Preference-based Conflict Resolution for Collaborative Configuration of Product Lines

Sabrine Edded, Sihem Ben Sassi, Raúl Mazo, Camille Salinesi, Henda Ben Ghézala

Preference-based Conflict Resolution for Collaborative Configuration of Product Lines

Sabrine Edded[1,2], Sihem Ben Sassi[2,3], Raul Mazo[1,4], Camille Salinesi[1], and Henda Ben Ghezala[2]
[1]*Centre de recherche en Informatique (CRI), Université Panthéon Sorbonne, Paris, France*
[2]*RIADI Lab., National School of Computer Sciences, Manouba University, Tunisia*
[3]*MIS Department - CBA, University of Taibah, P.O. Box 344, Al-Madinah, KSA*
[4]*Lab-STICC, ENSTA Bretagne, Brest, France*
{*Sabrineedded@gmail.com, Sihem.BenSassi@gmail.com, Raul.Mazo@univ-paris1.fr, Camille.Salinesi@univ-paris1.fr, Henda.Benghezala@ensi.rnu.tn*

Keywords: Product Lines, Collaborative Configuration, Conflict Resolution, Minimal Correction Subsets.

Abstract: In the context of Product lines, the collaborative configuration process gets complicated when the configuration decisions of involved stakeholders are contradictory, which may lead to conflicting situations. Although considerable research has been devoted to collaborative configuration, little attention has been paid to conflict resolution. Moreover, most of existing approaches rely on a systematic process which constraint decisions of some stakeholders. In this paper, we propose a new collaborative configuration approach which allows conflict resolution based on stakeholders preferences expressed through a set of substitution rules. Based on such preferences, we delete the minimal set of conflicting configuration decisions which are identified using the Minimal Correction Subsets (MCSs) computing algorithm. An illustrating example and a tool prototype are presented to evaluate the applicability of our approach.

1 INTRODUCTION

The configuration process is a crucial step in Software Product Line Engineering (SPLE) that refers to feature selection from a product line model (Clements and Northrop, 2001). The larger the product line model, the more configurations can be created. On industrial scale product line models, it cannot be expected that configuration activities are handled by single user solely responsible of all configuration decisions (Mendonca et al., 2008). This is not only because of the number of decisions to make, but also because of the multidisciplinary nature of product line models (engineering, marketing, etc). Sharing configuration decisions between stakeholders is highly desirable in order to cope with the issues of the configuration process which is referred to as collaborative configuration process.

Real life product lines models can contain up to 10,000 features (Batory et al., 2006). In such a case, the configuration becomes an arduous and error prone task. Ensuring compromise between all stakeholders configuration decisions during conflict resolution presents a critical issue. Potentially, the more there are constraints, the higher the effort is to resolve conflicts. Moreover, resolving such situations can not be considered effective if it does not take into account the configuration decisions of stakeholders.

To cope with conflict resolution issues, several proposals such as those proposed by (Czarnecki et al., 2005),(Mendonca et al., 2007),(Mendonca et al., 2008), (Rabiser et al., 2009) and (Hubaux et al., 2010) adopted a staged configuration principle where configuration decisions are refined on multiple steps by different stakeholders and arranged by a workflow. The workflow approach has a significant disadvantage; namely the configuration process lacks flexibility. Such a method poses several obstacles for interactive negotiation of the different requirements and in many cases, choices made in earlier stages overlay those in later phases. In addition, activities of some stakeholders would have side effects on those performed by other stakeholders, sometimes through propagation that can be hard to anticipate and solve beforehand.
The resulting lack of flexibility is likely to be a limitation when resolving conflicts as the adopted conflict resolution strategy will not ensure the satisfaction of all stakeholders and consider their preferences.

We thus in this paper, propose a new product line collaborative configuration approach to improve the conflict resolution process, by allowing stakeholders to express preferences through a set of subsitution rules and then deleting the minimal set of conflicting configuration decisions according to these preferences. Major collaborative configuration issues are discussed such as the lack of configuration process flexibility and the systematic strategy of conflict resolution. Finally, evidence of the feasibility of the proposed approach is shown through an illustrative example and by introducing a support tool.

The remainder of this paper is organized as follows. Section 2 provides preliminaries. Section 3 is devoted to the proposed collaborative configuration approach. Section 4 illustrates the approach using an example on the web portal domain. Section 5 covers related work and Section 6 concludes the paper.

