On Studying Bad Practices in Configuration UIs

Tony Leclercq, Maxime Cordy, Bruno Dumas, Patrick Heymans
University of Namur, Belgium
first.last@unamur.be

ABSTRACT
In today's businesses, configurators are essential tools that allow customers to personalise a product to their specific needs. Being often the first contact between the retailer and the customer, it is important for a configurator to provide a smooth user experience. However, these software have been studied mostly from the reasoning point of view, that is, how to represent the logical relations between the configuration options and how to automatically ensure that the user makes no configuration error. In this paper, we focus instead on HCI and analyse the occurrence of 11 design flaws that occur in 28 car configurators. These flaws includes violations of general HCI principles as well as lacks of essential, configurator-specific functionalities. Our results indicate that most of the studied configurators contain defects in their UIs including, surprisingly, violations of some basic principles of HCIs. Still, the most frequent problems are inappropriate implementations of configurator-specific functionalities.

ACM Classification Keywords
H.5.2. User Interfaces: Graphical user interfaces (GUI), Configurator, Evaluation, Intelligent system

INTRODUCTION
The avenue of mass customisation – an established marketing and engineering paradigm that unifies mass production and customisation [11] – has yielded new habits in customers' behaviour. In constant search of the ideal product (i.e. good or service) that satisfies their particular needs, they hunger for the freedom of tailoring sellers' catalog into a personalised offer. It is therefore not surprising that personalisation is considered as one of the hottest topics in the retail industry [13]. Still, to be successful personnalisation must be supported by a smooth user experience (UX) that offers the highest customisation capability while hiding the inherent complexity [12].

Configurator is the typical software support for mass customisation. It consists in an interactive application where the user specifies its requirements by selecting options and setting parameter values. As a result she obtains the relevant products that can address these needs. For example, Figure 1 shows a

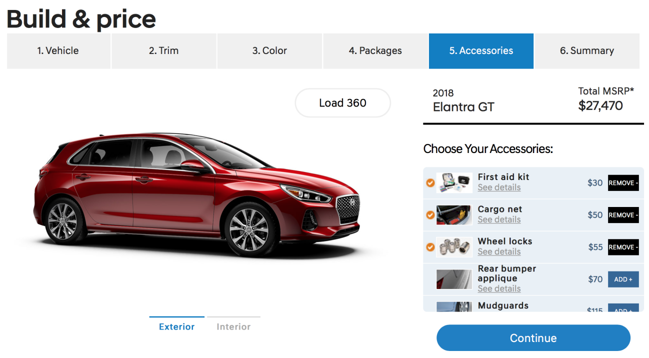
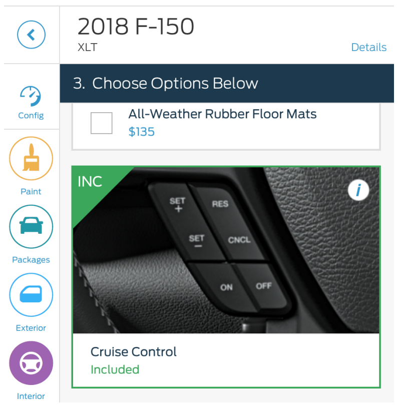


Figure 1. Some car configurator

car configurator where customers can choose between various options like trim, motor, colour, accessories, etc. Beneath the user interface lies an intelligent system that provides effective guidance by preventing errors and providing automated support for the configuration task [5, 8]. An error is typically an attempt to include incompatible options or parameter values in the configuration. When the user makes an error, the configurator should alert her and explain why the configuration is invalid. On the other side, when previous choices make mandatory to select or to exclude an option, or when a parameter accepts only one value, the configurator should *propagate* (i.e., force) this choice so as to simplify the task of the user. For instance, selecting a comfort trim automatically includes a cruise control system; see Figure 2). It is again recommended to explain the cause of the propagation, in case the user disagrees with this choice. When errors or undesired propagations occur, the configurator should help *repair* the configuration [19, 20], that is, automatically compute what must be changed to fix the error or undo the propagation, respectively. An example of repair system is shown in Figure 3. We see that the selected wheels require a particular suspension system and specific tires, as well as removing the off-road accessory pack.

