Belief-Driven Software Product Line Development and Evolution

Anonymous Author(s)

ABSTRACT

The planning, realization, and release of a Software Product Line (SPL) are driven by features. Therefore, many high-level decisions about the evolution of an SPL are made at the feature level. However, a feature can involve many stakeholders with different expertise, and taking their opinions into account to make the right decisions is not trivial. In this paper, we propose using belief uncertainty in conjunction with feature models to assist in the evolution of SPLs by explicitly quantifying opinions. We outline three evolution scenarios in which subjective logic can be used to represent the opinions of stakeholders and explain in detail how to use subjective logic to make decisions in the context of the next release problem. We illustrate our ideas with a Smartwatch SPL. Finally, we discuss different ways of combining the opinions of stakeholders depending on the situation, the goals and the risks that can be assumed.

CCS CONCEPTS

• **Software and its engineering** → **Software product lines**; **Software evolution**; *Collaboration in software development*.

KEYWORDS

Decision making support, feature model, software product line, subjective logic, uncertainty

ACM Reference Format:

Anonymous Author(s). 2023. Belief-Driven Software Product Line Development and Evolution. In *Proceedings of 27th ACM International Systems and Software Product Lines Conference (SPLC'23)*. ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The planning, realization and release of a *software product line* (SPL) centers around features [2]. In fact, *feature models* have become the main artifact that guides the whole SPL process [28], specifying the variability of the system as a collection of common and variable features. As a result, it is therefore not surprising that many high-level decisions about the evolution of an SPL are made at the granularity of features [22]. For example, at some point, a decision might be needed on whether or not to add a certain feature to the set of planned features of the SPL (aka *evolution of the variability* [1, 22]). Another question might be to identify which features should be realized for the coming release (aka *the next release problem* [4, 21, 37]). Or, finally, one might want to decide on how many products to sell to end users, that is, whether to allow end users to customize their products as much as the realization allows, or whether to reduce the number of possible products artificially (aka *variability reduction* [12, 34]).

To make the right decisions, many SPL stakeholders with different expertise might be involved. This includes obviously software developers that are experts in different domains, but also hardware experts, financial planners, human resources, marketers, etc. Taking the opinions of that many people into account to make the right decisions is not trivial. Conflicting opinions often arise among stakeholders due to their differing opinions (or beliefs) about a specific statement, making it difficult to reach a consensus. Moreover, in an SPL, features often have dependencies between them, and decisions about specific features must consider those dependencies, increasing the uncertainty of appropriate decisions.

In this paper, we propose a novel approach that uses *belief uncertainty* [36] in conjunction with feature models to explicitly quantify the opinions of stakeholders regarding the evolution of an SPL. We outline three scenarios in which *subjective logic* [17], a formalism for reasoning under belief uncertainty, is used to represent and combine the opinions of stakeholders. We explain how to use subjective logic to make decisions in the context of the next release problem, and how to use the appropriate fusion operators to reach a consensus depending on the desired decision strategy.

The remainder of the paper is structured as follows. Section 2 briefly presents the necessary background on belief uncertainty and subjective logic. Section 3 explains the three feature model evolution scenarios that we cover and illustrates them by means of a *Smartwatch* SPL. Section 4 applies our approach to the next release problem, again in the context of the *Smartwatch* SPL. Section 5 presents related work, and the last section draws some conclusions.

2 UNCERTAINTY AND SUBJECTIVE LOGIC

Uncertainty [20] is defined as the state that involves imperfect and/or unknown information. It applies to predictions of future events, estimations, physical measurements, or unknown properties of a system. Uncertainty can take different forms. For example, *measurement uncertainty* [36] refers to the inability to know with complete precision the value of a quantity. It can be due to different causes, such as unreliable data sources and communication networks; tolerance in the measurement of the values of the physical elements; estimates due to the lack of accurate knowledge about certain parameters, or the inability to determine whether a particular event has actually happened or not. *Belief uncertainty* [36] is a particular type of uncertainty where an agent is uncertain about an statement. For instance, it can capture the inability to decide whether something is true or not (i.e., a Boolean predicate), and depends on the individual stating such uncertainty.

In [36] different kinds of uncertainty were identified and classified. The authors also analyzed how uncertainty is represented in software models and used in the context of model-based software engineering (MBSE). In many occasions, belief uncertainty is expressed by *probabilities* (interpreted in Probability theory [8] or in Uncertainty theory [20]), *possibilities* (in Fuzzy set theory [39]), *plausibilities* (in the Dempster–Shafer theory of evidence [31]) or *opinions* (in subjective logic [17]). In this paper, we focus on subjective logic [17] which is a type of probabilistic logic that explicitly takes uncertainty into account.

