Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Colla-Config: A stakeholders preferences-based approach for product lines collaborative configuration[☆]

Sihem Ben Sassi[a,c,*], Sabrine Edded[b,a], Raúl Mazo[d,e], Henda Ben Ghezala[a], Camille Salinesi[b]

[a] RIADI Lab., National School of Computer Sciences, Manouba University, Tunisia
[b] Centre de Recherche Informatique (CRI), Université Paris 1 Panthéon-Sorbonne, Paris, France
[c] CBA, University of Taibah, Al-Madinah, Kingdom of Saudi Arabia
[d] Lab-STICC, ENSTA Bretagne, Brest, France
[e] GIDITIC, Universidad Eafit, Medellin, Colombia

ARTICLE INFO

Article history:
Received 11 February 2022
Received in revised form 19 October 2022
Accepted 2 December 2022
Available online 7 December 2022

Keywords:
Software product lines
Collaborative configuration
Stakeholder preferences
Usability tool test

ABSTRACT

During collaborative configuration of software product lines (SPL), multiple stakeholders contribute together in building a single product specification. Conflicting situations can arise during the configuration process due to contradictions between some/all stakeholders' configuration choices. Detecting and resolving such situation rise two major challenges: choosing which choices to omit, and taking stakeholders' preferences into account. Several SPL collaborative configuration approaches are available. However, they either do not present detailed information on the strategies for conflict resolution, or they rely on a systematic process that resolves conflicts by prioritizing configuration decisions made at earlier stage, constraining therefore some of stakeholders' choices. The lack of flexibility may hinder conflict resolution as choices taken at earlier stages overlay those at later phases. To mitigate these limitations, we propose a new collaborative configuration approach (Colla-Config) that provides a preference-based conflict resolution method within a free-order configuration process; each stakeholder expresses his/her preferences through a set of substitution rules, and freely makes his/her configuration decisions towards the desired product, without being constrained by the configuration decisions made by the other ones. To assess the feasibility and the usability of the proposed approach, we conducted a usability test designed following the ISO/IEC 25062:2006 Common Industry Format for usability tests. Results of the experiments provide preliminary evidence of the approach feasibility and the tool ability to properly support the SPL collaborative configuration.

1. Introduction

In Software Product Lines Engineering (SPLE) the first step of product derivation consists in identifying the features of the desired product from a product line model (Clements and Northrop, 2001). The collection of selected features must comply with product line model constraints and stakeholders requirements (Salinesi et al., 2010). At the industrial scale, product line models may include hundreds of features (Shahin et al., 2021). However, in some cases, they could be larger than expected and may contain thousands of features such as the automotive product line model cited in the work of Pett et al. (2019), and the linux kernel which is one of the largest software product lines currently available, reaching 21,000 features (Nieke et al., 2022). In such cases, the number of potential configurations can be large too. Therefore, and also because of the multidisciplinary nature of product lines models (engineering, marketing, etc.), it is hard to think of a single stakeholder solely handling all the configuration activities (Mendonca et al., 2008). Sharing configuration activities between many stakeholders is therefore very useful to put up with the eventual issues of configuration process, which is called collaborative configuration process.

Several works already propose collaborative configuration approaches such as Czarnecki et al. (2005), Mendonca et al. (2007, 2008), Rabiser et al. (2009), Hubaux. et al. (2010), Junior et al. (2011), Stein et al. (2014), Ochoa and González-Rojas (2016), Pereira (2017) and Le (2021). Each of them relies on a different way to carry out configuration, and to manage conflicting situations. Some of these approaches, such as Czarnecki et al.

[☆] Editor: Raffaela Mirandola.
[*] Corresponding author at: RIADI Lab., National School of Computer Sciences, Manouba University, Tunisia.

E-mail addresses: Sihem.BenSassi@ensi.rnu.tn (S. Ben Sassi), Sabrineedded@gmail.com (S. Edded), Raul.mazo@ensta-bretagne.fr (R. Mazo), Henda.benghezala@ensi.rnu.tn (H. Ben Ghezala), Camille.salinesi@univ-paris1.fr (C. Salinesi).

https://doi.org/10.1016/j.jss.2022.111586
0164-1212/© 2022 Elsevier Inc. All rights reserved.

(2005), Mendonca et al. (2007, 2008) and Rabiser et al. (2009), rely on a pre-designed process to ensure coordination of configuration activities. Such a method lacks flexibility as configuration choices made in earlier steps limit those in later phases and a backtracking may be occasionally required to cope with constrained decisions, which makes it difficult to reach a valid configuration agreed by all stakeholders (White et al., 2009; Stein et al., 2014; Edded et al., 2019). Therefore, the adopted conflict resolution strategy does not fairly consider the choices of all stakeholders. Besides, it completely ignores their conflict resolution preferences. Other approaches such as Junior et al. (2011) and Holl et al. (2012) allow a free-order collaborative configuration. However, they do not consider stakeholders' preferences and their conflict resolution processes is a win–lose strategy. Finally, few approaches make a step forward towards a consensus-based conflict resolution strategy that takes stakeholders preferences into account within a flexible configuration process e.g. Stein et al. (2014) and Ochoa et al. (2015). Nevertheless, these adopted strategies do not always generate a solution that takes into account the preferences of all stakeholders as pointed in Le (2021). In general, most of the existing collaborative configuration approaches mainly focus on the configuration process and lack flexibility by ignoring preferences in their conflict resolution strategy as identified in Edded et al. (2019).

This work proposes a collaborative product line configuration approach, called Colla-Config, that relies on "free order" configuration process, where stakeholders freely make their configuration decisions towards the desired product without being constrained by the configuration decisions made by other stakeholders. Colla-Config offers a new preference-based conflict resolution strategy, where stakeholders' preferences are elicited under the form of substitution rules to be considered if one or more of their configuration decisions could not be retained because of a conflict. Afterward, the minimal set of conflicting configuration choices is computed using the MCS (Minimal Correction Subset) algorithm (Liffiton and Sakallah, 2008). To evaluate the proposed approach, we created a software tool that implements the Colla-Config approach and used it to conduct a usability test, so as to gain insights into how Colla-Config supports collaborative configuration, mainly preference-based conflict resolution. Eleven Ph.D students took part and were asked to carry out a series of actions to configure a Web portal product line. During the experiment, we collected data to calculate the tool effectiveness and efficiency and measure the participants satisfaction. This article extends our earlier paper, presented at the 15th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2020) (Edded et al., 2020). In comparison to the conference publication (8 pages, double-columns), this article provides more details about the Colla-Config approach, especially (i) the formalization of the whole process, (ii) how the Minimal Unsatisfiable Subsets (MUSs) are computed in order to obtain the MCSs, and (iii) a better illustration through a running example. In addition, unpublished evaluation of the Colla-Config approach extends our prior work through (1) a usability test conducted on the tool supporting the approach, (2) a computation of the complexity of the presented algorithms, and (3) a discussion of the scalability of the proposed solution.

The remainder of this paper is organized as follows. Section 2 is dedicated to defining the main concepts related to collaborative configuration of product lines and to the context of this work. Section 3 presents the running example used to illustrate the proposed approach, and motivates this work through various problematic scenarios based on the same example. Section 4 presents the Colla-Config approach. Section 5 reports a usability test conducted to evaluate the proposed approach. Section 6 discusses the scalability of the approach. Section 7 is dedicated to threats to validity. Section 8 positions our contribution to related works. Finally, Section 9 concludes the paper with a discussion on future works and open issues.

2. Background and context

In the last three decades, various means and techniques have been proposed to represent software product lines, intended especially to capture the variability aspect (Feichtinger et al., 2021). Among them, feature model is one of the most popular one; we can also cite orthogonal variability model and UML extended by variability stereotypes. A comprehensive list of the plethora of notations is given in the tertiary study carried out by Raatikainen et al. (2019). In this work, we are interested in SPLs represented through feature models. We introduce in this section the main concepts related to software product lines collaborative configuration, as well as the key ones related to this work while presenting the context.

2.1. SPL collaborative configuration main concepts

Feature model: Feature models are proposed within a domain analysis method called FODA, (Feature-Oriented Domain Analysis) (Kang et al., 1990), as a means to represent commonalities and variabilities of product families. A feature model is composed of a set of features related by relationships (also called "dependencies"), namely mandatory, optional, inclusive choices (OR) and exclusive choices (XOR). Furthermore, a set of constraints can be defined to express inclusion (resp. exclusion) relationship between features in order to capture domain constraints: a feature requires that another feature be included (resp. excluded) when it is itself chosen in the product (Benavides et al., 2010).

Stakeholder: In the context of product lines configuration, any person directly or indirectly involved in the configuration process is a stakeholder. Stakeholders may have various *roles*, such as product manager, costumer, software engineer, technical experts, domain expert, etc.

Product configuration: Product configuration refers to a decision-making process in which a set of features are chosen to meet individual needs. These features must comply with both the constraints of the product line model and with the requirements of the user (Clements and Northrop, 2001). The *configuration process* is therefore defined as the set of activities that consists in specifying a valid product from a product line model consistently with user requirements (Deelstra et al., 2005).

Collaborative configuration: It is defined as a shared coordinated activity that involves a set of stakeholders who decide about the set of features of the desired product based on their domain of expertise (Edded et al., 2019).

Free-order configuration process: It refers to a collaborative configuration process where stakeholders freely make their configuration choices towards the desired product, in the sense that they are not constrained by the configuration decisions made by other stakeholders.

Predefined order configuration process: In this approach a decision-making order between stakeholders is set in advance. Therefore decisions made by some stakeholders constrain decisions made by other stakeholders afterwards.

Workflow-based configuration process: It refers to a process that relies on a predefined process where configuration activities are coordinated and assigned to stakeholders. In this approach, each stakeholder configures a specific module according to his/her expertise.

Conflict: It refers to a situation of inconsistency where the requirements of two or more stakeholders contradict each other, or do not respect the constraints of the product line model overall (Osman et al., 2009).

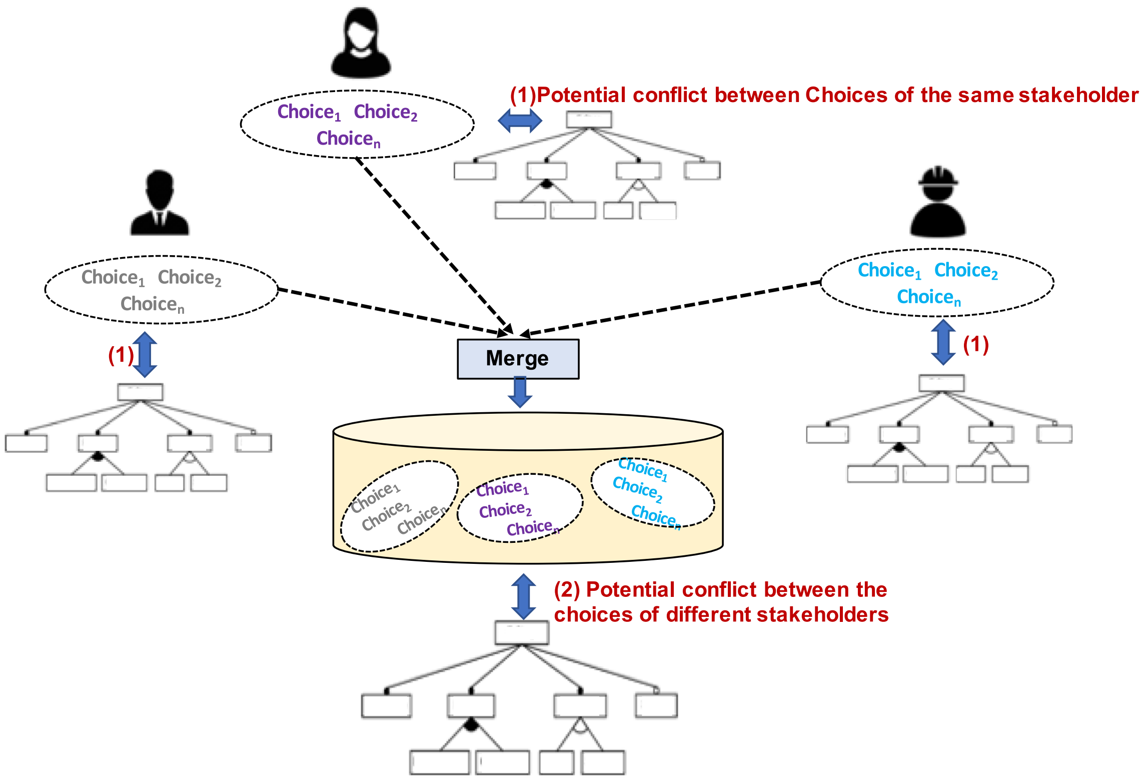

(1)Potential conflict between Choices of the same stakeholder
Choice1 Choice2 Choicen
(1)
Merge
(2) Potential conflict between the choices of different stakeholders

Fig. 1. Conflict situations occurrence in collaborative configuration of product lines.

2.2. Context

During the collaborative configuration of product lines, each stakeholder expresses his/her configuration choices by selecting the desired features from a product line model. These choices are then merged in order to check their consistency and derive the desired product. However, disagreement situations may arise when the configuration choices of stakeholders do not respect the constraints of the model. As illustrated in Fig. 1, these situations may arise at two different times:

- *(1):* When the stakeholder expresses choices that individually violate the constraints of the product line model.
- *(2):* When all stakeholders' configuration choices are merged. Disagreement situations may therefore arise due to: (i) the contradiction of two or more of these choices, or (ii) the violation of the constraints of the product line model by these choices.

These situations of disagreement represent conflicts. According to Roschelle and Teasley (1995), a conflict occurs when two or more choices cannot be taken into account together in a single decision-making process. Generally, a conflict resolution strategy cannot be considered as consensual if it does not take into account *preferences* of the different stakeholders and does not try to find a compromise between as many of them as possible (Stein et al., 2014). Preferences may be defined as a kind of soft requirements provided by the stakeholder, in addition to his/her configuration choices, to reflect a wish or a special interest. They are used in case of conflict detection in stakeholders merged configuration choices, meaning that one or more configuration choices could not be retained. They influence the process of conflict resolution by causing some configuration choices (i.e. features de/selection from the configuration) to be favored. Preferences can be expressed in various ways, for example, in terms of hard and soft requirements as proposed by Bagheri et al. (2010), in terms of functional and non-functional requirements as proposed by Soltani et al. (2012), or else through *substitution rules* as we propose in this work. Substitution rules encompass an alternative configuration scenario allowing to resolve conflicts while taking into account stakeholders wishes or interests. For example, the "simplest product" rule allows to deselect the maximum number of features from the merged configuration to obtain a consistent one without dissatisfying the stakeholder having chosen that rule. Indeed, stakeholders *satisfaction* is of paramount importance when collaboratively configuring a product within SPLE. It can be ensured, among others, by considering stakeholders conflict resolution preferences in deriving the desired product (Edded et al., 2019). Therefore, the collaborative configuration approach proposed in this paper is based on a conflict resolution strategy that relies on stakeholders preferences.