2 PRELIMINARIES

To understand our approach, the knowledge about conflict within collaborative configuration of product lines and Minimal Correction Subsets (MCSs) is important. They are briefly discussed in this section.

2.1 Conflict definition

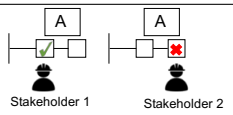
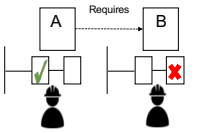
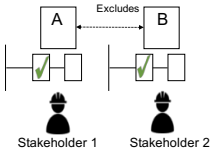
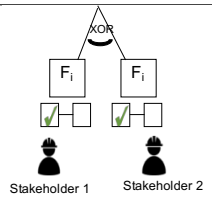
In the context of product line collaborative configuration, conflict management is of paramount importance. According to (Mendonca et al., 2007), conflict occurs when two or more features contain explicit or implicit dependencies rely on each other's decision state. Likewise, (Osman et al., 2009) outlined that a conflict occurs when two or more configuration decisions assigned to different stakeholders cannot be true at the same time.
Formally a conflict can be defined as follows:
Definition. For a configuration *Co* composed of a set of configuration decisions $\{Cd_i\}$, a subset $Cflt \subseteq$ Co is a conflict, if *Co* is unsatisfiable and $\forall\, Cd_i \in Cflt$, $Co \backslash \{Cd_i\}$ is satisfiable.

2.2 Conflict types

Conflicts can be categorized into two different types as illustrated in Table 1.

- ***Explicit Conflict:*** represents the case where the configuration decisions about the same feature made by two or more stakeholders are contradictory (feature selected by a stakeholder and undesired by another).

Table 1: Conflict types in collaborative configuration.

Conflict type	Illustration
Explicit	
Implicit	
Implicit	
Implicit	

- ***Implicit Conflict:*** represents the case where the configuration decisions of different stakeholders do not respect the constraints of the model / domain of the product line. Here, three situations are distinguished:
  - *Situation 1:* conflict occurs when a feature A selected by a stakeholder 1 *requires* a feature B which undesirable by stakeholder 2.
  - *Situation 2:* conflict occurs when a feature A *excludes* feature B and both are selected by two stakeholders
  - *Situation 3:* conflict occurs when two or more features are alternative and all are selected by different stakeholders.

2.3 Minimal Correction Subsets

In the proposed approach we make use of MCSs computing algorithm to identify the minimal set of conflicting configuration decisions to be deleted.
According to (Liffiton and Sakallah, 2008), a MCS is an irreducible subset of constraints whose removal makes satisfiable a constraint program.
Formally an MCS can be defined as follows:
Definition. A subset of constraints $M \subseteq$ C is a MCS, if $C \backslash M$ is satisfiable and $\forall\, C_i \in M, C \backslash (M \backslash \{C_i\})$ is unsatisfiable.
Computing MCSs has been the subject of several re-

search works in the fields of Boolean satisfiability and constraint satisfaction problems. The goal is to find minimal sets of clauses whose removal renders given unsatisfiable Conjunctive Normal Formula (CNF) satisfiable.

Different approaches have been proposed over the years for computing MCSs such as those proposed by (Bailey and Stuckey, 2005), (Liffiton and Sakallah, 2008) and (Marques-Silva et al., 2013). Most of these approaches consist on making iterative calls to a SAT solver to check the satisfiability of different sub-formulas. Generally, these algorithms handle a triplet $\{S, U, C\}$ of F, where S is a satisfiable sub-formula, C contains clauses which are inconsistent with S (i.e. $\forall\, c \in C,\ S \cup \{c\} \models \bot$), and U contains the remaining clauses of F. For a first call, the solver computes a complete assignment μ satisfying F_H in order to initialize S (resp. U) containing the satisfied clauses (resp. unsatisfied) by μ, and C as the empty set. Note $F_H \subseteq S$. In the end, S (C) will be an MCS of F and U will become empty.

MaxSAT (Liffiton and Sakallah, 2008), represents the most widely adopted approach for computing MCSs that consists of finding an assignment that satisfies the maximum number of clauses of an unsatisfiable formula (Marques-Silva et al., 2013). Therefore, finding MCSs is related to the MaxSAT (or MaxCSP) problem, which consists in finding a minimal subset of assignment satisfying the different clauses of unsatisfiable CNF. This represents an optimal solution to MaxSAT.

