The Journal of Systems and Software 219 (2025) 112235

Contents lists available at ScienceDirect

The Journal of Systems & Software

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

Leveraging belief uncertainty for informed decision making in software product line evolution☆

Jose-Miguel Horcas [a], Lola Burgueño [a,*], Jörg Kienzle [a,b]

[a] ITIS Software, Universidad de Málaga, Spain
[b] McGill University, Montreal, Canada

ARTICLE INFO

Dataset link: https://github.com/atenearesearchgroup/fms-subjectivelogic

Keywords:
Decision making support
Feature model
Software product line
Belief uncertainty
Subjective logic
Evolution
Next release problem
Variability reduction

ABSTRACT

Software Product Lines (SPL) are not static software artifacts, but they evolve over time. The planning, realization, and release of a SPL requires many high-level decisions involving many different stakeholders with different expertise. Taking their opinions into account to make the right decisions is not trivial. Currently there are no mechanisms to assist stakeholders in the decision making process in an informed manner. In this paper, we propose the use of belief uncertainty in conjunction with feature models to assist in the evolution of SPLs by explicitly quantifying opinions and their associated uncertainty. We present a novel approach in which subjective logic is used to represent the opinions of stakeholders in three evolution scenarios, namely feature model evolution, next release problem and variability reduction. We apply our approach to the evolution of the Xiaomi MiBand SmartWatch SPL over the time period from July 2014 to October 2023. We present an implementation of our approach and evaluate its scalability.

1. Introduction

The planning, realization and release of a *software product line* (SPL) centers around features (Apel et al., 2013). In fact, *feature models* have become the main artifact that guides the whole SPL process (Raatikainen et al., 2019), 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 (Marques et al., 2019). 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* Ali et al., 2019; Marques et al., 2019). Another question might be to identify which features should be realized for the coming release (aka *the next release problem* Bagnall et al., 2001; Mamun et al., 2016; Ullah et al., 2010). 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* Hentze et al., 2022; Sundermann et al., 2021b).

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. Conflicts often arise among stakeholders due to their differing opinions (or beliefs) about a specific statement, making it difficult to reach a consensus. The uncertainty associated to an opinion cannot be neglected. Having a measure of the degree of uncertainty associated with a resulting decision can be useful, for example, to discard any decision whose degree of uncertainty is above a given threshold. This cannot be achieved with standard probabilistic logic, but it is possible with our proposal. Moreover, in an SPL, features often have dependencies between them, and decisions about specific features must consider those dependencies, increasing their uncertainty.

In this paper, we extend our initial ideas presented in our short SPLC paper (Burgueño et al., 2023) where we proposed a novel approach that uses *belief uncertainty* (Troya et al., 2021) in conjunction with feature models to explicitly quantify the opinions of stakeholders regarding the evolution of an SPL. We outlined how *subjective logic* (Jøsang, 2016), a formalism for reasoning under belief uncertainty, can be used to represent and combine the opinions of stakeholders and explained how we envision it could be used to make decisions in the context of the next release problem. In this paper, we extend these initial ideas as follows: *(i)* we now apply our approach to *three different problems*, namely feature model evolution, next release problem, and variability reduction,

☆ Editor: Professor Laurence Duchien.

* Corresponding author.
E-mail address: lolaburgueno@uma.es (L. Burgueño).

https://doi.org/10.1016/j.jss.2024.112235
Received 27 February 2024; Received in revised form 1 August 2024; Accepted 27 September 2024
Available online 16 October 2024

(ii) we illustrate our approach by applying it to a industrial example, i.e., the Xiaomi MiBand SmartWatch SPL as it evolved between July 2014 and October 2023, *(iii)* we have *implemented a tool* and have evaluated our approach in terms of feasibility and scalability, and *(iv)* we have *proposed literals to ease the provision of opinions* using subjective logic for stakeholders if desired.

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 on the *Xiaomi Smartwatch SPL*. Section 4 applies our approach to the three evolution scenarios, again in the context of the *Smartwatch* SPL, and presents our proposed opinion literals. Section 5 provides an overview of our proof-of-concept implementation in Python and discusses feasibility and scalability issues of our approach. Section 6 presents related work, and Section 7 draws some conclusions and outlines the future work.

2. Uncertainty and subjective logic

This section introduces the main concepts of belief uncertainty and subjective logic that we use throughout the paper.

2.1. Types of uncertainty

Uncertainty (Liu, 2010) 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 be classified into two categories according to its nature: *aleatory*, when it is due to probabilistic variability or randomness; and *epistemic*, when it is due to lack of knowledge. While aleatory uncertainty is irreducible, epistemic uncertainty can be reduced with additional information or knowledge.

Uncertainty can take different forms *according to its source*. For example, *measurement uncertainty* (Joint Committee for Guides in Metrology, 2008) 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* (Troya et al., 2021) is a particular type of *epistemic* 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, by definition, has a subjective nature.

In Troya et al. (2021) 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 De Finetti, 2017 or in Uncertainty theory Liu, 2010), *possibilities* (in Fuzzy set theory Zimmermann, 2001), *plausibilities* (in the Dempster–Shafer theory of evidence Shafer, 1976) or *opinions* (in subjective logic Jøsang, 2016).

2.2. Belief uncertainty and subjective logic

In this paper, our focus is on belief uncertainty, and we use *subjective logic* (Jøsang, 2016), which is a type of probabilistic logic, to capture the beliefs of stakeholders.

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) the degree of *disbelief* d that the statement is true (i.e., the belief 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 (Burgueño et al., 2022) defined as a 4-tuple (b, d, u, a), and satisfying $b + d + u = 1$, and $b, d, u, a \in [0, 1]$.

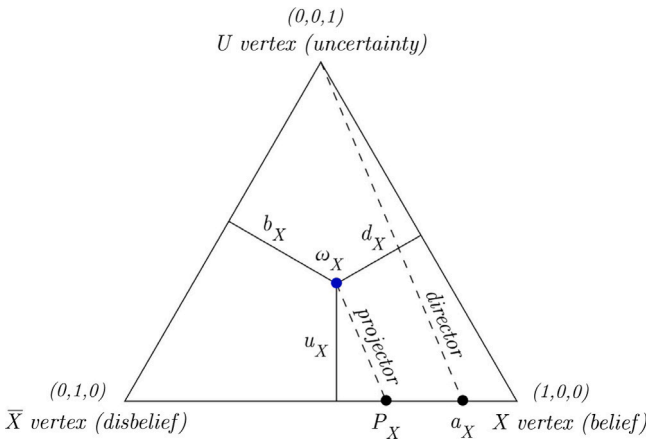

Fig. 1. A subjective Boolean opinion represented using Barycentric coordinates (Jøsang, 2016).

Intuitively, the *base rate* of an opinion represents the *objective* probability that can be assigned to the statement using *a priori* evidences or statistical estimates, whilst the other elements of the tuple represent the *subjective* degrees of belief, disbelief, and uncertainty about the statement assigned by an agent (whom we will call *stakeholder*). Thus, regardless of the value of the prior probability, different stakeholders can express their subjective opinions about the statement, including their degree of uncertainty.[1] With this definition, the type `Boolean` can be embedded into `SBoolean` as follows: the Boolean value `true` is represented as `SBoolean` $(1, 0, 0, x)$, and `false` as `SBoolean` $(0, 1, 0, x)$, where the base rate can take any value $\in$ [0,1].

Opinions can be represented on an equilateral triangle using Barycentric coordinates as shown in Fig. 1 (Jøsang, 2016). A point inside the triangle represents a (b_x, d_x, u_x) triple. Vertices at the bottom represent absolute opinions, and the vertex at the top represents the vacuous opinion ($u_x = 1$). Dogmatic opinions belong to the baseline ($u_x = 0$), and correspond to probabilities. The base rate a_x, or prior probability, is shown along the baseline, too.

The *projected probability* (or *projection*) of an opinion is defined as $P_x = b_x + a_x u_x$. Graphically, it is formed by projecting the opinion ω_x onto the base, parallel to the base rate director line. The projected probability permits combining the objective and subjective values into one single probability, which modifies the prior base rate according to the opinion of the agent.

Let us illustrate the use of `SBoolean` with an example. Hana needs to assign a cost to an NFC chipset. She has been provided an estimated cost of 5K EUR. Based on the previous interactions with the company providing the estimate, the statistics show that in 95% of the cases the final price was accurate, and only in 5% of the cases the final price was higher or lower. However, given the current political situation worldwide and the fact that the price of hardware components has increased, she does not fully trust the quote. Therefore, she expresses her opinion on the fact that the cost will be the one provided as `SBoolean(0.6, 0.1, 0.3, 0.95)`, which means that she trusts the original prediction (0.95) 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* (Jøsang, 2016) (or *projection*) of an `SBoolean` opinion is defined as $P = b + a \times u$, and allows us to move

[1] Note that the benefits of subjective logic as opposed to Kleene logic (Kleene, 1938) is that Kleene logic does not include base rates and therefore probability projections cannot be derived.

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 price of the NFC chipset is 0.885 (88.5%).

Subjective Logic comes with the traditional logical operators (`and`, `or`, `implies`, etc.) (Jøsang, 2016), which are used to combine the opinions of the same person/agent about different statements. In previous work (Muñoz et al., 2020b), we implemented for the type `SBoolean` the basic operations `not`, `and`, and `or`; and secondary operators `implies`, `equivalence`, and `xor` of the traditional Boolean algebra by extending them to subjective logic.

Apart from the traditional logical operators, Subjective logic implements *fusion* operators for combining the subjective opinions of *different* people/agents about the same statement. The goal of these fusion operators is to produce a single opinion that better reflects the collection of opinions, or is closer to the truth than each opinion in isolation. We present them in detail in the following subsection.

2.3. Fusion operators

Subjective logic comes with a set of *fusion operators* Jøsang (2016), Jøsang et al. (2017), Van Der Heijden et al. (2018) that can be used to *combine opinions* of different agents about the same statement. Each fusion operator is designed for a specific purpose and scenario. Depending on the situation, the person in charge of merging the opinions needs to decide which fusion operator is the most suitable for each particular case.

In the following, we present a brief description of the fusion operators and illustrate their behavior by merging opinions $w_i = (b_i, d_i, u_i, a_i)$. The merged opinion will be denoted $w = (b, d, u, a)$. The situations in which they are applicable are also discussed.