3. Motivating example

We choose to motivate the current work using the example of a Web portal product line, illustrated in Fig. 2, simplified from the Web portal feature model proposed in Mendonca et al. (2008). The same example will serve as a running example to illustrate the Colla-Config process.

As depicted in Fig. 2, each Web portal must necessarily have a Web server ("WebServer") and may have one or more optional features among the following "DriverProg", "Security" and "Performance". Each of these features has a set of sub-features to which it is related thanks to one of the relationships: OR, XOR and optional. For example, "XML" and "Database" are related to "DriverProg" with XOR and therefore, only one of them may be selected if the parent feature ("DriverProg") is selected in a configuration. As to "DataStorage", "Data Transfer" and "UserAuth", they are related to "Security" with an OR relationship. Therefore when "Security" is selected, at least one of them must be selected

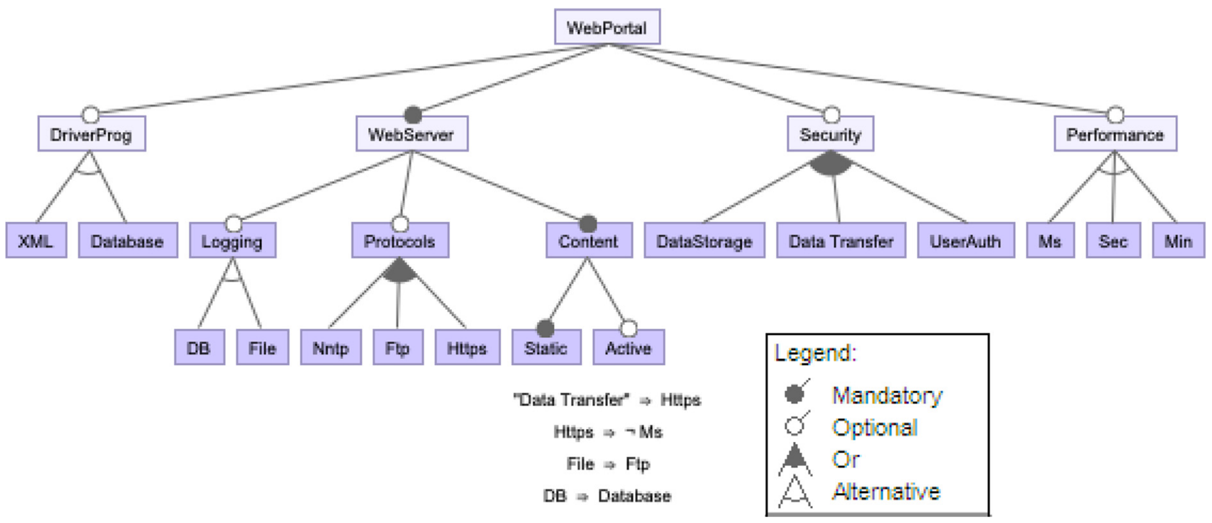


Fig. 2. Reduced model of the Web portals line (Mendonca et al., 2008).

too. Furthermore, the Web portal feature model comes with a set of constraints of the two types: "require" (e.g. "File" requires "Ftp") and "exclude" (e.g. "Https" excludes "Ms").

When a single stakeholder configures the product line, the configuration, made up of the set of selected features, should comply with the constraints in order to be valid and free of inconsistent choices. For example, the configuration *{WebServer, Content, Logging, DB, File, Security, Data Transfer, performance, Ms, Sec}* is not valid for many causes. Indeed, (i) "Static" is a mandatory feature, but it is not included in the configuration, (ii) "DB" and "File" are related with XOR relationship, therefore only one of them should be included; assume that "DB" is retained (iii) "Data Transfer" requires "Https" and "DB" requires "Database", however the latter are not selected; assume now that they are selected; (iv) "Ms" and "Sec" are related with XOR while they are both selected, besides that "Https" excludes "Ms" while Ms is selected; assume now that "Ms" is deselected; and (v) "Database" requires "DriverProg" to be selected and "Https" requires that "Protocols" is selected; assume that both are now selected. The consistent final configuration is therefore *{WebServer, Content, Static, Logging, DB, DriverProg, Database, Security, Data Transfer, Protocols, Https, Performance, Sec}*. The requirements we just discussed, leading to compliance with constraints, are easily ensured thanks to "constraint propagation" within a stakeholder configuration (Czarnecki and Kim, 2005).

Assuming now that two stakeholders are collaboratively configuring the Web portal line, each of them is responsible of modules within his/her expertise. The first one has a role of security engineer; he/she is therefore focusing on security aspects and selects "Https" as protocol, "DB" as logging system and imposes "UserAuth" to access the Web portal. The second stakeholder is an expert in user experience (UX expert); he/she is, therefore, interested in performance and interface design; he/she selects "XML" for its capabilities, "Ms" as performance requirement and demands an "Active" content to ensure information freshness. While each configurations per se, after including mandatory features and applying the constraints propagation, is valid, the configuration resulting from the union of the two stakeholders choices has several problems: (1) "DB", selected by the security engineer, requires to include the feature "Database" in the configuration. However, the UX expert has already selected the "XML" feature which cannot be selected at the same time as "Database"; hence "XML" selection is canceled and "Database" feature is added. The UX expert, noticing that "XML" is no longer included in the configuration, re-selects the feature. By doing so, the feature "Database" is canceled because of the XOR relationship. Since "DB" requires "Database", the latter is, one more time, selected. This scenario is repeated until the two stakeholders decide to modify their conflicting choices. (2) Similarly, "Https" selected by the security engineer excludes "Ms", selected by the UX expert, because of the exclude constraint that relates the two features. When the latter re-selects "Ms" after noticing the modification, the exclude constraint makes "Https" cancel "Ms" another time.

A similar deadlock may be observed in the case where the configuration process involves two stakeholders, say a Web master and a security engineer, that may select any feature from the whole model (i.e. no module assignment). Assume the Web master chooses "Ftp" as protocol, whereas "Ftp" is an undesired feature for security reasons. Hence, the security engineer deselects it and cancels the choice of the Web master. The latter may re-select the feature to enforce including it, while the security engineer, convinced by his/her choice, de-selects it another time. The scenario can be repeated until the two stakeholders decide to modify their conflicting choices.

Let us now consider the case where the configuration process is done according to a predefined order. Assume the product manager is responsible of the "WebServer" module and he/she is the first stakeholder to start the configuration; he/she selects the features "DB", "FTP" and "Active". The security engineer can now start making his/her choices regarding the "Security" module; he/she selects "Data Transfer" and "UserAuth". Now comes the turn the UX expert in charge of "Performance" and "DriverProg" modules; he/she wants to select "Ms" and "XML". However, both features cannot be considered because of: (i) the constraint propagation: "Ms" is excluded by "Https", which is included in the configuration as it is required by "Data Transfer" selected by the second stakeholder; (ii) XOR relationship and constraint propagation: "XML" is related to "Database" with XOR, while "Database" is already selected by constraint propagation as it is required by "DB", which is selected by the first stakeholder. The third stakeholder (i.e. the UX expert) sees himself/herself confronted with a situation where either he/she accepts the configuration as such while being completely dissatisfied as it does not include his/her desired features, or selects "Ms" and "XML" and challenges the other stakeholders choices. In such case, the process loops again to hand over to the product manager then to the security engineer followed by the UX expert turn, till a consensus is reached either by modifying the choices or using external arbitration. Otherwise, the Web portal will never be built.

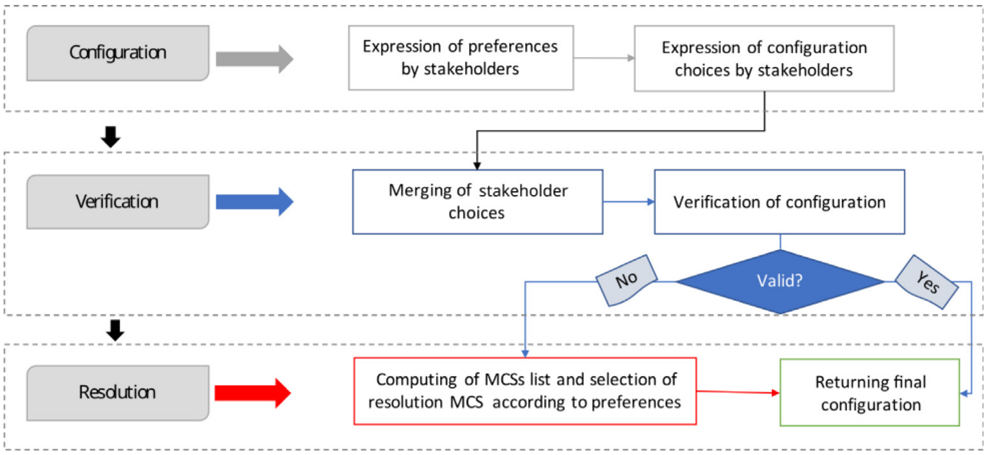

Configuration
Expression of preferences by stakeholders
Expression of configuration choices by stakeholders
Verification
Merging of stakeholder choices
Verification of configuration
No
Valid?
Yes
Resolution
Computing of MCSs list and selection of resolution MCS according to preferences
Returning final configuration

Fig. 3. Colla-Config process.

4. The Colla-Config approach

The proposed approach allows a flexible configuration where stakeholders freely express their configuration choices toward the whole same feature model without being forced to follow a specific predefined order. As explained in Section 3, each single configuration is valid in the sense that it respects the feature model along with its constraints (e.g. includes all mandatory features, respects XOR relationships as well as require and exclude constraints). The Colla-Config approach also allows a dynamic preference-based conflict resolution. These preferences are expressed through a predefined set of alternative configuration scenarios representing *substitution rules* introduced in Section 2.2.

A summary of the proposed approach is depicted in Fig. 3. The approach encompasses three main steps: (1) collaborative configuration step during which the stakeholders express their preferences regarding the conflict resolution by selecting substitution rules, and make their choices regarding the desired product features, (2) configuration verification step during which the total configuration resulting from stakeholders' ones is checked to detect conflicts, and (3) conflict resolution step consisting in identifying the different minimal combinations of conflicting configuration choices that allow to resolve all conflicts and then choose one of them according to substitution rules in order to obtain a final configuration that fairly takes into account all stakeholders conflict resolution preferences. If no configuration allowing to reach a consensus exists, a configuration respecting a prioritized stakeholder configuration choices is identified and returned as final product specification. These steps are detailed in the following subsections.

4.1. Collaborative configuration step

As mentioned earlier, stakeholders involved in the collaborative configuration of a product line may have the role of product manager, CEO of a company, customer, software engineer, technical expert, user, seller, and so on. One of them has an authority to designate the prioritized stakeholder. The role along with the status (i.e. expert or novice) constitute the main information of the profile representing the stakeholder. During the first step of the proposed approach, each stakeholder expresses his/her preferences by selecting one or more substitution rules. As the latter represent alternative configuration scenarios, to be considered if the stakeholder's initial configuration choices could not be totally or partially retained, a stakeholder may choose not to explicitly express them. In that case, we can consider that he/she will be satisfied with the solution that takes into account the preferences of the other stakeholders in case of conflict.

We propose a list of five generic substitution rules derived from known defaults in consumer choice that are determined by marketing studies on consumer preferences and considered by product lines researchers while configuring and deriving products (Donkers et al., 2020; Ziadi and Jezequel, 2006). These rules are described as follows :

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

After rules selection, the stakeholders express their configuration choices by specifying desired (F_i) and undesired ($\neg F_i$) features. In the context of the proposed approach, a configuration choice permits stakeholders to explicitly express both, the list of desired features and the list of undesired ones.

Illustration with the running example

Using the running example presented in Section 3, let us assume that three stakeholders are collaboratively configuring the Web portal line. First, each stakeholder selects the desired substitution rules: the first stakeholder *(STK1)* chooses SR1 representing the most complete product, stakeholder two *(STK2)* does not choose any rule. The third stakeholder *(STK3)* chooses SR5 that permits prioritizing his/her explicit choices. Then, the different stakeholders express their configuration choices by specifying the list of desired and undesired features which are presented in the third column of Table 1 that provides an example of configuration scenario.

The consistency of each stakeholder's configuration is ensured by the automatic constraint propagation. For example, when *STK1* selects the feature "Https", the feature "Ms" is automatically considered as undesired because of the `exclude` constraint between "Https" and "Ms" as depicted in Fig. 2.

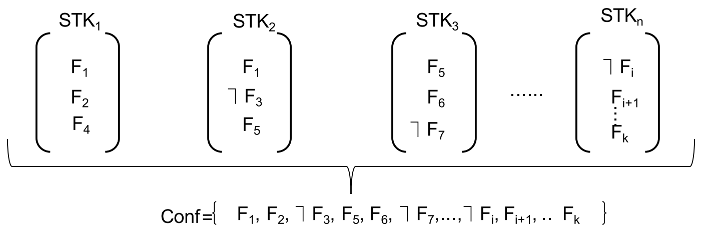


Fig. 4. Configuration choices merging.

Table 1
Web portals line configuration scenario example.

Stakeholder	Selected rule	Configuration choices
STK1	SR1	Active, Protocols, Https, ¬*Ms* (by constraint propagation)
STK2	–	¬*Active*, Performance, Ms, ¬*Sec* (by constraint propagation), ¬*Min* (by constraint propagation, ¬*Https* (by constraint propagation))
STK3	SR5	Active, Performance, Sec, ¬*Ms* (by constraint propagation), ¬*Min* (by constraint propagation)
Total configuration		**WebServer** ∧ **Content** ∧ **Static** ∧¬*Active* ∧ Active ∧ Protocols ∧ Https ∧ ¬*Https* ∧ Performance ∧ ¬*Ms* ∧ Sec ∧¬*Min*∧ Ms ∧¬*Sec*

4.2. Configuration verification step

During the verification step, the configuration choices expressed by the different stakeholders go through a verification process to detect conflicts. As depicted in Fig. 3, choices out of a collection of features (F_i) that characterizes the product line are first merged. The merging of all stakeholders (STK_i) choices gives the total configuration (Conf) as shown in Fig. 4. The latter is composed of: (1) all mandatory features of the model without redundancy, (2) all common choices (desired and undesired non mandatory features) of all stakeholders without redundancy, and (3) the non common remaining choices of all stakeholders. The total configuration is then checked to ensure the consistency of the different configuration choices against the constraints of the product line model. If the configuration is consistent, it is validated and returned as the final product specification. Otherwise, a conflict resolution process is performed.

We categorized potential conflicts into two main types as follows (Edded et al., 2020):

- ***Explicit Conflicts:*** occur when the configuration choices about the same feature made by two or more stakeholders are contradictory. In other words, the same feature is explicitly desired by a stakeholder and undesired by another.
- ***Implicit Conflicts:*** occur when the configuration choices of different stakeholders are formally inconsistent with the constraints of the product line features model. Three simple situations can be distinguished, for example:
  - A feature F_i selected by a stakeholder 1 *requires* a feature F_j which is undesirable by stakeholder 2.
  - A feature F_i *excludes* a feature F_j and both are desired by two stakeholders.
  - Two or more features are *alternative (XOR)* and at least two of them are desired by different stakeholders.

