A Unified Vision of Configurable Software

Houssem CHEMINGUI [1,2], Inès GAM [1,2], Raúl MAZO [3,4], Henda BEN GHEZALA [2], Camille SALINESI [1]

1. CRI Laboratory, Paris 1 Panthéon Sorbonne University, 90 rue de Tolbiac, 75013 Paris - France
2. RIADI Laboratory, ENSI Manouba University, 2010 Manouba - Tunisia
3. Lab-STICC, ENSTA Bretagne, 2 rue François Verny, Brest - France
4. GIDITIC, Universidad Eafit, Carrera 49 N° 7 Sur-50, Medellin - Colombia

RÉSUMÉ. En pratique, la configuration logicielle est une tâche difficile et sujette à erreurs en raison du grand nombre d'exigences et de contraintes à satisfaire simultanément. Les parties prenantes se trouvent confrontés à des problèmes de rigueur et de passage à l'échelle lors de la configuration de logiciels: les méthodes employées pour spécifier les systèmes à configurer ne permettent pas de maîtriser de manière formelle et systématique un nombre important de décisions complexes. Cependant, des approches visant à surmonter ces verrous existent et ont été publiées, mais dans certains domaines; peuvent-elles être adaptées pour toute sorte de logiciels configurables? Les défis scientifiques de la configuration logicielle peuvent ils ainsi être transposés? Cet article aborde ces questions à travers un cadre unificateur de configuration identique et adaptée à différents cas d'utilisation et contextes.

ABSTRACT. In practice, software configuration is error-prone due to the plethora of requirements and constraints to satisfy at the same time. Practitioners face awkward scalability issues when configuring large variability-based software. Indeed, standard variability modeling methods such as feature and even decision models fail in mastering a suitable configuration process implying a huge panel of complex decisions. However, published solutions, aiming to overcome these obstacles, exist but they have been designed in separate ways; can they be adapted for all sorts of configurable software? Can the scientific challenges of software configuration be transposed? This paper addresses these issues through a unified framework of configuration encompassing different use cases and contexts.

Mots-clés : Configuration logicielle, passage à l'échelle, défis scientifiques, cadre unificateur

KEYWORDS: Software configuration, scalability issues, scientific challenges, unified framework

DOI:10.3166/RCMA.25.1-n © 2020 Lavoisier AR_DOI

1. Introduction

Developing configurable software such as ERP, SPL or COTS aims to boost productivity and thus maximize revenues. However, uncareful management of the configuration process (Mittal et al., 1989) creates serious vulnerability and inconsistency problems in the software. The huge number of decisions and incomprehensible interactions between them present a major problem. Unfortunately, this problem is common for all configurable software systems even worse in large scale variability based software (Chemingui et al., 2019).

Currently, a panoply of research solutions aiming to improve the configuration process exists in literature, nevertheless, these solutions are designed in separate ways. For instance, solutions that master SPL configuration fail to be adapted to master COTS configuration. Therefore, thinking about a systematically extending and adaptation of solutions to other kinds of configurable systems of software leads this research to the following research questions:

RQ1: What are the open research challenges of unified software configuration?

RQ2: What are the common configuration assets of configurable systems?

This paper advocates a unified vision that can be formalized through a conceptual framework considering all sorts of configurable software in a consistent way. The motivation of this unified vision is to address common configuration issues and challenges with an identical series of methods, techniques and tools. The paper recognizes that in practice software configuration occurs in specific contexts, with different problems, goals, applications and use cases.

2. Challenges of Software Configuration

Considering all sorts of configurable software conjointly, leads this research to be focusing on topical and thought-provoking challenges related to design solutions and scientific research potentials.

2.1. Design solutions and industry need

In practice, configuration process models fail to scale up to the huge panel of variants that are encountered in real life software configuration settings. For instance, it becomes impractical to make an extensive list of all possible decision processes leading to final configurations of an ERP. Even worse, it is quickly impractical to match these decisions with the actual collection of requirements of the future users. At the level of an enterprise, collections can be large and up to several thousands. In fact, major issues of configurable software are related to scalability enforcement and control. Besides, stakeholders usually misunderstand the configuration variants, their semantics and their independencies. This unfortunately common issue in configurable software can be formalized as the mismatch. In the case of COTS, this mismatch in form, topic, content, and level of abstraction between the requirements causes complex matching problems (Zoukar et al., 2004). Failing to properly match users' and COTS requirements results is failing to meet organizational expectations, to user dissatisfaction, and in the end to high risks of project failure. In practice, COTS implementation guides hardly provide any support to guide the design of a suitable solution that takes this issue into account in an equilibrated way. The solutions that are proposed to guide the configuration process are most often driven by the solution; other approaches such as decision models, and goal driven approaches are still subject of research. A compendium about scalability issues of configurable systems are discussed in our previous works (Chemingui et al., 2019). In other works, we showed that it is also possible to address scalability issues in a similar way to the approaches used for SPL (Software Product Lines) configuration (Mazo et al., 2014).

