A Collaborative-Based Recommender System for Configuration of Extended Product Lines

Juliana Alves Pereira
Otto-von-Guericke-University of Magdeburg, Germany
Email: juliana.alves-pereira@ovgu.de

***Abstract*—Product Line (PL) configuration practices have been employed by industries as a mass customization process. However, due to the NP-hard nature of the process, performance concerns start to be an issue when facing large-scale configuration spaces. The aim of my doctoral research is therefore to propose an efficient collaborative-based recommender system that provides accurate and scalable solutions to users. To demonstrate the efficiency of the proposed recommender system, I will conduct series of experiments on real-world extended PLs. In addition, I plan empirically to verify through a user case study the usability of the proposed approach. My expected contribution is to support the adoption of PL configuration practices in industrial scenarios.**

I. Introduction

Product Line (PL) has been introduced as a set of products that share a set of common features [3]. Kang et al. [6] introduce *features* as product functional requirements that are important to stakeholders. PL has proven to be an efficient and effective strategy for mass customization by exploiting large-scale reuse. However, although customization has been extensively studied over last past decades, it remains a source of concerns. In the context of PLs, additional constraints (emerging from large-scale variability spaces) add one more layer of complexity to the customization process, *a.k.a.* configuration process. In this circumstances, to assist users in dealing with the increased complexity and variability related concerns, an easy and comprehensive configuration process becomes crucial.

Previous researchers have proposed several *interactive* and *automatic* PL configuration approaches. Despite their maturity, the current process of configuration is still not able to fully use the power of customization when dealing with the variability of highly configurable PLs. On the one hand, the amount and complexity of options presented by *interactive* configurators may exceed the capability of a user to identify an appropriate configuration. On the other hand, *automatic* approaches have focused on techniques to derive configurations in a single step, not allowing users to *interact* with the configuration process. Furthermore, since those techniques can provide a set of feasible solutions, users may not know which one would be the better choice. Finally, improvements related to scalability and performance are still needed.

Based on the identified limitations, the envisioned contribution of my doctoral research is therefore to fill the gap in the literature by proposing a more efficient interactive configuration process for highly configurable PLs. To achieve that goal, I will adopt a collaborative-based recommender system that relies on past configurations from previous users to generate personalized recommendations for a current user. Moreover, this system follows providing visual support that allows users to focus on a reduced amount of information from valid and relevant parts of the configuration space. To this end, I aim at answering the following three research questions (*RQs*):

RQ1. (*Effectiveness* and *Performance*) Can a collaborative-based recommender system support configuration in realistic PL configuration scenarios? *RQ1.1* What is the impact of different recommender algorithms on the quality of recommendations? *RQ1.2* In which point of the configuration process can the recommender algorithms better support its users? *RQ1.3* Can the proposed recommender algorithms automatically support the complete PL configuration process in a reasonable computational time?

RQ2. (*Integration*) How can a state-of-the-art PL configurator tool be integrated with the proposed recommender system? *RQ2.1* What are necessary visualizations for the configurator tool while configuring products through the use of the proposed recommender system?

RQ3. (*Usability*) To what degree the proposed recommender system assist users related to state-of-the-art tools without a collaborative-based support?

To address *RQ1–RQ3*, I will investigate how efficient and effective the proposed recommender system is in supporting the *interactive* and *automatic* configuration process (*cf.* Sec. III-A).

II. State of the Art

On the *interactive* PL configuration scenario, users configure personalized products by consecutively selecting desired features based on their individual needs. In this context, there are many approaches[1] that aim to guide users into a valid configuration, ensuring that any partially configured product is in accordance with the PL constraints. However, when using those approaches, features of no importance to the stakeholders also need to be taken into account [9]. On this scenario, as most features are interdependent, users must understand the impact of their gradual selections in order to make valid decisions. This may lead to delays due to users' exploration of choices at each step of the configuration process. Thus, especially when dealing with large PLs with complex dependencies, additional support is needed to guide the users through the configuration

[1]For a survey on those approaches, I refer to my previous work [8].

process and allow them to focus on valid and relevant parts of the configuration space.