Illustration with the running example
The configuration choices of the three stakeholders are merged to obtain the total configuration as presented in the last row of Table 1. The consistency of the total configuration is checked against the dependency constraints of the Web portal feature model. Therefore, three conflicting situations are detected as presented in Table 2.

Table 2
Conflicting situations.

Violated constraint	Conflict type
(1) Https vs. Ms	Implicit
(2) Active vs. ¬*Active*	Explicit
(3) Ms vs. Sec	Implicit

The first conflict is implicit as the *exclude* constraint between "Https" and "Ms" is violated when *STK1* selected "Https" and implicitly (through constraint propagation) discarded "Ms" (*{¬ Ms})* which is selected by *STK2*. The second conflict is explicit and occurs because of the disagreement between *STK1* who selected the feature "Active" and *STK2* who does not want that feature (*{¬ Active}*). The third conflict is implicit because the *XOR* constraint between "Sec" and "Ms" is violated when *STK2* selected "Ms" and *STK3* selected "Sec".

Since a configuration is constituted by the set of configuration choices, it may be formulated as a Conjunctive Normal Form (CNF), where each configuration choice is represented as single clause. This actually is a Boolean formula and refers to a Boolean satisfiability problem (abbreviated SAT), where it is question of asking whether the variables of a given Boolean formula can be consistently replaced by the values TRUE or FALSE in such a way that the formula evaluates to TRUE. If this is the case, the formula is satisfiable. Each configuration should therefore be checked with the help of some method to make sure it is free from conflicts (i.e. consistent). This may be implemented using Boolean formulas satisfiability checker tools known as SAT solvers. The Colla-Config verification method is based on MCSs (Minimal Correction Subsets), presented in the following sub-section.

4.3. Conflict resolution step

The resolution strategy involves: (1) identifying the different minimal combinations of conflicting configuration choices, which removal resolves all detected conflicts; and (2) choosing one of these combinations of choices to be deleted according to stakeholders preferences.

As depicted in Fig. 3, the set of minimal combination of choices to remove is identified by computing the list of the minimal correction subsets using MCS computing algorithm. The concept

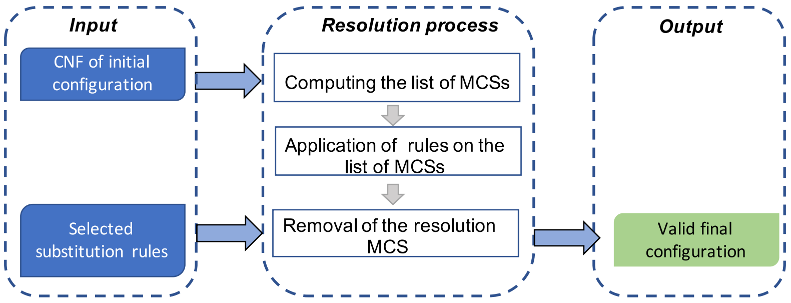

Input
CNF of initial configuration
Selected substitution rules
Resolution process
Computing the list of MCSs
Application of rules on the list of MCSs
Removal of the resolution MCS
Output
Valid final configuration

Fig. 5. Conflict resolution process.

of "Minimal Correction Subset" was originally proposed in Liffiton and Sakallah (2008) where it is defined as an irreducible subset of constraints which removal makes a constraint program satisfiable.

The resolution process is schematized in Fig. 5, it consists of three steps: (1) computing the list of MCSs, (2) applying the substitution rules on the list of MCSs obtained, and (3) removing the resolution MCS from the initial configuration to validate it. The different steps are detailed in the sequel.

4.3.1. Computing the minimal correction subsets

To identify the minimum correction subsets, we rely on Liffiton and Sakallah (2008) algorithm, which allows to compute MCSs from a list of MUSs (Minimal Unsatisfied Subsets) representing the list of detected conflicts in our case. In the context of Constraint Satisfaction Problem, where it is question to solve the satisfiability of a system composed of a set of constraints over variables, a MUS is a subset of those constraints, that is unsatisfiable and minimal; in the sense that removing any one of its elements makes the remaining set of constraints satisfiable (Liffiton and Sakallah, 2008). Therefore, solving a conflict problem consists in finding the set of MUSs and neutralizing them by removing at least one constraint from each MUS. Since a MCS is an irreducible subset of constraints whose removal makes a constraint problem satisfiable, a MCS must therefore contain at least one constraint from each MUS in that constraint problem. As proposed in Liffiton and Sakallah (2008), computing a list of MCSs consists in walking through the list of MUSs and identifying the different combinations formed by a set of constraints from each MUS (taking one constraint from each MUS) that makes the constraints problem satisfiable. These combinations constitute the set of MCSs. The details of the MCSs computing algorithm are presented in Liffiton and Sakallah (2008).

The MCSs computing algorithm is adopted in Colla-Config to compute all possible combinations of choices (MCSs) which deletion resolves all the detected conflicts. The MCSs computing algorithm requires as input the list of MUSs. The latter corresponds in Colla-Config to the list of detected conflicts.

Therefore, to identify conflicts, we propose an algorithm (Algorithm 1) that allows computing the list of MUSs based on stakeholders configuration choices list, the constraints of the product line and the number of these constraints (lines 1–4). As specified in Algorithm 1, computing the list of MUSs consists in walking through the constraints list of the product line (line 6) to identify violated ones in order to deduce the corresponding MUSs from the clauses of these constraints.

Line 7 of Algorithm 1 checks if the violated constraint is either *exclude* or a *XOR*. These two types of constraints represent a mutual exclusion between two or many features. Therefore, as described in lines 8–15 of Algorithm 1, the features of the constraint are browsed to determine the MUS which corresponds to the absence ($\neg F_j$) and presence (F_j) of each of these features.

In the case where the violated constraint corresponds to a *require* dependency, as described in line 17 of Algorithm 1, the conflict occurs if some stakeholder does not want the required feature ($\neg F_j$). Therefore, as indicated in lines 18–25 of Algorithm 1, the features of the constraint are browsed to identify the corresponding MUS, which is $\{F_j, \neg F_j\}$.

Once the list of MUSs is obtained, MCSs are then computed according to the algorithm proposed by Liffiton and Sakallah (2008).

Algorithm 1 Compute_MUSs: Computing MUSs from configuration conflicts.

```
Require:
 1: $List\_Choice$: list of stakeholders' choices.
 2: $Constraint\_List$: list of product line constraints.
 3: $Nb\_Cst$: number of constraints.
Ensure:
 4: $List\_MUS$: list of conflicts (MUSs).
 5: Begin
 6:   for $i \leftarrow 1$ to $N$ do
 7:     if $(Stateof(Constraint_i) = Violated)$ AND
        $(Typeof(Constraint_i) = exclude$ OR $Typeof(Constraint_i) = XOR)$ then
 8:       $j \leftarrow 1$
 9:       $m \leftarrow Nb\_features(Constraint_i)$
10:       while $(j < m)$ do
11:         if $(F_j \in Constraint_i)$ AND $(F_j \in List\_Choice_i)$ then
12:           $List\_MUS \leftarrow List\_MUS \cup \{F_j, \neg F_j\}$
13:         end if
14:         $j \leftarrow j + 1$
15:       end while
16:     end if
17:     if $(Stateof(Constraint_i) = Violated)$ AND
        $(Typeof(Constraint_i) = require)$ then
18:       $j \leftarrow 1$
19:       $m \leftarrow Nb\_features(Constraint_i)$
20:       while $(j < m)$ do
21:         if $(F_j$ IN $\{required\_feature(constraint_i)\})$ AND $(F_j \in$
            $List\_Choice_i)$ then
22:           $List\_MUS \leftarrow List\_MUS \cup \{F_j, \neg F_j\}$
23:         end if
24:         $j \leftarrow j + 1$
25:       end while
26:     end if
27:   end for
28:   $Return(List\_MUS)$
29: End
```

Illustration with the running example

Based on the conflictual situations (i.e. violated constraints) presented in Table 2, the list of corresponding MUSs is computed using Algorithm 1 as illustrated in Table 3.

Table 3
Detected conflicts.

Violated constraint	Corresponding MUS
(1) Https vs. Ms	{Https, ¬*Https*}
	{Ms, ¬*Ms*}
(2) Active vs. ¬*Active*	{Active, ¬*Active*}
(3) Ms vs. Sec	{Ms, ¬*Ms*}
	{Sec, ¬*Sec*}

To resolve these conflicts, MCSs are computed using the obtained MUSs list. The whole MCSs list is presented in Table 4 which represents all the possible combinations of conflicting configuration choices. Removing the configuration choices of any combination among them from the total configuration resolves all the detected conflicts.

4.3.2. Application of substitution rules on MCSs list

The decision of which MCS to choose (resolution MCS) to resolve the identified conflict(s) is based on stakeholders' preferences expressed through the proposed list of substitution rules. This consists in applying the set of selected rules on the obtained list of MCSs where each rule allows to keep a set of specific MCSs as presented in Table 5. In fact, applying SR1 on the list of MCSs results in retaining those that eliminate the minimum number of features; that is to say, those including the minimum number of F_i, and therefore, the maximum number of $\neg F_j$. Similarly, SR2 keeps MCSs with the maximum number of F_i and the minimum number of $\neg F_j$. As to SR3 and SR4, they make use of the configuration repository where each configuration is related to a feature model, and associated to the list of stakeholders involved in that configuration. SR3 considers the past configurations of the product line related to the same stakeholder; it retains the MCSs that keep the choices related to the past configuration. SR4 keeps the configuration choices of a stakeholder with similar profile (role and status). Finally, SR5 retains the MCS that does not eliminate the stakeholder's choices including desired and undesired features.

In order to identify the resolution MCS, we propose Algorithm 2 that permits applying the list of selected rules on the MCSs list. The algorithm requires as input the list of substitution rules that are selected by the stakeholders, as well as the list of computed MCSs; it returns the MCS that resolves the conflicts. As described in lines 5–8 of Algorithm 2, the different selected substitution rules (`S_rules`) are applied one by one on the obtained MCSs list (`MCS_list`) according to the correspondence list presented in Table 5.

A substitution rule may return several MCSs or none. Therefore, line 9 checks the commonality between all returned lists present in `Result_list`.

If there are common MCSs, then the list of common MCSs (`Com_list`) is computed (line 10). If there is one common MCS, this one is returned as resolution MCS (`R_MCS`) (lines 11–12). If several MCSs are common (line 13), a priority order is then assigned by an authority (e.g. product manager) to the stakeholders involved in the conflict.

The priority order serves as a second resolution alternative to be executed if the selected rules return several or none common MCSs. In this case, as described in line 14, the resolution MCS is selected among the common list (`Com_list`) with respect to the configuration choices of the stakeholder who has the highest priority.

If there are no common MCSs (line 16), the resolution MCS is selected among the initial set of resulting MCSs (`Result_list`) with respect to the configuration choices of the prioritized stakeholder (line 17).

Algorithm 2 Apply_SubstitutionRules: Substitution rules application.

```
Require:
 1: $S\_rules$: list of rules selected by stakeholders.
 2: $MCS\_list$: list of computed MCSs.
Ensure:
 3: $R\_MCS$: conflict resolution MCS.
 4: Begin
 5:   for each (Rule ∈ S_rules) do
 6:     $Result \leftarrow apply(Rule, MCS\_list)$
 7:     $Result\_list \leftarrow Result\_list \cup Result$
 8:   end for
 9:   if ($hasCommon(Result\_list) = True$) then
10:     $Com\_list \leftarrow ComputeCommonList(Result\_list)$
11:     if ($Size(Com\_List) = 1$) then
12:       $R\_MCS \leftarrow Com\_list$
13:     else if ($Size(Com\_list) > 1$) then
14:       $R\_MCS \leftarrow Get\_Priority\_Based(Com\_list)$
15:     end if
16:   else
17:     $R\_MCS \leftarrow Get\_Priority\_Based(Result\_list)$
18:   end if
19:   $Return(R\_MCS)$
20: End
```

The list of choices contained in the retained resolution MCS are simply removed from the initial configuration. Once the conflict is resolved, the configuration is then validated and returned as the final specification.

Illustration with the running example

The rules selected by stakeholders are applied on the MCSs list according to Algorithm 2 to identify the suitable MCS. As presented in Table 6, *STK1* chose SR1-most complete product which corresponds to MCS that eliminates the minimum number of features i.e *{¬Https, Ms, ¬Active, ¬Sec}*. *STK3* chose SR5 which prioritizes his/her explicit choices which are "Active" and "Sec". Therefore, the MCS that respects these choices is the one that keeps these two features which are: *{¬Https, Ms, ¬Active, ¬Sec}* and *{Https, Ms, ¬Active, ¬Sec}*.

Here, we have one common MCS which is *{¬Https, Ms, ¬Active, ¬Sec}*, as described in line 12 of Algorithm 2, the common MCS is selected as *resolution MCS*. Therefore, as shown in Table 6, *{¬Https, Ms, ¬Active, ¬Sec}* is chosen as resolution MCS and the configuration choices contained in this MCS are then removed from the initial configuration. The final valid configuration includes to following features *{WebServer, Content, Static, Active, Protocols, Https, Performance, Sec}*

5. Usability evaluation

This section presents a usability test of the tool[1] supporting our approach. The tool allows users to configure product lines feature models according to Colla-Config process; namely, (i) express their preferences by selecting one or more substitution rules, (ii) express their configuration choices by specifying the desired and the undesired features, (iii) visualize the result of conflict resolution according to their preferences, and (iv) visualize the final valid configuration. Two main modules make up the tool: (1) configuration verification module, responsible for merging user choices and checking the resulted configuration consistency, and (2) conflict resolution module, responsible for computing the MCSs list and applying the substitution rules selected by the users

[1] Available at https://github.com/Sihembs/CollaConfig-Tool.

Table 4
MCSs list.

{Https, Ms, Active, Sec}	{Https, Ms, Active, ¬ Sec}	{Https, Ms, ¬ Active, Sec}
{Https, Ms, ¬ Active, ¬ Sec}	{Https, ¬ Ms, Active, Sec}	{Https, ¬ Ms, ¬ Active, Sec}
{¬ Https, Ms, Active, Sec}	{¬ Https, Ms, Active, ¬ Sec}	{¬ Https, Ms, ¬ Active, Sec}
	{¬ Https, Ms, ¬ Active, ¬ Sec}	

Table 5
List of substitution rules and the corresponding MCS.

Substitution rule	Corresponding MCS
SR1 most complete product	MCS that eliminates the minimum number of features
SR2 simplest product	MCS that eliminates the maximum number 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 choices of similar profile
SR5 prioritize my explicit configuration choices	MCS that respects configuration choices explicitly made by the stakeholder

Table 6
Substitution rules application.

