Reviewer 2
Next, we provide answers (in blue text) to the comments/remarks pointed out by the reviewers (in black text).

Thanks for the careful reading and for the comments and suggestions. Below please find how we deal with them.

Summary
The paper under review introduces a probabilistic process algebra SPLA^P that extends SPLA, a process algebra for software product line (SPL) models that follows the feature-oriented domain analysis approach (FODA) [28]. SPLA has been specified by the authors in a preceding paper [1]. The authors claim that the use of their probabilistic extension of eases testing, analysis, and tailoring software products, e.g., by prioritizing feature combinations that are most likely to be included in the SPL. The operational semantics of process algebra terms is provided by a kind of probabilistic transition system. Compatibility to the semantics of [1] is shown when abstracting away probabilistic annotations. A case study is carried out using an own implementation of the formal framework.

Evaluation
The paper addresses an important field of SPL engineering, namely including stochastic information about features, their composition, and testing into the development process of SPLs. Unfortunately, the paper lacks of a justification of the probabilistic semantics, misses comparison to related work, and does not show feasibility of the approach. Hence I recommend to reject the paper.

Comments
1) There is no explanation or justification why the chosen operational semantics is suitable for its purpose. Intuitively I would expect that the probability attached to some A-transition corresponds to the probability of the event of composing feature A to the current feature combination. However, when, e.g., considering the term t1=A;tick /\ (B;tick /\ C;tick), after composing A, there is only a probability of 1/4 each for composing B and C. Hence, the probability of having the only possible feature combination ABC is less than 1, which let me conclude that the semantics is not appropriate. Furthermore, equality of process algebra terms is only considered with respect to their products. I argue that, as the semantics is an operational one, it would be more appropriate to take (probabilistic) bisimulation into account. The example term t2=(A;tick /\ B;tick) /\ C;tick then shows that the operational semantics is not associative (opposed to the claim that the proof of [1] takes over): choosing A in t1 has probability of 1/2, whereas it has probability of 1/4 in t2. The reason for these pathologies is in my opinion the somehow arbitrary choice of dividing probabilities in conjunctions by 2 (rules [con1], [con4] etc.). Please correct me if I am wrong, but in either case, there should be more explanations including examples where probabilities are involved (there is no example at all).

The transition system obtained with term operational semantics is “observed”, similarly as for classical trace (and not bisimulation) equivalence, only for the sets of features. The proof of the following result is an immediate consequence of Lemmas 3-12 (see Appendix A of the paper) select (each set being the labels of the transitions of a trace, thus abstracting also from trace label ordering) and the probabilities associated to them. For this reason, the conjunction operator is indeed associative: technically associativity is an immediate consequence of Proposition 3 and Definition 6 (an item about conjunction in both of them). We have rewritten the proof of Proposition 3 (including the case of conjunction, Lemma 8) more in detail trying to make it clearer. We also have included the example indicated by the reviewer in the paper. The 1/2 that is used in the operational semantics of /\ to assign probabilities to left-hand and right-hand branches is not critical: as a matter of fact we

could have chosen any other distribution factor p in one branch and 1-p in the other branch (anyway both the features on the left and on the right of /\ will have to be selected in the end). The choice of 1/2 w.r.t. any other p (with 0<p<1) just has the advantage that it makes, at least, commutativity of /\ obvious. In the example that the reviewer comments, the only product (set of features) of t2=(A;tick /\ B;tick) /\ C;tick is [A, B, C] with probability 1, that coincides with the set of probabilistic products obtained for t1=A;tick /\ (B;tick /\ C;tick). Additionally, we have put some examples of the operational semantics of terms in the paper.

2) Furthermore, for me it is not clear what the probabilities attached to transitions actually mean. The only modeling formalism for probabilistic SPLs is provided by SPLA^P terms. In [1] however, a translation of FODA diagrams to SPLA terms has been given. For me it is not clear how a corresponding translation would provide an SPLA^P term as the authors do not deal with probabilistic variants of feature models at all.

FODA does not have probabilities so their models are translated into SPLA [1]. In order to be translated to SPLA^P, it should be necessary to introduce the probabilities in FODA designs: in the optional feature (probability here determines the presence or not of such a feature, and corresponds to the frequency with which such a feature is selected) and in the choice operator (probability here determines whether the left-hand or right-hand features are selected, and corresponds to the frequency with which the corresponding selection is made).