On the *automatic* PL configuration scenario, users specify product requirements and the configuration system provides feedback. *Automatic* approaches aim to deal with *Extended Product Lines* (EPLs). An EPL extends PL by adding *Non-Functional Properties* (NFPs) as feature attributes [3]. The specification of EPLs allows the verification of product' requirements. The requirements are verified by aggregating the NFPs of all selected features and feature interactions. I identify two groups of *automatic* configuration approaches: *feature-based* and *configuration-based* approaches. While *feature-based* approaches predict the utility of each feature for the users, *configuration-based* approaches predict the utility of an entire set of features, which forms a valid configuration.

Galindo et al. [5] propose a *configuration-based* approach, named Invar, which provides to the users a dynamic decision model with a set of questions and a defined set of possible answers. In a similar scenario, other authors [1][2][12] propose *product-based* and *feature-based* approaches where users are constantly asked to compare a pair of features and identify their relevance in terms of satisfying given NFPs. However, while question-based decision may often introduce some vague descriptions and even misleading information to questionnaires, pair-wise based decision may introduce inconsistencies in the feature ranking. Moreover, if the features, or even the set of answers, are of equal (or no) interest to the user, no support is provided to guide the selection process. Furthermore, as one feature may contribute to many NFPs, the amount and complexity of information presented to users can be overwhelming to them identify an appropriate choice.

Further *configuration-based* approaches have used constraint programming and evolutionary algorithms to automatically derive a configuration in a single step[2]. Although constraint programming approaches guarantee the optimality of the generated configuration, due to the NP-hard computational complexity of finding an optimal variant, exact approaches have inefficient exponential time. On this scenario, evolutionary algorithms have been deeper studied in order to manage large configuration spaces, deriving near optimal solutions in an efficient polynomial time. However, when using such approaches, the users are not able to interact with the configuration process. Moreover, the specification of multiple requirements may lead to conflicting solutions. In this context, the current approaches neither guide the users on choosing a suitable solution nor offer further support for specification of stakeholders' preferences.

In general, the definition of product requirements differs depending on the chosen configuration approach. Through a systematic literature review, I recognize three main types of product' requirements: *resource constraints*, *stakeholders' preferences*, and *optimization objective*.

Resource constraints. Definition of decision rules with regards to product limitations and stakeholders' needs.

[2]For a detailed review of those approaches, I refer to my work [7].

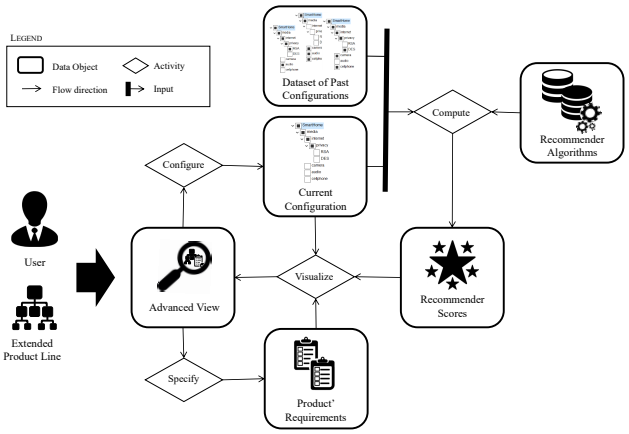

Figure 1. An overview of the configuration process.

Stakeholders' preferences. Specification of NFPs relative importance for stakeholders.

Optimization objectives. Definition of maximization or (and) minimization criteria over NFP values.

While the most of constraint programming and evolutionary algorithms consider *optimization objective(s)* and *resource constraints* as product requirements, the use of dynamic decision models allows the specification of *stakeholders' preferences*. Thus, an approach that offers a complete support for those requirements is still missing in the literature and need to be further explored [7].

III. The Proposed Approach

Personalized recommender systems have proved to be an appropriate mean to assist users in finding information and making decisions. I therefore propose a collaborative-based personalized recommender system for guiding a user through the PL configuration process by directing the order of selecting features and predicting which of them are more useful. The workflow in Figure 1 illustrates this process.