In this paper, we will make use of (Liffiton and Sakallah, 2008) approach to compute MCSs for conflict resolution within collaborative configuration of Product lines.

3 PROPOSED APPROACH

To cope with collaborative configuration issues, we propose a new configuration approach where stakeholders freely express their configuration decisions and conflicts are resolved based on their preferences.

A summary of the proposed approach is depicted in Figure 1. The approach encompasses four main steps: *(1)* the substitution rules selection, *(2)* collaborative configuration, *(3)* configuration verification and *(4)* conflict resolution.

3.1 Substitution rules selection

Substitution rules represent a recovery plan that permit stakeholders to express their preferences if one or more of their configuration decisions could not be retained in case of conflict.

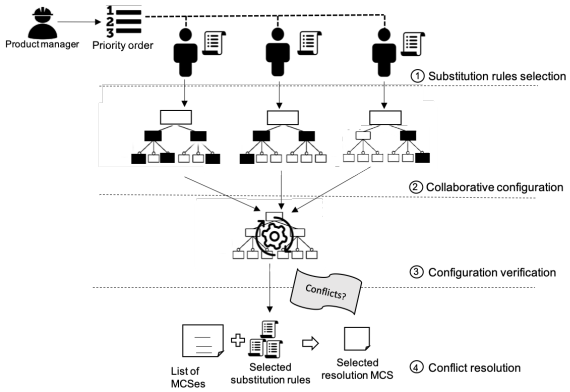


Figure 1: Overview of the proposed collaborative configuration approach.

The proposed list is composed of five substitution rules described as follows :

- SR1. *Most complete product*: includes the maximum as possible of features selected by the different stakeholders.
- SR2. *Simplest product*: includes the minimum as possible of features selected by the different stakeholders.
- SR3. *Product similar to a past configuration*: previous configuration that contains the same features selected by the current stakeholder.
- SR4. *Product configured by a similar profile*: two profiles are similar when two stakeholders share the same concern (engineers, project managers, managers, marketing, sales, customers, users, etc.). Therefore, the configuration decisions made by the current stakeholder will be aligned with those made by the stakeholder with the same profile.
- SR5. *Prioritize my explicit configuration decisions*: considering configuration decisions explicitly expressed through feature selection.

During the first step of the proposed approach, each stakeholder selects the set of desired substitution rules. Each rule refers to the removal of a specific MCS as presented in Table 2. In the context of the proposed approach, a MCS represents the set of configuration decisions whose removal makes satisfiable the current configuration.

In case of conflict, the selected rules are applied on the list of computed MCSs to identify the resolution MCS which is the one common among all these rules. Moreover, a priority order is also assigned by the product manager to the different stakeholders which serves as a secondary MCS selection criterion if the

selected rules do not return any common MCS.

3.2 Collaborative configuration

During the configuration process, the different stakeholders freely express their configuration decisions towards the desired product without being constrained to the configuration decisions made by the other stakeholders.

In the context of the proposed approach, a configuration decision permits stakeholders to explicitly express both, the list of desired features and the list of undesired features. Formally a configuration decision (Cd) can be defined as follows:
$Cd \in \{F, \neg F\}$, $Cd = F$, if the feature is *desired* by the stakeholders, $Cd = \neg F$, if the feature is *undesired* by the stakeholder.
The final configuration encompasses all the mandatory features that any derived product should contain and all the partial configurations made by the different stakeholders. Formally a partial configuration of a given feature model *(fm)* can be defined as follows: $Sh\_cf = \langle fm , \{Cd_{Sh}\}\rangle$, where $\{Cd_{Sh}\}$ represents the set of the decisions made by a stakeholder (Sh). The consistency of each partial configuration is ensured through an automatic propagation of constraints.
The final configuration, can be formally defined as follows: $Cf_{tot} = \langle fm, \{Sh\_cf\}\rangle$, where $\{Sh\_cf\}$ represents the set of all the partial configurations.

3.3 Configuration verification

During the verification step, the partial configurations of all stakeholders are merged and checked against the list of conflict types described in Section 2.