In this respect, configurable systems need a deeper focus on their design and user understanding, not solely to know who they are, but to dive deeper into their motivations and their exhibited behaviors. A matching between user requirements, context information and the system model seems mandatory. As far as we know, there are no efficient software resources that blend this matching with the different configuration areas. It seems interesting to think about a software standard with a universal dimension that is able to overcome scalability and flexibility problems. Independently of its nature, this solution can be a programming language, a design notation or a complete method that is able to support broaden system contexts and particular user properties.

According to our previous experiences with industrials in automotive (Dumitrescu et al., 2013), electronics (Triki et al., 2015) and health (Djebbi et al., 2007) domains, we suggest that expected guidance solutions should consider design fuzziness to meet the user satisfaction policies. For example, what principles should be taken into account throughout the design? How configuration information and constraints can be better presented? What users need to a better understanding of complex configuration alternatives? Regardless of the size of variants, design has to support easy but not abstract data allowing to infer rapidly the plethora of configurations when high constraints occur. On the one hand, a focus on the delivery time will have a significant impact in the configuration management. Moreover, design has to provide simplified mechanisms to reveal configuration use cases and scenarios. It is highly recommended to follow domain processes, divide tasks and create views automatically from a big range of artefacts and interests. On the other hand, the solution supporting tool will provide simple ways to express variants and their interdependencies allowing an easier process of configuration. For instance, incorporating Human Machine Interface (HMI) solutions such as dynamic forms and 3D views increases the user comfort and decreases the configuration complexity.

2.2. Scientific Research Potentials

From an organizational point of view, a collaborative research focus may conduct the investigation of cited challenges in real environments. Methods such as Participatory action research (Chevalier et al., 2019) and ethnography Research (Holland et al., 2004) are recommended to achieve a better understanding of configuration design and potential users interests. Too often, research involvement allows to motivate real technology underpinnings and its usefulness to overcome scalability problems in this case. In fact, there is a significant gap between what industry needs and what academic research focuses on. Usually, the interaction between industrial actors and academics faces several barriers since companies keep secret their strategic activities. Consequently, the fear of breaching confidentiality impacts significantly the research valorization. The lack of collaboration may cause trustworthiness shortcomings making it more complex to valorize research outcomes and relate best practices. As a result, proposed methods and tools coming from academic research are making the same mistakes with unforeseen incompatibilities.

3. Common considerations of Software Configuration

At first sight, suitable solutions for all configurable software need to (i) provide exact information about product constituents and their interactions; (ii) meet non-functional properties such as performance, ease of use and robustness, (iii) forcecast information even about next constituents to configure.

Our expertise with very large variability systems such as Electronic Parking Brake Systems, the Automatic Lighting System, or the French railways operated by SPLs, COTS and ERPs, showed that design methods are still suffering from scalability. The most widely used variability modeling methods were *feature models* (Kang et al., 1990) *use cases* (Jacobson et al., 1993), *decision models* (Quinlan, 1990), *goal models* (Barron et al., 2001) and *constraint programming* (Rossi et al., 2006). In front of scalability problems, resolution mechanisms including alternative structures of the model and estimation of the configuration errors were proposed to conduct a significant software reconfiguration (Giese et al., 2006). The configuration process can be streamlined by a series of interfaces aiming to a prealable preparation of a simple process, but a large number of variants still impossible to manage. Adding to that, software approaches such as feature toggles (Schermann et al., 2018) and versioning (Sigal et al., 1999) were widely adopted by developers to control combinatorial explosions. The main usage of these approaches is to avoid conflict that can arise when merging changes in software. Although this also can lead to technical debts that arise due to constraints violation after a feature has been switched on/off. Frequently, constraint violations disturb other parts of the system.

However, the literature reveals a prominent focus on configuration considerations but in different ways that are specific to each software kind, while in fact the deep problems are similar in nature. The idea put forward is to focus on the common configuration considerations, think about a unified conceptual framework propeling potential generic solutions. It seems interesting to emphasize, that independently of the variability modeling method and the kind of software you are dealing with, there are common considerations of configuration. The landscape to sketch aims to substantiate that software configuration handles the same assets independently of variability modeling methods. In the aforementioned variability methods, *composition, and order* present recurrent activities' patterns during a configuration process. Moreover, one of the key principles of the configuration processes is to guarantee that all *domain constraints* are verified. In fact, variability constraints can have cascading effects during configuration processes, and they are the source of complex problems that raise exponentially with the number of optional assets in all variability based systems. For example, a SPL Eshop can be composed of several functionalities such as a search menu, a payment system and a security mode. Purchases can be paid through a credit card and/or a bank transfer. Security mode can be High or Standard. Selecting the credit card system requires a high security mode. In instance, configuring the security mode before selecting the payment system is error prone and leads to undo choices. Consequently, following a configuration order that respects the constraint domains is necessary. In fact, when the number of the eshop functionalities increases, the number of potential eshops increases and user decisions to be made increases also.