- **Belief Constraint Fusion (BCF).** This operator is used when the stakeholders have committed their choices and will not change their minds, even if the result is that no consensus will be reached. It computes the overlapping beliefs (the relative *Harmony*) and the non-overlapping beliefs (the relative *Conflict*) of the opinions to fuse. More precisely, in the case of two opinions w_x and w_y, it computes $Har = b_x b_y + b_x u_y + b_y u_x$ and $Con = b_x d_y + b_y d_x$. Then, in the general case, the components of the fused opinion are defined as follows:

$$b = \frac{Har}{1 - Con}, d = 1 - b - u - a, u = \frac{u_x u_y}{1 - Con}, a = \frac{a_x(1 - u_x) + a_y(1 - u_y)}{2 - u_x - u_y}$$

  The divisor $(1 - Con)$ normalizes the belief mass and uncertainty, and ensures additivity. The constraint fusion operator is commutative and non-idempotent. Associativity is preserved when the base rate is equal for all agents. The result is undefined in case of totally conflicting opinions.
- **Consensus & Compromise Fusion (CCF).** This is applicable when stakeholders (possibly non-expert) have dependent opinions about the same fact (e.g., in a survey). It preserves shared beliefs and transforms conflicting opinions into vague beliefs. It is suitable for situations in which consensus is sought if it exists, and a vague opinion is acceptable in case of diverging opinions. This operator assumes *dependence* between opinions[2] and it is *idempotent* with the vacuous opinion as neutral element. It uses a three-steps process: (1) consensus; (2) compromise; (3) merge. The consensus step determines the shared beliefs and disbeliefs between the two opinions. The compromise step redistributes conflicting residual beliefs and disbeliefs to produce the so-called *compromise* belief. Finally, the merge step distributes that compromise between the shared belief and the uncertainty to compute the resulting opinion.[3] This operator is applicable in situations where (possibly non-expert) agents have dependent opinions about the same fact, such as when they are asked to give their opinions in a survey (Jøsang, 2016).
- **Averaging Belief Fusion (ABF).**
  This operator assumes that the agents' opinions are *dependent*. It is commutative, idempotent, and non-associative, and does not have a neutral element, i.e., every opinion, even a vacuous one, influences the fused result. This situation applies when all agents observe the same situation at the same time, and all opinions should be taken into account, for example, when a jury tries to reach a verdict after having observed the court proceedings.
  Basically, this operator computes the average of the opinions by weighing the belief of each opinion with the product of the uncertainty of the rest. In the case of three non-dogmatic opinions w_x, w_y and w_z, it is expressed as follows:

$$b = \frac{b_x u_y u_z + b_y u_x u_z + b_z u_x u_y}{u_x u_y + u_x u_z + u_y u_z}, d = 1 - b - u - a$$
$$u = \frac{3 u_x u_y u_z}{u_x u_y + u_x u_z + u_y u_z}, a = \frac{a_x + a_y + a_z}{3}$$

  If dogmatic opinions are present, this operator computes the average of the beliefs and disbeliefs of these opinions only, resulting in a dogmatic opinion too (Jøsang et al., 2017).
- **Weighted Belief Fusion (WBF).** This operator is similar to the ABF operator, but gives more weight to those opinions with less uncertainty i.e., the smaller the value of the uncertainty mass, the higher the weight—unlike ABF where all opinions have the same weight even when some of them are very uncertain. This operator has the vacuous opinion as neutral element and, like ABF, it is *idempotent*.
  To illustrate how WBF works, the following expression computes the WBF of three non-dogmatic opinions w_x, w_y and w_z (van der Heijden et al., 2018):

$$b = \frac{b_x(1 - u_x)u_y u_z + b_y(1 - u_y)u_x u_z + b_z(1 - u_z)u_x u_y}{u_x + u_y + u_z - 3u_x u_y u_z}, d = 1 - b - u - a$$
$$u = \frac{(3 - u_x - u_y - u_z)u_x u_y u_z}{u_x + u_y + u_z - 3u_x u_y u_z}, a = \frac{a_x(1 - u_x) + a_y(1 - u_y) + a_z(1 - u_z)}{3 - u_x - u_y - u_z}$$

- **Aleatory Cumulative Belief Fusion (ACBF).** This operator is applied when the evidences are independent, that is, the amount of evidence increases when more stakeholders give their opinion. This operator is suitable when giving opinions about a variable governed by a frequentist process, such as flipping a coin. Like the ABF, the ACBF operator calculates the mean of the opinions by weighting the belief of each opinion with the product of the uncertainty of the others. However, it modulates this mean by subtracting the product of the uncertainties. Thus, in the case of three non-dogmatic opinions, their aleatory cumulative fusion can be expressed as follows:

$$b = \frac{b_x u_y u_z + b_y u_x u_z + b_z u_x u_y}{u_x u_y + u_x u_z + u_y u_z - 2u_x u_y u_z}, d = 1 - b - u - a$$
$$u = \frac{u_x u_y u_z}{u_x u_y + u_x u_z + u_y u_z - 2u_x u_y u_z}, a = \frac{a_x + a_y + a_z}{3}$$

  Again, if dogmatic opinions are present, this operator computes the cumulative fusion of these opinions only, resulting in a dogmatic opinion too (Jøsang et al., 2017).

[2] Opinions are said to be *dependent* when they are based on the same or dependent evidence; and they are said to be *independent* when they are based on different or independent evidence.

[3] Due to the length of the algorithm that computes this operator we do not present it here but refer the reader to Jøsang (2016) (chapter 12).

Table 1
Fusion operators' properties.

	Belief Constraint Fusion (BCF)	Cumulative Belief Fusion ([A&E]CBF)	Averaging Belief Fusion (ABF)	Weighted Belief Fusion (WBF)	Consensus & Compromise Fusion (CCF)
Agents' willingness to compromise	–	✓	✓	✓	✓
Dependence between opinions	✓	–	✓	✓	✓
Vacuous opinion is neutral element	✓	✓	–	✓	✓
Preserve shared beliefs; conflicting opinions turned into vague beliefs	–	–	–	–	✓

- **Epistemic Cumulative Belief Fusion (ECBF).** This extends the ACBF operator . First, it calculates ACBF and then applies uncertainty maximization to the result. The *uncertainty-maximized* opinion $\ddot{w}$ of an opinion w is the opinion with maximum uncertainty that still preserves the same projected probability as w (Jøsang, 2016). Unlike ACBF, the ECBF is used for facts whose uncertainty is of epistemic nature.

More details on which fusion operator can be applied in each situation are introduced in Section 4. The interested reader can consult the formal definitions of the operators in Jøsang (2016) with some extensions in Jøsang et al. (2017), Van Der Heijden et al. (2018).

Since different opinions can be fused in various ways depending on how the fusion situation needs to be handled, we identified in Burgueño et al. (2022) the following characteristics that can be helpful to determine which operator to use in a specific case.

- Willingness to compromise. There is room for compromise even in case of totally conflicting opinions.
- Dependence between opinions. The opinions were formed witnessing the same events at the same time.
- Vacuous opinion as neutral element (idempotence). Fusion of vacuous opinions (i.e., those with uncertainty mass $u_x = 1$) have no effect in the result.
- Preserve shared beliefs, and conflicting opinions are turned into vague belief. In order to find a compromise, it is possible to turn conflicting opinions into vague beliefs.

Table 1 maps the fusion operators to these characteristics.

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 $FM_{realized}$, i.e., a model that contains all the features that have already been 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, which we call $FM_{planned}$. It holds that $FM_{reduced} \subseteq FM_{realized} \subseteq FM_{planned}$.

Fig. 2 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 Fig. 2)[4]:

1. **Feature Model Evolution** (Ali et al., 2019; Marques et al., 2019): 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 planned feature model ($FM_{planned,current}$) as a basis, and suggesting *edits* (Thüm et al., 2009) 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. As result, we obtain the next planned feature model ($FM_{planned,next}$).
2. **Next Release Problem** (Bagnall et al., 2001; Mamun et al., 2016; Ullah et al., 2010): 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 in the current realized feature model ($FM_{realized,current}$) (i.e., features that are part of $FM_{planned}$ and are not part of $FM_{realized}$) should the development team work on next, obtanining as result the next realized feature model ($FM_{realized,next}$).
3. **Variability Reduction** (Hentze et al., 2022; Sundermann et al., 2021b): 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 the current realized feature model ($FM_{realized,current}$) to reduce the variability (e.g., making an optional feature mandatory or deciding which additional cross-tree constraints to add), obtaining as result a reduced feature model ($FM_{reduced,current}$). Reducing the variability of a SPL simplifies maintainability and quality assurance (Bagheri and Gasevic, 2011), 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 by end users", to "Distribution expert: high maintenance cost", to "Developer: unclear how to implement efficiently".

3.2. Xiaomi Mi Band SPL example

The Xiaomi Mi Band is a wearable activity tracker that resembles a bracelet and can be worn on either wrist. The first version of the Mi Band was released in July 2014, followed by the Mi Band 1S in November 2015. There have been 17 models released so far, with the latest two being the Mi Band 8 and Mi Band 8 NFC in October 2023.

From the very beginning, Mi Band offered multiple features to users, e.g., a `FitnessMonitor` or `SleepTracker`. Since all Mi Band models provide these features, they are marked as mandatory. The Mi Band 1S added an additional feature, the `HeartRateSensor`, which therefore shows up as the first optional feature. As the SPL evolves, some features are also mutually exclusive. For example, over time, Xiaomi has evolved the Mi Band's support for the Bluetooth Communication Protocol from Bluetooth 4.0 to Bluetooth Low Energy

[4] In the following, we use the second subscript (*current* or *next*) in the feature model names FM to refer to the current or the next feature model in the evolution scenarios.

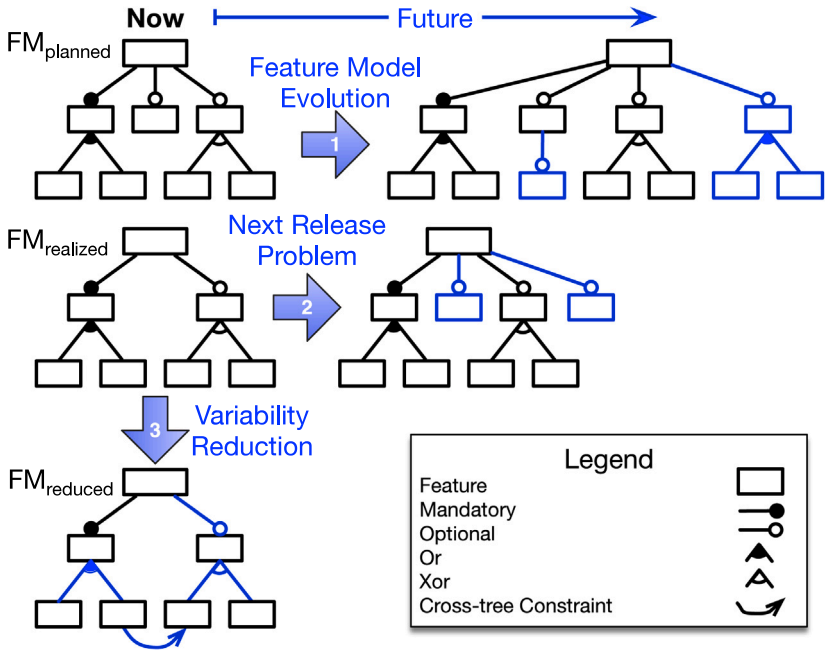

Fig. 2. SPL evolution scenarios for variability models.

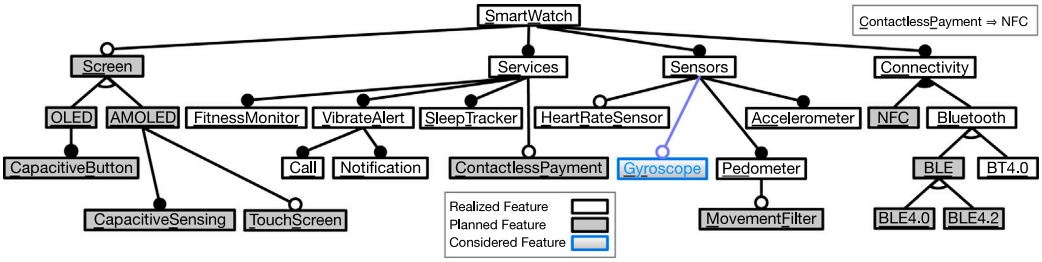

Fig. 3. Mi Band 1S evolution example.

