Runtime Collaborative-Based Configuration of Software Product Lines

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

***Abstract*—Software Product Line (SPL) configuration practices have been employed by industries as a mass customization process. However, the inherent variability of large SPLs leads to configuration spaces of exponential sizes. Thus, scalability and performance concerns start to be an issue when facing runtime environments, since it is usually infeasible to explore the entire configuration space exhaustively. In this context, the aim of my research is therefore to propose an efficient collaborative-based runtime approach that relies on recommender techniques to provide accurate and scalable configurations to users. To demonstrate the efficiency of the proposed approach, I conduct series of experiments on real-world SPLs. In addition, I plan empirically verify through a user case study the usability of the proposed approach. My expected contribution is to support the adoption of SPL configuration practices in industrial scenarios.**

***Keywords*-software product lines; configuration; recommender systems; collaborative-based recommendations**

I. Research Problem and Motivation

SPLs provide configuration options to adjust a software to different platforms and stakeholders' needs. Previous works have proposed several *interactive* and *automatic* SPL configuration approaches. However, the current configuration process is still limited when dealing with the variability of highly configurable systems. 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 process. Moreover, since those techniques can provide a set of feasible solutions, users may not know which one would be the better choice. Furthermore, due to the NP-hard nature of the process, improvements related to scalability and performance are still needed.

Based on the identified limitations, the envisioned contribution of my research is therefore to fill the gap in the literature by proposing a polynomial-time configuration approach for runtime configurable systems. To achieve that goal, I adopt a collaborative-based recommender system that relies on configurations from a set of historical users to generate runtime personalized configurations to a current user.

To illustrate this scenario, consider the example of Facebook, an online social media. Facebook has hundreds of features and relationships, leading to an exponential size configuration space. In this context, the proposed approach aims to automatically adjust the software features based on *external* and *internal* system settings (*i.e.* implicit information). *External* system settings are product resource constraints that limits the environment context, such as *network access*. *Internal* system settings are preferences options that allow users to customize their Facebook environment, such as *privacy options*. The set of *internal* and *external* settings is known as product' requirements. Besides the product requirements and SPL constraints, the proposed approach relies on explicit information from other users. For instance, *friends* and *apps*, to suggest new Facebook features to current users. To achieve this goal, I developed the prototype PLUS[1] (*Product Line configUration System*). The prototype is currently able to suggest relevant features to users in a matter of milliseconds. In addition, It has a good performance already at the initial configuration stage. The contribution of my work demonstrates the effectiveness of collaborative-based recommender techniques to support runtime SPL configuration.

II. State of the Art

On the *interactive* SPL configuration scenario, users configure personalized products by consecutively selecting desired features based on their individual needs and SPL constraints. In this context, there are many approaches[2] that aim to guide users into a valid configuration, ensuring that any partially configured product is in accordance with the SPL constraints. However, when using those approaches, features of no importance to the stakeholders also need to be taken into account [7]. 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 SPLs with complex dependencies, additional support is needed to guide the users through the configuration process and allow them to focus on valid and relevant parts of the configuration space.

On the *semi-automatic* SPL configuration scenario, Galindo et al. [4] propose a dynamic decision model with a set of questions and a defined set of possible answers. In a similar scenario, other authors [1][2][10] propose a pair-wise based decision approach where users are constantly asked to compare a pair of features and identify their relevance in terms of satisfying given non-functional requirements. However, while question-based decision may often introduce some vague descriptions and even misleading information to questionnaires,

[1]http://wwwiti.cs.uni-magdeburg.de/∼jualves/PLUS/

[2]For a survey on those approaches, I refer to my previous work [6].

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 non-functional requirements, the amount and complexity of information presented to users can be overwhelming to them identify an appropriate choice.

On the *automatic* SPL configuration scenario, approaches[3] have used constraint programming and evolutionary algorithms to automatically derive a configuration in a single step that is in accordance with users product requirements. 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 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.

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 to guide users through the SPL configuration process by directing the order of selecting features and predicting which of them are more useful.