4. Conclusions

One of the challenging problems facing the software configuration community today is how to unambiguously define solutions for scalability in such a way to achieve consistency, flexibility and preference considerations. Furthermore, how to define solutions in a rich, generic and unified form allowing to consider all sorts of configurable software by combining advantages and avoiding drawbacks? All knowledge acquired in the different fields of configuration enhancement must be extended, adapted, and shared in a transparent manner. Therefore, there is a clear need to ease the cooperation of scientifics and industrials by creating a common ground of concepts and knowledge sharing. To meet these challenges, an efficient research valorization is needed to meaningfully conduct a collaborative enhancement of the configuration process **[RQ1]**. With the wisdom of hindsight, this vision paper assumes that all sorts of configurable software can be analyzed and improved in a unified way. Generally, configurable software aims to reach valid products implying common configuration assets to deal with namely composition, order and domain constraints and obviously other particular assets such as subtyping and cardinalities **[RQ2]**.

This research is under deep technical experiments that are supported by the PHC UTIQUE project N° 16G/1416 (called CONFIGURE) and the european project N° 15010 (called REVaMP^2).

References

Astesana, J. M., Cosserat, L., & Fargier, H. (2010, October). Constraint-based vehicle configuration: A case study. In 2010 22nd IEEE International Conference on Tools with Artificial Intelligence (Vol. 1, pp. 68-75). IEEE.

Barron, K. E., & Harackiewicz, J. M. (2001). Achievement goals and optimal motivation: testing multiple goal models. Journal of personality and social psychology, 80(5), 706.

Chemingui, H., Gam, I., Mazo, R., Salinesi, C., & Ghezala, H. (2019). Product Line Configuration Meets Process Mining. In Proceeding of the 11th International Conference on ENTERprise Information Systems (pp. 199-210)

Chevalier, J. M., & Buckles, D. J. (2019). Participatory action research: Theory and methods for engaged inquiry. Routledge.

Djebbi, O., Salinesi, C., & Diaz, D. (2007) Product Line Requirements Matching and Deriving: the RED-PL Guidance Approach.

Dumitrescu, C., Mazo, R., Salinesi, C., & Dauron, A. (2013, August). Bridging the gap between product lines and systems engineering: an experience in variability management for automotive model based systems engineering. In Proceedings of the 17th International Software Product Line Conference (pp. 254-263). ACM.

Giese, H., & Tichy, M. (2006, September). Component-based hazard analysis: Optimal designs, product lines, and online-reconfiguration. In International Conference on Computer Safety, Reliability, and Security (pp. 156-169). Springer, Berlin, Heidelberg.

Holland, D., & Leander, K. (2004). Ethnographic studies of positioning and subjectivity: An introduction. Ethos, 32(2), 127-139.

Jacobson, I. (1993). Object-oriented software engineering: a use case driven approach. Pearson Education India.

Kang, K., Cohen, S., Hess, J., Novak, W., Peterson, S., 1990. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR21, SEI, Carnegie Mellon University

Mazo, R., Assar, S., Salinesi, C., & Hassen, N. B. (2014). Using Software Product Line to improve ERP Engineering: literature review and analysis. Latin American Journal of Computing Faculty of Systems Engineering Escuela Politécnica Quito-Ecuador, 1(1), 10.

Mittal, S., & Frayman, F. (1989, August). Towards a Generic Model of Configuration Tasks. In IJCAI (Vol. 89, pp. 1395-1401).

Quinlan, J. R. (1990). Decision trees and decision-making. IEEE Transactions on Systems, Man, and Cybernetics, 20(2), 339-346.

Rossi, F., Van Beek, P., & Walsh, T.. (2006). Handbook of constraint programming. Elsevier.

Schermann, G., Cito, J., & Leitner, P. (2018). Continuous experimentation: challenges, implementation techniques, and current research. Ieee Software, 35(2), 26-31.

Sigal, A. D., Bien, D., & Pissarra, A. (1999). U.S. Patent No. 5,881,292. Washington, DC: U.S. Patent and Trademark Office.

Triki, R., Salinesi, C., & Mazo, R. (2015, May). Three strategies to specify multi-instantiation in product lines. In 2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS) (pp. 211-216). IEEE.

Zoukar, I., & Salinesi, C. (2004, April). Matching ERP Functionalities with the Logistic Requirements of French Railways: A Similarity Approach. In ICEIS (3) (pp. 444-450).