BLEv4.0, BLEv4.2, BLEv5.0, BLEv5.1 and BLEv5.2. Because each model comes with only one Bluetooth implementation, the corresponding features in the feature model are in an *xor* relationship as shown in the feature model for Mi Band 2. At times there are also functional dependencies between features. For example, starting with Mi Band 3, `ContactlessPayment` is offered as an optional feature, but in order to support it, `NFC` must also be present. These feature dependencies are specified in the model with the cross-tree constraint e.g., `ContactlessPayment` ⇒ `NFC`.

Fig. 3 shows an example of the *Xiaomi Mi Band SPL* that we use in the following to illustrate our three scenarios. The whole SPL of the *Xiaomi Mi Band* is presented in Section 5.

Let us imagine we are in November 2015 when Mi Band 1S was just released. The feature model of the Mi Band product line at that point is represented using white features in Fig. 3. Note that there is only one optional feature, the `HeartRateSensor (HRS)`, and therefore only two possible products (the Mi Band 1 and the Mi Band 1S). This feature model represents $FM_{reduced}$ as well, i.e., all the white features have been implemented, and they are part of at least one of the two products. Both the white and gray features in Fig. 3 represent $FM_{planned}$. In other words, in November 2015, the features highlighted in gray represent the set of features that have not been realized yet, but that are very likely going to be part of a future Mi Band product.

Feature Model Evolution: According to our three evolution scenarios, the Mi Band 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 could propose that maybe in a long-term future it would make sense to incorporate a new gyroscope sensor into the SPL in order to improve step counting (see blue `Gyroscope` feature in Fig. 3).

Next Release Problem: 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 the $FM_{realized}$ for Mi Band 2). Here decisions may involve deciding the incorporation of an independent feature (e.g., `MovementFilter`), 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 `Screen` feature and its variants).

Variability Reduction: Finally, the third scenario is illustrated in the Mi Band 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.

4. Belief uncertainty applied to SPL

In this section, we illustrate our decision-making approach based on subjective logic using the three evolution scenarios presented above using the Mi Band SPL in three separate sections. Finally, we define Likert-type scales to ease the definition of opinions for non-expert stakeholders.

4.1. Scenario 1: Feature model evolution

Fig. 3 shows the plans for the Mi Band SPL ($FM_{planned}$) as they could have been in November 2015 when Mi Band 1S had just been released. As explained before, the features shown in white are already realized (i.e., from Mi Band 1 and 1S), whereas the gray features represent those features that are currently being considered for the near future (which we took from Mi Band 2 and 3, and even some of Mi Band 4). This feature model is intentionally not set in stone, but continuously evolves to keep up with technological developments and market trends. For instance, the marketing team could propose that in a long-term future it would make sense to incorporate a new gyroscope sensor into the SPL in order to improve step counting (see blue `Gyroscope` feature in Fig. 3).

4.1.1. Deciding on an independent feature

To decide whether the `Gyroscope` feature should be included in the set of planned features, representatives from the marketing, hardware, and software departments sit together and provide their opinions on the feature.

The marketing stakeholder Bob thinks that `Gyroscope`, which improves the precision of the `FitnessMonitor` and in particular step counting, is an interesting feature for physically active customers. It should 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(0.9, 0.0, 0.1, 0.5)`. That is, Bob has a belief of 0.9 on the fact that the `Gyroscope` should be included, a disbelief of 0, and an uncertainty of 0.1. 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 `Gyroscope` feature because it will have a considerable impact on power consumption. She knows that there is new hardware in development that also improves step counting precision and would affect the battery less. However, whether or not that technology will be available in time is uncertain. She gives her opinion on the `Gyroscope` feature 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 should not be too difficult to support the `Gyroscope` feature. Hence, it could make sense to add it to $FM_{planned}$ to investigate the feature's software requirements further. Their opinion therefore is `SBoolean(0.4, 0.05, 0.55, 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). The result of fusing the three opinions using the ECBF operator is: `SBoolean(0.45, 0.0, 0.55, 0.5)`. The three stakeholders decide that a feature will be included in the plan only when the projection of the fused opinion (i.e., the probability) is higher than 0.6. A lower threshold value would imply a low confidence and hence a high risk of wasting time investigating the planned feature, which they are not willing to take. The projection of the fused opinion in this case is 0.72. Therefore, the three stakeholders decide that the `Gyroscope` feature should be added to $FM_{planned}$.

4.2. Scenario 2: Next release problem

In the second scenario, the stakeholders need to make decisions about what features to implement and include in the next release (i.e., which gray features in Fig. 3 to be included in $FM_{realized}$ for Mi Band 2). Here decisions may involve: (i) deciding the realization of an independent feature (e.g., `MovementFilter`), (ii) deciding on a feature with dependencies on other features (e.g., to incorporate the `ContactlessPayment` feature, the `NFC` feature must be implemented too), or (iii) deciding on a group of related features, such as a variation point and its variants (e.g., the development of the `Screen` feature and its variants).

From a decision-making process point of view, when deciding on an independent feature for the next release problem, we face the same case as presented in Section 4.1.1. For instance, a rather straightforward decision is to determine whether `MovementFilter` should be realized or not. Here again, we can imagine that stakeholders from marketing, hardware, and software provide their opinions, e.g. `SBoolean(0.7, 0.1, 0.2, 0.5)`, `SBoolean(0.3, 0.5, 0.2, 0.5)`, and `SBoolean(0.7, 0.0, 0.3, 0.5)`. Again, we are facing a case of epistemic uncertainty, and the *Epistemic Cumulative Belief Fusion* (ECBF) fusion operator is applied, resulting in `SBoolean(0.38, 0.0, 0.62, 0.5)` with a projection of 0.69.

The two remaining cases (ii) and (iii) are not so straightforward, hence we present them in Sections 4.2.1 and 4.2.2, respectively. Furthermore, it could be the case that the stakeholders do not only want to decide on the individual realization or not of (groups of) features, but need rank all the possible options to determine which ones to realize based on the available development resources. We present this case in Section 4.2.3.

4.2.1. Deciding on a feature with cross-tree constraints

Xiaomi has suggested studying the viability of realizing contactless payment, but as Fig. 3 shows, the realization of the feature `ContactlessPayment` (CP) requires the realization of the feature `NFC`.[5] 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 two features are presented in Table 2.

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

$$\begin{aligned}&\texttt{FUSION}(OpEng_1CP \wedge OpEng_1NFC,\\ &\quad OpEng2CP \wedge OpEng1NFC,\\ &\quad OpEng3CP \wedge OpEng3NFC)\end{aligned}$$

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

[5] For space reasons, we use abbreviations (see underlined letters in Fig. 3 instead of the long feature names for the Mi Band SPL.

Table 2
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)

Table 3
Fused opinions for CP ∧ NFC.

Operator	Fused Opinion	Projection
AverageBeliefFusion	(0.715, 0.206, 0.078, 0.25)	0.734787
WeightedBeliefFusion	(0.724, 0.203, 0.073, 0.25)	0.742006
ConsensusAndCompromiseFusion	(0.524, 0.190, 0.286, 0.25)	0.595724

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 3 presents the fused opinions for all three operators.

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

4.2.2. Dealing with feature relationships and complex constraints

Our approach can also handle more complex decisions involving multiple features. Each stakeholder has to express an independent opinion on each feature involved in the decision, and then we apply the boolean formula that corresponds to the feature relationships and constraints between the features specified in the feature model. As such, we can deal with feature model parent–child constraints (*mandatory*, *optional*, *or*, *xor*), the standard cross-tree constraints (requires and *excludes*), as well as any other more complex constraints that can be mapped to a boolean formula that uses the boolean operators *and*, *or*, and *not*.

For instance, in the Mi Band SPL, the move to Mi Band 2 required a more complex decision to be made. Mi Band 1 and 1S did not have a screen. For the next meeting with its main shareholders, Xiaomi needs to determine whether the `Screen` feature and its variants should be implemented or not for Mi Band 2. We can observe in Fig. 3 that there are different options: the screen type could be `AMOLED`, which has capacitive sensing, or `OLED`, which is a lot cheaper but needs a `CapacitiveButton` for capturing input. With an `AMOLED` screen, it would even potentially be possible to go for a full-fledged `Touchscreen`.

Once everyone gives their opinion on each feature of the tree, the individual opinion of that person on the fact that one way or another `Screen` is included in the next realization is calculated with the following formula:

$$(\texttt{Scr} \wedge \texttt{OLED}) \vee (\texttt{Scr} \wedge \texttt{AMOLED}) \vee (\texttt{Scr} \wedge \texttt{AMOLED} \wedge \texttt{TS})$$

The formula is a disjunction of all possible configurations of the subtree. Since the `Screen` is the parent of an `xor` group it cannot 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 4.2.1. In summary, the general opinion of the marketing department states that it is important to implement `Screen`, but they do not have an opinion on whether it should be `OLED` or `AMOLED`. On the other hand, the hardware department prefers `OLED` because it is cheaper, while the software team has a slight preference for `AMOLED` and, if the decision is to go for `AMOLED`, they are against the `TS`, mainly because the team is currently operating at full capacity and they do not have the resources to do additional work. Table 4 presents these opinions.

After applying the formula above and fusing the opinions with the CBF operator, 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 `Screen` will be presented to the shareholders.

The next step is to decide on the technical level how exactly the `Screen` is going to be implemented. For this, each possible branch of the tree needs to be explored. Table 5 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.

Based on these results, the features `Screen` and `OLED` are clearly the most likely ones to be realized, but `AMOLED` and `TouchScreen` could also be considered.

4.2.3. 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 *the set of features* that should be realized for the next release, taking into account the available development resources. To this aim, 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, in our Mi Band SPL, if we use the previously calculated opinions for the features `MF`, `NFC`, `CP`, `Scr`, `OLED`, `AMOLED` and `TS`, and add to that the fused opinions for `BLE` SBoolean(0.95, 0.0, 0.05, 0.5), `BLE4.0` SBoolean(0.9, 0.0, 0.1, 0.5) and `BLE4.2` SBoolean(0.85, 0.0, 0.15, 0.5), then the resulting prioritized feature list for the next release is:

1. 0.975: BLE
2. 0.972: Scr
3. 0.926: BLE and BLE4.0
4. 0.914: Scr and OLED (and hence also CB)
5. 0.902: BLE and BLE4.2
6. 0.900: MF
7. 0.762: NFC
8. 0.746: Scr and AMOLED
9. 0.600: Scr and AMOLED and TS
10. 0.589: CP (knowing that NFC is already realized)

With this information, and information on how much resources are needed to realize each feature and how many resources are available, the decision on which features to implement for the next release can be made. In our example, given the development constraints and the fact that there is a clear gap between the projected probabilities of `MF` and `NFC`, Xiaomi decided to realize the top 6 features for Mi Band 2.

4.3. Scenario 3: Variability reduction

Assuming we are approaching June 2016 and the development of the 6 new features (`BLE`, `Scr`, `BLE4.0`, `BLE4.2`, `OLED` (and hence also `CB`), and `MF`) are well underway. With those features, and the optional feature `HRS` from Mi Band 1S, $FM_{realized}$ of the Mi Band SPL (white and gray features of Fig. 3) now has technically 24 different

Table 4
Opinions for the Screen Features.