Stakeholder	Selected rule	Application result	Resolution MCS
STK1	SR1	{¬Https, Ms, ¬Active, ¬Sec}	
STK2	–	–	{¬Https, Ms, ¬Active, ¬Sec}

STK3	SR5	{¬Https, Ms, ¬Active, ¬Sec}
		{Https, Ms, ¬Active, ¬Sec}	

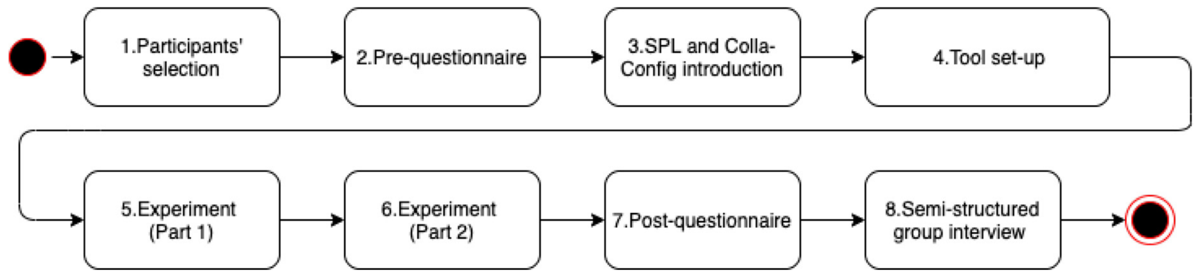


Fig. 6. Colla-Config usability test process.

in order to determine the final valid configuration. Moreover, the Colla-Config tool makes use of a database to store users profile information along with their past configurations. As to feature models, they are stored in XML format.

The main idea of the usability evaluation is to test the tool usability to support the Colla-Config approach, and to gain insight into how easy or difficult it is to follow and understand the Colla-Config approach using the tool. The experimentation process comprises four main activities which are: (1) research question formulation, (2) experimental protocol definition, (3) experimental protocol execution, and (4) results interpretation. To guide the usability evaluation, we defined the following research question. **RQ**. What is the level of usability of Colla-Config in supporting the collaborative configuration of product lines?

In order to address this question, we conducted a usability test by following the ISO/IEC Common Industry Format (CIF) for usability tests (ISO/IEC, 2006). The following subsections detail the evaluation process: the two first ones focus on the protocol along with its execution, and the last one is dedicated to the results.

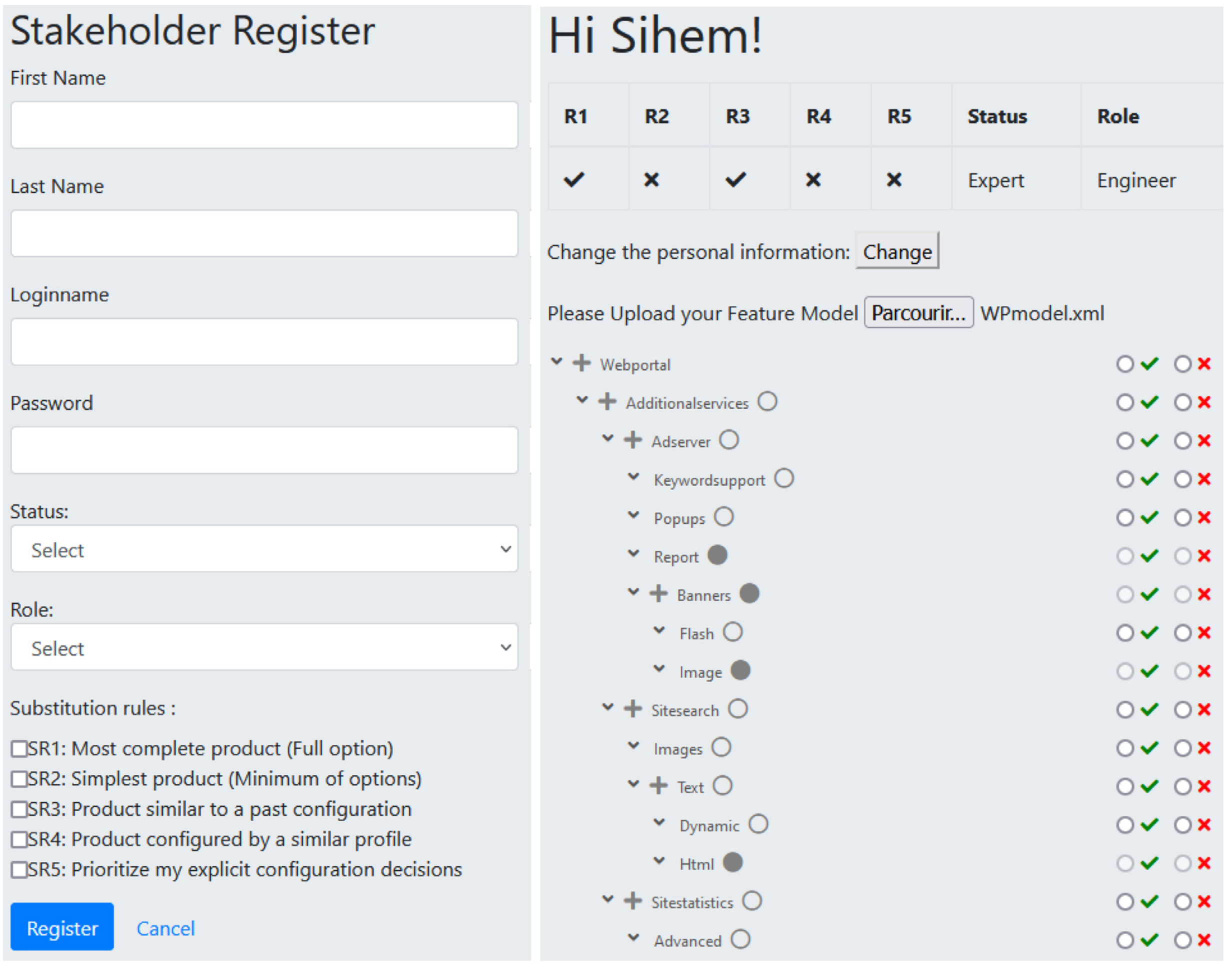
5.1. Usability test process

The experimentation was planned for 3 h,[2] during which eleven Ph.D. students in computer science accepted to participate in the evaluation process. As Fig. 6 shows, the usability test was designed as a process with eight sequential activities.

[2] All the experiment artifacts are available on this link https://bit.ly/3zhs60j.

Participants' selection. Eleven Ph.D. students from different research laboratories *(CRI in France, RIADI in Tunisia and GIDTIC in Colombia)* voluntarily participated in this test. All Ph.D. students had a varied knowledge about SPLE. Some of them had followed a product lines lecture while others were working on SPLE on their PhD thesis.

Pre-questionnaire (15 min). We requested the participants to complete a pre-questionnaire related to their background and experience with SPLE. The pre-questionnaire showed that participants had basic knowledge on product line collaborative configuration but lacked knowledge on conflict management. In fact, we asked them about their number of years of experience in the field of SPLE, and their knowledge about product lines configuration, collaborative configuration and conflict notions. The last question of the questionnaire was open-ended about conflict resolution concept (see Appendix A). Collected data revealed that (i) 73% of the participants are new to SPLE (less than one year experience), 18% of them (two participants) have an experience between 1 and 5 years, and only one participant has between 5 and 10 years of experience (ii) most of them (82%) either heard about product lines configuration or know what it is, and (iii) 91% of them answered correctly to the question related to collaborative configuration. However, 63% of them were not able to identify the answer related to conflict definition. Regarding the open-ended question about conflict resolution, few participants recognized that they "don't know" what it is; some of them tried to guess or give some useful information as the following reported answers: "Naively, it would appear that if the conflict is due to two choices introducing incompatibilities as a consequence of their inclusion, then the solution would either involve prioritizing one over the



Stakeholder Register

First Name

Last Name

Loginname

Password

Status: Select

Role: Select

Substitution rules :

- ☐SR1: Most complete product (Full option)
- ☐SR2: Simplest product (Minimum of options)
- ☐SR3: Product similar to a past configuration
- ☐SR4: Product configured by a similar profile
- ☐SR5: Prioritize my explicit configuration decisions

Hi Sihem!

R1	R2	R3	R4	R5	Status	Role
✓	✗	✓	✗	✗	Expert	Engineer

Change the personal information: Change

Please Upload your Feature Model Parcourir... WPmodel.xml

- Webportal
  - Additionalservices
    - Adserver
      - Keywordsupport
      - Popups
      - Report ●
      - Banners ●
        - Flash
        - Image ●
    - Sitesearch
      - Images
      - Text
        - Dynamic
        - Html ●
    - Sitestatistics
      - Advanced

(a) (b)

Fig. 7. Configuration interface of Colla-Config.

other, or it would involve trying to find alternative choices that remove the conflict through some analysis of the adequacy of the alternatives. How to do so in practice, however, is not something I'm familiar with". and "I think that conflict resolution in the collaborative configuration allows to choose one solution among others". Only one participant gave an interesting answer: "Conflict resolution is to apply algorithms to find consistency sets and suggest and guide the configuration process". Finally, it is worth mentioning that a high experience with SPLE does not mean a knowledge about conflicts in collaborative configuration.

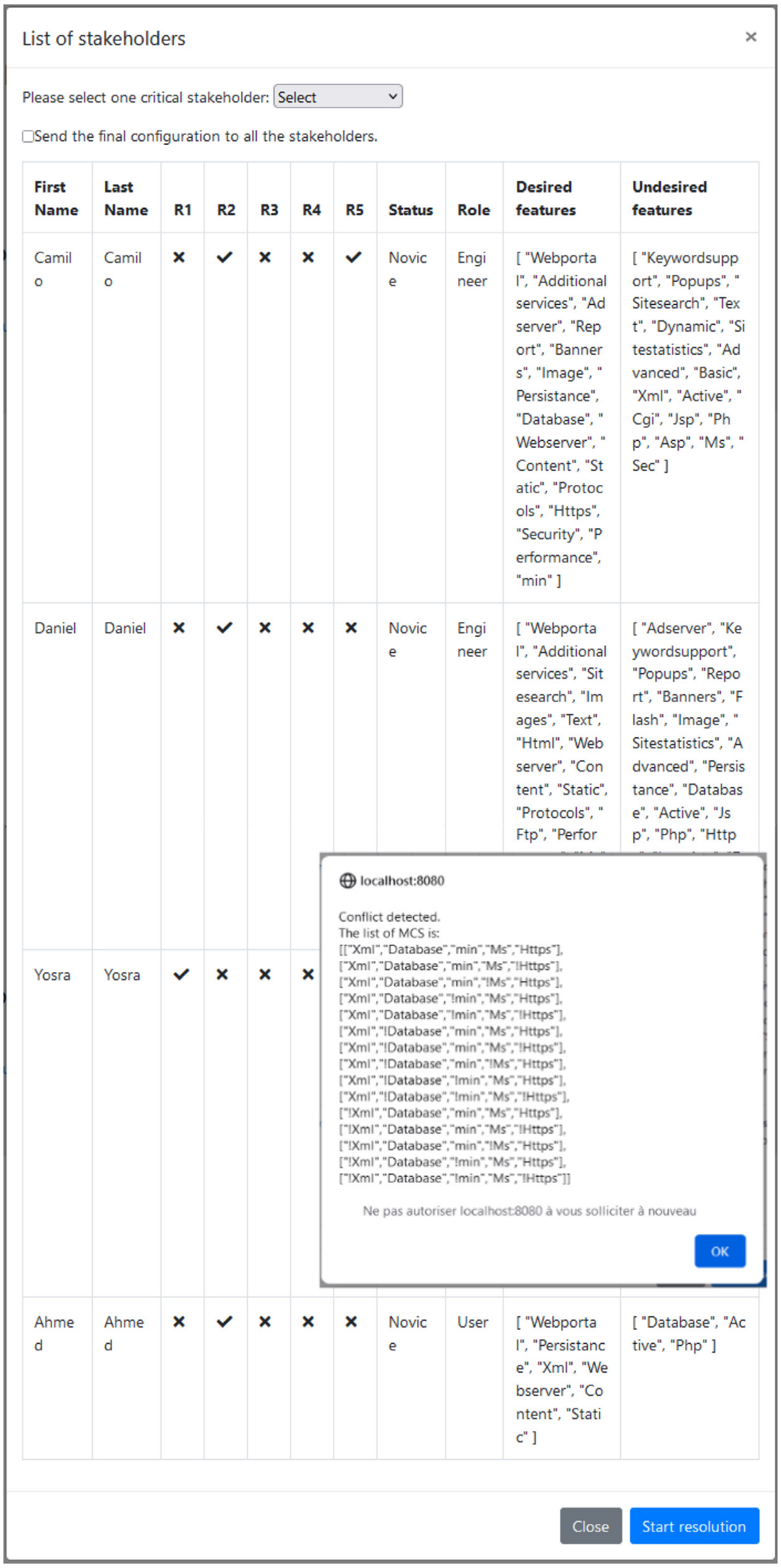
SPL and Colla-Config introduction (1 h). We designed a presentation session about the main concepts needed to conduct this experiment. This introduction was important for the participants where topics such as product line engineering, feature modeling and product line collaborative configuration were introduced. We also developed a small example of the use of Colla-Config to explain its principle.

Tool set-up (15 min). During this session, the participants were introduced to a document that presented a series of steps to set up a collaborative configuration using the Colla-Config tool.

Part one of the experiment (Limit. 40 min). We shared with all participants a document with the material needed for this first part of the experiment. They were then requested to complete four tasks, while an administrator was observing and guiding the participants during the experiment. The tasks in the experiment part were product configuration: (i) registering and selecting desired substitution rules, (ii) configuring the Web portal model, (iii) detecting and resolving conflicts, and (iv) getting the final result.

The different participants used the interface (a) and (b) of Fig. 7 to select rules and express their configuration choices. The participant designated as product manager was in charge of selecting the prioritized participant and launching the conflict detection and resolution process using the interface presented in Fig. 8.

Part two of the experiment (Limit. 50 min). This second part of experiment consisted in configuring a product using another different collaborative configuration approach. Indeed, we need participants to be able to express their opinion about the Colla-Config tool and their satisfaction about the resolution strategy it implements. This could not be done properly if they have not tried another way to do it. We chose to use the method from Mendonca et al. (2008), which consists in configuring a product in a sequential way according to a workflow-based configuration plan, where each participant configures a module of the Web portal model. This configuration principle called "staged configuration" was initially proposed in Czarnecki et al. (2005) and then improved in Mendonca et al. (2008), which proposes a detailed feature model decomposition process and the creation of a configuration plan. In this part of the experiment, the tasks were: (i) assigning the modules to the participants, (ii) configuring the assigned module, and (iii) resolving conflicts and returning the product specification.


List of stakeholders ×

Please select one critical stakeholder: Select

☐ Send the final configuration to all the stakeholders.