The process starts by specifying product' requirements through an *advanced view*. Based on those requirements, visualization mechanisms[3] are employed to deal with the information overload resulting from current approaches. The visualization mechanisms narrow the configuration space of possible feature selections down to the relevant ones by explicitly excluding features if not required in a specific context and ranking features such that the most relevant ones are easily accessible. Moreover, to provide an optimized guidance for the user, relevant features are also *scored* with respect to their relevance by the *recommender algorithms*. Features relevance are predicted from users' implicit requirements and gradual feature selections (*i.e. current configuration*), based on EPL constraints and explicit content from a dataset of *past configurations*. Then, each time the user configure a particular feature, decision propagation strategies are applied to automatically validate the *current configuration* (*cf.* [9] for

[3]For a preliminary implementation of those mechanisms, I refer to [9].

more details) and the predictions are updated. To compute the predictions, extra information related to each past configuration are also considered, such as selection order of features, customers' market domain, features' frequency of use, features' last known defects, and others. Moreover, I also plan to consider features' creation date to be capable of making predictions for new features through the identification of similarities over NFPs. Consequently, a user is always provided with a limited set of permitted, necessary and relevant choices.

The recommender system creates an *interactive* perspective for users in order to learn their preferences and provide new recommendations. Moreover, the proposed system can be successfully *automated* to support large-scale PLs. Finally, through the use of this system, configuration update and upgrade can also be supported from a *current configuration*.

A. Research Methodology

In this section, I outline how I will investigate and evaluate the research questions presented in Sec. I.

RQ1. Since there is no single recommender algorithm that performs the best in all applications, *RQ1.1* follows analyzing the accuracy results from different state-of-the-art collaborative-based recommender algorithms (*i.e. neighbourhood-based Collaborative Filtering (CF)*, *CF-Hoeffding*, *CF-shrinkage*, *CF-significance weighting*, *average similarity*, *matrix factorization*, and *tensor factorization*) to realistic PL configuration scenarios. Accuracy evaluates how well the proposed algorithms are capable of understanding the preferences of the users and giving recommendations that are in accordance with the decisions that the users actually have in mind. To analyze the accuracy, I plan to perform an offline evaluation splitting the dataset into training and test through the use of the leave-one-out evaluation protocol. According to this protocol, one configuration is left out from the training set and used for testing. The remaining ones are given to the algorithm as training data. This simulates the behavior of a real system, where a user logs in and carries out a new configuration. The data of the past configurations are available to the system and only the new configuration should be predicted. To perform well, a recommender system has to recommend the features that were used in the left-out test configuration (*cf.* [10] for more detail).

RQ1.2 follows investigating if useful feature recommendations can be performed with a reasonable amount of selected features. By answering this question, I will gain a deeper understanding of the *automation level* of the proposed algorithms. Finally, since recommender algorithms are frequently intended to work on very large datasets, the performance of the recommender is essential. Thus, *RQ1.3* follows analyzing the impact of the proposed algorithms on configuration performance.

In summary, the main purpose of *RQ1* is to evaluate which algorithms are more effective and efficient in producing a PL configuration. My assumption is that the combination of various recommender algorithms may outperform single methods in terms of accuracy and performance.

RQ2. Although there are several *interactive* configurator tools, they do not implement further configuration support to answer industry needs (*cf.* Sec. II). Thus, based on a preliminary survey of configurator tools [8] and two empirical user studies [4][11], I decided to follow extending a state-of-the-art tool FeatureIDE [13] with the proposed recommender system. FeatureIDE is an open-source Eclipse-based tool which widely covers all phases of software PL development. Besides being integrated with several programming and composition languages, FeatureIDE provides the key functionality of typical tools, such as PL editor, analyzer, and configurator. Moreover, FeatureIDE is actively used by industry practitioners and academic researchers. By integrating the proposed recommender system with FeatureIDE, I can provide to users a most efficient and complete environment.