	Marketing	Hardware	Software
Scr	(0.98, 0.01, 0.01, 0.5)	(0.7, 0.1, 0.2, 0.5)	(0.6, 0.2, 0.2, 0.5)
OLED	(0.95, 0.02, 0.03, 0.5)	(0.9, 0.0, 0.1, 0.5)	(0.8, 0.1, 0.1, 0.5)
AMOLED	(0.9, 0.05, 0.05, 0.5)	(0.5, 0.4, 0.1, 0.5)	(0.3, 0.5, 0.2, 0.5)
TS	(0.9, 0.1, 0.0, 0.5)	(0.7, 0.2, 0.1, 0.5)	(0.2, 0.6, 0.2, 0.5)

Table 5
Opinions for the Branches of the Screen Subtree.

	Marketing	Hardware	Software
Scr ∧ OLED	(0.944, 0.030, 0.026, 0.250)	(0.713, 0.100, 0.187, 0.250)	(0.553, 0.280, 0.167, 0.250)
Scr ∧ AMOLED	(0.901, 0.060, 0.039, 0.250)	(0.407, 0.460, 0.133, 0.250)	(0.240, 0.600, 0.160, 0.250)
Scr ∧ AMOLED ∧ TS	(0.816, 0.154, 0.030, 0.125)	(0.315, 0.568, 0.117, 0.125)	(0.073, 0.840, 0.087, 0.125)

	Fused	Projection
Scr ∧ OLED	(0.885, 0.000, 0.225, 0.250)	0.914
Scr ∧ AMOLED	(0.661, 0.000, 0.339, 0.250)	0.746
Scr ∧ AMOLED ∧ TS	(0.542, 0.000, 0.458, 0.125)	0.600

configurations. It does not make sense to have that many versions of Mi Band on the market, and hence the variability of the SPL needs to be reduced to a more manageable size.

Again, subjective logic can help with the required decision making. We suggest that in this case, one needs to gather the opinions of the relevant stakeholders on the different *possible products*. Two of the 24 possible products correspond to Mi Band 1 and Mi Band 1S, i.e., they have already been released, which leaves 22 new possible configurations to consider. For instance, one could imagine running market surveys to gather the opinions of different stakeholder groups on specific products, and then combine the opinions using the appropriate fusion operator. In the end, the products are ranked according to their projection, and depending on how many different configurations the company is willing to support, the top x products are released.

For example, for determining the variability reduction for the Mi Band 2 release, a market study could have resulted in the following ranked list of products:

1. (0.920, 0.000, 0.080, 0.5) → 0.960: Scr, OLED, CB, HRS, BLE, BLE4.0
2. (0.880, 0.020, 0.100, 0.5) → 0.930: Scr, OLED, CB, HRS, BLE, BLE4.2
3. (0.880, 0.050, 0.070, 0.5) → 0.915: Scr, OLED, CB, MF, BLE, BLE4.0
4. (0.860, 0.050, 0.090, 0.5) → 0.905: Scr, OLED, CB, MF, BLE, BLE4.2
5. (0.750, 0.080, 0.170, 0.5) → 0.835: Scr, OLED, CB, BLE, BLE4.2
6. (0.760, 0.090, 0.150, 0.5) → 0.835: Scr, OLED, CB, BLE, BLE4.0
7. (0.680, 0.260, 0.060, 0.5) → 0.710: HRS, BLE, BLE4.0
8. (0.650, 0.270, 0.080, 0.5) → 0.690: HRS, BLE, BLE4.2
9. (0.590, 0.320, 0.090, 0.5) → 0.635: Scr, OLED, CB, MF, HRS, BLE, BLE4.2
10. (0.580, 0.350, 0.070, 0.5) → 0.615: Scr, OLED, CB, MF, HRS, BLE, BLE4.0
11. (0.540, 0.410, 0.050, 0.5) → 0.565: MF, HRS, BLE, BLE4.0
12. (0.520, 0.390, 0.090, 0.5) → 0.565: MF, BLE, BLE4.2
13. (0.390, 0.290, 0.320, 0.5) → 0.550: BLE, BLE4.2
14. (0.480, 0.450, 0.070, 0.5) → 0.515: MF, BLE, BLE4.0
15. (0.420, 0.460, 0.120, 0.5) → 0.480: BLE, BLE4.0
16. (0.380, 0.580, 0.040, 0.5) → 0.400: MF, HRS, BLE, BLE4.2
17. (0.220, 0.720, 0.060, 0.5) → 0.250: Scr, OLED, CB, MF, HRS, BT40
18. (0.200, 0.750, 0.050, 0.5) → 0.225: Scr, OLED, CB, HRS, BT40
19. (0.170, 0.725, 0.105, 0.5) → 0.223: Scr, OLED, CB, MF, BT40
20. (0.150, 0.750, 0.100, 0.5) → 0.200: Scr, OLED, CB, BT40
21. (0.120, 0.820, 0.060, 0.5) → 0.150: MF, HRS, BT40
22. (0.100, 0.850, 0.050, 0.5) → 0.125: MF, BT40

In the end, for the Mi Band 2, the top 3 products of this list were released in June 2016. All new products featured a `Screen`, and none of the new products used the outdated Bluetooth `BT4.0`. The *Chinese Edition* was released with `HRS` and `BLE4.0`, the *International Edition* also had `HRS` but with `BLE4.2`, and the *Indian Edition* did not have `HRS`, but `MF` instead, with `BLE4.0`. This variability reduction was achieved by adding the constraints $\texttt{BT4.0} \Leftrightarrow \neg\texttt{Screen}$, $\texttt{BLE} \Rightarrow \texttt{HRS} \vee \texttt{MF}$, and $\texttt{MF} \Rightarrow \neg\texttt{HRS} \wedge \texttt{BLE4.0}$. These constraints, resulting in a new $FM_{reduced}$ shown in the third feature model of Fig. 6, reduced the variability significantly, bringing the total number of products to be maintained to a more manageable number of 5.

4.4. Likert-type scales to represent opinions

Expressing an opinion as an `SBoolean` can be daunting, especially for non-experts when conducting market surveys. In those contexts, Likert scales have shown to be very effective. As part of this work, we therefore also propose two Likert-type scales, where textual, easy-to-understand literals are mapped to corresponding subjective logic values.

Initially, a stakeholder has to determine whether their opinion carries uncertainty or not. If not, then they should express their opinion using the following "Confident" five-point Likert scale, which defines: *Certain*, *In Favor*, *Neutral*, *Against* and *Impossible*.

Table 6 shows the 5-point scale textual literals and some possible `SBoolean` values (carrying no uncertainty) and projections.

On the other hand, if the stakeholder feels somehow unsure about their opinion, they should express their opinion using the following "Hesitant" seven-point Likert scale, which defines: *Certain*, *Probable*, *Possible*, *Uncertain*, *Improbable*, *Unlikely* and *Impossible*. Tables 7 and 8 show two possible ways of mapping the 7-point scale literals to corresponding `SBoolean` values with different degrees of uncertainty. While the projections of the values are very similar, the opinions in Table 7 carry more uncertainty than those in Table 8. The two cases are graphically illustrated using Barycentric coordinates in Fig. 4. It would be the job of an expert to decide on which mapping to use depending on the situation.

Note that we have created these literals with a base rate of 0.5 to represent a binary domain, i.e., a case in which the agents give their subjective opinion about a fact that can take two values that are equiprobable. Different situations would require the base rate to be adapted accordingly.

5. Proof of concept and validation

In this section, we present an implementation of our approach as a proof of concept and evaluate its applicability and scalability by using a real-world industrial example with 27 feature models representing the current nine versions of the Smartwatch SPL of Xiaomi Mi Band. The artifacts, including the feature models and information on how to replicate the results, are available online.[6] In this section, we seek to answer the following research questions (RQs):

[6] Artifact: https://github.com/atenearesearchgroup/fms-subjectivelogic.

Table 6
Possible Equivalence between Degrees of Certainty and Subjective Logic Values. Five-point "Confident" Scale for Opinions without Uncertainty with Base Rate of 0.5.

Literal	Subjective opinion value	Proj.	Explanation
Certain	`SBoolean(1.00, 0.00, 0.00, 0.5)`	1.00	The stakeholder considers the fact to be true.
In favor	`SBoolean(0.75, 0.25, 0.00, 0.5)`	0.75	The stakeholder is in favor of the fact.
Neutral	`SBoolean(0.50, 0.50, 0.00, 0.5)`	0.50	The stakeholder is neutral about the fact.
Against	`SBoolean(0.25, 0.75, 0.00, 0.5)`	0.25	The stakeholder is against the fact.
Impossible	`SBoolean(0.00, 1.00, 0.00, 0.5)`	0.00	The stakeholder considers the fact to be false.

Table 7
Possible Equivalence between Degrees of Certainty and Subjective Logic Values. Seven-point "Hesitant" Scale, Highest Degree of Uncertainty with Base Rate of 0.5.

Literal	Subjective opinion value	Proj.	Explanation
Certain	`SBoolean(1.00, 0.00, 0.00, 0.5)`	1.00	The expressed fact is considered to be true.
Probable	`SBoolean(0.67, 0.00, 0.33, 0.5)`	0.83	The expressed fact is probably true.
Possible	`SBoolean(0.33, 0.00, 0.67, 0.5)`	0.66	The expressed fact is possibly true.
Uncertain	`SBoolean(0.00, 0.00, 1.00, 0.5)`	0.50	There is uncertainty about the expressed fact.
Improbable	`SBoolean(0.00, 0.33, 0.67, 0.5)`	0.34	The expressed fact is possibly false.
Unlikely	`SBoolean(0.00, 0.67, 0.33, 0.5)`	0.17	The expressed fact is likely false.
Impossible	`SBoolean(0.00, 1.00, 0.00, 0.5)`	0.00	The expressed fact is considered to be false.

Table 8
Possible Equivalence between Degrees of Certainty and Subjective Logic Values. Seven-point "Hesitant" Scale, Lower Degree of Uncertainty with Base Rate of 0.5.

Literal	Subjective opinion value	Proj.	Explanation
Certain	`SBoolean(1.00, 0.00, 0.00, 0.5)`	1.00	The expressed fact is considered to be true.
Probable	`SBoolean(0.75, 0.05, 0.20, 0.5)`	0.85	The expressed fact is probably true.
Possible	`SBoolean(0.45, 0.15, 0.40, 0.5)`	0.65	The expressed fact is possibly true.
Uncertain	`SBoolean(0.20, 0.20, 0.60, 0.5)`	0.50	There is uncertainty about the expressed fact.
Improbable	`SBoolean(0.15, 0.45, 0.40, 0.5)`	0.35	The expressed fact is possibly false.
Unlikely	`SBoolean(0.05, 0.75, 0.20, 0.5)`	0.15	The expressed fact is likely false.
Impossible	`SBoolean(0.00, 1.00, 0.00, 0.5)`	0.00	The expressed fact is considered to be false.