First Name	Last Name	R1	R2	R3	R4	R5	Status	Role	Desired features	Undesired features
Camilo	Camilo	✗	✓	✗	✗	✓	Novice	Engineer	["Webportal", "Additional services", "Adserver", "Report", "Banners", "Image", "Persistance", "Database", "Webserver", "Content", "Static", "Protocols", "Https", "Security", "Performance", "min"]	["Keywordsupport", "Popups", "Sitesearch", "Text", "Dynamic", "Sitestatistics", "Advanced", "Basic", "Xml", "Active", "Cgi", "Jsp", "Php", "Asp", "Ms", "Sec"]
Daniel	Daniel	✗	✓	✗	✗	✗	Novice	Engineer	["Webportal", "Additional services", "Sitesearch", "Images", "Text", "Html", "Webserver", "Content", "Static", "Protocols", "Ftp", "Perfor	["Adserver", "Keywordsupport", "Popups", "Report", "Banners", "Flash", "Image", "Sitestatistics", "Advanced", "Persistance", "Database", "Active", "Jsp", "Php", "Http
Yosra	Yosra	✓	✗	✗	✗					
Ahmed	Ahmed	✗	✓	✗	✗	✗	Novice	User	["Webportal", "Persistance", "Xml", "Webserver", "Content", "Static"]	["Database", "Active", "Php"]

```
localhost:8080

Conflict detected.
The list of MCS is:
[["Xml","Database","min","Ms","Https"],
["Xml","Database","min","Ms","!Https"],
["Xml","Database","min","!Ms","Https"],
["Xml","Database","!min","Ms","Https"],
["Xml","Database","!min","Ms","!Https"],
["Xml","!Database","min","Ms","Https"],
["Xml","!Database","min","Ms","!Https"],
["Xml","!Database","min","!Ms","Https"],
["Xml","!Database","!min","Ms","Https"],
["Xml","!Database","!min","Ms","!Https"],
["!Xml","Database","min","Ms","Https"],
["!Xml","Database","min","Ms","!Https"],
["!Xml","Database","min","!Ms","Https"],
["!Xml","Database","!min","Ms","Https"],
["!Xml","Database","!min","Ms","!Https"]]

Ne pas autoriser localhost:8080 à vous solliciter à nouveau
```

Fig. 8. Conflict detection and resolution interface of Colla-Config.

Here, the participants configured the assigned modules by selecting the desired features (undesired features are not considered in Mendonca et al. approach). The participants designated a product manager who was in charge of assigning the modules to the participants and detecting conflicts afterward. Fig. 9 shows the feature model which is divided into nine configuration spaces (i.e. one or more modules) and the stakeholders that are responsible for configuring each one of them. This configuration process should conform to the defined plan specifying that the stakeholders Sh3 and Sh11 are responsible for the Ws configuration space,

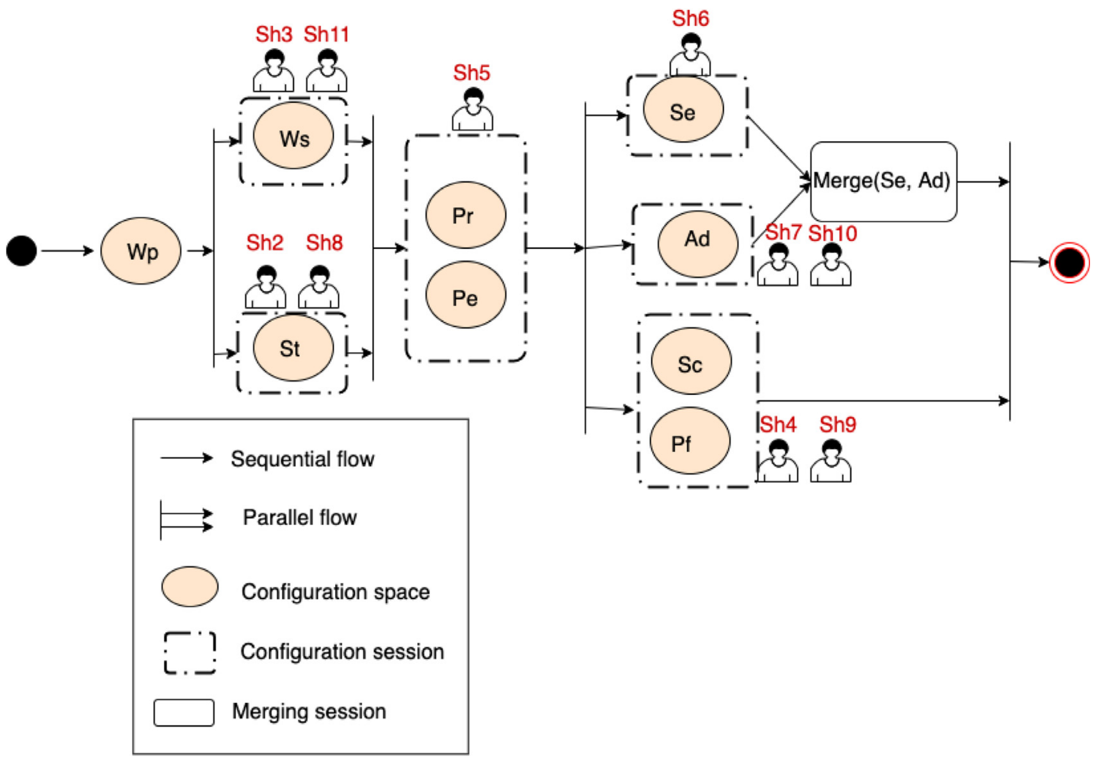


Fig. 9. Configuration plan according to Mendonca et al. (2008) approach.

while Sh2 and Sh8 are responsible for St. All these stakeholders cannot start the configuration process till the product manager finishes configuring the Wp space. In turn, the stakeholder Sh5, responsible for Pr and Pe spaces, needs to wait until Sh2, Sh8, Sh3 and Sh11 finish their configuration. The remaining of the figure is read similarly. As proposed in Mendonca et al. (2008), conflicts were systematically resolved following the configuration plan order depicted in Fig. 9.

Post-questionnaire (15 min). The participants were submitted to a post-questionnaire with questions about (i) the experiment environment, (ii) the overall satisfaction, (iii) the tool performance, (iv) general concerns, and (v) specific questions about the Colla-Config principle. The post-questionnaire is made up of 15 questions (c.f. Appendix B). Six among them are used as a source of information for threats to validity presented in Section 7. The 9 questions about satisfaction are based on four usability dimensions: (1) 2 questions related to ease of use, (2) 2 questions related to ease of learning, (3) 2 questions related to ease of remembering, and (4) 3 questions related to subjective satisfaction. These questions are formulated with 5 point likert scale suitable for measuring such characteristics (Nielsen, 2010), where 1 and 2 ratings express negative opinion or judgment, a 3 value rating generally expresses neutrality or moderation, while 4 and 5 express positive opinion or judgment answers.

Semi-structured interview (30 min). We asked participants four open questions about the tool usability, and we recorded their answers. The questions were:

- What do you like most about the tool, about the collaborative configuration approach, and about the conflict resolution approach?
- What do you do not like about the tool, about the collaborative configuration approach, and about the conflict resolution approach?
- In your opinion, what would be the main area to improve the tool and the collaborative configuration process with automatic conflict resolution?
- In your opinion, what would be the novelty to be created in the tool (the development axis to be brought to the tool) and in the collaborative configuration process?

All responses were analyzed to answer the research question using thematic analysis driven by an inductive approach as we aimed at knowing independent thoughts of each participant about the proposed collaborative configuration approach, the conflict resolution strategy and his/her experience with the tool as well as his/her opinion about the novelty and area of improvement. First, we went through all the collected data. Then, we highlighted expressions and sentences to come up with codes. The latter were clustered into themes, which were afterward reviewed. Finally, we draw the conclusions.

5.2. Metrics

Usability has three main attributes which are effectiveness, efficiency, and satisfaction. The metrics used to measure them and adopted in the current work are detailed in the following.

- ***Effectiveness***: is the ability to produce a desired result or the ability to produce the desired output. To measure effectiveness, we recorded completion rate, errors and assists.

  1. *Completion rate*: consists in measuring the rate of completion of each task of the configuration process by the different participants. It is generally identified as a binary measure of the success of a task (coded 1) or the failure of a task (coded 0). The completion rate of a task is therefore calculated by dividing the number of participants who successfully completed

the task by the total number of participants as shown by expression (1).

$$Completion\_Rate_{task} = \frac{number\ of\ participants\ who\ successfully\ completed\ the\ task}{total\ number\ of\ participants} \quad (1)$$

The completion rate per participant is computed as the number of tasks he/she successfully completed divided by the number of tasks he/she undertook, as shown by expression (2).

$$Completion\_Rate_{participant} = \frac{number\ of\ tasks\ successfully\ completed\ by\ the\ participant}{total\ number\ of\ tasks} \quad (2)$$

The overall completion rate is the average of completion rates per participant as given by expression (3).

$$Completion\_Rate = \frac{\sum_{i=1}^{Nb\_participants} Completion\_Rate_{participant_i}}{total\ number\ of\ participants} \quad (3)$$

2. *Errors*: are defined as a task completed wrongly or not completed. Therefore, for each participant, we identified the list of accomplished tasks and those not accomplished, and therefore recorded the number of errors. The overall errors metric value is calculated as the average of recorded participants errors as shown by expression (4).

$$Errors\_Number = \frac{\sum_{i=1}^{Nb\_participants} Errors\ number_{participant_i}}{total\ number\ of\ participants} \quad (4)$$

3. *Assists*: are defined as verbal help given by the administrator to guide the participants to complete the different tasks. We therefore recorded the number of assists per participant, namely the number of asked questions and help requested. The overall assists metric value is computed as the average number of assists for all participants (c.f. expression (5))

$$Assists\_Number = \frac{\sum_{i=1}^{Nb\_participants} Assists\ number_{participant_i}}{total\ number\ of\ participants} \quad (5)$$

- ***Efficiency***: is the ability to produce the desired product without wasting resources and/or time. To measure the efficiency, we recorded task time and completion rate efficiency.

(1) *Task time*: is the amount of time used to complete each task by each participant, defined for a task as $Task\_Time = End\_Time - Start\_Time$. The mean time for a $Task_j$ is computed as the average of the times spent by participants to complete the task in question as given by expression (6), where $St_{ij} = 1$ if the $Task_j$ is completed by the $participant_i$ and $St_{ij} = 0$ if $Task_j$ is not completed by the same participant.

$$Mean\_Task\_Time_j = \frac{\sum_{i=1}^{Nb\_participants} Task\_Time_j\ of\ the\ participant_i \times St_{ji}}{total\ number\ of\ participants} \quad (6)$$

The overall mean task time is calculated as the average of mean time of all tasks as shown by expression (7).

$$Mean\_Task\_Time = \frac{\sum_{j=1}^{Nb\_tasks} Mean\_Task\_Time_j}{total\ number\ of\ tasks} \quad (7)$$

The mean total task time is calculated as the average of the total time spent by each participant in completing all tasks divided by the number of participants as given by the expression (8).

$$Mean\_Total\_Task\_Time = \frac{\sum_{i=1}^{Nb\_participants} \sum_{j=1}^{Nb\_tasks} Task\_Time_j\_participant_i}{total\ number\ of\ participants} \quad (8)$$

(2) *Completion rate efficiency*: is measured as shows expression (9) by dividing the mean completion task rate (c.f. expression (3)) by the mean task time (c.f. expression (7)).

$$Completion\_Rate\_Efficiency = \frac{Completion\_Rate}{Mean\_Task\_Time} \quad (9)$$

- ***Satisfaction***: we used the post-questionnaire results and measured the different participants perception of ease of use, ease of learning, ease of remembering, and subjective satisfaction. Since we use a 5 point scale, the score of each dimension is calculated as follows: first the average of responses values to each question related to the dimension answers are computed for each participant. Then, the average for all participants related to each dimension is calculated as illustrated by the expression (11).

$$Score\_Dimension\_participant = \frac{\sum_{i=1}^{Nb\_Q\_dimension} Score\_Question_i}{number\ of\ dimension\ questions} \quad (10)$$

$$Score\_Dimension = \frac{\sum_{j=1}^{Nb\_participants} Score\_Dimension\_participant_j}{total\ number\ of\ participants} \quad (11)$$

Moreover, we took advantage of the semi-structured interview results.

5.3. Results

Table 7 summarizes the experiment tasks and provides few more details about the activities asked to both participants and product manager. The main results using the metrics presented in Section 5.2, are described in the following. The first sub-section covers the effectiveness and the efficiency, while the second one is dedicated to satisfaction results.

Performance Results. Both the ten participants and the product manager, successfully completed each of the tasks that were assigned to them. Therefore, the $Completion\_Rate_{task} = 100\%$ according to the expression (1); the $Completion\_Rate_{participant} = 100\%$ according to the expression (2); and the overall $Completion\_Rate = 100\%$ according to the expression (3). There were no errors since all participants completed the asked tasks properly as illustrated in Table 8. Yet, while six of the ten participants completed all tasks without assistance, four of them asked for help. All the requests were related to $Task2_{part1}$ (configuration according to Colla-Config approach, c.f. Table 7), which required

Table 7
Experiment tasks.

Experiment part	Task_id	Executed by	Task activities
Part1: related to Colla-Config approach	$Task1_{part1}$	Participant	**Register**: record task start time, enter personal information and select desired substitution rules, record task end time
	$Task2_{part1}$	Participant	**Configure**: record task start time, log-in, load the Web Portal model XML file, select desired and undesired features, take screen-shots of the configuration, save and close, log-out, record task end time
	$Task3_{part1}$	Product manager	**Detect and resolve conflicts**: record task start time, log-in, "Merge" participants configurations, select prioritized participant, "Start Resolution", visualize the displayed list of possible corrections, visualize the retained correction, visualize the final configuration, record task end time
	$Task4_{part1}$	Participant	**Get the final result**: record task start time, log-in, load the Web Portal model XML file, visualize the final configuration, record task end time
Part2: related to Mendonca et al. (2008) approach	$Task1_{part2}$	Product manager	**Assign configuration modules**: record task start time, assign each participant a module to configure, communicate the configuration workflow to all participants, record task end time
	$Task2_{part2}$	Participant	**Configure**: record task start time, log-in, load the Web Portal model XML file, select desired features related to the assigned modules only, take screen-shots of the configuration, communicate the configuration choices to the product manager, save and close, log-out, record task end time
	$Task3_{part2}$	Product manager	**Resolve conflicts according to the configuration workflow**: record task start time, 1. Check participants choices and identify the conflicts according to the workflow, communicate to participants the outcome of the conflict resolution process, communicate to participants the final configuration, record task end time

Table 8
Participants' performance result summary.

