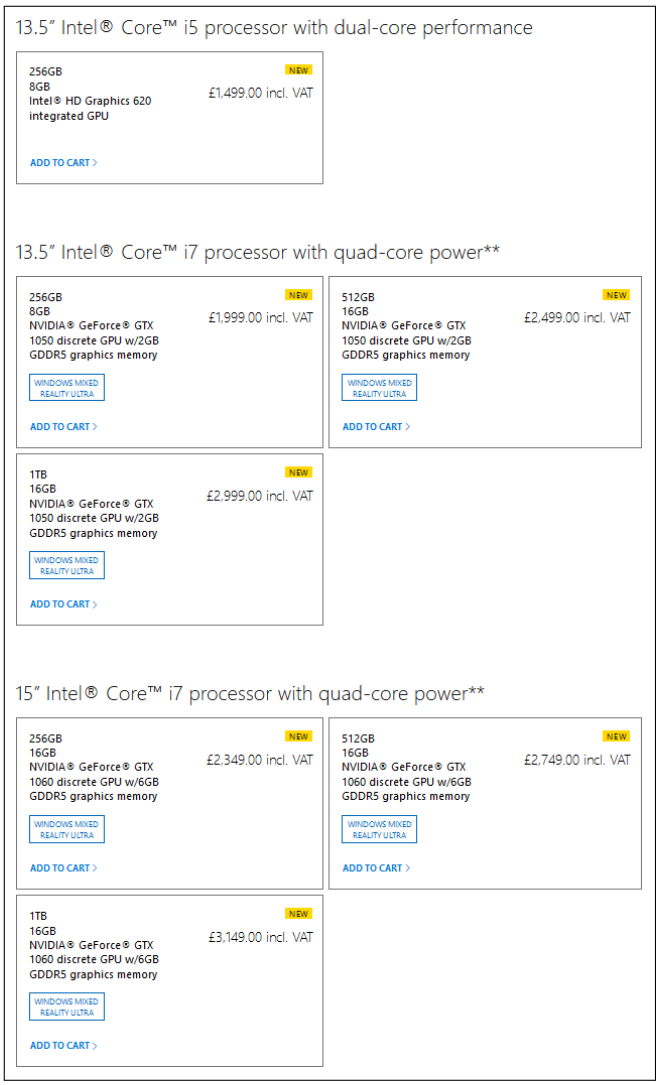

Figure 6. Microsoft's product selection opposed to configuration.

compound options. Furthermore, compound options may complicate the handling of dependencies to options in other parts of the configurator. For instance, a simple rule requiring another option may then need to be duplicated for all occurrences in compound options.

3.5 Continuing with Invalid States

A further strategy to deal with conflicting options is to let users continue with the configuration process although the current configuration is invalid. That is, users may even select an option that is not allowed according to dependencies and already selected options. The action required by the user to fix the configuration is basically postponed to a later point during the configuration process.

As illustrated in Figure 7, Lenovo's configurators issue a warning to the user once a conflict was detected. The user can continue with the configuration process, but the same warning appears after every selection until the problem is fixed. Unfortunately, the only hint on how to solve this problem is the written-down dependency along with one of the interacting configuration options. Thus, the dependency may happen to be written next to the option whose selection lead to the problem or next to any other option.

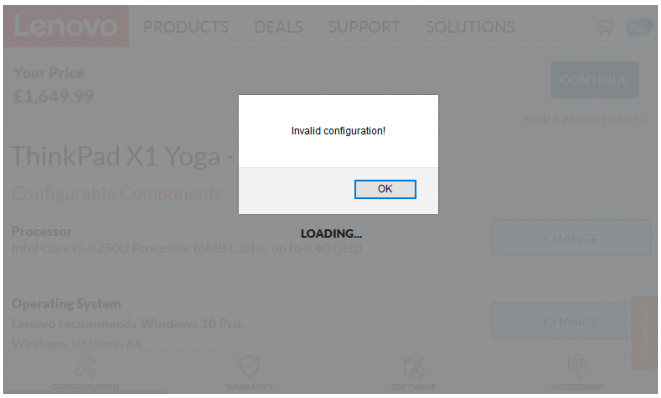

Figure 7. Warning for invalid states in Lenovo's ThinkPad configurator.