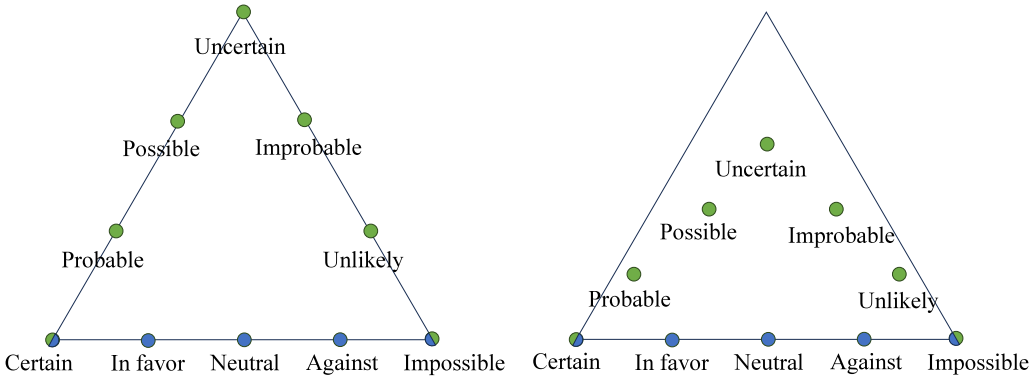

Fig. 4. Literals represented using Barycentric coordinates (Green: Opinions with higher degree on uncertainty (left), with lower degree of uncertainty (right). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

- **RQ1:** *How feasible and applicable is the proposed approach in practice?*
  Rationale: The objective is to analyze the viability and applicability of the approach in practice. We first provide an implementation of the three evolution scenarios using open source technologies. Then we analyze and synthesize an industrial SPL and its evolution between July 2014 and October 2023 consisting of 27 feature models corresponding to the nine versions of the Xiaomi Mi Band. Finally, we apply our implementation to the complete Xiaomi SPL.
- **RQ2:** *Does the approach scale to SPLs of realistic size?*
  Rationale: Scalability is a well-known challenge in the context of SPLs, as already the presence of a relatively small number of features can lead to a combinatorial explosion of possible products. It is therefore imperative for SPL approaches to demonstrate that they do not suffer severe performance degradation when applied to SPLs of realistic size.

5.1. Open-source implementation

Fig. 5 presents an overview of the implementation of our approach in Python. We rely on *Flama*,[7] a Python framework for the automated analysis of feature models (Galindo et al., 2023), and on a Python library of *Uncertainty Datatypes*[8] that provides support to work with

[7] Flama: https://flamapy.github.io/.
[8] Uncertainty Datatypes: https://github.com/atenearesearchgroup/uncertainty-datatypes-python.

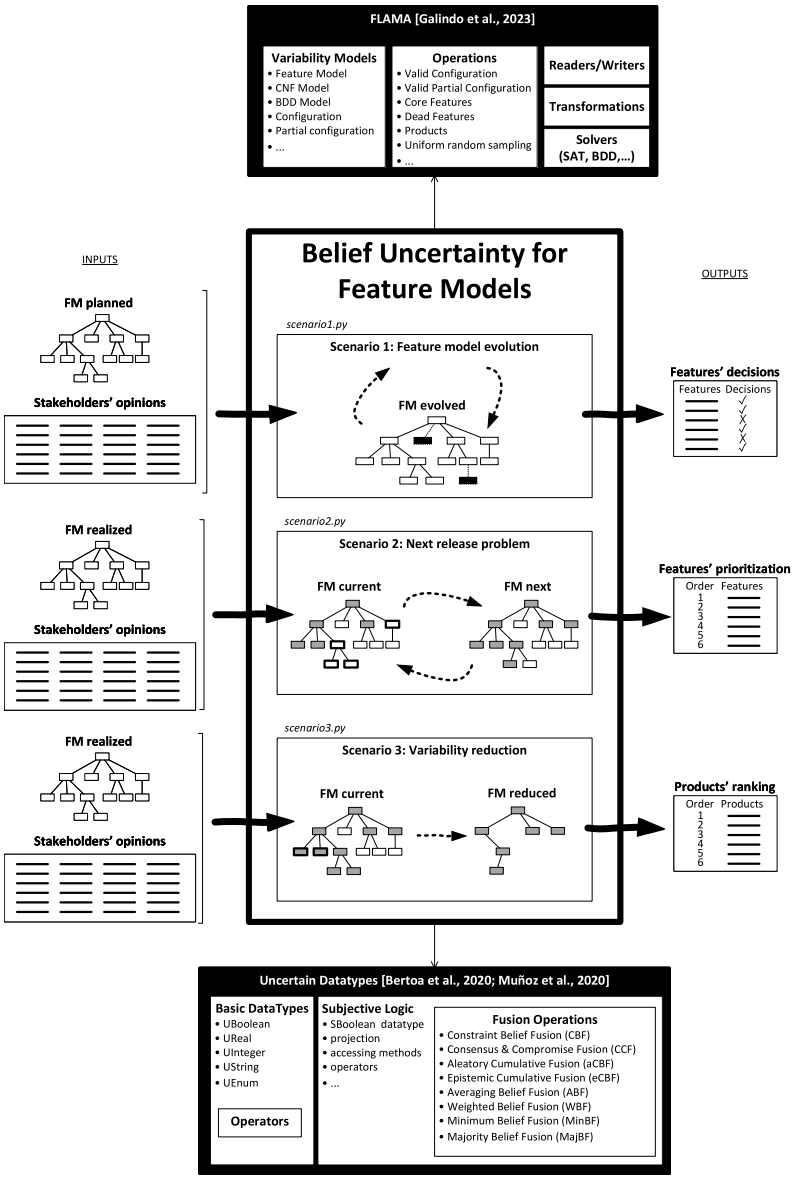

Fig. 5. Implementation architecture.

`SBoolean` values to express opinions and fusion operators (Bertoa et al., 2020; Fernández-Candel et al., 2024; Muñoz et al., 2020a).

Our implementation includes three scripts to support the three evolution scenarios presented throughout the paper:

- **Scenario 1: Feature Model Evolution**. The script takes the stakeholder's opinions about features to be included or removed in $FM_{planned}$ of the SPL, and returns the fused opinions of the stakeholders for each feature along with its projection and the decision to be made according to a given threshold.
- **Scenario 2: Next Release Problem**. It takes a feature model and the stakeholder's opinions about features to be implemented for the next release, and returns fused opinions for each feature taking into account the related/dependent features. The script groups the features according to their dependencies. First, it shows the feature with cross-tree constraints, then the group of related features in the tree, and then the independent features. Finally, it shows a ranking of the features prioritization to be implemented according to the opinions.
- **Scenario 3: Variability Reduction**. It takes a feature model, generates a sample of products of a given size, and determines those products that should be realized in the following release of the SPL according to the fused opinions of the stakeholders. The script outputs a list of products ordered by the projection of the fused opinions of the stakeholders (i.e., a ranking of products). For each product, only the variable features of the products are shown for easy inspection.

The feature models are encoded in the Universal Variability Language (UVL) (Sundermann et al., 2021a) format that is currently riding high in the SPL community thanks to the MODEVAR initiative (Benavides et al., 2019). The stakeholder's opinions are given in a *.csv* file, and for each feature the opinions can be specified as an `SBoolean` vector (e.g., `(0.67, 0.00, 0.33, 0.50)`), or as literal (e.g., `PROBABLE`) as defined in Section 4.4. In addition, the fusion operator to use when fusing the opinions can be specified, too.

5.2. The smartwatch SPL of Xiaomi Mi Band

We have analyzed and synthesized a real-world industry example of an SPL: the SPL of Xiaomi for its Mi Band smartwatch. We have chosen this SPL because despite none of the authors are affiliated

Table 9
Evolution of the Xiaomi SPL for the MiBand products for versions 1 to 8.

SPL Model	$FM_{realized}$									$FM_{reduced}$		
	DC (withdomain constraints)				AC (with all constraints)							
	#F	#DC	#P	T_{gen} (s)	#C	$\#P_{all}$	T_{gen} (s)	$\#P_{voted}$	T_{rank} (s)	#F	#C	$\#P_{rel}$
MiBand 1	12	0	1	0.003	0	1	0.003	1	N/A[a]	12	0	1
MiBand 1S	13	0	2	0.003	0	2	0.003	1	N/A[a]	13	0	2
MiBand 2	21	0	24	0.003	0	24	0.003	22	0.003	21	3	5
MiBand 3	24	1	108	0.004	3	24	0.003	19	0.002	24	7	7
MiBand 4	31	1	1,344	0.024	7	72	0.003	65	0.007	31	12	9
MiBand 5	34	1	8,064	0.330	12	64	0.004	55	0.006	34	15	11
MiBand 6	35	1	16,128	1.073	15	22	0.004	11	0.002	35	16	13
MiBand 7	36	1	20,160	2.178	16	24	0.003	11	0.002	36	17	15
MiBand 8	37	1	24,192	2.828	17	21	0.004	6	0.002	37	17	17

#F: Number of features. #DC: Number of domain constraints. #C: Number of total constraints (domains and additional constraints). #P: Number of products. T_{gen}: time to execute algorithm.

[a] N/A: Not applicable (because there is only one product to be decided).

with Xiaomi and considering the inability to contact or obtain the necessary information from any of its employees, the required SPL information regarding the different features available in each product version as well as the possible constraints between the features are publicly available in the company's official documentation and the associated websites.[9] Thus, the features of the Mi Band and the products that were released, are real and have been extracted from the publicly available specifications released by Xiaomi. However, the stakeholders involved in the decision making and their opinions used in the evolution scenarios are made up.

We have analyzed the evolution of the current nine versions of Mi Band (1, 1s, 2, 3, 4, 5, 6, 7, and 8), and synthesized the variability, resulting in a total of 27 feature models. Concretely, there are three feature models for each version of Mi Band: $FM_{planned}$, $FM_{realized}$, and $FM_{reduced}$. Figs. 6 and 7 illustrate the evolution of the feature models. Each feature model represents the $FM_{reduced}$ of version X, where $X \in \{1, 1s, 2, 3, \dots, 8\}$. $FM_{realized}$ of version X is the same feature model, but only with the constraints imposed by the domain (marked in bold), i.e., without those constraints imposed by the reduction of the variability. Finally, the constraints added explicitly for variability reduction are marked with a green +).

5.3. Experimentation setup

Scenario setup. To analyze the scalability of our approach, we focus on scenario 3, which is the most challenging one regarding scalability. This is because scenario 3 generates the list of new products introduced in $FM_{realized}$ compared to the previous version, and it requires stakeholders to give their opinions about those products in order to rank them and decide which ones are released in the market. For the evolution of the different versions in the Xiaomi SPL we therefore need to distinguish between $FM_{realized,DC}$ (the feature model that contains only domain constraints), and $FM_{realized,AC}$ (the feature model that contains also the constraints introduced by the variability reduction of the previous release). The products from the latter comprise the new products only, and hence are those to be ranked. We use 3 stakeholder opinions as votes to be considered for each product in all cases. The `SBoolean` values of those opinions were artificially created by the authors, since the specific values do not affect the scalability experimentation.

Parameter settings. We perform 30 runs and calculate the medians, means, and standard deviations for the execution time of generating all products of $FM_{realized,DC}$, and for the generation and ranking of the products (based on the stakeholder's opinions) of $FM_{realized,AC}$. Table 9 reports the medians. To generate all products we rely on the *products* operation available in *Flama*, and concretely, in the implementation based on the *Glucose3* SAT solver (Audemard and Simon, 2018) of the *PySAT* library (Ignatiev et al., 2018).

Execution setup. The experiments were performed on a desktop computer with Intel Core i7-4770 CPU @ 3.4 GHz, 16 GB RAM, Windows 11, and Python 3.10.

5.4. Results and discussion

Table 9 shows the evolution of the Xiaomi SPL for the nine existing versions of the Mi Band smartwatch. Since we do not have insight into the strategic planning of the company, we cannot know $FM_{planned}$, i.e., we are unaware of features that Xiaomi might have planned at some point, but never realized in any product. The table therefore only lists in the first column $FM_{realized,DC}$ (with only domain constraints), in the second column $FM_{realized,AC}$ (including also the constraints introduced by the variability reduction of the previous release) and in the last column $FM_{reduced}$. $FM_{planned}$ would be at least as big as $FM_{realized,DC}$, but would include other considered features.