Thus, *RQ2.1* follows investigating which would be necessary visualizations for the configurator tool by analyzing two main questions from an empirical user study: Which elements of the configuration process should be visualized? How to visualize those elements?

On the *interactive* support scenario, I aim at proposing a visualization mechanism that shows to the user the impact, in terms of PL constraints and NFPs, of each relevant feature over a set of target product requirements, and functional and non-functional dependencies. Moreover, relevant features will be scored (*e.g.* five-stars classification) and ranked (*e.g.* top-10 features) according to their importance for the user. This is particularly helpful to users when it is not clear either which feature selection fulfills their requirements better, or which feature selection defines a valid configuration.

On the *automatic* support scenario, the user will be able to automatically configure a complete product in any point of the configuration process. Consequently, the algorithm selects the maximum number of features with higher predictions by the recommender system that are in accordance with the PL constraints and product requirements. This is especially applicable when configuring large-scale PLs in which the interactive configuration is recognized to be a time-consuming and tedious task.

RQ3. I plan to conduct an empirical user study of the proposed approach by comparing the extended tool with existing tools in the literature, such as [1][2][5][12]. I aim at investigating the improvements in terms of *efficiency*, *efficacy*, *acceptance* and *usability* of the proposed tool. I expect that the use of a collaborative-based recommender system: *(a)* makes the completion of the process easier and reduce the time taken to complete the configuration process, *(b)* enhances the desirability of the end product, and *(c)* reduce the mental burden to a more manageable level. To this end, given a target product specified through a requirement document, I plan to verify assumption *(a)* by investigating whether the recommendations are adopted, measuring the number of configuration changes, and the time spent to configure the product (for configurations that satisfy more then 80% of the specified requirements). Regarding assumption *(b)*, I will analyze if the recommended

features from the proposed approaches meet the target product requirements. Finally, assumption *(c)* investigates whether the recommender system combined with visualization mechanisms reduce considerably the amount of information presented to the user in each step of the interactive configuration process.

In particular, I am interested in showing that the proposed integrated environment compare favorably with existing ones in terms of prediction accuracy and performance, providing to the user a most efficient PL configuration process.

IV. Work Progress

My PhD research started in January 2015, since then I published six and submitted two papers that are closely related to my topic. First, I conducted a systematic literature review on the PL configuration domain to increase the understanding of the fundamental research issues in this field. A journal version of this study is under development. Second, to overcome one of my findings, I introduced the adoption of a collaborative-based personalized recommender system for non-EPLs configuration [10]. I presented an initial formalization of three recommender algorithms: *neighbourhood-based CF*, *average similarity*, and *matrix factorization*. The algorithms were validated against two real-world software PLs (i.e. ERP System and E-Agribusiness), currently the largest available datasets of PL configuration already cited in the literature. While the ERP System dataset provides 1.653 features and 171 past configurations in the business management domain, the E-Agribusiness dataset provides 2.008 features and 5.749 past configurations in the e-commerce agribusiness domain. Both PLs have a very high degree of variability which would be hard for a user to go through without any additional support.

To draw conclusions from the algorithms' effectiveness, I compare them with a baseline random recommender algorithm[4]. The experiments show that two of the three proposed recommendation algorithms (*i.e. neighbourhood-based collaborative filter* and *matrix factorization*) clearly and consistently outperform the baseline recommender in finding relevant features. Moreover, it has a good performance already at the initial stage of the configuration process (*i.e.* with just 10% of selected features). For a detailed discussion of the results, I refer the reader to [10]. An extended journal version of this paper considering three more algorithms (i.e. *CF-Hoeffding*, *CF-shrinkage*, and *CF-significance weighting*) is under development.

Recently, I extended FeatureIDE with the Matrix Factorization recommender algorithm [10] and some visualization mechanisms [9]. As a next step, I will extend the proposed approach to consider others well-established state-of-the-art recommender algorithms and the notation of EPLs. By extending a PL, I aim to work with the three types of recognized product' requirements (*cf.* Sec. II). In this context, I also plan to investigate how requirements specification can influence the accuracy and performance results from proposed algorithms. Finally, after extending FeatureIDE with the most efficient recommender system, I will perform an empirical user evaluation of the tool.