The configuration verification is ensured by a SAT solver. Therefore, the configuration is formulated as CNF where each configuration decision is represented as single clause. In case of conflict detection, the MAXSAT algorithm of (Liffiton and Sakallah, 2008) is used to compute the list of all the possible MCSs for a given unsatisfiable configuration.
Considering the set of substitution rules selected by stakeholders, the MCS that better meets these rules is then chosen to resolve the detected conflict(s).

3.4 Conflict resolution

To resolve conflicts, we propose an algorithm that permits identifying the resolution MCS that better satisfies stakeholders preferences expressed through selected substitution rules.

We have chosen to compute MCSs as they permit finding minimal sets of conflicting configuration decisions whose removal renders the configuration satisfiable. Thus, we can ensure in part the satisfaction of stakeholders by removing the minimal set of their inconsistent choices.

The inputs of the proposed algorithm (Algorithm 1) are the *list of rules selected by stakeholders* (`S_rules`), *list of computed MCSs* (`MCS_list`). As output, the algorithm delivers the MCS of conflict resolution (`R_MCS`) according to the set of substitution rules selected by stakeholders.

The algorithm applies at first the list of selected rules (`S_rules`) on the list of computed MCSs (`MCS_list`) according to the correspondence list presented in Table 2. A substitution rule may return none or many MCSs. Therefore, (`Result_list`) contains the different lists of MCSs returned by each rule. In such a case, it is possible that (`Result_list`) contains common MCS(es) between all the resulting MCSs lists. Then, the Boolean function *(hasCommon)* checks the commonality between all the returned lists present in (`Result_list`). If there is commonality (*hasCommon(Result_list)= True*), then the function (*ComputeCommonList(Result_list)*) returns the list of common MCSs (`Com_list`).

If there is one common MCS (*Size(Com_list)=1*), this one is returned as solution (`R_MCS`). If more than one MCS are common (*Size(Com_list)>1*), the function (*Get_Priority_Based*) is applied on the set of common MCSs (`Com_list`) to select a MCS that better respects the configuration decisions of the prioritized stakeholder.

In the case where there is no common MCSs (*hasCommon(Result_list)= False*), the function (*Get_Priority_Based*) is applied on the initial set of resulting MCSs (`Result_list`) to select a MCS that better respects the configuration decisions of the prioritized stakeholder.

4 ILLUSTRATIVE EXAMPLE

In the previous section, we introduced our collaborative configuration approach. In this section, we describe a simple example that illustrates the proposed approach namely how conflicts can be resolved during product line collaborative configuration using our algorithm. This example involves the configuration of a website product line (Figure 2) based on an adapted version of the website feature model proposed by (Mendonca et al., 2008). We pick one configuration scenario from this domain which is described in Table 3. Considering that three stakeholders are sharing the configuration of the website model. First, a priority order is assigned to the different stakeholders. After-

Table 2: List of substitution rules and the corresponding MCS.

Substitution rule	Corresponding MCS
SR1 most complete product	MCS that eliminates the minimum of features
SR2 simplest product	MCS that eliminates the maximum of features
SR3 product similar to a past configuration	MCS that respects features present in a similar past configuration
SR4 product configured by a similar profile	MCS that respects configuration decisions of similar profile
SR5 prioritize my explicit configuration decisions	MCS that respects configuration decisions explicitly made by the stakeholder