Measure	Task completion rate	Errors number	Assists number	Mean total task time	Mean task time	Completion rate efficiency
10 participants	100%	0	0.4	17.2	4.3	24.04%
Product manager	100%	0	2	47	15.66	6.38%

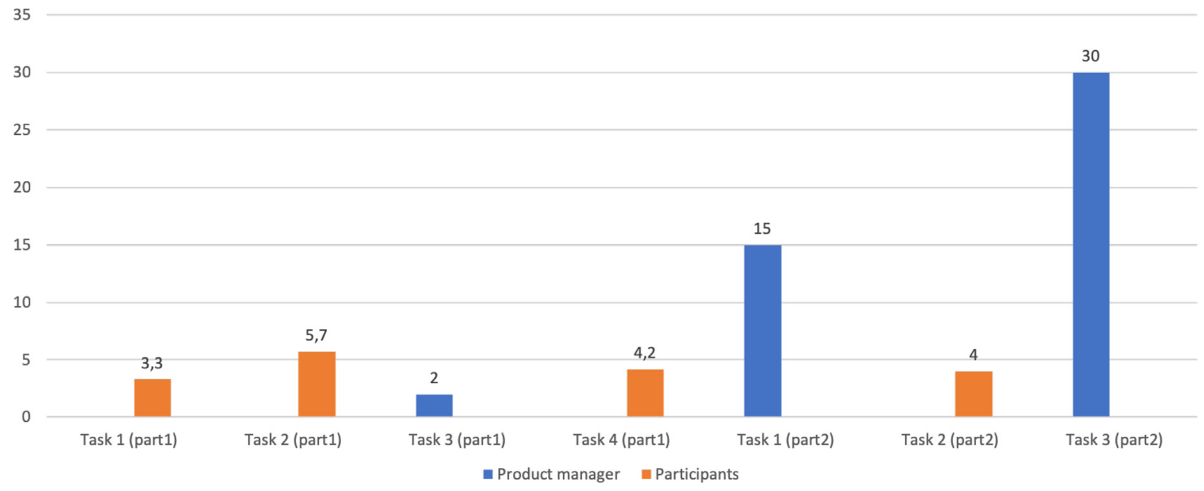


Fig. 10. Participants' average time (minutes) to complete each task.

in average 6 min to be completed by participants as shows Fig. 10. The *Assists_Number* for participants is therefore equal to 0.4 by applying the formula of the expression (5). As to the product manager, *Assists_Number* = 2 since he requested assistance for $Task1_{part2}$ (configuration modules assignment) as well as for $Task3_{part2}$ (conflict resolution) which was the most complicated task. As a matter of fact, the product manager spent a mean of 30 min to complete this task (see Fig. 10). Fig. 10 also shows that the product manager spent only 2 min in the development of $Task3_{part1}$. This task was about selecting the prioritized participant and merging choices to start conflict detection and resolution; $Task2_{part2}$ focused on configuring only the assigned module, the participants only spent approximately 4 min in selecting the set of desired features. As presented in Table 8, the *Mean_task_time* is equal to 4.3 min, while the mean time to complete all tasks, computed using expression (8), was approximately 17 min for each participant and 47 min for the participant who took the role of product manager. The completion rate efficiency is obtained by applying the expression (9); it is equal to around 24% task completion per minute for the participants and 15% for the product manager.

In the context of product lines collaborative configuration, there are no previous works that tested the usability of their tools, neither benchmarks do exist to the best of our knowledge. In such case, we may adopt a 78% for the completion rate metric as benchmark (Sauro and Lewis, 2012). The completion rate of all Colla-Config tasks undertaken by all participants is 100%. This high completion rate is partly owing to the way the test is

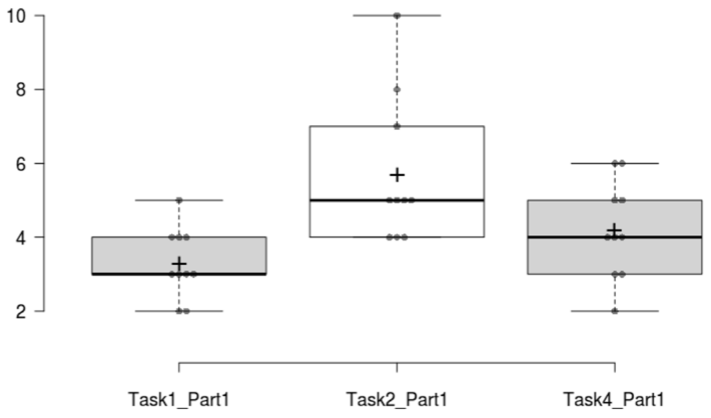


Fig. 11. Colla-Config tasks completion time box plot.

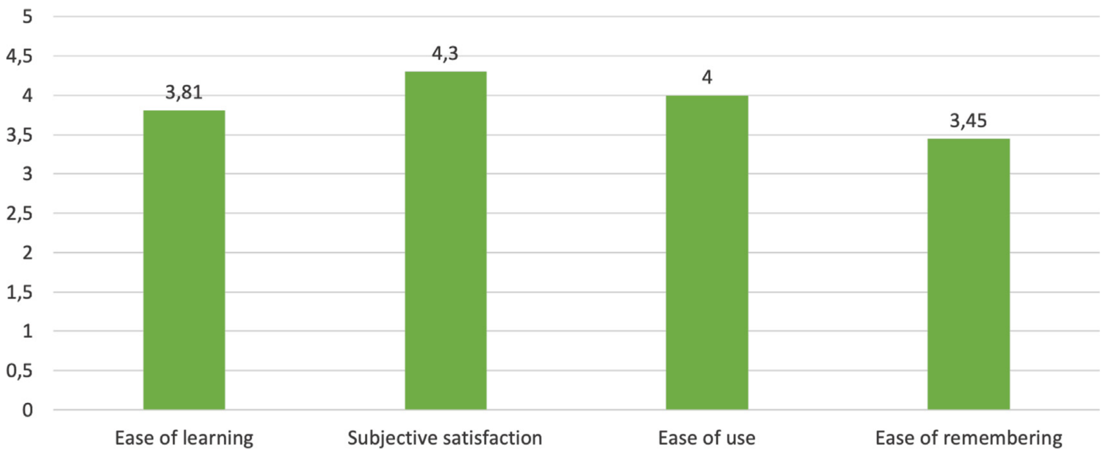


Fig. 12. Participants' satisfaction question average results.

designed; especially, (i) we planned an example of Colla-Config use and explained its principle during the "SPL and Colla-Config introduction" activity, and (2) we shared documents explaining the experiment (part one and part two) steps, which contributed to make the tasks easy to perform by the participants. According to Sauro and Lewis (2012), the completion rate of 100% means that we can be 95% confident between 75% and 100% of all users will be able to complete all tasks when using our tool for collaborative configuration of product lines. However, the small number of participants calls for caution before making any inference. Even though no errors in accomplishing the tasks were recorded while the acceptable average error rate per task is 0.7 (Sauro, 2010), 40% of participants asked assistance for the same task ($Task2_{part1}$) among the four ones; this means that more information and clear instructions should be incorporated as help provided to the users of the related task interface. Overall, we conclude that the tool is able to produce the desired output.

Fig. 11 illustrates the box plot chart for the three tasks related to Colla-Config and undertaken by the ten participants. The fourth one was the responsibility of the product manager who took 2 min to accomplish it. The geometric mean, as recommended to be considered in the case of small size samples (Sauro and Lewis, 2010), is 3.17 min for $Task1_{part1}$, 5.43 min for $Task2_{part1}$, and 4.00 min for $Task3_{part1}$. Fig. 11 shows that all participants executed the tasks in a duration less than two times the geometric mean. Compared to an expert, it usually takes normal users two or three times longer to perform a task (Bevan, 2006). In the case of Colla-Config, the expert completed $Task1_{part1}$ in 1 min, $Task2_{part1}$ in 2 min, and $Task3_{part1}$ in 2 min. All participants performed the tasks in the benchmark range, except three ones. These latter were among the four participants who asked assistance for the task $Task2_{part1}$ and the time of helping them was included in their recorded task time.

The completion rate efficiency is related to productivity by providing the rate at which services are produced. The measure, expressed as percent task completion per minute, trades off time against accuracy and completeness; but it is not always easy to interpret (Bevan, 2006). In this case of Colla-Config related experiment, we estimate that a completion rate efficiency around 25% for participants and 50% for the product manager are very acceptable, knowing that all participants were novice to SPL collaborative configuration. Overall, we conclude that the tool is able to produce the desired output without wasting time and/or resources.

Satisfaction Results. As mentioned earlier, 9 questions from the post-questionnaire were formulated according to 5 point likert scale to measure the ease of use, ease of learning, the ease of remembering and the subjective satisfaction about the Colla-Config tool, respectively 2, 2, 2 and 3 questions. In such scale, an average of 4 or higher is considered as a "good value". We analyzed the 9 questions and computed the average score of each dimension according to the expression (11). The result is presented in Fig. 12, showing that the highest satisfaction result was about the *subjective satisfaction* regarding Colla-Config with a mean of 4.3 and the *ease of use* of Colla-Config with a mean of 4. As to ease of learning and ease of remembering, the average scores are respectively 3.81 and 3.45. This means that the Colla-Config team is recommended to focus in a future design of the tool on aspects that improve these two characteristics.

It is also worth highlighting that for questions about the configuration principle and conflict resolution, ten out of eleven participants (90%) preferred the configuration without role assignment and without pre-defined order (i.e. Colla-Config principle) to the workflow-based configuration (i.e. Mendonca et al. (2008) principle). Moreover, nine out of eleven participants (81%) chose the Colla-Config conflict resolution strategy.

Finally, the semi-structured interview responses showed that in general, the participants liked the ease of use of the tool and appreciated the way the Colla-Config approach worked. They mentioned that "it is a good strategy to consider users preferences and ensure fairness between their choices in the conflict resolution process". There were also some recommendations to improve the tool and the approach regarding: (i) interactivity: most of the participants (six out of eleven participants i.e. 54%) suggested improving the tool so that it allows users to interact in real time and share their choices at each step of the configuration; for example, one participant indicates that "it would be nice to have a mechanism to pose objections to the final product if one feels a choice that was retained should not have been selected ..., which perhaps might ask other participants to then reevaluate their choices when faced with these objections"; (ii) conflict resolution: some participants (two out of eleven participants i.e. 18%) suggested to give users the opportunity to review their choices before proceeding with automatic resolution, which may reduce the number of detected conflicts; we report as example the following statement: "I think a possible avenue for future expansion (and which would be particularly useful when working on configurations for very large feature models) would be to have real-time indications of how the other choices that have already been made would affect the choices one does, as it would be perhaps be best to be aware of the conflicts before one even completes or finishes the configuration"; another interesting statement was: "stakeholders that are slightly motivated by selecting an attribute (they consider it is better to have it, but not essential at all) could reconsider their choice if they know that it considerably adds constraints on possible features. This might reduce the number of conflicts beforehand"; (iii) accessibility: some participants (three out of eleven participants i.e. 27%) also proposed to transform the tool into a Web application to easily open the application and avoid some installation settings; an example of such suggestions are the following: "to make it a web-based application with synchronous configurations" and "it can be hosted in a web application or mobile phone application to let users using it".

6. Scalability evaluation

In the previous section, we experimentally evaluated the Colla-Config approach by conducting a usability test designed according to ISO/IEC 25062:2006 Common Industry Format for usability tests. The results showed a preliminary evidence of (i) the approach feasibility, and (ii) the tool ability to properly support the SPL collaborative configuration. In this section, we discuss the complexity of the presented algorithms and the whole system to provide an idea about the scalability of the proposed solution. To that end, we wrapped up the core process of Colla-Config in the Algorithm 3, requiring as input (i) the product line to configure, as a tree of features, (ii) the product line and related domain constraints list, (iii) the list of stakeholders involved in the collaborative configuration, and (iv) the list of substitution rules among them each stakeholder selects the ones that express his/her preferences. As output, the algorithm provides a valid configuration that takes into account the users expressed preferences and roles.

The parameters that may influence the performance of the provided solution and with which we will compute and express the complexity of the algorithms are:

- *Nb_F*: the number of features of the product line to configure.
- *Nb_Cst*: the number of constraints including the product line model constraints as well and the related domain ones.
- *NbF_Cst*: the number of features by constraint.
- *Nb_Users*: the number of users (stakeholders) involved in the collaborative configuration process.
- *Nb_Rules*: the number of substitution rules.

Algorithm 3 CollaConfig: Core process of Colla-Config approach.

```
Require:
1: PL_Fs: the product line to configure as a tree of features.
2: Constraint_List: list of product line constraints.
3: User_List: list of users (stakeholders) involved in the configuration.
4: SubstitutionRules_List: list of substitution rules among which
   users choose their preferences.
Ensure:
5: Valid_Config: list of features constituting the valid configuration according to users preferences.
   Parameters:
6: Nb_F: number of features of the product line.
7: Nb_Cst: number of constraints.
8: NbF_Cst: number of features by constraint. Nb_Users: number of users (stakeholders).
9: Nb_Rules: number of substitution rules.
10: Begin
11:     for each($user_i \in User\_List$) do
12:         $L\_SubstitutionRules_i \leftarrow Select\_SR(SubstitutionRules\_List)$
13:         $L\_Choice_i \leftarrow Configure(PL\_Fs)$
14:     end for
15:     $S\_rules \leftarrow MergeSRules(\{L\_SubstitutionRules_i\}\ i = 1..Nb\_Users)$
16:     $List\_Choice \leftarrow MergeConfigs(\{L\_Choice_i\}\ i = 1..Nb\_Users)$
17:     if ($Verify(List\_Choice) =$ false) then
18:         $List\_MUS \leftarrow Compute\_MUSs(List\_Choice, Constraint\_List, Nb\_Cst)$
19:         $MCS\_list \leftarrow Compute\_MCSs(List\_MUS)$
20:         $Res\_MCS \leftarrow Apply\_SubstitutionRules(S\_rules, MCS\_list)$
21:         $Valid\_Config \leftarrow (List\_Choice, Res\_MCS)$
22:     else
23:         $Valid\_Config \leftarrow (List\_Choice)$
24:     end if
25:     return($Valid\_Config$)
26: End
```

The lines 11 to 14 of Algorithm 3 are related to the configuration step, were each user freely selects none, one or some of substitution rules allowing him/her to express his/her preferences, represented by $L\_SubstitutionRules_i$ list, and freely configures the product line where the chosen features are represented by L_Choice_i list. This step does not induce any computation complexity as it consists in building lists by walking through the set of rules and the xml feature model tree. Hence, we focus on the following steps regarding the complexity of the solution.