Letting users continue with invalid states seems to be a flexible way for configuration, as users can decide when to handle conflicts. In particular, a user can focus on the most relevant options first and then solve conflicts with the remaining options. However, especially problematic is that postponing a fix can result in multiple conflicts, which are even harder to fix. Furthermore, it seems that tool support for invalid states is not straightforward, as we have only found this strategy in the two configurators by Lenovo and Toyota, of which neither provided useful support.

3.6 Subsequent Configuration Steps

Configurators with many options often split the decision process over several steps. The decisions in one configuration step are confirmed before continuing with subsequent steps. While configurators typically allow to go back and forth in those steps, the configuration decisions of subsequent configuration steps may be reverted in this process. In some configurators, certain steps can even be skipped based on the configuration in prior steps.

For instance, the BMW configurator consists of seven steps and depending on the choice of engine, a further additional step called *Model Variants* may appear (cf. Figure 8). However, in the studied configurator, the additional step was only used to show an inclusive option called *Sport package*, which cannot be deselected.

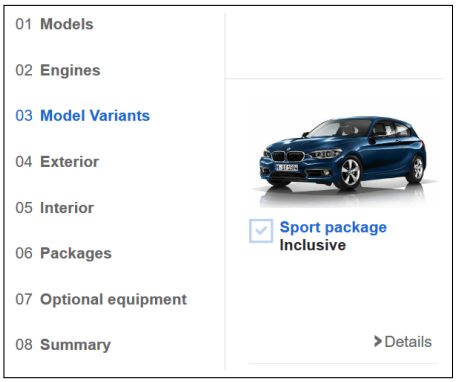

Figure 8. Configuration steps in the BMW configurator.

Configuration steps seem to increase the overview over complex configuration processes with hundreds of features. Hiding irrelevant configuration steps is also an interesting concept to deal with constraints. Nevertheless, hiding certain steps may also be a challenge for users that are actually looking for a given step and do not understand why it is not visible. In addition, users could be unaware of configuration steps that would be helpful in their decision making process. Furthermore, although guiding the user and hiding options, configuration steps do not solving the problem of conflicting configuration options. Even worse, constraints between different configuration steps could be a particular challenge for users, if they have to switch back and forth over and over again.

3.7 Automated Reconfiguration

Another strategy to deal with constraints is to automatically select and deselect options after every user decision. In particular, constraints are used to identify conflicting decisions. Those conflicts are then resolved automatically, if possible.

Automated reconfiguration is applied by the BMW configurator. If already chosen configuration options are affected by the current decision, a configuration assistant is shown to the user summarizing the automated changes (cf. Figure 9). There is also an automatic mode in which the configuration assistant will not inform users after every reconfiguration. Unfortunately, we were not able to provoke a situation in which the configurator obviously has to chose one of several alternative fixes. Such a situation would be interesting to see whether some kind of user interaction is supported in this process.

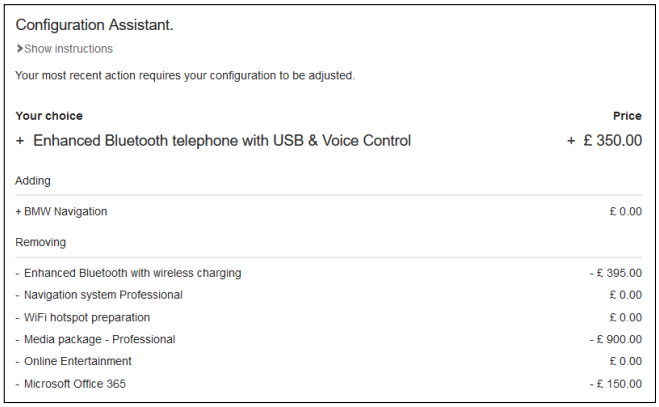

Figure 9. Automated reconfiguration in the BMW configurator.