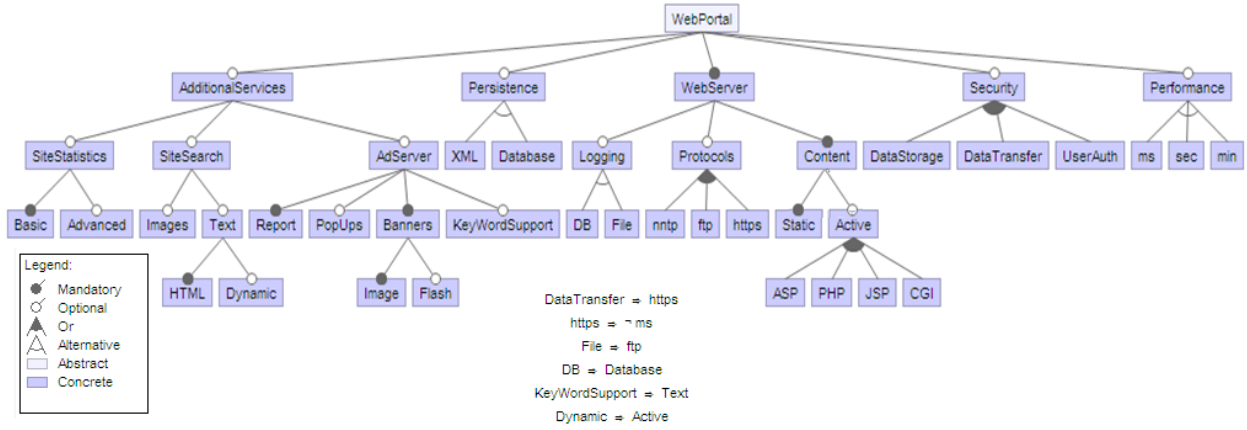


Figure 2: Web Portal feature model.

Algorithm 1 Conflict Resolution algorithm.

Require:
S_rules: list of rules selected by stakeholders.
MCS_list: list of computed MCSs.
Ensure:
R_MCS: conflict resolution's MCS.

for each (Rule ∈ S_rules) do
$Result \leftarrow apply(Rule, MCS\_list)$
$Result\_list \leftarrow Result\_list \cup Result$
end for
if $(hasCommon(Result\_list) == True)$ then
$Com\_list \leftarrow ComputeCommonList(Result\_list)$
if $(Size(Com\_List) == 1)$ then
$R\_MCS \leftarrow Com\_list$
else if $(Size(Com\_list) > 1)$ then
$R\_MCS \leftarrow Get\_Priority\_Based(Com\_list)$
end if
else
$R\_MCS \leftarrow Get\_Priority\_Based(Result\_list)$
end if
$Return(R\_MCS)$

ward, each stakeholder selected the desired substitution rules, stakeholder one *(Sh1)* chose SR2 representing the simplest product, stakeholder two *(Sh2)* didn't choose any rule, stakeholder three *(Sh3)* chose SR5 that permits prioritizing his/her explicit decisions.

In the next step, stakeholders start configuring the desired web site. Therefore, each stakeholder express his configuration decisions by specifying the list of desired features and the list of undesired features.
The consistency of each partial configuration is ensured by the automatic propagation of constraints. For example, when *(Sh1)* selected the feature "Https", the feature "Ms" is automatically discarded because of the constraint `exclude` between these two features as depicted in Figure 2.
As illustrated in last row of Table 3, in addition to the mandatory features *(Webserver, Content and Static)*, the total configuration encompasses all the partial configuration made by the different stakeholders.

Afterward, the total configuration consistency is checked against the feature model dependency constraints depicted in Figure 2.
After the verification of the obtained configuration, three conflicting situations detected are respectively presented in Table 4.
The first conflict occurs because "Https" *excludes* "Ms" where *Sh1* explicitly selected "Https" ($\{Https\}$) and implicitly (through constraint) discarded "Ms" ($\{\neg\, Ms\}$) and *Sh3* explicitly selected "Ms" ($\{Ms\}$)and implicitly discarded "Https" ($\{\neg\, Https\}$).
The same for the second and third conflict where "XML" and "Ms" are respectively alternative features *(XOR)* with "Database" and "Sec".

Table 3: Configuration scenario.

Stakeholder	Priority order	Substitution rule	Configuration decisions
Sh1	2	SR2	$\neg Active$, Protocols, Https, $\neg Ms$
Sh2	3	None	$\neg Active$,Persistence, XML,$\neg Database$, Performance, Sec,$\neg Ms$, $\neg Min$
Sh3	1	SR5	$\neg Active$,Persistence, Database, $\neg XML$, Performance, Ms, $\neg Sec$, $\neg Min$, $\neg Https$
Total configuration			WebServer $\wedge$ Content $\wedge$ Static $\wedge\neg Active\wedge$ Security $\wedge$ Datatransfer $\wedge$ Protocols $\wedge$ Https $\wedge$ $\neg Https$ $\neg Ms\wedge$ Persistence $\wedge$ XML $\wedge\neg Database\wedge$ Performance $\wedge$ Sec $\wedge\neg Min\wedge$ Database $\wedge\neg XML\wedge$ Ms $\wedge\neg Sec$

Table 4: Detected conflicts.