In subjective logic, each opinion about a statement is composed of four values: (1) the degree of *belief* b that the statement is true; (2)

SPLC'23, August 28 – September 1, 2023, Tokyo, Japan
2023. ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

the degree of *disbelief* d that the statement is false; (3) the degree of *uncertainty* u about the statement (i.e., the amount of uncommitted belief); and (4) the *base rate* a or *prior probability* of the statement (i.e., the objective probability). The four values are represented in an `SBoolean` vector [6] defined as a 4-tuple (b, d, u, a), and satisfying $b + d + u = 1$, and $b, d, u, a \in [0, 1]$.

Let us illustrate the use of `SBoolean` with an example. Hana checks the weather forecast for Tokyo and it says that the probability of rain is 45%, but she knows that the website she is checking is not always very accurate. Therefore, she expresses her opinion on the fact that the following day will rain as `SBoolean(0.6, 0.1, 0.3, 0.45)`, which means that she trusts the original prediction (0.45) with a degree of belief of 60%, she thinks that it might be wrong with a degree of disbelief of 10%, and she expresses her uncertainty with a 30%.

The *projected probability* [6] (or *projection*) of an `SBoolean` opinion is defined as $P = b + a \times u$, and allows us to move from opinions expressed in subjective logic to a single value opinion, (i.e., a probability). For example, the projection of Hana's opinion on the weather forecast is 0.735 (73.5%).

Subjective logic comes with a set of *fusion operators* that can be used to *combine opinions* of different agents about the same statement. Which fusion operator to apply depends on several factors, such as, whether one wants to minimize risk when dealing with contradicting opinions. We will introduce some operators later in Section 4. The interested reader can consult the formal definitions of fusion operators in [17, 18, 38].

3 DECISION MAKING IN SPL DEVELOPMENT

During the development of an SPL, there will be a moment in time where there is a variability model of the current state of the SPL, that is, the features that have already been realized: $FM_{realized}$. In the case where marketing or other reasons warrant that not all possible products of the SPL are made available to end users in a release, there might also be a variability model $FM_{reduced}$ that encodes the (artificially) reduced configuration variability for end users. Most likely there is also a variability model of the planned features of the SPL for future releases: $FM_{planned}$. It holds that $FM_{reduced} \subseteq FM_{realized} \subseteq FM_{planned}$.

Figure 1 depicts the evolution of those three variability models over time. The feature planning ($FM_{planned}$), the feature realization ($FM_{realized}$), and the feature release ($FM_{reduced}$) of the SPL can evolve at different speeds, and moving from one variability model to the next requires decisions to be made.

3.1 Evolution Scenarios for Feature Models

We identified three different kinds of evolution scenarios for the variability models that require collaborative decision-making among stakeholders (highlighted with blue arrows in Figure 1):

(1) **Feature Model Evolution [1, 22]**: How should the plan of the variation of the SPL evolve?

$$FM_{planned,current} \rightarrow FM_{planned,next}$$

Scenario 1 consists of using the current plan $FM_{planned}$ as a basis, and suggesting *edits* [35] such as adding new features, including new relationships or updating existing ones, adding cross-tree constraints if necessary, and potentially even removing planned features that have not been realized yet.

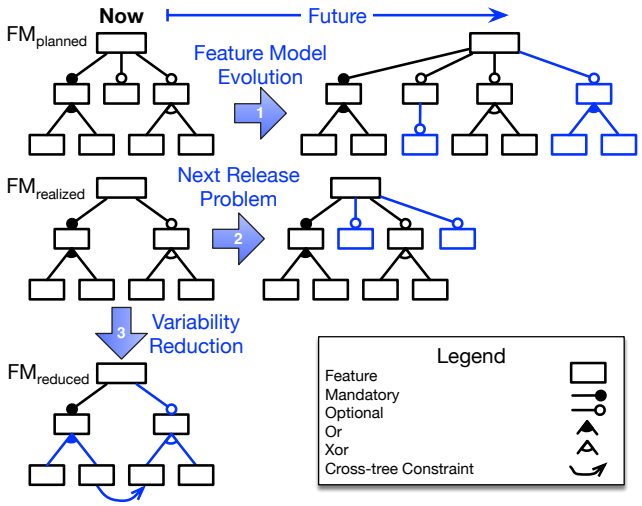

Figure 1: SPL evolution scenarios for variability models.

(2) **Next Release Problem [4, 21, 37]**: Which features should be implemented next?

$$FM_{realized,current} \rightarrow FM_{realized,next}$$

Scenario 2 consists in choosing which planned features that have not been realized yet (i.e., features that are part of $FM_{planned}$ and are not part of $FM_{realized}$) should the development team work on next.

(3) **Variability Reduction [12, 34]**: How can the configuration space of the current realized SPL be reduced to a reasonable number of possible configurations for the current release?

$$FM_{realized,current} \rightarrow FM_{reduced,current}$$

Scenario 3 consists in choosing which edits to apply to $FM_{realized}$ to reduce the variability (e.g., making an optional feature mandatory or deciding which additional cross-tree constraints to add). Reducing the variability of a SPL simplifies maintainability and quality assurance [3], as fewer products need to be considered.