First, we can observe how the SPL has evolved from 12 features in the first version (`Mi Band 1`) to 37 features in the most recent version (`Mi Band 8`). In general, only one to three new features are added from one version to the next (see #F), except for the evolution from `Mi Band 1S` to `Mi Band 2` where eight new features were added, and from `Mi Band 3` to `Mi Band 4` with seven new features. Note also that there is only one cross-tree constraint imposed by the domain in $FM_{realized,DC}$ (see #DC constraints), namely `ContactlessPayment` → `NFC`, originally added in `Mi Band 3` (Fig. 6).

The evolution of $FM_{realized,DC}$ represents a set of products ranging from 1 to 24,192 products. Despite the fact that the analysis of all these products by generating them from the feature model only takes 2.8 s for the largest $FM_{realized,DC}$ (column T_{gen}), the realization of all these products in the real world is not viable. To decide which new products are released on the market, stakeholders only need to give their opinions about the *new features* involved in each new version, and make decisions about the new possible realized products ($\#P_{voted}$ of $FM_{realized,AC}$, which is determined by $\#P_{all}$ - $\#P_{rel}$ of the previous release). The generation and ranking of these products do not pose a scalability issue (see T_{gen} and T_{rank}), as the number of realized products is very restricted by the additional constraints added. For instance, of the 24,192 possible products from $FM_{realized,DC}$ in `Mi Band 8`, the stakeholders only need to decide among 6 products from $FM_{realized,AC}$. Generating the possible 21 products and ranking the 6 new ones only takes four respectively two milliseconds.

The biggest sets of products that had to be voted on were during the variability reduction for `Mi Band 4`, where 65 new possible products had to be ranked, and for `Mi Band 5`, which introduced 55 new possible products. Providing votes for that many products can be tedious for

[9] Xiaomi website: https://www.mi.com/.

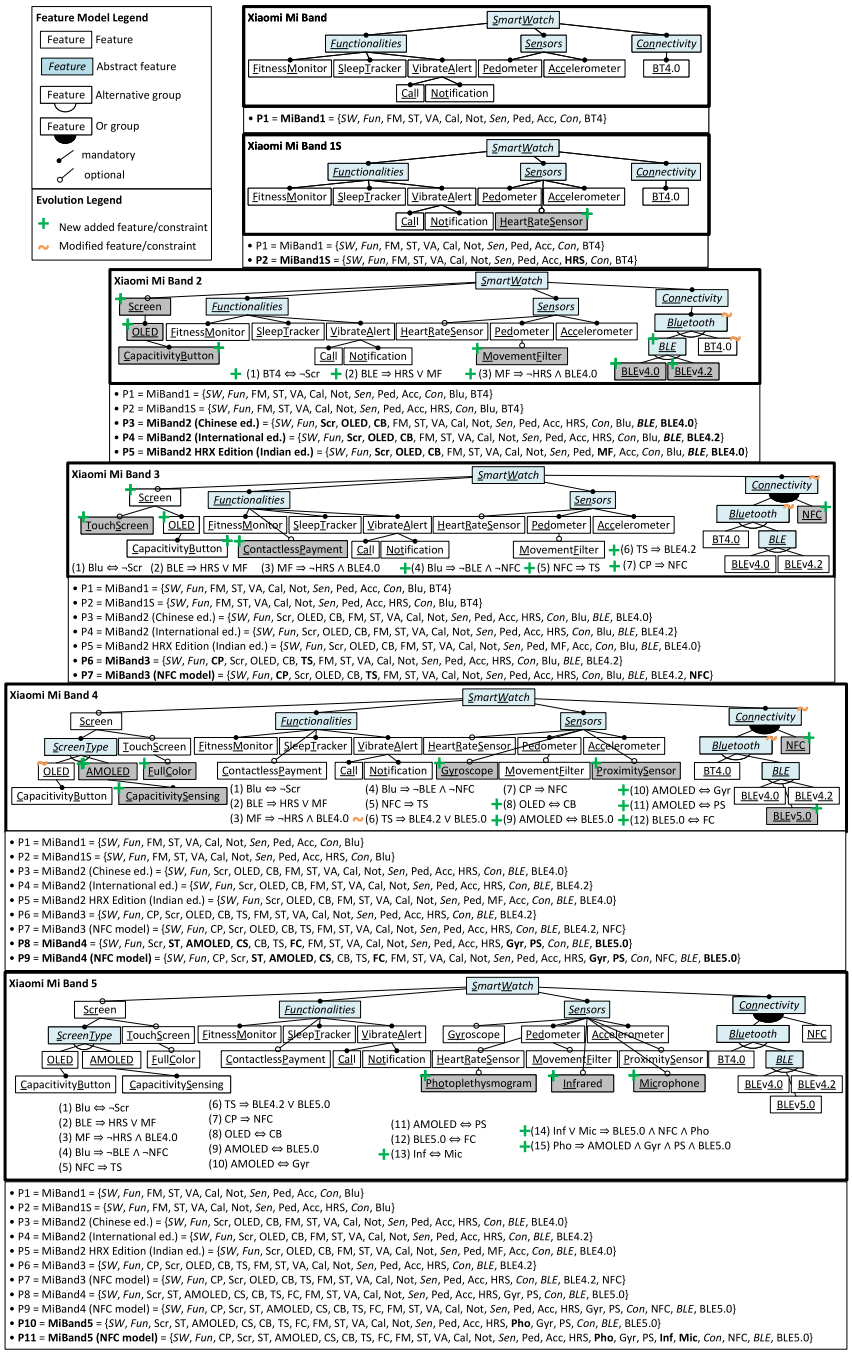

Fig. 6. Evolution of the smartwatch SPL of Xiaomi Mi Band (versions 1–5).

stakeholders, even when using literals for voting. In this case, one could use the already provided feature opinions (when voting on the next features to realize in scenario 2) to determine a reasonably sized subset of the possible products that will be used for voting during variability reduction (scenario 3). The reasoning behind this sampling heuristic is that the products containing the features that were the most popular ones to be realized are also most likely going to be the most popular products.

The final number of products released ranges from 1 to 17 different smartwatches (see #P of $FM_{reduced}$), and Xiaomi usually released only two new products for each new version (a Chinese version with the NFC feature and an international version without NFC), with the exception of Mi Band 2, where a specific version for India was released.

5.5. Answers to research questions

RQ1: How feasible and applicable is the proposed approach in practice?

We demonstrated the feasibility of our approach by implementing tool support for all the evolution scenarios using open source technologies. The applicability of our proposed approach is determined by how well it manages the complexity of realistic SPLs in terms of total number of features and number of evolving features between releases.

In our approach stakeholders only need to provide their opinions on features that evolve between releases (e.g., new features, deprecated features, changing features) and do not need to take into account the entire feature set of the SPL. Our algorithms that take into account opinions and feature dependencies likewise only depend on the number

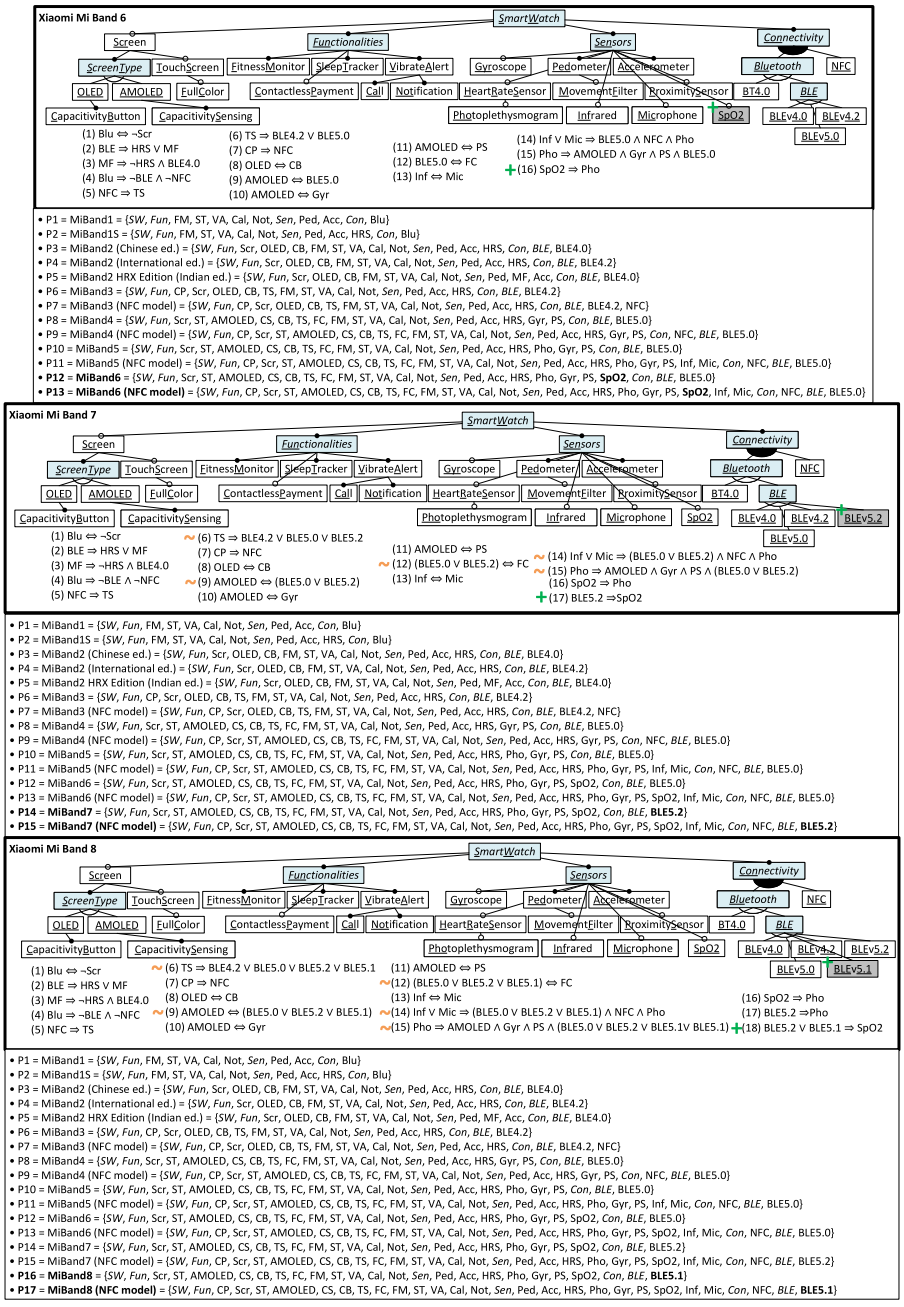

Fig. 7. Evolution of the smartwatch SPL of Xiaomi Mi Band (versions 6–8).

of evolving features. This targeted focus ensures that the approach remains manageable and effective in real-world scenarios as long as the number of evolving features is reasonable.

RQ2: Does the approach scale to SPLs of realistic size?

We conclude that our approach is scalable regarding the size of the SPL. Despite the fact that Scenario 3 requires to generate products which is one of the most costly analysis operation in SPLs, in our approach not all products are required to be generated. As explained in Section 5.1, "we generate a sample of products of a given size", and thus, we can always consider a reasonable sample size that makes sense to be manually analyzed and voted by the stakeholders.

5.6. Threats to validity

While using a real-world industrial example for validation offers rich insights, it is imperative to acknowledge potential threats to validity inherent in this approach.