Violated Constraint	Detected Conflict
1) Https VS Ms	{Https, $\neg Https$} {Ms, $\neg Ms$}
2) XML VS Database	{XML,$\neg XML$ } {Database, $\neg Database$}
3) Ms VS Sec	{Ms, $\neg Ms$} {Sec, $\neg Sec$}

To resolve these conflicts, MCSs are computed. The whole list is presented in Table 5 which represents all the possible combinations of conflicting configuration decisions whose removal resolves all the detected conflicts.

The rules selected by stakeholders are then applied on the MCSs list to decide which MCS to choose. As presented in Table 6, *(Sh1)* chose SR2-simplest product which corresponds to MCS that eliminates the maximum of features i.e {*Https, Ms, XML, Database, Sec* }. *(Sh3)* chose SR5 which prioritizes his explicit choices which are "Ms" and "Database". Therefore, the MCS that respects these choices is the one who keeps these two features which is: {*Https,* ¬ *Ms, XML,* ¬ *Database, Sec*} .

The obtained results are different, according to the proposed algorithm, the *resolution MCS* is selected based on the priority order assigned to stakeholders. As shown in Table 3, (Sh3) is the prioritized stakeholder, therefore, {Https, ¬ Ms, XML, ¬ Database, Sec} is selected as the resolution MCS and the configuration decisions contained in this MCS are then removed from the current configuration.

Here, stakeholders are aware of the MCS that has been selected to resolve conflicts and could be satisfied in the sense that the resolution relies on the substitution rules they selected whatever the eliminated configuration decisions.

To evalute the feasibility of the proposed approach, a tool called Colla-Config tool is implemented using micro-service based web application architecture where the front end is implemented using Vue.js framework and the backend in java.

The tool offers two different user access. (1) for stakeholders where they can select the desired subsitution rules and make configuration decisions as depicted in Figure 3. (2) for product manager where he handles participating stakeholders (see Figure 4) through assigning priority order, checking the consistency of the total configuration, running the conflict resolution and delivery of the final validation configuration to stakeholders as depicted in Figure 5 .

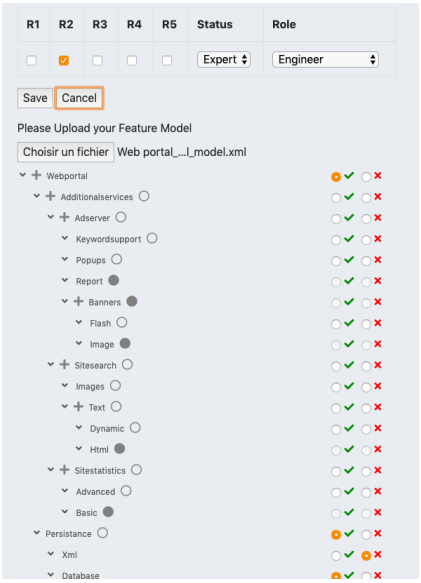


Figure 3: Stakeholders configuration interface.

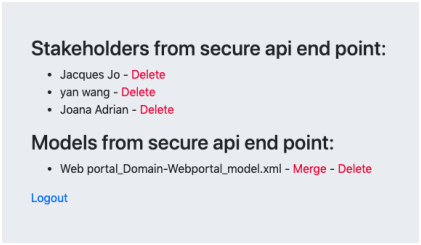


Figure 4: Product manager interface part 1.

The example presented above was run under the implemented tool where computing MCSs and selection of resolution MCS costs 3659ms. This initial experiment was undertaken in the following environment: Laptop with MacOS MOjave Ultimate of 64 bits, processor Intel Core i7, CPU 2.2 GHz, and RAM memory of 8,00 GB.

Table 5: List of computed MCSs.

{Https, Ms, XML, Database, Sec }	{Https, Ms, XML, Database, ¬ Sec }	{Https, Ms, XML, ¬ Database, Sec }
{Https, Ms, XML, ¬ Database, ¬ Sec }	{Https, Ms, ¬ XML, Database, Sec }	{Https, Ms, ¬ XML, Database, ¬ Sec }
{Https, ¬ Ms, XML, Database, Sec }	{Https, ¬ Ms, XML,¬ Database, Sec}	{Https, ¬ Ms, ¬ XML, Database, Sec }
{¬ Https, Ms, XML, Database, Sec}	{¬ Https, Ms, XML, Database, ¬ Sec}	{¬ Https, Ms, XML, ¬ Database, Sec }
{¬ Https, Ms, XML, ¬ Database, ¬ Sec }	{¬ Https, Ms, ¬ XML, Database, Sec}	{¬ Https, Ms, ¬ XML, Database, ¬ Sec}