The decision making for these three scenarios is non-trivial, as different stakeholders have different views on the SPL features depending on the scenario. For example, when deciding on what features to realize next (Scenario 2), opinions of different stakeholders could range from "Marketing: in high demand from end users", to "Distribution expert: high maintenance cost", to "Developer: unclear how to implement efficiently".

3.2 Smartwatch SPL Example

Figure 2 presents a *Smartwatch SPL*. A smartwatch is a wearable computer in the form of a watch. It offers multiple characteristics to users such as `ContactlessPayment` or an `ActivityTracker`, which can provide `ContinousMonitoring` and/or `OnDemand` monitoring. All these features are optional, but in order to support them, the SPL provides several `Sensors` and `Connectivity`. For example, to include the `ActivityTracker` feature, an `Accelerometer` or an `HeartRateSensor` are required, as specified in the cross-tree constraint `ActivityTracker` ⇒ `Accelerometer` ∨ `HeartRateSensor`. The

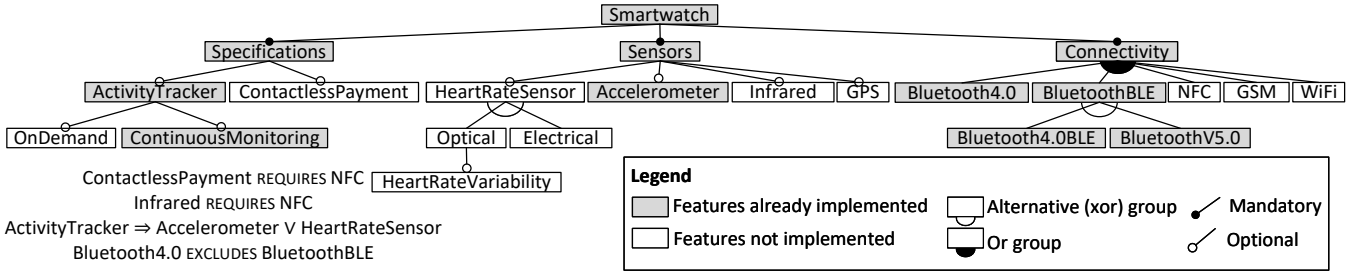

Figure 2: A feature model for a Smartwatch SPL.

HeartRateSensor can be either Electrical or Optical, the latter can also include the HeartRateVariability feature to compute fluctuations on the user's heartbeats. Other available sensors in the SPL are the Infrared and the GPS, while for connectivity the SPL provides NFC, GSM, WiFi, and different versions of Bluetooth (Bluetooth4.0 and the low energy versions Bluetooth4.0BLE or BluettothV5.0). Other cross-tree constraints in the SPL specify that in order to have the ContactlessPayment or the Infrared sensor, the NFC feature must be included as well. Furthermore, only one version of Bluetooth can be present in a smartwatch.

The complete feature model of Figure 2 represents $FM_{planned}$ for the Smartwatch SPL. In the current state of the SPL, only those features highlighted in gray are already implemented, representing $FM_{realized}$. Different versions of the smartwatch have been released in the market including only those implemented features, thus $FM_{reduced} = FM_{realized}$. According to our three evolution scenarios, the Smartwatch SPL can first evolve by adding new features to $FM_{planned}$ for the future releases of the smartwatch or by removing features that stakeholders think should never be realized. For instance, the marketing team proposes to incorporate a new SpO2 sensor into the SPL in order to track the blood oxygen saturation level of the user, while the distribution experts propose to remove the WiFi feature from the SPL because WiFi consumes a lot of energy and does not make sense for a smartwatch with other types of connectivity. Hence, that feature will most likely never be implemented in any release. In the second scenario, the stakeholders need to make decisions about what features to implement and include in the next release (i.e., features to be included in $FM_{realized}$). Here decisions may involve deciding the incorporation of an independent feature (e.g., GPS), deciding on a feature with dependencies on other features (e.g., to incorporate the ContactlessPayment feature, the NFC feature must be implemented too), or deciding on a group of related features, such as, a variation point and its variants (e.g., the development of the HeartRateSensor feature and its variants). Finally, the third scenario is illustrated in the Smartwatch SPL when the number of possible products of $FM_{realized}$ is too high and it does not make sense to have that many versions of the smartwatch on the market. In this case, it needs to be decided which specific products can be omitted from being released by reducing the variability. For example, the distribution experts propose the exclusion of the Bluetooth4.0 feature already implemented in the SPL from any smartwatch version of the current release because there is no technical support for such an old version of Bluetooth in the new version of the operating system. This decision reduces the variability exposed by the SPL in $FM_{reduced}$ and the number of products to be maintained is more manageable.

4 BELIEF UNCERTAINTY APPLIED TO SPL

We illustrate our decision-making approach based on subjective logic using the second evolution scenario (the next release problem), while the decisions for the other scenarios can be modeled similarly.