Configuration knowledge refers to the set of rules that determine what is a valid configuration. It originates from rational (technical, legal, mathematical) and subjective (marketing, aesthetics) knowledge. The acquisition of this knowledge is a critical analysis activity for a successful configurator [2, 6], as erroneous or incomplete knowledge can have drastic consequences on the products that will be built afterward. Once the configuration knowledge is defined, one can develop the back-end part of the configurator responsible for checking errors, as well as computing propagations and repairs. A standard way

WII'18, March 11, 2018, Tokyo, Japan



2018 F-150
XLT
Details

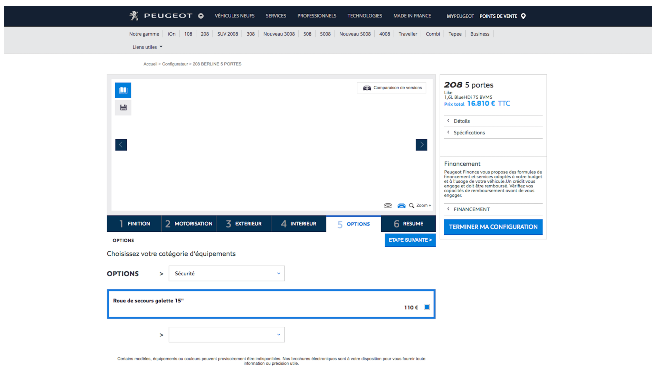
3. Choose Options Below

All-Weather Rubber Floor Mats
$135

INC

Cruise Control
Included

Figure 2. Propagation of mandatory option

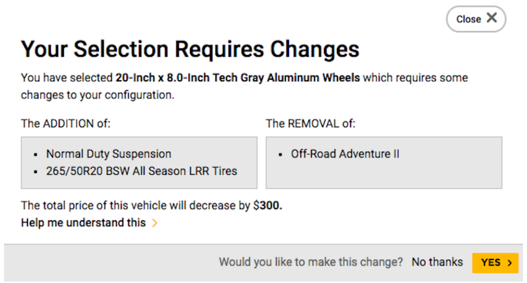


Choisissez votre catégorie d'équipements

OPTIONS > Sécurité

Roue de secours galette 15" 110 €

Figure 4. Violations of basic ergonomic guidelines: empty controls, empty boxes and wrong control placements



Close ✕

Your Selection Requires Changes

You have selected **20-Inch x 8.0-Inch Tech Gray Aluminum Wheels** which requires some changes to your configuration.

The ADDITION of:	The REMOVAL of:
• Normal Duty Suspension • 265/50R20 BSW All Season LRR Tires	• Off-Road Adventure II

The total price of this vehicle will decrease by **$300.**
Help me understand this ›

Would you like to make this change? No thanks **YES ›**

Figure 3. An instance of repair where some options must be added and others removed

to do that is to implement it from scratch in an established programming language like Java or PHP. However, this results in scattering the knowledge across the code base, which leads to increased risks of bugs and poor maintainability [1]. An alternative solution is to encode it as logical rules in a *configuration engine* [14, 7, 3], i.e., a dedicated software component that implements generic algorithms to reason on configuration knowledge. Such engines thus provide the configuration functionality at no cost and regardless of the actual application domain.

While efficient configuration engines exist, they are only a part of the challenges in configurator engineering. Indeed, configurators belong to a class of interactive applications where UX is of paramount importance [16] and relies on specific functionalities of these intelligent systems [12]. However, there exist neither established nor de-facto HMI standards for configurators [17, 9]. Even worse, recent studies [1] showed that many public configurators do not follow general HMI guidelines, resulting in weak usability. This reveals that designing HMI guidelines for configurators is a essential element of the envisioned *"body of knowledge dedicated to the engineering of configurators"* [1, 3].

A prerequisite to creating these guidelines is to understand what existing configurators do wrong. Therefore, pursuing previous research [12, 1], in this paper we analyse to what extent existing configurators contain HCI design flaws. We focus more particularly on car configurators, the most popular kind of configurators according to the Cyledge's database statistics [4]. Car configurators are excellent candidates: they commonly provide a high number of options (which makes both presentation and reasoning challenging), they are directly aimed at customers (which makes UX a primary concern), and we can find very different UI design choices across different brands and countries. More precisely, we check whether 11 UX design flaws occur in 28 configurators we selected across different brands and countries. These 11 flaws comprise 4 general HMI anti-patterns, 4 specific to navigation-based interfaces, and 3 related to configurator-specific functionalities. Our results reveal that 75% of the configurators show more than three types of UX flaws, half of them more than six. This is mostly due to missing or badly implemented configuration-specific functionalities, although we also observe the violation of general HCI principles. This corroborates the need for HCI guidelines specific to configurator engineering.