3) There has been plenty of work done in the area of probabilistic SPLs and I miss a comparison with the existing approaches to estimate the contribution of the paper. For instance, the aim of [19] and the paper under review is highly related - what is the benefit of using the new approach presented in the paper instead of [19]? Furthermore, a reference to [a] is missing. There, a probabilistic feature model has been presented that supports dynamic (probabilistic) SPLs. I think that the intention of this paper is somehow covered by their approach as the stepwise feature compositions could be encoded into the (operational/automata-based) “feature controller” formalism in [a]. There is also an implementation of the theoretical framework of [a] published as [b] that could be mentioned.

In the new version of the paper, we have added a new section (Section 2) with an extended the related work as requested. Specifically related to the reference [29] one of the interesting aspects presented in our probabilistic extension is that any of the referenced research articles manage to describe in their work the use of multisets. Also, they do not explicitly work on the translation of FODA to represent probabilities and they do not introduce the notion of hiding those not needed features to calculate the probability of a specific feature.

4) The case study is not convincingly showing applicability of the approach. First, it is not clear how the randomly generated feature models include stochastic information. Correct me if I am wrong, but I understood that “the probability of having a mandatory feature is 20%” means that during the generation process of BeTTy the probability of generating a mandatory feature is 1/5. I could not find any hint where the annotated probabilities in the process terms come from and hence, I cannot evaluate the results. Furthermore, it is questionable to argue about runtimes of around 20ms - to make more reliable statements, the number of features should be increased to lower potential impacts of other processes or side effects.

This is a good point. Since this concept was not clearly described in the first version of the paper, we have re-written this part to clarify the values for configuring BeTTy and the obtained results.

In this case, it is important to differentiate the probabilities defined in BeTTy, which are used to generate a feature model, and the probability - calculated from the model - to have a feature in a final product. Let suppose that we configure BeTTy to generate a feature model using a probability of 0.2 for having a mandatory feature, which means that 20% of the generated features are mandatory. However, this does not imply that these features are part of the 20% of the generated products, because the probability of having a feature in a final product depends of the position of this feature in the model. For example, if a given mandatory feature is placed in a choose-one relationship, it is possible that the other branch (the branch that does not contain the mandatory feature) is used to generate the final product, discarding the branch containing the mandatory feature.
Additionally, we have repeated the experiment 10 times, where a new model is randomly generated and processed. Table 1 shows the obtained results from each execution.

Minor comments
- Definition 2.1 is missing (cf. Definition 5).

Fixed

- Reference [8] and [9] are the same.

Fixed

- Page 3 presents a long chain of references concerning probabilistic systems without any comment or explanation. It would be helpful to describe the relation of these publications to the presented paper.

Fixed

- Please explain in detail why the choose-one operator \/_p is n-ary. Furthermore, include the n-ary interpretation into your semantics (currently only binary).

We have removed the comments about the choice being n-ary. We could have defined n-ary (by assigning a probability distribution to the n options), but we think the notation would have been more complex to follow.

- The semantics is not uniquely defined. I assume that you look for the smallest probabilistic transition relation fulfilling the rules of Figure 2.

Fixed. We have introduced them properly in a definition.

- The statement that the rules of Figure 2 are in essential the same as in [1] is not correct as it is: the rule [req2] has been changed in [8] and taken from there.

The change in [8] was a minor change and we proved that both rules are equivalent: the change does not affect the semantics. We have preferred to keep such a change in this version. We have added a comment in the paper.

- Lemma 2: Q already fixed.

Fixed.

- Check typos in Proposition 5

Fixed.

References

[a] Clemens Dubslaff, Christel Baier, Sascha Klüppelholz: Probabilistic Model Checking for Feature-Oriented Systems. Trans. Aspect-Oriented Software Development 12: 180-220 (2015)

[b] Philipp Chrszon, Clemens Dubslaff, Sascha Klüppelholz, Christel Baier: ProFeat: feature-oriented engineering for family-based probabilistic model checking. Formal Asp. Comput. 30(1): 45-75 (2018).

These works have been cited in the new version of the paper.