In line 15, the lists of selected substitution rules chosen by the users ($\{L\_SubstitutionRules_i\}$) are merged to form a unique list with no duplications (*S_rules*); this implies to go through each $L\_SubstitutionRules_i$ having as a maximum *Nb_Rules* elements. The complexity of this step is therefore given by expression (12).

$$Compl1 = O(Nb\_Rules \times Nb\_Users) \tag{12}$$

Similarly, the merged list of users configurations (*List_Choice*) is obtained by walking through each user choices L_Choice_i having as maximum elements the number of the product line features which is Nb_F. Hence, the complexity of the line 16 statement is as shown by expression (13).

$$Compl2 = O(Nb\_F \times Nb\_Users) \quad (13)$$

The statement of line 17 consists in checking the consistency of the configuration. This goes back to the idea of detecting if there are conflicts in the configuration ($List\_Choice$); in other words and in the worst case, it is question of checking that each constraint is verified and does not imply a conflict. Since some constraints, such as XOR ones, may involve a set of features (NbF_Cst) and each feature requires walking through the configuration $List\_Choice$, the complexity of this step is given by expression (14).

$$Compl3 = O(Nb\_Cst \times NbF\_Cst \times Nb\_F) \quad (14)$$

The statement of line 18 corresponds to an invocation of the Algorithm 1 (Compute_MUSs()) which computes the list of MUSs $List\_MUS$ by walking through the list of constraints features against the configuration ($List\_Choice$) in the same manner as detecting conflicts of the previous step. The complexity of this step is therefore given by expression (15).

$$Compl4 = O(Nb\_Cst \times NbF\_Cst \times Nb\_F) \quad (15)$$

As to the statement of line 19, it consists in computing MCSs from $List\_MUS$. First, it is question of building a list of involved features based on the MUSs list. The number of elements is less or equal to the number of constraints multiplied by the maximum of constraints features numbers ($Nb\_Cst \times NbF\_Cst$). Then, the MCSs are progressively constructed by walking through the set of MUSs features and adding to the preceding sets one *feature* and its negation $\neg feature$. The complexity of such process is as shown by expression (16). As the number of MCSs is atmost equal to $2^{Nb\_Cst \times NbF\_Cst-1}$, we introduce a new parameter Nb_MCSs to make the discussion easier to follow.

$$Compl5 = O(2^{Nb\_Cst \times NbF\_Cst}) = O(Nb\_MCSs) \quad (16)$$

The next step consists in applying the substitution rules S_rules to the MCS_List as described by the Algorithm 2 (Apply_SubstitutionRules()); mainly MCSs are browsed to be checked against each rule. Hence, the complexity of this step is given by the expression (17).

$$Compl6 = O(Nb\_Rules \times Nb\_MCSs) \quad (17)$$

The proposed solution complexity is the sum of the complexities given in expressions (12), (13), (14), (15), (16), and (17) as shows expression (18).

$$\begin{aligned} &Colla - Config\ Complexity \\ &\quad = Compl1 + Compl2 + Compl3 + Compl4 + Compl5 + Compl6 \\ &\quad = O(Nb\_Rules \times Nb\_Users) + O(Nb\_F \times Nb\_Users) \\ &\quad + O(Nb\_Cst \times NbF\_Cst \times Nb\_F)+ \\ &O(Nb\_Cst \times NbF\_Cst \times Nb\_F) + O(Nb\_MCSs) \\ &\quad + O(Nb\_Rules \times Nb\_MCSs) \end{aligned} \quad (18)$$

The final complexity, obtained after simplification (c.f. expression (19)), shows that it is composed of four terms.

$$\begin{aligned} &Colla - Config\ Complexity = O(Nb\_Rules \times Nb\_Users) \\ &\quad + O(Nb\_F \times Nb\_Users)+ \\ &O(Nb\_Cst \times NbF\_Cst \times Nb\_F) + O(Nb\_Rules \times Nb\_MCSs) \end{aligned} \quad (19)$$

Aside from the fourth term, the complexity of the three first ones is polynomial. Indeed, for product lines to be configured like some industrial ones represented by a model including 20,000 features or more, with a great number of constraints, involving a great number of users having the possibility to choose some among a great number of substitution rules to express their preferences, the performance of Colla-Config part depending on such parameters remains very acceptable: the complexity of the first term ($O(Nb\_Rules \times Nb\_Users)$) is quadratic; the complexity of the second one ($O(Nb\_F \times Nb\_Users)$) is quadratic; and the complexity of the third term ($O(Nb\_Cst \times NbF\_Cst \times Nb\_F)$) is cubic. Hence, their influence is represented in an increasing variation that is to the most cubic; meaning that the complexity of this part remains under control.

Table 9
Colla-Config execution time against the number of violated constraints.

Nb of violated constraints	nb of MCSs	Colla-Config runtime (in s)
1	3	0.018
2	9	0.020
3	15	0.022
4	30	0.023
5	98	0.025
6	189	0.026
7	396	0.032
8	792	0.043
9	1584	0.067
10	3168	0.153
11	6336	0.369
12	12 672	1.096
13	25 344	3.626
14	50 688	12.867
15	101 376	47.673

As to the fourth term of the expression (19), it depends on Nb_MCSs. MCS computing is based on Liffiton and Sakallah (2008) idea as mentioned earlier in the paper. Such problem is known to be exponential since it builds a set of combinations based on a given set of features. As we demonstrated above, it is tightly related to the number of constraints (Nb_Cst) along with the number of features by constraint (NbF_Cst). The former may vary a lot depending on the product line domain and the model itself. For example, Pett et al. (2019) were interested in two industrial product lines. The number of constraints of the first one represents only 7% of the number of features, while it reaches 180% of the number of features for the second one. In the proposed solution, we reduce the number of computed MCSs by eliminating any MCS that violates one of the constraints. Moreover, constraints generally involve only a pair of features, which means that $O(Nb\_MCSs)$ can be written as $O(Nb\_MCSs) = (2^{Nb\_Cst})$. With this practical considerations, we hope that the whole solution scales for real industrial product lines models. As rudimentary experimentation, we took the case of a product line model with 42 features, 5 substitution rules and 4 users. We varied the configurations such that one new conflict is added, i.e. a new constraint is violated, involving at least one new feature. For each execution, we recorded the number of violated constraints, the number of computed MCSs and the execution time needed to compute the MCSs and resolve the conflicts (c.f. Table 9). With the used model, we were able to have till 15 conflicts. Obtained results, although limited, provide an idea about how Colla-Config tool performs on a Core i5 CPU @ 1.60 GHz computer. More experimental evaluations should be carried out to determine the scale of the execution time for real industrial product lines models while varying all system parameters, including the number of constraints. To that end, a dataset of such product lines models along with configuration scenarios should be carefully constructed, and then experimentations should be carried on an appropriate computer (i.e. server) with high processing speed and enough resources.

7. Threats to validity

Various factors, such as plan, design, variables and metrics, may cause threats to the validity of a study. We therefore use the four types of threats to validity proposed by Cook and Campbell (1979) to identify and discuss the relative threats to our study.

- *Conclusion validity.* Conclusion validity refers to the belief in the ability to derive conclusions from the relationships between the experiment, represented by the independent variables, and the outcomes, represented by the dependent variables. Threats to conclusion validity in this work are about:
  - **Statistical validity.** This threat is due to the weakness of the statistical tests. However, given that the main purpose of the conducted experiment is to study the behavior and opinions of tool users, qualitative research methods are well suited. Moreover, the analysis of the collected data still depends on our interpretation. The work was performed by two researchers, and the result was carefully checked by two other researchers as well.
  - **Fishing for the result.** This threat arises from the fact that the researchers, fishing for results that conform to their hypotheses, could unintentionally draw conclusions that are not correct for the study setup and design. We minimized this threat by measuring the three usability attributes effectiveness, efficiency and satisfaction through defined metrics. The main conclusions are drawn based on the interpretation of quantitative results.
- *Internal validity.* Refers to the impact of external experiment variables on the design and results of the experiment. This potential threat was mitigated by defining and validating our experimental protocol and carefully following a well-structured usability testing process using the SO/IEC CIF format for usability tests (ISO/IEC, 2006). The internal validity mainly addresses the following threats:
  - **Participants sample.** The ISO/IEC CIF for usability tests states that "*eight or more subjects are recommended*" (ISO/IEC, 2006). We ensured this guideline by carrying out the experiment with 11 participants.
  - **Participants selection.** This threat is related to the fact that participants selection for a study may affect the results and their interpretation. Even though the group of subjects that participate in a study is always heterogeneous as it is also the case of this work (e.g. male and female, different countries), we tried to limit this threat by considering only Ph.D students, not expert in collaborative configuration of product lines, so that the dominant factor is the experiment itself and not the difference between participants.
  - **Model size.** The configured model is a basic model that was chosen for two reasons: (i) the model was also used in the collaborative configuration approach proposed by Mendonca et al. (2008) to explain the principle of model decomposition and the corresponding configuration plan generation. Since the second part of the experiment is about configuring a product using this approach, it was preferable to adopt the same decomposition/plan to ensure that the approach was correctly applied; (ii) the participants are novice to collaborative configuration; therefore, a simple model is appropriate to carry out the configuration as simply as possible. We are aware that the validity of the evaluation results might be threatened by the simplicity of the chosen feature model, and that considering more complex models from different domains might decrease this threat. However, on the one hand the chosen feature model covers the aspects we seek to evaluate, and on the other hand, the evaluation results of more complex models still depend on the way each participant decides to configure (i.e. chosen features).
  - **Insufficiency of skills to perform the tasks.** This threat was mitigated by the results of the participants pre-questionnaire and the dedicated training.
  - **Participants' expectations.** The expectations of the participants could skew results. This threat was mitigated by the varied post-questionnaire questions that address different dimensions of usability assessment.
- *Construct validity.* Refers to the generalization of experimental results to a concept or theory (Wohlin et al., 2012). In this case, construct validity concerns the following threats:
  - **Insufficiency of pre-operational explanation of concepts.** In order to avoid this threat, all the concepts necessary to understand the different experiment steps were well defined and presented to participants. The participants were also assisted throughout the experiment to answer their questions.
  - **Theory definition.** This threat refers to the fact that the measured variables may not actually measure the conceptual variable. This experiment is about measuring the usability of Colla-Config tool and it is relied on the ISO/IEC 25062:2006 standard that adopts the ISO 9241–11 standard specification of the three metrics effectiveness, efficiency and satisfaction as usability attributes. Moreover, a recent systematic review about usability models and standards revealed that efficiency, satisfaction and effectiveness along with learnability are the commonly addressed usability attributes (Sagar and Saha, 2017); we measured the learnability as one of satisfaction dimensions.
  - **Experimenter expectations.** This type of threats refers to the possibility that the behavior of the experimenter might convey his or her hypothesis or influence participant behavior in other ways. This threat has two sources in the case of this experiment: (1) participants may have prior knowledge or guess that Colla-Config was developed by the team, and this could bias their choice, and (2) the experiment is implicitly driven by the team toward Colla-Config strategy choice. To overcome these potential threats, (i) we carefully followed the well-structured usability testing process using the SO/IEC CIF format for usability tests (ISO/IEC, 2006), (ii) we collected data before and after test session; we used different ways to collect data: i.e. closed-ended and open-ended questions, and (iii) we used in addition to the post-questionnaire, a semi-structured interview which allows flexibility and where confidentiality was emphasized to the respondent in the initial invitation and at the start of the interview.
- *External validity.* Refers to the ability to generalize the results of the experiment to other situations and larger population. ISO/IEC CIF (ISO/IEC, 2006) specifies that 8 or more participants are recommended to conduct the experiment. We carried out the experiment with 11 participants. A threat related to the representation of the population may arise here. Ph.D students from different countries were selected to participate to the experiment. All of them have knowledge about product lines but lack knowledge about conflict

resolution. Other subjects, representing users, who are not familiar with using computers and interested in collaboratively configuring a product line, or those who are not familiar with chosen design interfaces (e.g. tree of features with radio-buttons, etc.) may need help and more time to complete the tasks. Therefore, the results may be different as they depend on participants' opinions about the tool. Another threat is about the representativity of the product line model subject of the experiment. We used a well-known model already used to illustrate previous works in the same context. Furthermore, it is possible to have all types of conflicts, meaning that it is able to cover the whole approach. Other larger models may need more time to be configured and relatively more time to generate the solution. How the model is configured is the choice of the stakeholder, therefore, the results still depend on the participants' experience and their expectations regarding the final product.

8. Related work

Collaborative configuration of product lines has appealed to several researchers that diversely addressed this topic; various studies focusing on different collaborative configuration aspects are proposed. Several collaborative configuration approaches already exist as we showed in Edded et al. (2020). They can be classified as follows:

Workflow-based approaches: this category of approaches mainly relies on predefined process where configuration activities are coordinated and assigned to stakeholders, each of them configures a specific module according to her/his expertise. In this category, conflicts can be systematically resolved by prioritizing configuration choices made at earlier stage. This resolution principle is adopted in some approaches such as Czarnecki et al. (2005) and Mendonca et al. (2008) where stakeholders' configuration decision making governs each other and thus influences which features are still available. These approaches may also make use of negotiation techniques especially when conflicting choices are shared between many stakeholders. Other approaches such as Mendonca et al. (2007) propose a resolution method that relies on conflicts anticipation and offers priority-based conflict resolution: role-based priority and decision set-based priority. In the former, priority is given to the highest ranked decision role, while in the latter, priorities of decision sets are sorted by the importance of the features they contain regarding the configuration process. Moreover, we find other collaborative configuration approaches such as Rabiser et al. (2009) and Xiong et al. (2012) which propose resolving conflicts by suggesting to stakeholders a set of correction values choices.

Free order-based approaches: In this category of approaches, stakeholders freely make their configuration choices without being constrained to a predefined decision-making order. Most of the identified approaches do not provide information about conflict management. Only a few ones explicitly propose a conflict resolution strategy such as the work of Junior et al. (2011) that makes use of software agents to provide personal assistance and propose resolution plans to stakeholders who freely accept anyone of the offered suggestions.

We also find some free order configuration approaches which propose resolving conflicts based on stakeholders' preferences such as Stein et al. (2014) and Ochoa et al. (2015). In Stein et al. (2014), preferences are expressed through hard and soft constraints and in case of conflict, the maintained decision is the one with the highest degree of expressed preference. In Ochoa et al. (2015), language-based criteria is used to find a non-conflicting set of configuration solutions that meets stakeholders preferences expressed in terms of non-functional properties.

Overall, an extensive study has been reported in Edded et al. (2019) that identifies the different existing collaborative configuration approaches, their salient characteristics and the differences between them through a classification framework.

The focus of the studies identified above was generally on analyzing stakeholder' choices to derive valid configuration, or on ensuring coordination during the configuration process. However, most of them either provide little information about conflict resolution process or they do not deal with a full preference-based resolution strategy.

The approach proposed in this paper focuses on both: (i) ensuring coordination within a flexible configuration process where stakeholders freely express their choices, and; (ii) resolving conflicts based on stakeholder's preferences. Moreover, our approach is one of the few that explicitly defines the concept of conflict and provide a full classification of all possible conflict types. We believe that the approach represents a considerable improvement over current conflict resolution policies by proposing a step forward to a full preference-based strategy.

9. Conclusion

This paper presents a new approach called Colla-Config for collaborative configuration within SPLE. This approach relies on a free order process to allow stakeholders freely expressing their choices. Current research conducted in the field showed that the most of existing collaborative configuration approaches adopt resolution strategies that do not consider the preferences of stakeholders. To cope with this issue, we proposed a preference-based conflict resolution strategy where stakeholders preferences are elicited through a set of predefined substitution rules. Based on the expressed preferences, the suitable minimal set of conflicting choices is removed from the configuration. To evaluate the approach, we carried a usability test with eleven Ph.D students. The results provided preliminary evidence that Colla-Config is a usable tool that properly supports the collaborative configuration. At the same time, the analyzed results revealed that the feature model configuration interface needs to be improved by incorporating clear instructions and help to make its use easier. Moreover, it is recommended to focus in a future design of the tool on aspects that improve the ease of learning and ease of remembering aspects.