Internal validity: A potential threat to validity is the researcher bias or subjective interpretation that may influence the results since the real opinions of the Xiaomi stakeholders were not used. In our case, this is not a problem, as the opinions used in the calculations in this paper are simply there for illustration purposes. The scalability of the calculations is not affected by the actual opinions. It could be possible, though, that the number of Xiaomi stakeholders consulted is significantly higher (we always used 3 stakeholder opinions in our illustrations). Again, this is not problematic, as our calculation algorithm complexity scales linearly with the number of opinions that need to be processed.

External validity: One such concern is the limited generalizability of findings, as case studies often focus on specific contexts and may not represent broader populations or circumstances. It is noteworthy that our validation employs a real-world industrial example centered on the Xiaomi SPL. This choice enhances the relevance and applicability of our findings within a tangible industrial setting, offering valuable insights

into practical challenges and solutions within the realm of software development. In terms of applicability, as shown in our validation, our approach does not need to consider all features of an SPL, but only those that change during evolution. Even in real-world SPLs such as the biggest known SPL reported (i.e., the Linux kernel with more than 18,000 features (She et al., 2010; Thüm, 2020)) the number of features that change and need to be consider from one version to another is not so high, especially when those features need to be manually analyzed by humans. Thus, we believe that the Xiaomi SPL example nicely showcases the applicability of our approach. A final threat is the ecological validity due to possible errors in the calculations of the experiment materials and tools used. To mitigate this threat we have relied on well-known libraries for subjective logic (Bertoa et al., 2020; Muñoz et al., 2020a) and tools for automated analysis of feature models such as Flama (Galindo et al., 2023).

6. Related work

This section discusses the related work on uncertainty and decision-making in software engineering specifically in the context of software product line engineering.

6.1. Stakeholder analysis and decision-making

There exist several decision-making techniques and tools to make the right decisions when stakeholders with different expertise are involved (Achimugu et al., 2014). Some of the more commonly used techniques are the *analytic hierarchical process* (AHP) (Zhang et al., 2014), the *WinWin* approach (Boehm and Kitapci, 2006), or the *Kano* model (Kano, 1984), among many others (Achimugu et al., 2014; Hudaib et al., 2018). These techniques are mainly employed as prioritization methods for requirements engineering. Before applying these techniques, stakeholders need to analyze the most important business factors, e.g., value, cost, risk, implementation effort, success, or urgency for each requirement (or feature in the context of SPLs). However, getting the stakeholders' input for those factors is not straightforward due to the existence of uncertainty, which is not considered in the classical techniques for decision-making and stakeholder analysis (Achimugu et al., 2014). Subjective logic is an alternative method that allows ranking requirements or features' priority taking into account the uncertainty in the analysis of the stakeholder for the different business factors.

Subjective logic in software engineering. Subjective logic has been applied in several domains, e.g., artificial intelligence (Hüllermeier and Waegeman, 2021), software engineering (Troya et al., 2021), or digital humanities (Martin-Rodilla and Gonzalez-Perez, 2019). In the software engineering domain Troya et al. (2021), subjective logic has been mainly applied in the requirements engineering context for different purposes (e.g., to check whether a product meets its specifications in natural language (Sree-Kumar et al., 2018), or to specify architectures requirements with incomplete information (Bertoa et al., 2020), among others). But subjective logic has also been applied to model-based software engineering (MBSE) to deal with belief uncertainty in domain models (Burgueño et al., 2022). Concretely, Bertoa et al. (2020) applied subjective logic to software models (e.g., primitive dataypes 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. To the best of our knowledge and according to the survey presented in Troya et al. (2021), subjective logic has not been applied before to SPLs. However, several works deal with uncertainty and probabilities in SPLs.

6.2. Uncertainty and decision-making in SPLs

Uncertainty and decision-making are present at different stages of the SPL process, including requirements elicitation (Famelis et al., 2017; Sree-Kumar et al., 2018; Sree-Kumar et al., 2021), product configuration (Czarnecki et al., 2008; Martinez et al., 2014; Mazo et al., 2014; Nöhrer and Egyed, 2013; Pereira et al., 2018; Rodas-Silva et al., 2019), and automated analysis of feature models (Heradio et al., 2019; Horcas et al., 2023). In fact, recently, Horcas et al. (2023) 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 simulation-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. The next release problem and evolution of feature models are only mentioned.

SPL requirements and modeling. Sree-Kumar et al. (2018), Sree-Kumar et al. (2021) 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. (2017) made a research vision about combining variability abstractions with *partial modeling* (Famelis et al., 2012), a technique for managing design uncertainty within a software model (Dhaouadi et al., 2021) that explicates decision points and represents the set of possible models that could be obtained by making decisions and solving uncertainty. In Almharat (2016), Almharat deals with the lack of probabilistic quantification of model's implications and decision support for reasoning under uncertainty. The work proposes (i) the construction of a Bayesian Belief Feature Model, capable of quantifying the uncertainty measures in model parameters, and (2) a mathematical reasoner for Uncertain Constraint Satisfaction Problems to satisfy the model constraints by considering probability weight for all involved parameters and quantify the actual implications of the problem constraints.

SPL configuration. Product configuration is the SPL process where decision-making techniques have been applied the most so far. The concept of *probabilistic feature models* (Czarnecki et al., 2008) 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)* (Martinez et al., 2014) 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 (Czarnecki et al., 2008; Martinez et al., 2014) 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. (2019) 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. (2014) 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 and Egyed (2013) 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. (2018) 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 sufficient personal experience to evaluate the complex technical properties of the features. The main drawback of these works (Mazo et al., 2014; Nöhrer and Egyed, 2013; Pereira et al., 2018; Rodas-Silva et al., 2019) is that they do not take into account the expertise of the domain experts, only the final users' rating (Rodas-Silva et al., 2019), users' needs and preferences (Pereira et al., 2018), or historical data from previous users' configurations (Mazo et al., 2014; Nöhrer and Egyed, 2013) are considered.

SPL evolution. Finally, in Sang Tran and Massacci (2014) 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. In Section 3, we have identified three evolution scenarios where subjective logic may improve the decision-making process by explicitly considering and reasoning about the existing uncertainty using SBoolean opinions.

7. Conclusions and future work

In this paper, we applied subjective logic to capture and address uncertainty when making decisions for SPL evolution. In particular, we described in detail three feature model evolution scenarios, and explained how stakeholder opinions can be quantified for decision-making in these scenarios using a real-world industrial example. Our approach considers feature dependencies and supports different decision strategies (e.g., risky vs. conservative) through the use of different fusion operators and different decision thresholds when making the final decision. Finally, we proposed two Likert-type scales – one carrying uncertainty and one without – with easy-to-understand textual literals to simplify voting using subjective logic for non-expert stakeholder.

We open a new window of research where belief uncertainty could become a new tool of great value in the SPL practitioner's toolkit whenever the opinions of different stakeholders have to be taken into account to make informed decisions.

As part of our future work, we plan to apply our approach to other real SPLs to study more thoroughly the impact of the size of the SPL on the performance and scalability of our approach. We would also like to perform empirical studies with users as stakeholders providing opinions in a real setting. This would allow us to study the usefulness and usability of our approach. Last, but not least, we would also like to integrate our approach into other tools such as FeatureIDE (Thüm et al., 2014) to make it available to a wider range of users.

Software artifact and open science

Artifact: https://github.com/atenearesearchgroup/fms-subjectivelogic

CRediT authorship contribution statement

Jose-Miguel Horcas: Writing – review & editing, Writing – original draft, Validation, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Data curation, Conceptualization. **Lola Burgueño:** Writing – review & editing, Writing – original draft, Validation, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Data curation, Conceptualization. **Jörg Kienzle:** Writing – review & editing, Writing – original draft, Validation, Software, Resources, Methodology, Investigation, Funding acquisition, Data curation, Conceptualization.

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.

Acknowledgments

This work was partially funded by the Spanish Government (Ministerio de Ciencia e Innovación–Agencia Estatal de Investigación), Spain under projects PID2021-125527NB-I00, TED2021-130523B-I00, *IRIS* (PID2021-122812OB-I00) (co-financed by FEDER funds), *Data-pl* (PID2022-138486OB-I00), and *TASOVA PLUS* research network (RED2022-134337-T); and by Junta de Andalucía, Spain under project QUAL21 010UMA. Funding, including open access charge: Universidad de Málaga/CBUA, Spain.

Data availability

Software Artifact: https://github.com/atenearesearchgroup/fms-subjectivelogic.

References

Achimugu, Philip, Selamat, Ali, Ibrahim, Roliana, Mahrin, Mohd Naz'ri, 2014. A systematic literature review of software requirements prioritization research. Inf. Softw. Technol. 56 (6), 568–585. http://dx.doi.org/10.1016/j.infsof.2014.02.001.

Ali, Nazakat, Hwang, Sangwon, Hong, Jang-Eui, 2019. Your opinions let us know: Mining social network sites to evolve software product lines. KSII Trans. Internet Inf. Syst. 13 (8), 4191–4211. http://dx.doi.org/10.3837/tiis.2019.08.021.

Almharat, Anas, 2016. Probabilistic Graphical Modelling for Software Product Lines: A Framework for Modeling and Reasoning under Uncertainty (Ph.D. thesis). University of East London.

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

Audemard, Gilles, Simon, Laurent, 2018. On the glucose SAT solver. Int. J. Artif. Intell. Tools 27 (1), http://dx.doi.org/10.1142/S0218213018400018.

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

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

Benavides, David, Rabiser, Rick, Batory, Don S., Acher, Mathieu, 2019. First international workshop on languages for modelling variability (MODEVAR). SPLC, In: 23rd International Systems and Software Product Line Conference, vol. A, ACM, http://dx.doi.org/10.1145/3336294.3342364, 46:1.

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

Boehm, Barry, Kitapci, Hasan, 2006. The WinWin approach: Using a requirements negotiation tool for rationale capture and use. In: Rationale Management in Software Engineering. Springer, pp. 173–190. http://dx.doi.org/10.1007/978-3-540-30998-7_8.

Burgueño, Lola, Horcas, José Miguel, Kienzle, Jörg, 2023. Development and evolution of software product lines driven by stakeholder beliefs. In: Proc. of the 27th ACM International Systems and Software Product Line Conference. ACM, pp. 34–40. http://dx.doi.org/10.1145/3579027.3608975.

Burgueño, Lola, Muñoz, Paula, Clarisó, Robert, Cabot, Jordi, Gérard, Sébastien, Vallecillo, Antonio, 2022. Dealing with belief uncertainty in domain models. ACM Trans. Softw. Eng. Methodol. http://dx.doi.org/10.1145/3542947.

Czarnecki, Krzysztof, She, Steven, Wasowski, Andrzej, 2008. Sample spaces and feature models: There and back again. In: 12th International Conference on Software Product Lines. SPLC, IEEE Computer Society, pp. 22–31. http://dx.doi.org/10.1109/SPLC.2008.49.

De Finetti, Bruno, 2017. Theory of Probability: A Critical Introductory Treatment, vol. 6, John Wiley & Sons, http://dx.doi.org/10.1002/9781119286387.

Dhaouadi, Mouna, Spencer, Kate M.B., Varnum, Megan H., Grubb, Alicia M., Famelis, Michalis, 2021. Towards a generic method for articulating design-time uncertainty. J. Object Technol. 20 (3), http://dx.doi.org/10.5381/jot.2021.20.3.a3, 3:1–14.