4.1 Decision Making about Features

4.1.1 Deciding on an independent feature. It needs to be decided whether the GPS feature should be realized or not. To make this decision, representatives from the marketing, hardware, and software departments sit together and provide their opinions on the feature.

The marketing stakeholder Bob thinks that GPS is an interesting feature for physically active customers that will increase the number of units sold according to a market study performed. This study was a poll sent to customers who have bought previous versions of the watch, so the results are not representative of the general population. Therefore, his opinion carries uncertainty and can be represented by the SBoolean(0.95, 0.0, 0.05, 0.5). That is, Bob has a belief of 0.95 on the fact that the GPS should be included, a disbelief of 0, and an uncertainty of 0.05. Note that the base rate is 0.5 because the a priori probability that a feature is included is 50% as there are only two options: it is included, or it is not included.

On the other hand, Alice, the representative of the hardware department, does not recommend the realization of the GPS feature because she knows that it will have a considerable impact on power consumption. She knows that there is new GPS hardware in development that would affect the battery less, but whether or not that technology will be available in time is uncertain. She gives her opinion as SBoolean(0.1, 0.7, 0.2, 0.5).

Finally, the software department representative Taylor thinks that from their department's point of view, it does not make a difference whether the watch has a GPS integrated or not. But they think that since it does not add any additional cost, it would be good to have it. Their opinion is SBoolean(0.3, 0.05, 0.65, 0.5).

To be able to use our approach to help make a decision, we need to fuse these three opinions, hence we need to choose the appropriate fusion operator. We are facing a case of epistemic uncertainty (vs. aleatory uncertainty). Since we have one person from each department giving their opinions based on their independent experiences and backgrounds, we can assume that there are no dependencies among these opinions. Therefore, the fusion operator that applies in this situation is *Epistemic Cumulative Belief Fusion* (ECBF). When fusing the three opinions using the ECBF operator, the result is: SBoolean(0.668, 0.0, 0.332, 0.5), whose projection (i.e., the probability) is 0.83. Therefore the three stakeholders decide that the GPS feature should be realized.

4.1.2 Deciding on a feature with cross-tree constraints. The smartwatch company has suggested to the hardware department to study the viability of realizing contactless payment. As Figure 2 shows, the realization of the feature `ContactlessPayment` (CP) requires the realization of the feature `NFC`. Instead of making an authoritarian decision, the manager of the department asks a group of three engineers for their opinions. The SBooleans that represent the engineers' opinions on these features are presented in Table 1.

Table 1: Opinions of the three engineers on each feature.

	Engineer 1	Engineer 2	Engineer 3
CP	(0.5, 0.1, 0.4, 0.5)	(0.9, 0.03, 0.07, 0.5)	(0.8, 0.15, 0.05, 0.5)
NFC	(0.5, 0.25, 0.25, 0.5)	(0.75, 0.2, 0.05, 0.5)	(0.94, 0.03, 0.03, 0.5)

To take into account the feature dependency, we need to aggregate the individual feature opinions using the logical and operator (∧). Only then can we combine the resulting opinions with the appropriate fusion operator. Therefore, the fused opinion is calculated using the following formula:

```
FUSION(OpEng1CP ∧ OpEng1NFC, OpEng2CP ∧ OpEng1NFC,
       OpEng3CP ∧ OpEng3NFC)
```

We are again facing a case of epistemic uncertainty. However, since the manager is asking for the opinion of three engineers that work in the same department, they all have more or less the same background and knowledge about the company and the team (i.e., in terms of subjective logic, they are observing the same fact), hence their opinions are considered *dependent*. There are three fusion operators that apply to this case: Average Belief Fusion (ABF), Weighed Belief Fusion (WBF) and Consensus and Compromise Fusion (CCF). The first one should be applied when every opinion carries the same weight (i.e., when one wants to give the same credibility to each person), the second should be applied when one wants to give more credibility to those people with stronger opinions (i.e., with less uncertainty), and the third one should be used when the desired behavior is that in the presence of divergent opinions the level of uncertainty in the resulting opinion is increased. While the operator CCF provides a more conservative opinion, the operators ABF and WBF are more "risky". Table 2 presents the fused opinions for these three operators.

Table 2: Fused opinions for `CP` ∧ `NFC`.

Operator	Fused Opinion	Projection
ABF	(0.715, 0.206, 0.078, 0.25)	0.734787
WBF	(0.724, 0.203, 0.073, 0.25)	0.742006
CCF	(0.524, 0.190, 0.286, 0.25)	0.595724

After checking the results, the manager observes that, even in the more conservative case, the engineers support the inclusion of contactless payment with almost 60% of confidence, hence he decides to suggest the implementation of both features.

4.1.3 Deciding on a group of related features. For the next meeting with its main shareholders, the smartwatch company needs to determine whether the `HeartRateSensor` feature and its variants are likely to be implemented or not. We can observe in Figure 2 that there are different options when it comes to integrating a heart rate sensor (`HRS`): it could be an electrical sensor (`E`), which is more accurate, or an optical sensor (`O`), which is cheaper. If an optical sensor is included, the software team could potentially implement a feature that measures the heart rate variability (`HRV`), which can detect heart problems.