In the Amazon configurator, we can distinguish available from unavailable combinations visually, as illustrated in Figure 10. Nevertheless, we can select also grayed-out options, such that a conflicting state arises. How the conflict is solved dependent on the latest decision. If a new size is selected that is not available for the current color selection, then another color that is still available is automatically selected. The selection appeared to be arbitrary, but it could be based on a recommender knowing more common combinations (e.g., red for smaller sizes and blue for larger ones). Interestingly, if a new color is selected the configurator does not automatically select another size, but instead resets the size selection. A reason for this different strategy might be that people can come to terms with another color, but typically are not willing to choose a t-shirt in a great color with a size that does not fit them.

One advantage of automated reconfiguration is that conflicts are directly solved when they occur. Furthermore, users may have the chance to confirm or abort the current decision propagation. However, an issue with this strategy is that reverting decisions that triggered automated reconfiguration may not revert the automatically introduced changes. This behavior can be confusing to the user and can

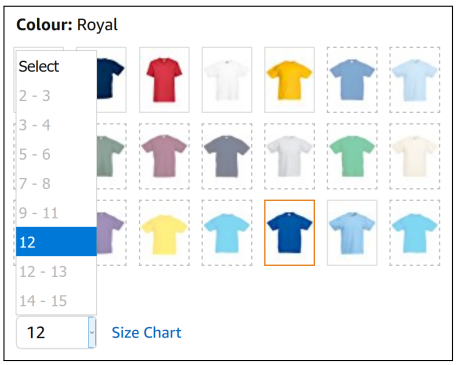

Figure 10. Automated reconfiguration in the Amazon configurator.

even introduce further conflicts. Furthermore, automated reconfiguration can interact with other applied strategies in undesired ways, which can again lead to the introduction of conflicts.

3.8 Interactive Resolution of Conflicts

Once a conflict occurred, it can either be resolved automatically, as discussed in Section 3.7, or with user interaction. With interactive resolution, we refer to a process in which the configurator informs users about conflicts in the current configuration and guides them to a resolution by proposing possible changes. Thus, interactive resolution is especially helpful if there are several meaningful resolutions to a conflict and the users input is required.

We experienced interactive resolution of conflicts in the configurators by Toyota and by HP Velotechnik. For Toyota, we identified a dependency between car colors and wheels. Black wheels are only available for three of six car colors, namely *White Flash*, *Electro Gray*, and *Bold Black*. Only when of those colors is selected, black wheels are visible (cf. hiding invalid combinations). However, when black wheels are selected, we can still choose one of the three incompatible car colors, as those are not hidden. In that case, the configurator selects the incompatible car color and shows the dialog for wheels in which the previously chosen wheel is hidden. If users select another wheel, the conflict is resolved. However, we experienced

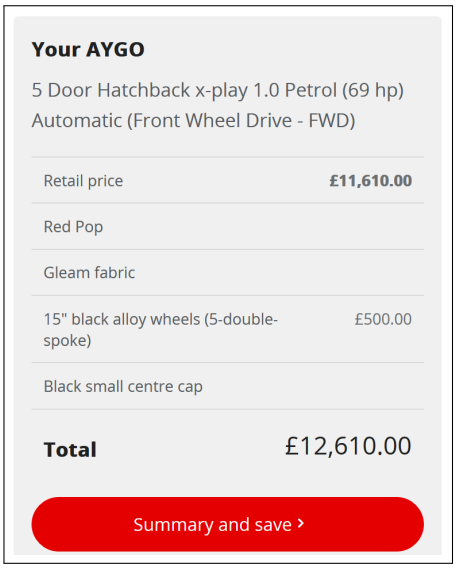

Figure 11. An unresolved conflict leading to a wrong price computation in the Toyota configurator.