Table 6: Substitution rules application

Stakeholder	Rule	Result	Common MCS
Sh1	SR2	{Https, Ms, XML, Database, Sec }	No
Sh2	None	-	
Sh3	SR5	{Https, ¬ Ms, XML, ¬ Database, Sec}	

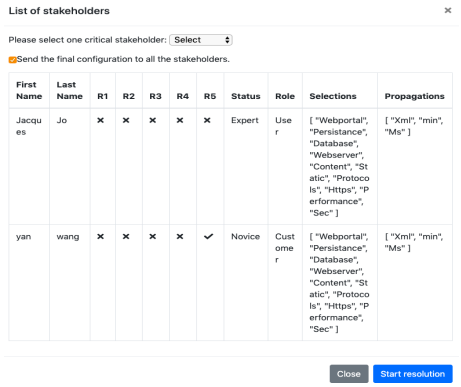


Figure 5: Product manager interface part 2.

5 RELATED WORK

According to a recent Systematic Mapping Study carried by (Edded et al., 2019), 41 primary studies address the collaborative configuration of product lines. Collaborative configuration approaches proposed by these studies can be categorized as follows:

Approaches relying on predefined process, also called workflow-based. In these approaches, configuration activities are coordinated and assigned to stakeholders where each of them is in charge of configuring a specific module of the feature model.

Some of these approaches, *systematically* resolve conflicts by *prioritizing configuration decisions made at earlier stage* such as the approaches proposed by (Czarnecki et al., 2005) and (Mendonca et al., 2008) where configuration decisions of some stakeholders govern the selection of other stakeholders and influence which features are still available. These approaches proposed also *negotiation session* as resolution method when the conflicting decisions are shared between many stakeholders.

Some other approaches such as the one proposed by (Mendonca et al., 2007) anticipates conflicts and identify two basic *priority schemes*: role-based and decision-set-based. In the first scheme, decision priority is given to the highest ranked decision role. In the second one, priorities are given to decision sets based on the importance of their contained features to the process.

Moreover, we find other collaborative configuration approaches such as (Rabiser et al., 2009), (Xiong et al., 2012) and (Hubaux et al., 2010) who resolve conflicts by directly suggesting to stakeholders a set of correction values called *Range Fixes*.

Approaches relying on free order process. In this category of approaches, stakeholders freely express their configuration decisions without imposing a selection order on them. Few of these approaches provide information about conflict management such as the approach proposed by (Junior et al., 2011) where personal assistance provided by software agents propose a *conflict resolution plans* and the stakeholders are free to accept anyone of the offered suggestions.

(J.Stein et al., 2014) also proposed a free order configuration approach where conflicts are resolved based on stakeholders preferences expressed through hard and soft constraints. The conflict resolution scenario consists in *relaxing conflicting hard constraint to a soft one* and deciding about the maintained decision through the higher expressed preference degree.

We also find the approach proposed by (Ochoa et al., 2015) that uses *language-based criteria* to find a non-conflicting set of solution configurations that are aligned to stakeholders preferences expressed in terms of non-functional properties.

However, many other approaches their main focus is on the configuration process and do not provide any information about conflict resolution process such as approaches proposed by (R.Dou et al., 2016) and (Pereira et al., 2018).

Our approach offers flexible configuration process where stakeholders freely express their choices. To the best of our knowledge, our approach is the first to allow expressing undesired features unlike existing approaches which only consider decisions about desired features. Moreover, our approach is one of the rare approaches which provide explicit information about conflict and its possible types. We believe

the approach represents a considerable improvement over current conflict resolution policies by proposing a full preference-based solution .

6 CONCLUSION

Current research carried in the field showed that the most of existing collaborative configuration approaches rely on a pre-designed processes where activities of some stakeholders would have side effects on those performed by other stakeholders. The resulting lack of flexibility is likely to be a limitation when resolving conflicts as the adopted resolution strategies do not consider the preferences of stakleholders thus do not ensure their satisfaction. Therefore, we presented in this paper a novel approach of collaborative configuration capable of resolving conflicts according to preferences of stakeholders.