Once someone gives their opinion on each feature of the tree, the individual opinion of that person on the fact that one way or another a heart rate sensor is included in the next realization is calculated with the following formula:

```
(HRS ∧ E) ∨ (HRS ∧ O) ∨ (HRS ∧ O ∧ HRV)
```

The formula is a disjunction of all possible configurations of the subtree. Since the `HRS` is the parent of an `xor` group it can not be realized on its own. Furthermore, the realization of a child feature always depends on the realization of the parent.

The different departments collect and aggregate the opinions of their members as described in Section 4.1.2. In summary, the general opinion of the marketing department states that it is important to implement the `HRS`, but they do not have an opinion on whether it should be electrical or optical. On the other hand, the hardware department prefers an optical sensor because it is cheaper, while the software team has a slight preference for an electrical sensor and, if the decision is to implement an optical sensor, they are against the `HRV`, mainly because the team is currently operating at full capacity and they do not have the resources to do additional work. Table 3 presents these opinions.

Table 3: Opinions for the Heart Rate Sensor features.

	Marketing	Hardware	Software
HRS	(0.98, 0.01, 0.01, 0.5)	(0.7, 0.1, 0.2, 0.5)	(0.6, 0.2, 0.2, 0.5)
E	(0.1, 0.1, 0.8, 0.5)	(0.15, 0.8, 0.05, 0.5)	(0.7, 0.1, 0.2, 0.5)
O	(0.1, 0.1, 0.8, 0.5)	(0.9, 0.05, 0.05, 0.5)	(0.7, 0.1, 0.2, 0.5)
HRV	(0.1, 0.1, 0.8, 0.5)	(0.2, 0.1, 0.7, 0.5)	(0.1, 0.8, 0.1, 0.5)

After applying the formula above and fusing the opinions with the CBF, the resulting opinion is `SBoolean(0.733, 0.000, 0.267, 0.508)` with projection `0.87`. Based on this result, it is decided that the heart rate monitor will be presented to the shareholders.

The next step is to decide on the technical level how exactly the `HRS` is going to be implemented. For this, each possible branch of the tree needs to be explored. Table 4 presents the opinions of each department for each branch, as well as the fused opinions for the branch using the CFB fusion operator and its projection.

Given these results, the company decides that the features `HRS` and `O` will be realized, while the features `E` and `HRV` will not.

4.2 Determining Feature Priorities

It is one thing to determine whether a feature should be realized or not, but in the end what needs to be decided is which feature(s) should be realized for the next release, taking into account the available development resources. Therefore, we suggest ranking the features based on the projection value of the fused opinions. The resulting list can then be used to prioritize feature realizations.

For example, for our smartwatch SPL the resulting prioritized feature list for the next release is:

(1) 0.87: `NFC`
(2) 0.83: `ECBF`
(3) 0.73: `CP` (assuming that `NFC` is already realized)
(4) 0.71: `HRS` and `O`

With this information, and information on how much resources are needed to realize each feature and how much resources are available, the decision on which features to implement for the next release can be made.

Table 4: Opinions for the branches of the Heart Rate Sensor subtree.

	Marketing	Hardware	Software	Fused	Projection
HRS ∧ E	(0.360, 0.109, 0.531, 0.250)	(0.127, 0.820, 0.053, 0.250)	(0.507, 0.280, 0.213, 0.250)	(0.000, 0.026, 0.974, 0.250)	0.24344
HRS ∧ O	(0.360, 0.109, 0.531, 0.250)	(0.702, 0.145, 0.153, 0.250)	(0.507, 0.280, 0.213, 0.250)	(0.614, 0.000, 0.386, 0.250)	0.710287
HRS ∧ O ∧ HRV	(0.167, 0.198, 0.635, 0.125)	(0.355, 0.231, 0.414, 0.125)	(0.075, 0.856, 0.069, 0.125)	(0.019, 0.000, 0.981, 0.125)	0.141517

5 RELATED WORK

Subjective logic has been applied in several domains such as artificial intelligence [16], software engineering [36], or digital humanities [23]. In the software engineering domain, Bertoa et al. [5] applied subjective logic to software models (e.g., primitive datatypes in UML/OCL models) to enrich them with individual opinions from experts and reach agreements about the uncertainty of the values obtained from physical measurements or user estimations. Burgueño et al. [6] proposed an approach to applying subjective logic to model-based software engineering (MBSE) dealing with belief uncertainty in domain models.