bugs when users decide to cancel the interactive process. We were actually able to configure a car with a wrong price computation (cf. Figure 11). If we continue and try to actually buy that car, the reason for the wrong computation becomes clear; the configurator automatically selected two wheel types, for which eight instead of four wheels are charged in terms of costs (cf. Figure 12). When we tried to fix the conflict manually, we were not even able to open the configuration dialog for wheels anymore. However, the Toyota configurator has a rather unique feature allowing users to undo and even redo previous decisions by going back and forth in the browser.

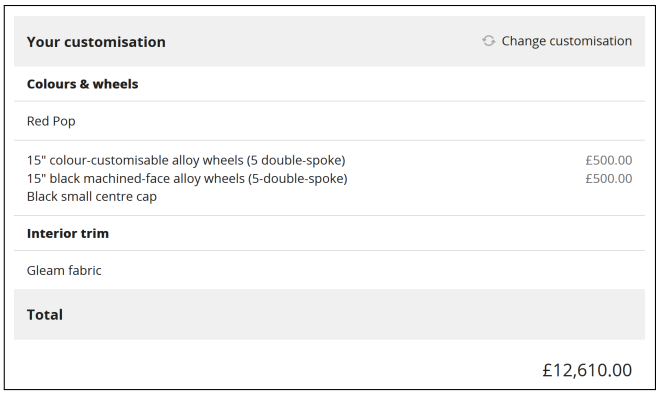

Figure 12. Two alternative wheels selected in the Toyota configurator.

For HP Velotechnik, interactive resolution is applied to all conflicts that occur between any pair of two categories. When opening the configurator, each category is set to the first option, whereas options are sorted in alphabetical order. When changing the tires to *Schwalbe Kojak* for the default configuration, a dialog opens asking the user to select one of two options as a drivetrain (cf. Figure 13). Although we opened the UK version, there is a hint in German saying that an unrelated drivetrain (i.e., not available and not selected as default) has some conflicts with particular brakes. After studying the configurator for a while, we found out that this hint is always shown if there is a conflict with drivetrains and the hint is most likely supposed to help with another conflict. Selecting one of those two options leads to another conflict which is resolved with the same technique. While, in this case, there are only two interactive resolution steps due to one selection, we experienced cascades of up-to five conflicts. With such cascades, there is a considerable burden on the user to understand why all those changes need be done. Configuration is especially challenging as users easily get lost which options were the default, which did they chose on purpose, and which options had to be selected in the interactive conflict resolution.

The advantage of resolving conflicts interactively is that the user is essentially making decisions on conflicting options. Hence, opposed to automatic reconfiguration users have the control over conflict resolution and opposed to the hiding of invalid combinations they can actually see and decide on all options. The downside is, however, that users have to take decisions in response to a conflict directly. In particular, this may lead to a frustrating cascade of numerous conflict resolutions, which may even need to be reverted, if the user is not satisfied with the options in one conflict resolutions. A postponed interactive resolution would probably need some kind of support for continuing with invalid states (cf. Section 3.5), which seems to be hard to be implemented properly.