Famelis, Michalis, Rubin, Julia, Czarnecki, Krzysztof, Salay, Rick, Chechik, Marsha, 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, pp. 93–100. http://dx.doi.org/10.1109/MODELS.2017.3.

Famelis, Michalis, Salay, Rick, Chechik, Marsha, 2012. Partial models: Towards modeling and reasoning with uncertainty. In: 34th International Conference on Software Engineering. ICSE, pp. 573–583. http://dx.doi.org/10.1109/ICSE.2012.6227159.

Fernández-Candel, Carlos J., Munoz, Paula, Troya, Javier, Vallecillo, Antonio, 2024. UTypes: A library for uncertain datatypes in Python. SoftwareX.

Galindo, José A., Horcas, José Miguel, Felfernig, Alexander, Fernández-Amorós, David, Benavides, David, 2023. FLAMA: A collaborative effort to build a new framework for the automated analysis of feature models. SPLC, In: 27th ACM International Systems and Software Product Line Conference, vol. B, ACM, pp. 16–19. http://dx.doi.org/10.1145/3579028.3609008.

van der Heijden, Rens Wouter, Kopp, Henning, Kargl, Frank, 2018. Multi-source fusion operations in subjective logic. In: Proc. of FUSION'18. IEEE, pp. 1990–1997. http://dx.doi.org/10.23919/ICIF.2018.8455615.

Hentze, Marc, Sundermann, Chico, Thüm, Thomas, Schaefer, Ina, 2022. Quantifying the variability mismatch between problem and solution space. In: 25th International Conference on Model Driven Engineering Languages and Systems. MODELS, ACM, pp. 322–333. http://dx.doi.org/10.1145/3550355.3552411.

Heradio, Ruben, Fernández-Amorós, David, Mayr-Dorn, Christoph, Egyed, Alexander, 2019. Supporting the statistical analysis of variability models. In: 41st International Conference on Software Engineering. ICSE, IEEE / ACM, pp. 843–853. http://dx.doi.org/10.1109/ICSE.2019.00091.

Horcas, José Miguel, Galindo, José A., Heradio, Ruben, Fernández-Amorós, David, Benavides, David, 2023. A Monte Carlo tree search conceptual framework for feature model analyses. J. Syst. Softw. 195, 111551. http://dx.doi.org/10.1016/j.jss.2022.111551.

Hudaib, Amjad, Masadeh, Raja, Qasem, Mais Haj, Alzaqebah, Abdullah, et al., 2018. Requirements prioritization techniques comparison. Mod. Appl. Sci. 12 (2), 62.

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

Ignatiev, Alexey, Morgado, António, Marques-Silva, João, 2018. PySAT: A Python toolkit for prototyping with SAT oracles. SAT, In: 21st International Conference on Theory and Applications of Satisfiability Testing, vol. 10929, Springer, pp. 428–437. http://dx.doi.org/10.1007/978-3-319-94144-8_26.

Joint Committee for Guides in Metrology, 2008. JCGM 100: Evaluation of Measurement Data - Guide to the Expression of Uncertainty in Measurement. Technical Report, JCGM.

Jøsang, Audun, 2016. Subjective logic - A formalism for reasoning under uncertainty. In: Artificial Intelligence: Foundations, Theory, and Algorithms. Springer, http://dx.doi.org/10.1007/978-3-319-42337-1.

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

Kano, Noriaki, 1984. Attractive quality and must-be quality. J. Jpn Soc. Qual. Control 31 (4), 147–156.

Kleene, S.C., 1938. On notation for ordinal numbers. J. Symbolic Logic 3 (4), 150–155. http://dx.doi.org/10.2307/2267778.

Liu, Baoding, 2010. Uncertainty theory. In: Uncertainty Theory: A Branch of Mathematics for Modeling Human Uncertainty. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 1–79. http://dx.doi.org/10.1007/978-3-642-13959-8_1.

Mamun, Abdullah Al, Djatmiko, Fahim, Das, Mridul Kanti, 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, pp. 581–585. http://dx.doi.org/10.1109/ICCITECHN.2016.7860263.

Marques, Maíra, Simmonds, Jocelyn, Rossel, Pedro O., Bastarrica, María Cecilia, 2019. Software product line evolution: A systematic literature review. Inf. Softw. Technol. 105, 190–208. http://dx.doi.org/10.1016/j.infsof.2018.08.014.

Martin-Rodilla, Patricia, Gonzalez-Perez, Cesar, 2019. Conceptualization and non-relational implementation of ontological and epistemic vagueness of information in digital humanities. Informatics 6 (2), http://dx.doi.org/10.3390/informatics6020020.

Martinez, Jabier, Ziadi, Tewfik, Mazo, Raúl, Bissyandé, Tegawendé F., Klein, Jacques, Traon, Yves Le, 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, pp. 50–59. http://dx.doi.org/10.1109/VISSOFT.2014.18.

Mazo, Raúl, Dumitrescu, Cosmin, Salinesi, Camille, Diaz, Daniel, 2014. Recommendation heuristics for improving product line configuration processes. In: Recommendation Systems in Software Engineering. Springer, pp. 511–537. http://dx.doi.org/10.1007/978-3-642-45135-5_19.

Muñoz, Paula, Burgueño, Loli, Ortiz, Victor, Vallecillo, Antonio, 2020a. Extending OCL with subjective logic. J. Object Technol. 19 (3), http://dx.doi.org/10.5381/JOT.2020.19.3.A1, 3:1–15.

Muñoz, Paula, Burgueño, Lola, Ortiz, Victor, Vallecillo, Antonio, 2020b. Extending OCL with subjective logic. J. Object Technol. 19 (3), http://dx.doi.org/10.5381/jot.2020.19.3.a1, 3:1–15. Special Issue dedicated to Martin Gogolla on his 65th Birthday.

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

Pereira, Juliana Alves, Matuszyk, Pawel, Krieter, Sebastian, Spiliopoulou, Myra, Saake, Gunter, 2018. Personalized recommender systems for product-line configuration processes. Comput. Lang. Syst. Struct. 54, 451–471. http://dx.doi.org/10.1016/j.cl.2018.01.003.

Raatikainen, Mikko, Tiihonen, Juha, Männistö, Tomi, 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.

Rodas-Silva, Jorge, Galindo, José Angel, García-Gutiérrez, Jorge, Benavides, David, 2019. Selection of software product line implementation components using recommender systems: An application to wordpress. IEEE Access 7, 69226–69245. http://dx.doi.org/10.1109/ACCESS.2019.2918469.

Sang Tran, Le Minh, Massacci, Fabio, 2014. An approach for decision support on the uncertainty in feature model evolution. In: IEEE 22nd International Requirements Engineering Conference (RE). pp. 93–102. http://dx.doi.org/10.1109/RE.2014.6912251.

Shafer, Glenn, 1976. A Mathematical Theory of Evidence, vol. 42, Princeton University Press.

She, Steven, Lotufo, Rafael, Berger, Thorsten, Wasowski, Andrzej, Czarnecki, Krzysztof, 2010. The variability model of the linux kernel. In: 4th International Workshop on Variability Modelling of Software-Intensive Systems. VaMoS, In: ICB-Research Report, vol. 37, Universität Duisburg-Essen, Linz, Austria, pp. 45–51, http://www.vamos-workshop.net/proceedings/VaMoS_2010_Proceedings.pdf.

Sree-Kumar, Anjali, Planas, Elena, Clarisó, Robert, 2018. Extracting software product line feature models from natural language specifications. SPLC, In: 22nd International Systems and Software Product Line Conference, vol. 1, ACM, pp. 43–53. http://dx.doi.org/10.1145/3233027.3233029.

Sree-Kumar, Anjali, Planas, Elena, Clarisó, Robert, 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, http://dx.doi.org/10.1145/3442391.3442407, 15:1–15:10.

Sundermann, Chico, Feichtinger, Kevin, Engelhardt, Dominik, Rabiser, Rick, Thüm, Thomas, 2021a. Yet another textual variability language?: A community effort towards a unified language. SPLC, In: 25th ACM International Systems and Software Product Line Conference, vol. A, ACM, pp. 136–147. http://dx.doi.org/10.1145/3461001.3471145.

Sundermann, Chico, Nieke, Michael, Bittner, Paul Maximilian, Heß, Tobias, Thüm, Thomas, Schaefer, Ina, 2021b. Applications of #SAT solvers on feature models. In: 15th International Working Conference on Variability Modelling of Software-Intensive Systems. VaMoS, ACM, http://dx.doi.org/10.1145/3442391.3442404, 12:1–12:10.

Thüm, Thomas, 2020. A BDD for Linux?: The knowledge compilation challenge for variability. In: Lopez-Herrejon, Roberto Erick (Ed.), SPLC, In: 24th ACM International Systems and Software Product Line Conference, vol. A, ACM, http://dx.doi.org/10.1145/3382025.3414943, 16:1–16:6.

Thüm, Thomas, Batory, Don S., Kästner, Christian, 2009. Reasoning about edits to feature models. In: 31st International Conference on Software Engineering. ICSE, IEEE, pp. 254–264. http://dx.doi.org/10.1109/ICSE.2009.5070526.

Thüm, Thomas, Kästner, Christian, Benduhn, Fabian, Meinicke, Jens, Saake, Gunter, Leich, Thomas, 2014. FeatureIDE: An extensible framework for feature-oriented software development. Sci. Comput. Program. 79, 70–85. http://dx.doi.org/10.1016/j.scico.2012.06.002.

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

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

Van Der Heijden, Rens W., Kopp, Henning, Kargl, Frank, 2018. Multi-source fusion operations in subjective logic. In: 21st International Conference on Information Fusion (FUSION). pp. 1990–1997. http://dx.doi.org/10.23919/ICIF.2018.8455615.

Zhang, Guoheng, Ye, Huilin, Lin, Yuqing, 2014. Quality attribute modeling and quality aware product configuration in software product lines. Softw. Qual. J. 22 (3), 365–401. http://dx.doi.org/10.1007/s11219-013-9197-z.

Zimmermann, Hans-Jürgen, 2001. Fuzzy Set Theory—And Its Applications. Springer Science & Business Media.

José-Miguel Horcas is an Associate Professor at ITIS Software, Universidad de Málaga, Spain. He earned his M.Sc. degree in 2012 and his Ph.D. in Computer Science in 2018, both from the same institution. As a dedicated member of the CAOSD research group, José-Miguel's research focuses on software product lines, with a particular emphasis on variability, configurability, quality attributes, and modularity. In 2021–2022, he worked with the DiversoLab research group at the University of Seville, furthering his expertise in software product lines. He has over 60 scientific publications and has received five

best paper awards at international conferences. For more information, visit his website: https://sites.google.com/view/josemiguelhorcas.

Lola Burgueño is an Associate Professor at ITIS Software, Universidad de Málaga, Spain. Her research interests lie in the fields of software engineering and model-based software engineering. She has made contributions to the application of artificial intelligence techniques to improve software development processes and tools, uncertainty management during the software design phase, and model-based software testing, among others. For more information, please visit http://lolaburgueno.github.io.

Jörg Kienzle is a researcher at ITIS Software, Universidad de Málaga, Málaga, Spain, and Full Professor at McGill University, Montréal, Québec, Canada, where he leads the Software Composition and Reuse lab (SCORE). His research interests include model-driven software development, software product lines, separation of concerns, reuse, software composition, and modularity. Further information about him can be found at https://djeminy.github.io Contact him at Joerg.Kienzle@uma.es or Joerg.Kienzle@mcgill.ca.