To the best of our knowledge and according to the survey presented in [36], subjective logic has not been applied before to SPLs. However, several works deal with uncertainty and probabilities in SPLs for decision-making at different stages of the SPL process, including requirements elicitation [10, 32, 33], product configuration [7, 24–27, 29], and automated analysis of feature models [13, 14]. In fact, recently, Horcas et al. [14] identify up to ten SPL problems where uncertainty is present in the decision-making process, including product configuration and reverse engineering of feature models. They propose a simulations-based framework to apply Monte Carlo methods which deal with the uncertainty of the large configuration spaces of SPLs. However, only five of the ten identified problems are modeled and solved, while the next release problem and evolution of feature models are only mentioned. Sree-Kumar et al. [32, 33] proposed a method to check whether a feature model meets the textual specifications of the SPL requirements. They assign a *confidence score* [0..1] to each feature and relationship in the feature model that measures the likelihood that the textual requirements identify this element as relevant in the SPL. However, such a confidence score is automatically assigned using natural language processing (NLP) based on the correctness (choice of representative elements) and completeness (no missing elements) of the feature model, and thus, the values do not consider subjective opinions from relevant stakeholders. Famelis et al. [10] made a research vision about combining variability abstractions with *partial modeling* [11], a technique for managing design uncertainty within a software model [9] that explicates decision points and represents the set of possible models that could be obtained by making decisions and resolving design uncertainty.

Product configuration is the SPL process where decision-making techniques have been applied the most so far. The concept of *probabilistic feature models* [7] was introduced to automate the choice propagation of features according to the constraints by applying an *entropy measure* to guide the configuration process. Also, *Feature relations graphs (FRoGs)* [24] show the impact of a given feature on all other features using a *confidence metric* that represents the probability of finding a product that violates a constraint. Both works [7, 24] also rely on historical data to extract probabilities without taking into account opinions from domain experts. In fact, historical data from previous users' configurations has been a wide source of knowledge to feed recommendation systems and guide decisions in the next release problem. Rodas-Silva et al. [29] propose a recommender system for the selection of the best components set to implement a given configuration of the SPL based on the users' rating of such components. Mazo et al. [25] proposed recommendation heuristics to prioritize choices and recommend candidate features to be configured. The purpose of their approach is to reduce the number of configuration steps and optimize the computation time required by the solver to propagate the configuration choices. Nöhrer et al. [26] investigate the ordering of the decisions when configuring a product to automatically optimize user guidance by reducing the number of decisions that need answering. Pereira et al. [27] proposed a feature-based personalized recommender system for product-line configuration that guides decision-makers in understanding users' needs and preferences. It focuses on decision-makers who lack the sufficient personal experience to evaluate the complex technical properties of the features. The main drawback of these works [25–27, 29] is that they do not take into account the expertise of the domain experts, only the final users' rating [29], users' needs and preferences [27], or historical data from previous users' configurations [25, 26] are considered.

Finally, in [30] the authors propose an approach for decision support on the uncertainty in feature model evolution. They assist the selection of an optimal configuration, which is a set of features to be implemented, and define two evolution models: *Evolution Possibility Model* (ePM) to describe potential possibilities a feature model could evolve, and *Evolutionary Feature Model* (eFM) to describe the feature model with all changes due to evolution incorporated. They use two analysis techniques to facilitate the decision support: *Survivability analysis* that answers whether a configuration (i.e., set of features) could survive during the expected evolution, and *repair cost analysis* that answers which configuration requires less effort to get repaired due to changes.

6 CONCLUSIONS AND FUTURE WORK

In this paper we outlined how the uncertainty that inevitably arises during the development and evolution of an SPL can be dealt with in a rigorous way using subjective logic. In particular, we described three feature model evolution scenarios, and explained in detail how to quantify the opinions of different stakeholders when deciding on which features to implement for the next release problem, and how to combine the opinions taking into account feature dependencies. Different decision strategies (e.g., risky vs. conservative) are supported through the use of different fusion operators. We envision that subjective logic could become a new tool of great value in the SPL practitioner's toolkit whenever opinions of different stakeholders have to be taken into account.

As future work we plan to investigate how our approach can consider complex constraints [19] beyond requires and excludes constraints and other variability modeling relationships [15].

REFERENCES

[1] Nazakat Ali, Sangwon Hwang, and Jang-Eui Hong. 2019. Your Opinions Let us Know: Mining Social Network Sites to Evolve Software Product Lines. *KSII Trans. Internet Inf. Syst.* 13, 8 (2019), 4191–4211. https://doi.org/10.3837/tiis.2019.08.021

[2] Sven Apel, Don S. Batory, Christian Kästner, and Gunter Saake. 2013. *Feature-Oriented Software Product Lines - Concepts and Implementation.* Springer. https://doi.org/10.1007/978-3-642-37521-7

[3] Ebrahim Bagheri and Dragan Gasevic. 2011. Assessing the maintainability of software product line feature models using structural metrics. *Softw. Qual. J.* 19, 3 (2011), 579–612. https://doi.org/10.1007/s11219-010-9127-2

[4] Anthony J. Bagnall, Victor J. Rayward-Smith, and Ian M. Whittley. 2001. The next release problem. *Inf. Softw. Technol.* 43, 14 (2001), 883–890. https://doi.org/10.1016/S0950-5849(01)00194-X