The process starts by analyzing implicit information through an *advanced view*. Based on those information, visualization mechanisms narrow the configuration space of possible features down to the relevant ones by explicitly excluding features if not required in a specific context (*i.e.* they do not satisfy the SPL constraints and product' requirements) and ranking features such that the most relevant ones are easily accessible. Features relevance are predicted from product requirements, SPL constraints and explicit content from other users (*i.e.* users similarity, selection order of features, features' frequency of use, and others). Moreover, I also consider *features' creation date* to be capable of making predictions for new features through the identification of similarities between the new feature and existing ones. Then, each time the user configure a particular feature, decision propagation strategies are applied to automatically validate the *current configuration* (*cf.* [7] for more details) and the predictions are updated. Consequently, a user is always provided with a limited set of permitted, necessary and relevant choices. According to our preliminary results, the proposed system can be successfully *automated* to learn users preferences and provide runtime configurations.

IV. Results

My research involved four phases. First, I conducted a systematic literature review on the SPL configuration domain to increase the understanding of the fundamental research issues in this field. Second, to overcome one of the SPL configuration issues found in the literature, I introduced the adoption of a collaborative-based personalized recommender system for SPL configuration. I presented an initial formalization of six recommender algorithms: *Neighbourhood-Based Collaborative Filtering (CF)*, *CF-Hoeffding*, *CF-Shrinkage*, *CF-Significance Weighting*, *Average Similarity*, and *Matrix Factorization*. Third, the algorithms were validated against two real-world SPLs, currently the largest available datasets of SPL configuration already cited in the literature [8]. To draw conclusions from the algorithms' effectiveness, I compare them with a baseline random recommender algorithm. I use 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. The data of the historical 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 based on the training data. The preliminary experiments show that three of the six proposed recommendation algorithms (*i.e. CF-Shrinkage*, *CF-Significance Weighting*, 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).

Fourth, as a result of previously empirical studies [3], [9], I developed a prototype with those recommender algorithms and some visualization mechanisms [7], [8]. As a next step, I will extend the proposed approach to consider others well-established state-of-the-art recommender algorithms. Moreover, I also pretend to use other real-world datasets of configurations to validate the proposed approach. Furthermore, I plan to investigate how different product requirements specification can influence the accuracy and performance results from proposed algorithms. Finally, I will perform an empirical user evaluation of the proposed approach.

V. Conclusion

In my research, I target an open research question in the SPL configuration domain: *Can a collaborative-based recommender system efficiently predict a suitable set of features from an SPL 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 a set of features according to product' requirements, 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 other users' configurations. I expect that the proposed approach provides to users an easy-to-understand SPL configuration process where the effort of decision-making is saved.

[3]For a survey on those approaches, I refer to my previous work [5].

Acknowledgment

I would like to thank Pawel Matuszyk and Sebastian Krieter for their collaboration and numerous fruitful discussions. 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] K. Constantino, J. A. Pereira, J. Padilha, P. Vasconcelos, and E. Figueiredo. An Empirical Study of Two Software Product Line Tools. In *Proceedings of the International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*, pp. 164–171, 2016.

[4] 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.

[5] 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 *Proceedings of the Workshop on Variability Modelling of Software-intensive Systems (VaMoS)*, pp. 5–12. ACM, 2017.

[6] J. A. Pereira, K. Constantino, and E. Figueiredo. A Systematic Literature Review of Software Product Line Management Tools. In *Proceedings of the International Conference on Software Reuse (ICSR)*, pp. 73–89. Springer, 2015.

[7] J. A. Pereira, S. Krieter, J. Meinicke, R. Schröter, G. Saake, and T. Leich. FeatureIDE: Scalable Product Configuration of Variable Systems. In *Proceedings of the International Conference on Software Reuse (ICSR)*, pp. 397–401. Springer, 2016.

[8] J. A. Pereira, P. Matuszyk, S. Krieter, M. Spiliopoulou, and G. Saake. A Feature-Based Personalized Recommender System for Product-Line Configuration. In *Proceedings of the International Conference on Generative Programming and Component Engineering (GPCE)*, pp. 120–131. ACM, 2016.

[9] 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 *Proceedings of the Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS)*, pp. 20–29. IEEE, 2013.

[10] L. Tan, Y. Lin, and L. Liu. Quality Ranking of Features in Software Product Line Engineering. In *Proceedings of the Asia-Pacific Software Engineering Conference (APSEC)*, volume 2, pp. 57–62. IEEE, 2014.