AN OVERVIEW OF UX ISSUES IN CONFIGURATORS

To carry out our study, we select a set of car configurators amongst the ten top-selling car manufacturers. For each manufacturer we choose the best selling car model, and for every model we consider the configurator of the US, UK and Belgium websites. This leads us to 28 configurators to evaluate (as two of the 30 candidates do not allow one to select any option).

Looking at these configurators, we rapidly note that basic ergonomic guidelines [18, 15, 16, 10] are not satisfied. Indeed, we observe occurrences of empty controls, empty boxes or wrong control placements; see Figure 4. Inappropriate choice of controls, inconsistent semantics grouping, and lack of emphasis were also very common. This is relatively surprising, as car configurators are popular and powerful marketing tools. This makes us consider general HCI principles as our first evaluation criteria. Our objective being more focused than a general-purpose HCI evaluation, we settle for only four violations we observed the most: absence of feedback, information overload, too many controls and bad ergonomics (see more in the next section).

In addition to these general considerations, we also observe issues related to the management of *navigation* within the UI. Figure 5 is a striking example that highlights multiple problems. First, we see that the interface does not display the step currently completed. Also it provides no visible way to

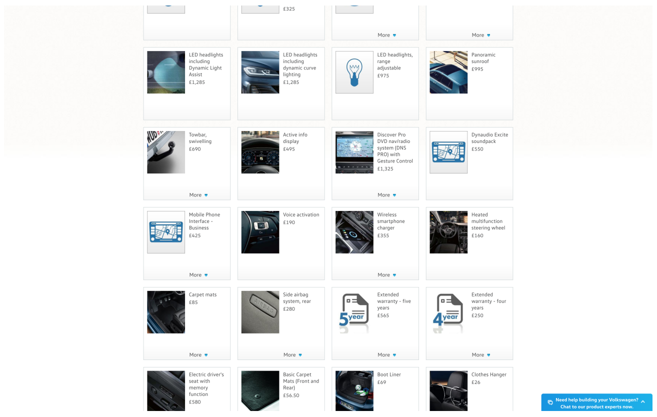
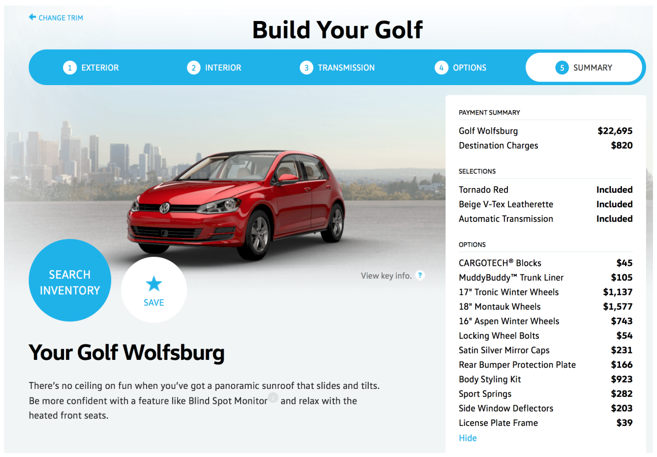

£325
More
More
LED headlights including Dynamic Light Assist £1,285
LED headlights including dynamic curve lighting £1,285
LED headlights, range adjustable £975
Panoramic sunroof £995
Towbar, swivelling £690
Active info display £495
Discover Pro DVD nav/radio system (DNS PRO) with Gesture Control £1,325
Dynaudio Excite soundpack £550
Mobile Phone Interface - Business £425
Voice activation £190
Wireless smartphone charger £355
Heated multifunction steering wheel £160
Carpet mats £85
Side airbag system, rear £280
Extended warranty - five years £565
Extended warranty - four years £250
Electric driver's seat with memory function £580
Basic Carpet Mats (Front and Rear) £56.50
Boot Liner £69
Clothes Hanger £26
Need help building your Volkswagen? Chat to our product experts now.