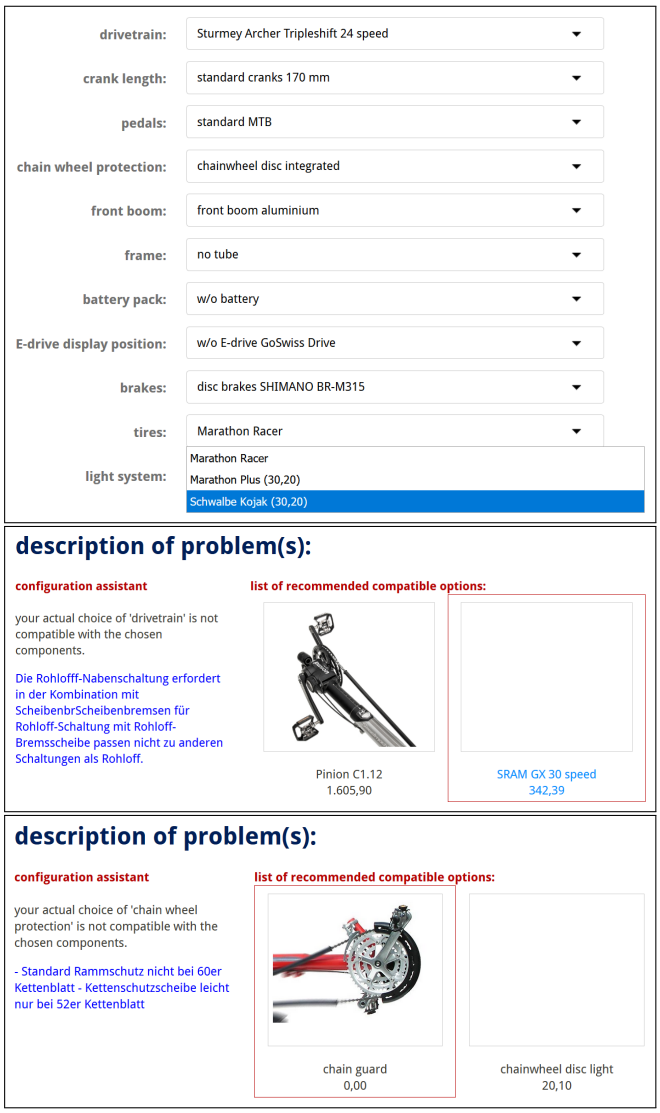

Figure 13. A cascade of conflicts in the HP Velotechnik configurator.

3.9 Combinations of Strategies

In the previous sections, we discussed each identified strategy independently. However, in practice, none of the studied configurators applies only a single strategy. Instead, we found that configurators apply between two and eight strategies, as illustrated in Table 1. HP Velotechnik, Ebay, and Amazon only apply two or three strategies. The configurators of Ebay and Amazon did only come with a few categories of alternative options. The most strategies are applied in the car configurators by Toyota and BMW, which also comprise the largest number of options and categories. The configurator by Toyota even applies all eight strategies.

We can further distinguish between strategies that are applied globally to all options or only locally to some options. Automatic deselection of alternative options is not only applied in all configurators, but also for all categories with alternative options. Similarly, configuration steps and default options have been applied only globally. In contrast, all other strategies have typically be applied locally to some options in the configurator. One exception is that the Ebay configurator uses hiding of invalid combinations in all parts, but there were only two categories of alternative features. Other exceptions

Strategy	BMW	Toyota	HP Velotechnik	Lenovo	Microsoft	Amazon	Ebay
Automatic Deselection in Alternatives	y	y	y	y	y	y	y
Default Configuration	y	y	y	y	y/n	y	n
Hiding Invalid Combinations	n	y	n	n	n	n	y
Compound Options	y	y	n	y	y	n	n
Continuing with Invalid States	n	y	n	y	n	n	n
Subsequent Configuration Steps	y	y	n	n	n	n	n
Automated Reconfiguration	y	y	n	n	y/n	y	n
Interactive Resolution of Conflicts	n	y	y	n	n	n	n

Table 1. Overview on configurators and their strategies.

are that BMW applies automated reconfiguration globally and HP Velotechnik uses interactive resolution for all categories.

The configurator by Microsoft is especially interesting as there are multiple links to configurators for the same product, which apply different strategies. For instance, there is a version of a configurator using default configurations and another version that opens without any decisions. Also, we experienced that the Microsoft configurator stopped using automated reconfiguration in April or May 2018.