[4]The efficiency of this algorithm is equivalent to the efficiency of a hypothetical, fully uninformed user without any support from a recommender.

V. Conclusion

In my doctoral research, I target an open research question in the PL configuration domain: *Can a collaborative-based recommender system efficiently predict a suitable set of features from an EPL based on implicit and (or) explicit information from users?* I aim to answer this question by providing a set of *interactive* and *automatic* visual mechanisms that are intended to support users selecting among a vast number of features, avoiding useless and invalid decisions. In contrast to the current literature, the proposed approach benefits from a simplified view of the configuration space by dynamically predicting the importance of the features. Moreover, this is the first approach that uses a collaborative-based recommender system that learns about the relevant features from past configurations. I expect that the proposed approach provides to users an easy-to-understand PL configuration process where the effort of decision-making is saved.

Acknowledgment

I would like to thank Gunter Saake and Myra Spiliopoulou for advising me on this thesis topic. Moreover, I gratefully acknowledge the financial support of the Brazilian National Council for Scientific and Technological Development (CNPq) grant 202368/2014-9.

References

[1] M. Asadi, S. Soltani, D. Gasevic, M. Hatala, and E. Bagheri. Toward Automated Feature Model Configuration with Optimizing Non-Functional Requirements. *Inform. and Software Technology*, 56(9):1144–1165, 2014.

[2] E. Bagheri and F. Ensan. Dynamic Decision Models for Staged Software Product Line Configuration. *Req. Engineering*, 19(2):187–212, 2014.

[3] D. Benavides, S. Segura, and A. Ruiz-Cortés. Automated Analysis of Feature Models 20 Years Later: A Literature Review. *Information Systems*, 35(6):615–708, 2010.

[4] K. Constantino, J. A. Pereira, J. Padilha, P. Vasconcelos, and E. Figueiredo. In *ENASE*, pp. 164–171.

[5] J. A. Galindo, D. Dhungana, R. Rabiser, D. Benavides, G. Botterweck, and P. Grünbacher. Supporting Distributed Product Configuration by Integrating Heterogeneous Variability Modeling Approaches. *Information and Software Technology*, 62:78–100, 2015.

[6] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, 1990.

[7] L. Ochoa, J. A. Pereira, O. González-Rojas, H. Castro, and G. Saake. A Survey on Scalability and Performance Concerns in Extended Product Lines Configuration. In *VaMoS*, pp. 5–12. ACM, 2017.

[8] J. A. Pereira, K. Constantino, and E. Figueiredo. A Systematic Literature Review of Software Product Line Management Tools. In *ICSR*, pp. 73–89. Springer, 2015.

[9] J. A. Pereira, S. Krieter, J. Meinicke, R. Schröter, G. Saake, and T. Leich. FeatureIDE: Scalable Product Configuration of Variable Systems. In *ICSR*, pp. 397–401. Springer, 2016.

[10] J. A. Pereira, P. Matuszyk, S. Krieter, M. Spiliopoulou, and G. Saake. A Feature-Based Personalized Recommender System for Product-Line Configuration. In *GPCE*, pp. 120–131. ACM, 2016.

[11] J. A. Pereira, C. Souza, E. Figueiredo, R. Abilio, G. Vale, and H. A. X. Costa. Software Variability Management: An Exploratory Study with Two Feature Modeling Tools. In *SBCARS*, pp. 20–29. IEEE, 2013.

[12] L. Tan, Y. Lin, and L. Liu. Quality Ranking of Features in Software Product Line Engineering. In *APSEC*, volume 2, pp. 57–62. IEEE, 2014.

[13] T. Thüm, C. Kästner, F. Benduhn, J. Meinicke, G. Saake, and T. Leich. FeatureIDE: An Extensible Framework for Feature-Oriented Software Development. *SCP*, 79(0):70–85, 2014.