Figure 5. All signs of navigation have disappeared



Your Golf Wolfsburg

There's no ceiling on fun when you've got a panoramic sunroof that slides and tilts. Be more confident with a feature like Blind Spot Monitor and relax with the heated front seats.

PAYMENT SUMMARY	
Golf Wolfsburg	$22,695
Destination Charges	$820
SELECTIONS	
Tornado Red	Included
Beige V-Tex Leatherette	Included
Automatic Transmission	Included
OPTIONS	
CARGOTECH® Blocks	$45
MuddyBuddy™ Trunk Liner	$105
17" Tronic Winter Wheels	$1,137
18" Montauk Wheels	$1,577
16" Aspen Winter Wheels	$743
Locking Wheel Bolts	$54
Satin Silver Mirror Caps	$231
Rear Bumper Protection Plate	$166
Body Styling Kit	$923
Sport Springs	$282
Side Window Deflectors	$203
License Plate Frame	$39

Figure 6. Invalid configuration: multiple wheel sizes are selected

go to the previous or the next step. The user has no idea of the progress she has already accomplished, and can neither consult nor preview the current state of her configuration. All these navigation-related issues constitute our second group of criteria.

Finally, we observe the lack of configuration-specific functionalities that are, however, deemed essential. For instance, Figure 6 depicts a configuration where three different sizes of wheels are selected. Errors are thus unchecked in this configurator. Configuration being a long and error-prone process, it is unrealistic not to help users complete their task. Explanation of errors, propagations and repairs – as presented in the previous section – are other facilitating functionalities. We argue that all configurators should implement them. Together with error checking, they are thus part of our third type of evaluation criteria. All our criteria are properly defined in the next section.

UX FLAWS: DEFINITIONS AND CRITERIA

Based on our observations, we elaborated a list of common design flaws typically found in configurators. We categorize them into three degrees of specifity. The first degree comprises violations of **general HCI principles** that any application should satisfy. The second concerns issues related to **navigation** and are thus commonly found in interactive applications that represent a process. The last degree is related to the specific functionalities found in configurators, like propagations and repairs.

General HCI. We selected four types of violations that we deem critical in configurators. We settle for those, as our aim is not to perform a thorough analysis but rather to study how often such criteria are unsatisfied in practice. These are:

- **Absence of feedback**. No feedback is given to the user. There is no apparent result to her action.
- **Information overload**. The quantity of displayed information is too high, or there is too much noise.
- **Too many controls**. The UI displays too many controls on a single screen, or the number of controls needed to perform a single action is too high.
- **Bad ergonomics**. The configurator does not follow established ergonomics guidelines. These include, e.g., misplaced controls and inappropriate colour or font size.

Navigation. Configuration being an interactive process, its navigation must be absolutely clear. We can see navigation from three points of view, i.e. in the UI, in the progress in the process, and in the configuration state. This leads us to four types of violations:

- **Unknown UI location**. The user is lost within the UI and has no idea what step she is currently completing.
- **Irregular navigation control**. In the UI, the user cannot see the control allowing her to move to the next step, because their position or shape change.
- **Unknown progress**. The progress in the process, i.e. what percentage of effort the user has already put in or remains to be done, is not displayed.
- **Unknown configuration state**. The current state of the configuration (e.g. what options were selected, how the final product looks like) is not displayed.

Configuration. Finally, the lack of essential configurator-specific functionalities can lead to bad UX, as reported by [12]. More precisely, we consider the following flaws:

- **Unchecked errors**. The configurator allows the user to make incompatible choices, which lead to invalid configurations and ultimately to infeasible products.
- **No explanation**. The configurator does not explain why errors are raised or why some options became mandatory or unavailable.
- **No repair**. The configurator does not provide any repair functionality that helps the user fixing an error or changing her configuration, e.g., to make available forbidden options.

RESULTS

For each of the 28 configurators, we check whether its UI exhibits the 11 aforementioned design flaw. We first observe the occurrence of each design flaws. Accordingly, Figure 7 presents for each flaw how many configurators exhibit it.