The experiments carried on were made to evaluate the feasibilty of the approach. We consider that in the future several controlled experiments must be conducted to assert the usefulness of this work.

ACKNOWLEDGMENTS

This work was financially supported by the REVAMP EOTP R5L220003801 project.

REFERENCES

Bailey, J. and Stuckey, P. J. (2005). Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In *Proceedings of the 7th International Conference on Practical Aspects of Declarative Languages*, PADL'05, pages 174–186, Berlin, Heidelberg. Springer-Verlag.

Batory, D., Benavides, D., and Ruiz-Cortes, A. (2006). Automated analysis of feature models: Challenges ahead. *Commun. ACM*, 49(12):45–47.

Clements, P. and Northrop, L. (2001). *Software Product Lines: Practices and Patterns*. Longman Publishing.

Czarnecki, K., Helsen, S., and Eisenecker, U. W. (2005). Staged configuration through specialization and multilevel configuration of feature models. In *Software Process: Improvement and Practice*.

Edded, S., BenSassi, S., Mazo, R., Salinesi, C., and BenGhezala, H. (2019). Collaborative configuration approaches in software product lines engineering: A systematic mapping study. *Journal of Systems and Software*, 158:110422.

Hubaux, A., Heymans, P., Schobbens, P.-Y., and Deridder, D. (2010). Towards multi-view feature-based configuration. In *Proceedings of the 16th International Working Conference on Requirements Engineering:Foundation for Software Quality (REFSQ'10)*, pages 106–112, Essen, Germany. Springer-Verlag.

J.Stein, I.Nunes, and E.Cirilo (2014). Preference-based feature model configuration with multiple stakeholders. In *18th International Software Product Line Conference*, pages 132–141, New York, NY, USA. ACM.

Junior, C. R. M., Cirilo, E. J., and de Lucena, C. J. (2011). Assisted user-guidance in collaborative and dynamic software product line configuration. In *Proceedings of the 14th Ibero-American Conference on Software Engineering*, pages 143–156.

Liffiton, M. H. and Sakallah, K. A. (2008). Algorithms for computing minimal unsatisfiable subsets of constraints. *J. Autom. Reason.*, 40(1):1–33.

Marques-Silva, J., Heras, F., Janota, M., Previti, A., and Belov, A. (2013). On computing minimal correction subsets. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 615–622. AAAI Press.

Mendonca, M., Bartolomei, T., and Cowan, D. (2008). Decision-making coordination in collaborative product configuration. In *ACM symposium on applied computing*, pages 108–113, New York, NY, USA. ACM.

Mendonca, M., Cowan, D., and Oliveira, T. (2007). Process-centric approach for coordinating product configuration decisions. In *Proceedings of the 40th Hawaii International Conference on System Sciences (HICSS'07)*, pages 1–10, Washington, DC, USA. IEEE Computer Society.

Ochoa, L., Gonzlez-Rojas, O., and Thm, T. (2015). Using decision rules for solving conflicts in extended feature models. In *Software Language Engineering, the 2015 ACM SIGPLAN International Conference*, pages 149–160, New York, NY, USA. ACM.

Osman, A., Somnuk, E., Phon-Amnuaisuk, and Ho, C. K. (2009). *Investigating Inconsistency Detection as a Validation Operation in Software Product Line*, pages 159–168. Springer Berlin Heidelberg, Berlin, Heidelberg.

Pereira, J. A., P.Matuszyk, S.Krieter, M.Spiliopoulou, and G.Saake (2018). Personalized recommender systems for product-line configuration processes. *Computer Languages, Systems & Structures*, 54:451–471.

Rabiser, R., Wolfinger, R., and Grunbacher, P. (2009). Three-level customization of software products using a product line approach. In *Proceedings of the 42nd IEEE Annual Hawaii International Conference on System Sciences*, pages 1–10.

R.Dou, Y.Zhang, and G.Nan (2016). Customer-oriented product collaborative customization based on design iteration for tablet personal computer configuration. *Computers & Industrial Engineering*, 99:474–486.

Xiong, Y., Hubaux, A., She, S., and Czarnecki, K. (2012). Generating range fixes for software configuration. In *Proceedings of the 34th International Conference on Software Engineering (ICSE'12)*, pages 58–68, Zurich, Switzerland. IEEE Computer Society.