[5] Manuel F. Bertoa, Lola Burgueño, Nathalie Moreno, and Antonio Vallecillo. 2020. Incorporating measurement uncertainty into OCL/UML primitive datatypes. *Softw. Syst. Model.* 19, 5 (2020), 1163–1189. https://doi.org/10.1007/s10270-019-00741-0

[6] Lola Burgueño, Paula Muñoz, Robert Clarisó, Jordi Cabot, Sébastien Gérard, and Antonio Vallecillo. 2022. Dealing with Belief Uncertainty in Domain Models. *ACM Trans. Softw. Eng. Methodol.* (jun 2022). https://doi.org/10.1145/3542947

[7] Krzysztof Czarnecki, Steven She, and Andrzej Wasowski. 2008. Sample Spaces and Feature Models: There and Back Again. In *12th International Conference on Software Product Lines (SPLC)*. IEEE Computer Society, 22–31. https://doi.org/10.1109/SPLC.2008.49

[8] Bruno De Finetti. 2017. *Theory of probability: A critical introductory treatment.* Vol. 6. John Wiley & Sons. https://doi.org/10.1002/9781119286387

[9] Mouna Dhaouadi, Kate M. B. Spencer, Megan H. Varnum, Alicia M. Grubb, and Michalis Famelis. 2021. Towards a Generic Method for Articulating Design-time Uncertainty. *J. Object Technol.* 20, 3 (2021), 3:1–14. https://doi.org/10.5381/jot.2021.20.3.a3

[10] Michalis Famelis, Julia Rubin, Krzysztof Czarnecki, Rick Salay, and Marsha Chechik. 2017. Software Product Lines with Design Choices: Reasoning about Variability and Design Uncertainty. In *ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. 93–100. https://doi.org/10.1109/MODELS.2017.3

[11] Michalis Famelis, Rick Salay, and Marsha Chechik. 2012. Partial models: Towards modeling and reasoning with uncertainty. In *34th International Conference on Software Engineering (ICSE)*. 573–583. https://doi.org/10.1109/ICSE.2012.6227159

[12] Marc Hentze, Chico Sundermann, Thomas Thüm, and Ina Schaefer. 2022. Quantifying the variability mismatch between problem and solution space. In *25th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. ACM, Montreal, Quebec, Canada, 322–333. https://doi.org/10.1145/3550355.3552411

[13] Ruben Heradio, David Fernández-Amorós, Christoph Mayr-Dorn, and Alexander Egyed. 2019. Supporting the statistical analysis of variability models. In *41st International Conference on Software Engineering (ICSE)*. IEEE / ACM, 843–853. https://doi.org/10.1109/ICSE.2019.00091

[14] José Miguel Horcas, José A. Galindo, Ruben Heradio, David Fernández-Amorós, and David Benavides. 2023. A Monte Carlo tree search conceptual framework for feature model analyses. *J. Syst. Softw.* 195 (Jan. 2023), 111551. https://doi.org/10.1016/j.jss.2022.111551

[15] José Miguel Horcas, Mónica Pinto, and Lidia Fuentes. 2023. A modular metamodel and refactoring rules to achieve software product line interoperability. *J. Syst. Softw.* 197 (2023), 111579. https://doi.org/10.1016/j.jss.2022.111579

[16] Eyke Hüllermeier and Willem Waegeman. 2021. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Mach. Learn.* 110, 3 (2021), 457–506. https://doi.org/10.1007/s10994-021-05946-3

[17] Audun Jøsang. 2016. *Subjective Logic - A Formalism for Reasoning Under Uncertainty.* Springer. https://doi.org/10.1007/978-3-319-42337-1

[18] Audun Jøsang, Dongxia Wang, and Jie Zhang. 2017. Multi-source fusion in subjective logic. In *20th International Conference on Information Fusion (FUSION)*. 1–8. https://doi.org/10.23919/ICIF.2017.8009820

[19] Alexander Knüppel, Thomas Thüm, Stephan Mennicke, Jens Meinicke, and Ina Schaefer. 2017. Is there a mismatch between real-world feature models and product-line research?. In *11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*. ACM, 291–302. https://doi.org/10.1145/3106237.3106252

[20] Baoding Liu. 2010. *Uncertainty Theory.* Springer Berlin Heidelberg, Berlin, Heidelberg, 1–79. https://doi.org/10.1007/978-3-642-13959-8_1

[21] Abdullah Al Mamun, Fahim Djatmiko, and Mridul Kanti Das. 2016. Binary multi-objective PSO and GA for adding new features into an existing product line. In *19th International Conference on Computer and Information Technology (ICCIT)*. 581–585. https://doi.org/10.1109/ICCITECHN.2016.7860263

[22] Maíra Marques, Jocelyn Simmonds, Pedro O. Rossel, and María Cecilia Bastarrica. 2019. Software product line evolution: A systematic literature review. *Information and Software Technology* 105 (2019), 190–208. https://doi.org/10.1016/j.infsof.2018.08.014