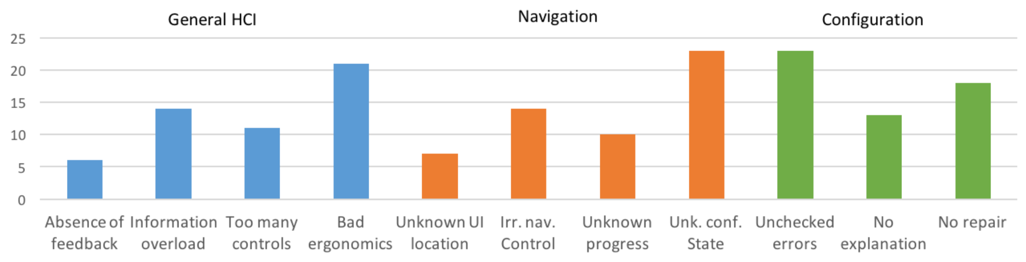


Figure 7. Number of configurators that exhibit each flaw

For general HCI principles, we observe that the lack of ergonomics is the most common pitfall, with as many as 21 configurators concerned with such issues. This is not totally surprising, as the ergonomics guidelines we consider are rather large, although we could expect better ergonomics in such customer-oriented applications. Information overload follows next, with half of the configurators being problematic. It is indeed common that car configurators have to deal with large amounts of options, hence the difficulty of presenting all these options while avoiding cognitive overload. 11 configurators contain too many controls. According to our observations, this is the result of overusing categorization to limit the number of options displayed to the user at once. Finally, only six configurators do not provide feedback. Overall, we notice that six configurators out of the 28 do not violate our general HCI principles, which may reveal an inherent difficulty to satisfying all these principles in this kind of application.

Regarding navigation issues, seven configurator UIs do not present clearly the navigation steps that the user has to follow. Furthermore, the navigation controls are irregular in 14 cases, while ten configurators do not display any progress. More importantly, most of the configurators (23 cases) do not show the current status of the configuration, which is increasingly important as the number of options gets higher. These navigation problems likely interfere with the user's experience; systematic solutions should thus be provided.

The most recurring problems, however, originate from configuration-specific functionalities. 23 out of 28 configurators do not systematically check the absence of errors, thereby authorising invalid configurations. We also identified 13 cases where explanations are not always given following an error or a propagation. Finally, 18 configurators do not provide automated repair each time they should.

Figure 8 gives us insights into the number of flaw types that occur in the configurators. We observe that half of the configurators violate more than six of our principles. This can be explained by the fact that the three configuration-specific functionalities are often missing altogether, while unknown configuration state and ergonomic problems are also pretty common. More surprisingly, only a quarter of the configurators comprise three violations or less. In the end, only one configurator does not contain any flaw. These observations tend to show the necessity of establishing guidelines to be followed by every configurator.

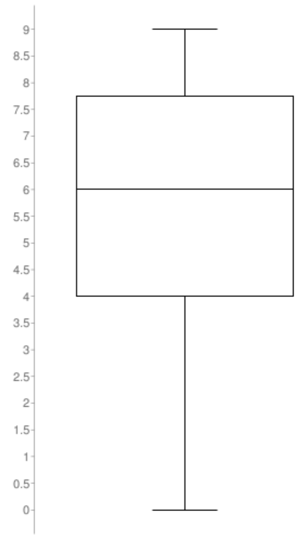


Figure 8. Number of flaw types occurring in configurators

These results clearly show that even though configurators are often infected by common UX flaws, the most important issues are specific to the configuration problem. These include not displaying the current configurator state and not providing essential configuration functionalities. Moreover, even if these functionalities were provided, there would remain the question of how to integrate them properly in a configuration UI.

CONCLUSION

Configurators are intelligent and highly-interactive systems that play a major role in today's businesses. Despite their importance, our study reveals major issues in regard to their UX. On the other side, no HCI standards exist for these applications, and only few research work were interested in configurator UX. This motivates our long-term objective to *design HCI guidelines specific to configurators*, taking into account all their functionalities and particularities. In the future, we plan to expand our study across multiple industries in order to generalise our conclusions. This way, we also hope to identify the sources of the different UX problems, thereby paving the way to our envisioned elaboration of guidelines.
Acknowledgement This work was partly supported by the European Commission (FEDER IDEES/CO-INNOVATION) and the Wallonia-Brussels Federation under the ARC programme.