The strategies that we identified can be further classified into strategies *avoiding* all or at least some conflicts and those strategies *resolving* conflicts. Default configurations, hiding invalid combinations, compound options, and configuration steps help to avoid conflicts. All other strategies let users make conflicting decisions and later solve them automatically or interactively.

4 Related Work

There exists a number of previous studies comparing web-based configurators [1, 14, 12]. Each of these studies considers different aspects of these tools. For instance, Streichsbier et al. focuses on the differing user interface of configurators for different domains [14]. They describe differences and similarities in the design of configurators, such as navigation buttons, product images, and prices displays, across multiple domains. Abbasi et al. compare 111 different configuration tools regarding the process of configuration and the internal behavior of configurators [1]. In their work, they consider the visual representation of options and constraints and the corresponding semantics behind it, constraint syntax and handling, and control flow of the configuration process. Pil and Holweg studied configurators of ten car manufacturers [12]. In addition to the external variety provided by the configurator, they also studied the internal variety by surveying the manufacturers. In contrast to these prior studies, we explicitly focus on the underlying strategies for handling conflicts and categorize these further.

Other work, such as Herrmann et al., Haag et al., and Hubaux et al. have looked at specific strategies that we cover in our paper [6, 7, 9]. In their paper, Herrmann et al. investigate the effects of starting the configuration process with a *default configuration*, which can be adapted by the user [7]. Although we do not investigate user behavior when using a particular conflict handling strategies, we aim to provide a broader view of currently used strategies on the web. Haag et al. consider different techniques for finding explanations in conflicting configurations [6]. This is corresponds to our identified strategy of *resolving conflicts interactively*. Amongst others, Hubaux et al. investigate which conflict handling strategies are being used in the two product lines Linux and eCos [9]. Although they are

more focused on the user perspective, they find that the Linux' configuration tool uses a mix of different strategies, such as *automated reconfiguration*, *hiding invalid combinations*, and *resolving conflicts interactively* and that eCos also allows to *continue with invalid states*.

Product configuration is very similar to software configuration [8], for which many tools exist [2, 11]. Berger et al. conducted a survey among practitioners and researchers with industrial experience on software product-line engineering and variability modeling [2]. Amongst others, they present data about the distribution of modeling and configuration tools that are used in practice and the scale of real-world variability models, regarding number of configuration options and dependencies. While, in this paper, we are also interested in dependencies that result from variability modeling, we foremost consider the strategies to enforce these dependencies during the configuration process.

5 Conclusion and Future Work

Product configurators are essential for the vision of mass customization. Customers use configurators to identify options and their valid combinations. A configurator is often the main source of knowledge being queried in the decision making process. We studied a corpus of seven configurators to understand how they handle dependencies between configuration options. Those dependencies can easily lead to decisions by a customer that are in conflict with each other. We argue that how a configurator supports users in avoiding or fixing conflicts is crucial for the success of mass customization.

In our study, we identified eight strategies to handle dependencies. Half of those strategies aim to prevent conflicting decisions whereas the rest help customers to resolve conflicts automatically or with their interaction. Most strategies are applied by several but not all configurators, whereas one strategy is applied by all configurators and all other strategies are applied by at least two configurators. Similarly, each studied configurator applies at least two strategies and we also found a configurator that applies all eight strategies. It seems that smaller configurators with fewer options and fewer dependencies tend to use fewer strategies and larger configurators apply numerous strategies.

While we discussed advantages and disadvantages of all strategies, it is an open research question which combinations of those strategies are best in which situations. Furthermore, it would be interesting to investigate whether there are further strategies and how each strategy can be supported with the state-of-the-art techniques for the implementation of configurators. In any case, studying further real-world configurators may give more insights.