[23] Patricia Martin-Rodilla and Cesar Gonzalez-Perez. 2019. Conceptualization and Non-Relational Implementation of Ontological and Epistemic Vagueness of Information in Digital Humanities. *Informatics* 6, 2 (2019). https://doi.org/10.3390/informatics6020020

[24] Jabier Martinez, Tewfik Ziadi, Raúl Mazo, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. 2014. Feature Relations Graphs: A Visualisation Paradigm for Feature Constraints in Software Product Lines. In *2nd IEEE Working Conference on Software Visualization (VISSOFT)*. IEEE Computer Society, Victoria, BC, Canada, 50–59. https://doi.org/10.1109/VISSOFT.2014.18

[25] Raúl Mazo, Cosmin Dumitrescu, Camille Salinesi, and Daniel Diaz. 2014. Recommendation Heuristics for Improving Product Line Configuration Processes. In *Recommendation Systems in Software Engineering*. Springer, 511–537. https://doi.org/10.1007/978-3-642-45135-5_19

[26] Alexander Nöhrer and Alexander Egyed. 2013. C2O configurator: a tool for guided decision-making. *Autom. Softw. Eng.* 20, 2 (2013), 265–296. https://doi.org/10.1007/s10515-012-0117-4

[27] Juliana Alves Pereira, Pawel Matuszyk, Sebastian Krieter, Myra Spiliopoulou, and Gunter Saake. 2018. Personalized recommender systems for product-line configuration processes. *Comput. Lang. Syst. Struct.* 54 (2018), 451–471. https://doi.org/10.1016/j.cl.2018.01.003

[28] Mikko Raatikainen, Juha Tiihonen, and Tomi Männistö. 2019. Software product lines and variability modeling: A tertiary study. *Journal of Systems and Software* 149 (2019), 485–510. https://doi.org/10.1016/j.jss.2018.12.027

[29] Jorge Rodas-Silva, José Angel Galindo, Jorge García-Gutiérrez, and David Benavides. 2019. Selection of Software Product Line Implementation Components Using Recommender Systems: An Application to Wordpress. *IEEE Access* 7 (2019), 69226–69245. https://doi.org/10.1109/ACCESS.2019.2918469

[30] Le Minh Sang Tran and Fabio Massacci. 2014. An Approach for Decision Support on the Uncertainty in Feature Model Evolution. In *IEEE 22nd International Requirements Engineering Conference (RE)*. 93–102. https://doi.org/10.1109/RE.2014.6912251

[31] Glenn Shafer. 1976. *A mathematical theory of evidence.* Vol. 42. Princeton university press.

[32] Anjali Sree-Kumar, Elena Planas, and Robert Clarisó. 2018. Extracting Software Product Line Feature Models from Natural Language Specifications. In *22nd International Systems and Software Product Line Conference (SPLC)*, Vol. 1. ACM, 43–53. https://doi.org/10.1145/3233027.3233029

[33] Anjali Sree-Kumar, Elena Planas, and Robert Clarisó. 2021. Validating Feature Models With Respect to Textual Product Line Specifications. In *15th International Working Conference on Variability Modelling of Software-Intensive Systems (VaMoS)*. ACM, Krems, Austria, 15:1–15:10. https://doi.org/10.1145/3442391.3442407

[34] Chico Sundermann, Michael Nieke, Paul Maximilian Bittner, Tobias Heß, Thomas Thüm, and Ina Schaefer. 2021. Applications of #SAT Solvers on Feature Models. In *15th International Working Conference on Variability Modelling of Software-Intensive Systems (VaMoS)*. ACM, Krems, Austria, 12:1–12:10. https://doi.org/10.1145/3442391.3442404

[35] Thomas Thüm, Don S. Batory, and Christian Kästner. 2009. Reasoning about edits to feature models. In *31st International Conference on Software Engineering (ICSE)*. IEEE, Vancouver, Canada, 254–264. https://doi.org/10.1109/ICSE.2009.5070526

[36] Javier Troya, Nathalie Moreno, Manuel F. Bertoa, and Antonio Vallecillo. 2021. Uncertainty representation in software models: a survey. *Softw. Syst. Model.* 20, 4 (2021), 1183–1213. https://doi.org/10.1007/s10270-020-00842-1

[37] Muhammad Irfan Ullah, Günther Ruhe, and Vahid Garousi. 2010. Decision support for moving from a single product to a product portfolio in evolving software systems. *J. Syst. Softw.* 83, 12 (2010), 2496–2512. https://doi.org/10.1016/j.jss.2010.07.049

[38] Rens W. Van Der Heijden, Henning Kopp, and Frank Kargl. 2018. Multi-Source Fusion Operations in Subjective Logic. In *21st International Conference on Information Fusion (FUSION)*. 1990–1997. https://doi.org/10.23919/ICIF.2018.8455615

[39] Hans-Jürgen Zimmermann. 2001. *Fuzzy set theory—and its applications.* Springer Science & Business Media.