REFERENCES

1. Ebrahim Khalil Abbasi, Arnaud Hubaux, Mathieu Acher, Quentin Boucher, and Patrick Heymans. 2013. The Anatomy of a Sales Configurator: An Empirical Study of 111 Cases. In *CAiSE'13*. Springer-Verlag, Berlin, Heidelberg, 162–177.

2. Liliana Ardissono, Alexander Felfernig, Gerhard Friedrich, Dietmar Jannach, Ralph Schäfer, and Markus Zanker. 2002. A Framework for Rapid Development of Advanced Web-based Configurator Applications. In *ECAI'02*. IOS Press, Amsterdam, The Netherlands, The Netherlands, 618–622.

3. Maxime Cordy and Patrick Heymans. 2018. Engineering Configurators for the Retail Industry: Experience Report and Challenges Ahead (to appear). In *SAC '18*. ACM.

4. Cyledge. 2017. Configurator Database. (2017). Retrieved July 24, 2017 from `http://www.configurator-database.com`

5. Alexander Felfernig, Lothar Hotz, Claire Bagley, and Juha Tiihonen. 2014. *Knowledge-based Configuration: From Research to Business Cases* (1 ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

6. Gerhard Fleischanderl, Gerhard E. Friedrich, Alois Haselböck, Herwig Schreiner, and Markus Stumptner. 1998. Configuring Large Systems Using Generative Constraint Satisfaction. *IEEE Intelligent Systems* 13, 4 (July 1998), 59–68.

7. Lothar Hotz, Alexander Felfernig, Andreas Günter, and Juha Tiihonen. 2014. A short history of configuration technologies. *Knowledge-based Configuration–From Research to Business Cases* (2014), 9–19.

8. Arnaud Hubaux, Yingfei Xiong, and Krzysztof Czarnecki. 2012. A User Survey of Configuration Challenges in Linux and eCos. In *VaMoS '12*. ACM, 149–155.

9. Tony Leclercq, Jean-Marc Davril, Maxime Cordy, and Patrick Heymans. 2016. Beyond De-Facto Standards for Designing Human-Computer Interactions in Configurators. In *EnCHIReS@EICS 2016, Bruxelles, Belgium*. 40–43.

10. Jakob Nielsen. 2005. Ten usability heuristics. (2005).

11. B.J. Pine and S. Davis. 1999. *Mass Customization: The New Frontier in Business Competition*. Harvard Business School Press.

12. Rick Rabiser, Paul Grünbacher, and Martin Lehofer. 2012. A Qualitative Study on User Guidance Capabilities in Product Configuration Tools. In *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering (ASE 2012)*. ACM, New York, NY, USA, 110–119. DOI: `http://dx.doi.org/10.1145/2351676.2351693`

13. RetailWeek. 2017. Personalisation. (2017). Retrieved July 24, 2017 from `https://www.retail-week.com/topics/technology/personalisation`

14. D. Sabin and R. Weigel. 1998. Product configuration frameworks-a survey. *IEEE Intelligent Systems and their Applications* 13, 4 (Jul 1998), 42–49. DOI: `http://dx.doi.org/10.1109/5254.708432`

15. Dominique L Scapin and JM Christian Bastien. 1997. Ergonomic criteria for evaluating the ergonomic quality of interactive systems. *Behaviour & information technology* 16, 4-5 (1997), 220–231.

16. Ben Shneiderman. 1997. *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (3rd ed.). Addison-Wesley, Boston, MA, USA.

17. C. Streichbier, P. Blazek, and F. Faltin. 2009. 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 *HICSS '09*. IEEE Computer Society, Washington, DC, USA, 1–7.

18. Jean Vanderdonckt. 1994. Guide ergonomique des interfaces homme-machine. *Presses Universitaires de Namur* (1994).

19. J. White, D. C. Schmidt, D. Benavides, P. Trinidad, and A. Ruiz-Cortés. 2008. Automated Diagnosis of Product-Line Configuration Errors in Feature Models. In *SPLC '08*. IEEE Computer Society, Washington, DC, USA, 225–234.

20. Yingfei Xiong, Arnaud Hubaux, Steven She, and Krzysztof Czarnecki. 2012. Generating Range Fixes for Software Configuration. In *ICSE '12*. IEEE Press, Piscataway, NJ, USA, 58–68.