REFERENCES

[1] Ebrahim Khalil Abbasi, Arnaud Hubaux, Mathieu Acher, Quentin Boucher, and Patrick Heymans, 'The Anatomy of a Sales Configurator: An Empirical Study of 111 Cases', in *Proc. Int'l Conf. Advanced Information Systems Engineering (CAiSE)*, pp. 162–177, Berlin, Heidelberg, (2013). Springer.

[2] Thorsten Berger, Ralf Rublack, Divya Nair, Joanne M. Atlee, Martin Becker, Krzysztof Czarnecki, and Andrzej Wąsowski, 'A Survey of Variability Modeling in Industrial Practice', in *Proc. Int'l Workshop Variability Modelling of Software-Intensive Systems (VaMoS)*, pp. 7:1–7:8, New York, NY, USA, (2013). ACM.

[3] Stanley M. Davis, *Future Perfect*, 1987.

[4] Rebecca Duray, Peter T. Ward, Glenn W. Milligan, and William L. Berry, 'Approaches to Mass Customization: Configurations and Empirical Validation', *J. Operations Management (JOM)*, **18**(6), 605–625, (2000).

[5] Cipriano Forza and Fabrizio Salvador, *Product Information Management for Mass Customization: Connecting Customer, Front-Office and Back-Office for Fast and Efficient Customization*, Palgrave Macmillan, New York, NY, USA, 2006.

[6] Albert Haag, Ulrich Junker, and Barry O'Sullivan, 'A survey of explanation techniques for configurators', in *Proceedings of ECAI-2006 Workshop on Configuration*, volume 41, p. 44.

[7] Andreas Herrmann, Daniel G. Goldstein, Rupert Stadler, Jan R. Landwehr, Mark Heitmann, Reto Hofstetter, and Frank Huber, 'The Effect of Default Options on Choice—Evidence from Online Product Configurators', *J. Retailing and Consumer Services*, **18**(6), 483–491, (2011).

[8] Arnaud Hubaux, Dietmar Jannach, Conrad Drescher, Leonardo Murta, Tomi Männistö, Krzysztof Czarnecki, Patrick Heymans, Tien N. Nguyen, and Markus Zanker, 'Unifying Software and Product Configuration: A Research Roadmap', in *Proc. Configuration Workshop (ConfWS)*, pp. 31–35, (August 2012).

[9] Arnaud Hubaux, Yingfei Xiong, and Krzysztof Czarnecki, 'A User Survey of Configuration Challenges in Linux and eCos', in *Proc. Int'l Workshop Variability Modelling of Software-Intensive Systems (VaMoS)*, pp. 149–155, New York, NY, USA, (2012). ACM.

[10] Cynthia Huffman and Barbara E. Kahn, 'Variety for Sale: Mass Customization or Mass Confusion?', *J. Retailing*, **74**(4), 491–513, (1998).

[11] Jens Meinicke, Thomas Thüm, Reimar Schröter, Fabian Benduhn, and Gunter Saake, 'An Overview on Analysis Tools for Software Product Lines', in *Proc. Workshop Software Product Line Analysis Tools (SPLat)*, pp. 94–101, New York, NY, USA, (2014). ACM.

[12] Frits K. Pil and Matthias Holweg, 'Linking Product Variety to Order-Fulfillment Strategies', *Interfaces*, **34**(5), 394–403, (October 2004).

[13] Giovani Da Silveira, Denis Borenstein, and Flávio S. Fogliatto, 'Mass Customization: Literature Review and Research Directions', *Int'l J. Production Economics*, **72**(1), 1–13, (2001).

[14] Clarissa Streichsbier, Paul Blazek, Fabian Faltin, and Wolfgang Frühwirt, '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', in *Proc. Hawaii Int'l Conf. System Sciences (HICSS)*, pp. 1–7, Washington, DC, USA, (2009). IEEE.

[15] Juha Tiihonen and Alexander Felfernig, 'An Introduction to Personalization and Mass Customization', *J. Intell. Inf. Syst.*, **49**(1), 1–7, (August 2017).