As a future work, we plan to develop more rigorous experiments: (i) to evaluate the performance of the Colla-Config tool for large sized feature models configured collaboratively by a large number of users and (ii) to develop an industrial case that provides a valuable evidence about the benefits and limitations of Colla-Config.

Regarding the approach itself, we plan to integrate a more advanced profile management that, besides collecting explicit information (e.g. user's role, status, expertise domain, default substitution rule), exploits user's past configurations and interactions to collect implicit information (i.e. preferences regarding the substitution rules, expertise level, expertise domain and interests) and to update his/her profile in the image of Yanes et al. (2017) work. This implicit information may be integrated within the preferences based conflict strategy.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have provided a link to data on GitHub

Appendix A. Pre-questionnaire

What is your level of experience in the following field: Software Product Lines Engineering (SPLE).

☐ Less than one year
☐ Between 1 and 5 years
☐ Between 5 and 10 years

Configuration of SPLE?

☐ It does not ring a bell
☐ I have heard about
☐ Yes, I know what it is about

Collaborative configuration of SPLE is:

☐ No idea
☐ Many people configuring a single product with each other
☐ Many people that each configure a product
☐ One person configuring multiple products
☐ One person configuring a single product

In collaborative configuration of SPLE, a conflict is:

☐ No idea
☐ Two or more choices that cannot be wrong at the same time in a configuration
☐ Two or more choices that cannot be true at the same time in a configuration
☐ Two or more choices that may or not be true at the same time in a configuration

What do you think conflict resolution in the collaborative configuration is all about?

Appendix B. Post-questionnaire

Ease of learning:

The tool is easy to use:

☐ Strongly disagree (strongly complicated)
☐ Disagree (very complicated)
☐ Neither agree nor disagree (complicated)
☐ Agree (easy)
☐ Strongly agree (very easy)

The configuration principle using the tool is clear:

☐ Strongly agree (very clear)
☐ Agree (clear)
☐ Neither agree nor disagree (slightly sophisticated)
☐ Disagree (sophisticated)
☐ Strongly disagree (very Sophisticated)

Ease of use:

On a scale of 1 to 5, how would you rate the speed of execution, the speed of the tool (the performance of the tool)?

☐ 5 (excellent)
☐ 4 (good)
☐ 3 (fair)
☐ 2 (poor)
☐ 1 (very poor)

On a scale of 1 to 5, how would you rate the ergonomics/ease of use of the tool?

☐ 5 (excellent)
☐ 4 (good)
☐ 3 (fair)
☐ 2 (poor)
☐ 1 (very poor)

Ease of remembering:

On a scale of 1 to 5, how strongly do you remember the configuration process principle?

☐ 5 (excellent)
☐ 4 (good)
☐ 3 (fair)
☐ 2 (poor)
☐ 1 (very poor)

On a scale of 1 to 5, how strongly do you remember the conflict resolution process principle?

☐ 5 (excellent)
☐ 4 (good)
☐ 3 (fair)
☐ 2 (poor)
☐ 1 (very poor)

Subjective satisfaction:

How did you find the experimentation process?

☐ Simple and easy to follow (5)
☐ Barely understandable (4)
☐ Complicated (3)
☐ Very complicated (2)
☐ Strongly complicated (1)

On a scale of 1 to 5, how strongly you are satisfied with the final product?

☐ 5 (Completely satisfied)
☐ 4 (Very satisfied)
☐ 3 (Moderately satisfied)
☐ 2 (Slightly satisfied)
☐ 1 (Not at all satisfied)

On a scale of 1 to 5, how strongly you are convinced of the principle of conflict resolution?

☐ 5 (Completely convinced)
☐ 4 (Very convinced)
☐ 3 (Moderately convinced)
☐ 2 (Slightly convinced)
☐ 1 (Not at all convinced)

Other questions:

Do you have specific examples of problems with response times and access to the tool?
Do you have specific examples of ergonomics/usability issues?
On a scale of 1 to 5, how strongly would you recommend using the tool?
Which configuration process did you like the most?
Which method of conflict resolution did you prefer?
Could you briefly explain what did you retain from the CollaConfig configuration principle and conflict resolution process in the experiment part 1?

References

Bagheri, E., Di Noia, T., Ragone, A., Gasevic, D., 2010. Configuring software product line feature models based on stakeholders' soft and hard requirements. In: Software Product Lines: Going beyond, the 14th International Conference. Springer-Verlag, Berlin, Heidelberg, pp. 16–31. http://dx.doi.org/10.1007/978-3-642-15579-6_2.

Benavides, D., Segura, S., Ruiz-Cortés, A., 2010. Automated analysis of feature models 20 years later: A literature review. Inf. Syst. 35 (6), 615–636. http://dx.doi.org/10.1016/j.is.2010.01.001, URL: https://www.sciencedirect.com/science/article/pii/S0306437910000025.

Bevan, N., 2006. Practical issues in usability measurement. Interactions 13 (6), 42–43. http://dx.doi.org/10.1145/1167948.1167976.

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

Cook, T., Campbell, D., 1979. Quasi-Experimentation: Design and Analysis Issues for Field Settings. Houghton Mifflin.

Czarnecki, K., Helsen, S., Eisenecker, U., 2005. Staged configuration through specialization and multilevel configuration of feature models. In: Software Process: Improvement and Practice. http://dx.doi.org/10.1002/spip.225.

Czarnecki, K., Kim, C.H.P., 2005. Cardinality-based feature modeling and constraints: A progress report. In: International Workshop on Software Factories. ACM, San Diego, California, USA, pp. 16–20.

Deelstra, S., Sinnema, M., Bosch, J., 2005. Product derivation in software product families: a case study. J. Syst. Softw. 74 (2), 173–194. http://dx.doi.org/10.1016/j.jss.2003.11.012, The new context for software engineering education and training.

Donkers, B., Dellaert, B.G., Waisman, R.M., Häubl, G., 2020. Preference dynamics in sequential consumer choice with defaults. J. Mar. Res. 57 (6), 1096–1112. http://dx.doi.org/10.1177/0022243720956642.

Edded, S., BenSassi, S., Mazo, R., Salinesi, C., BenGhezala, H., 2019. Collaborative configuration approaches in software product lines engineering: A systematic mapping study. J. Syst. Softw. 158, 110422. http://dx.doi.org/10.1016/j.jss.2019.110422.

Edded, S., BenSassi, S., Mazo, R., Salinesi, C., BenGhezala, H., 2020. Preference-based conflict resolution for collaborative configuration of product lines. In: Proceedings of the 15th International Conference on Evaluation of Novel Approaches to Software Engineering, ENASE 2020, Prague, Czech Republic, May 5-6, 2020. SCITEPRESS, pp. 297–304. http://dx.doi.org/10.5220/0009346102970304.

Feichtinger, K., Meixner, K., Rabiser, R., Biffl, S., 2021. A systematic study as foundation for a variability modeling body of knowledge. In: 2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). pp. 25–28. http://dx.doi.org/10.1109/SEAA53835.2021.00012.

Holl, G., Grunbacher, P., Elsner, C., Klambauer, T., 2012. Supporting awareness during collaborative and distributed configuration of multi product lines. In: Proceedings of the 19th Asia-Pacific Software Engineering Conference - Volume 01. APSEC '12, IEEE Computer Society, Washington, DC, USA, pp. 137–147. http://dx.doi.org/10.1109/APSEC.2012.41.

Hubaux., A., Heymans, P., Schobbens, P., 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). Springer-Verlag, Essen, Germany, pp. 106–112. http://dx.doi.org/10.1007/978-3-642-14192-8_12.

ISO/IEC, 2006. Software Engineering—Software Product Quality Requirements and Evaluation (SQuaRE). Technical Report, Common Industry Format (CIF) for usability test reports.

Junior, C.M., Cirilo, E., Lucena, C., 2011. Assisted user-guidance in collaborative and dynamic software product line configuration. In: 14th Ibero-American Conference on Software Engineering and 14th Workshop on Requirements Engineering, CIbSE 2011. pp. 143–156.

Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S., 1990. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report, Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, http://dx.doi.org/10.21236/ada235785.

Le, V.-M., 2021. Group recommendation techniques for feature modeling and configuration. pp. 266–268. http://dx.doi.org/10.1109/ICSE-Companion52605.2021.00123.

Liffiton, M.H., Sakallah, K.A., 2008. Algorithms for computing minimal unsatisfiable subsets of constraints. J. Autom. Reason. 40 (1), 1–33. http://dx.doi.org/10.1007/s10817-007-9084-z.

Mendonca, M., Bartolomei, T., Cowan, D., 2008. Decision-making coordination in collaborative product configuration. In: ACM Symposium on Applied Computing. pp. 108–113. http://dx.doi.org/10.1145/1363686.1363715.

Mendonca, M., Cowan, D., Oliveira, T., 2007. Process-centric approach for coordinating product configuration decisions. In: Proceedings of the 40th Hawaii International Conference on System Sciences (HICSS'07). pp. 1–10. http://dx.doi.org/10.1109/hicss.2007.27.

Nieke, M., Sampaio, G., Thüm, T., Seidl, C., Teixeira, L., Schaefer, I., 2022. Guiding the evolution of product-line configurations. Softw. Syst. Model. 21 (1), 225–247. http://dx.doi.org/10.1007/s10270-021-00906-w.

Nielsen, J., 2010. Chapter 1 - what is usability? In: User Experience Re-Mastered. Morgan Kaufmann, Boston, pp. 3–22. http://dx.doi.org/10.1016/B978-0-12-375114-0.00004-9.

Ochoa, L., González-Rojas, O., 2016. Program synthesis for configuring collaborative solutions in feature models. In: On the Move to Meaningful Internet Systems: OTM 2016 Workshops. pp. 98–108. http://dx.doi.org/10.1007/978-3-319-55961-2_10.

Ochoa, L., González-Rojas, O., Thüm, T., 2015. Using decision rules for solving conflicts in extended feature models. In: Proceedings of the 2015 ACM SIGPLAN International Conference on Software Language Engineering. In: SLE 2015, ACM, New York, NY, USA, pp. 149–160. http://dx.doi.org/10.1145/2814251.2814263.

Osman, A., Phon-Amnuaisuk, S., Ho, C.K., 2009. Investigating inconsistency detection as a validation operation in software product line. In: Lee, R., Ishii, N. (Eds.), Software Engineering Research, Management and Applications 2009. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 159–168. http://dx.doi.org/10.1007/978-3-642-05441-9_14.

Pereira, J.A., 2017. Runtime collaborative-based configuration of software product lines. In: Proceedings of the 39th International Conference on Software Engineering Companion. In: ICSE-C '17, pp. 94–96. http://dx.doi.org/10.1109/icse-c.2017.154.

Pett, T., Thüm, T., Runge, T., Krieter, S., Lochau, M., Schaefer, I., 2019. Product sampling for product lines: The scalability challenge. In: Proceedings of the 23rd International Systems and Software Product Line Conference - Volume a. SPLC '19, Association for Computing Machinery, New York, NY, USA, pp. 78–83. http://dx.doi.org/10.1145/3336294.3336322.

Raatikainen, M., Tiihonen, J., Männistö, T., 2019. Software product lines and variability modeling: A tertiary study. J. Syst. Softw. 149, 485–510. http://dx.doi.org/10.1016/j.jss.2018.12.027.

Rabiser, R., Wolfinger, R., Grünbacher, 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. pp. 1–10.

Roschelle, J., Teasley, S.D., 1995. The construction of shared knowledge in collaborative problem solving. pp. 69–197. http://dx.doi.org/10.1007/978-3-642-85098-1_5.

Sagar, K., Saha, A., 2017. A systematic review of software usability studies. Int. J. Inf. Technol. 1–24. http://dx.doi.org/10.1007/s41870-017-0048-1.

Salinesi, C., Mazo, R., Diaz, D., Djebbi, O., 2010. Using integer constraint solving in reuse based requirements engineering. In: Proceedings of the 18th IEEE International Requirements Engineering Conference (RE). pp. 243–251. http://dx.doi.org/10.1109/re.2010.36.

Sauro, J., 2010. A Practical Guide to Measuring Usability: 72 Answers to the Most Common Questions About Quantifying the Usability of Websites and Software. Measuring Usability LLC.

Sauro, J., Lewis, J.R., 2010. Average task times in usability tests: What to report? In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI '10, Association for Computing Machinery, New York, NY, USA, pp. 2347–2350. http://dx.doi.org/10.1145/1753326.1753679.

Sauro, J., Lewis, J.R., 2012. Quantifying the User Experience, Practical Statistics for User Research. Morgan Kaufmann, http://dx.doi.org/10.1016/C2010-0-65192-3.

Shahin, R., Hackman, R., Toledo, R., Ramesh, S., Atlee, J.M., Chechik, M., 2021. Applying declarative analysis to software product line models: An industrial study. In: 2021 ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems (MODELS). pp. 145–155. http://dx.doi.org/10.1109/MODELS50736.2021.00023.

Soltani, S., Asadi, M., Gašević, D., Hatala, M., Bagheri, E., 2012. Automated planning for feature model configuration based on functional and non-functional requirements. In: Software Product Line, the 16th International Conference. SPLC '12, ACM, New York, NY, USA, pp. 56–65. http://dx.doi.org/10.1145/2362536.2362548.

Stein, J., Nunes, I., Cirilo, E., 2014. Preference-based feature model configuration with multiple stakeholders. In: Proceedings of the 18th International Software Product Line Conference. pp. 132–141. http://dx.doi.org/10.1145/2648511.2648525.

White, J., Dougherty, B., Schmidt, D.C., Benavides, D., 2009. Automated Reasoning for Multi-Step Feature Model Configuration Problems. Carnegie Mellon University, USA, pp. 11–20.

Wohlin, C., Runeson, P., Hst, M., Ohlsson, M.C., Regnell, B., Wessln, A., 2012. Experimentation in Software Engineering. Springer Publishing Company, http://dx.doi.org/10.1007/978-3-642-29044-2, Incorporated.

Xiong, Y., Hubaux, A., She, S., Czarnecki, K., 2012. Generating range fixes for software configuration. In: Proceedings of the 34th International Conference on Software Engineering (ICSE'12). IEEE Computer Society, Zurich, Switzerland, pp. 58–68. http://dx.doi.org/10.1109/icse.2012.6227206.

Yanes, N., Ben Sassi, S., Hajjami Ben Ghezala, H., 2017. Ontology-based recommender system for COTS components. J. Syst. Softw. 132, 283–297. http://dx.doi.org/10.1016/j.jss.2017.07.031.

Ziadi, T., Jezequel, J.-M., 2006. Software product line engineering with the UML: Deriving products. In: Software Product Lines. Springer Berlin Heidelberg, pp. 557–588. http://dx.doi.org/10.1007/978-3-540-33253-4_15.