Towards Estimating and Predicting User Perception on Software Product Variants

Jabier Martinez, Jean-Sébastien Sottet, Alfonso García Frey, Tegawendé Bissyandé, Tewfik Ziadi, Jacques Klein, Paul Temple, Mathieu Acher, Yves Le Traon

Towards Estimating and Predicting User Perception on Software Product Variants

Jabier Martinez[1], Jean-Sébastien Sottet[2], Alfonso García Frey[4], Tegawendé F. Bissyandé[3], Tewfik Ziadi[1], Jacques Klein[3], Paul Temple[5], Mathieu Acher[5], and Yves le Traon[3]

[1] Sorbonne University, UPMC, Paris, France: `name.surname@lip6.fr`
[2] Luxembourg Institute of Science and Technology: `name.surname@list.lu`
[3] University of Luxembourg, Luxembourg: `name.surname@uni.lu`
[4] Yotako, Luxembourg: `alfonso@yotako.io`
[5] Univ Rennes, Inria, CNRS, IRISA, France: `name.surname@irisa.fr`

Abstract. Estimating and predicting user subjective perceptions on software products is a challenging, yet increasingly important, endeavour. As an extreme case study, we consider the problem of exploring computer-generated art object combinations that will please the maximum number of people. Since it is not feasible to gather feedbacks for all art products because of a combinatorial explosion of possible configurations as well as resource and time limitations, the challenging objective is to rank and identify optimal art product variants that can be generated based on their average likability. We present the use of Software Product Line (SPL) techniques for gathering and leveraging user feedbacks within the boundaries of a variability model. Our approach is developed in two phases: 1) the creation of a data set using a genetic algorithm and real feedback and 2) the application of a data mining technique on this data set to create a ranking enriched with confidence metrics. We perform a case study of a real-world computer-generated art system. The results of our approach on the arts domain reveal interesting directions for the analysis of user-specific qualities of SPLs.

Keywords: Software Product Lines, quality attributes, quality estimation, computer-generated art, product variants

1 Introduction

With the whole myriad of software alternatives that exist today, the adoption of a software product is eventually dependent on users' subjective perception, beyond the offered functionalities. Being able to apprehend, estimate and predict this perception on the software as a whole will be an important step towards efficient software production. Unfortunately, subjective perception of a software product is hard to formalize. It cannot even be computed with a simple formula based on the perception of its components: melting good ingredients indeed does not necessarily produce a good recipe. Systematically predicting the subjective appreciation of software is therefore a research direction that is still at its early stages. An extreme case where appreciation is important, is when the intention of the product is just about "beauty": this is the case of computer-generated art. By studying computer-generated art we can infer insightful techniques for estimating the perception of other types of software involving user-specific quality aspects.

The practice of computer-generated art, also known as generative art, involves the use of an autonomous system that contributes to the creation of an art object, either in its whole, or in part by reusing pieces of art from a human artist, or using predefined algorithms or transformations [4]. This art genre is trending in the portfolio of many artists and designers in the fields of music, painting, sculpture, architecture or literature [4]. Apart from art installations, we consume computer-generated art systems in our daily life as in videogames, cinema effects, screen-savers or visual designs. A computer-generated art setup is driven at its core by a software system that implements the algorithms for "creating" the artworks. The development of such software presents the same challenges as any software development project. In general, the autonomous systems for computer-generated art rely to some degree to a randomization step in the generation algorithms. However, when no relevant stochastic component is introduced and the creation is not limited to a unique art object, computer-generated art allows one to derive different art objects in a predefined and deterministic fashion, giving rise to a *family of art products*. Usually, because of the combinatorial explosion of all possible art objects, not all of them will actually be created. Besides, a number of them may not reach the aesthetic quality desired by the users. Exploring the art product family to find the "best" products is thus challenging for computer-generated art practitioners.

We note that the challenges for exploring generative art products boil down to traditional Software Product Line Engineering (SPLE). It thus seems opportune to leverage SPL techniques for the design and implementation of generative art systems. Based on our experiments with a generative art case study, we devise an SPLE approach capable of dealing with subjective opinions and end-user concerns. SPLs are built following two main activities: variability modeling and product derivation. A given variability model defines a set of valid combinations of features, referred to as configurations. The product derivation step then assembles different assets related to the features of a specific configuration to yield a product instance. As an SPL can lead to high number of product variants, the challenge thus becomes to produce only those that are aligned with certain quality attributes. In our case study, this quality attribute is the appreciation of beauty which is not known a priori and difficult to formally define.

We propose a two-phase approach for leveraging user feedbacks and *ranking* all the possible art products based on the calculated estimations. Instead of continuously picking random products for requesting feedback, we rely on interactive genetic algorithms [22] in the first phase to explore the configuration space. The implemented fitness function, which is the genetic algorithm operator that drives the evolution, is not automatically calculated but provided interactively by users. This phase yields a subset that already tries to converge towards optimal or suboptimal products. In the second phase, we go beyond any other evolutionary art approach or analogue SPL technique, and we propose a method to infer the estimation on user perception for all art product variants applying data mining interpolation techniques based on a defined distance function between any pair of configurations.

As validation, we use real-world data from an artistic installation created in collaboration with a professional artist. We built a generative art system for landscape paintings, where each painting is generated by assembling visual components. To formalize the variability in the artist style, we rely on a feature model [14]. The derivation process is then performed using SPL techniques treating painting compositional elements as software reusable components.

The main contribution is to ***empirically study whether we can predict the like/dislike user perception of a software, built by an SPL assembling perceivable components***. The related contributions are:

– We discuss how user feedback can be leveraged in the computer-generated art system derivation processes.
– We propose an approach for ranking all art products using user feedback on a relevant subset of products. This ranking is ordered by the score estimation. For each rank item we propose a confidence metric for the prediction.
– We propose a validation on the feasibility of the approach after applying it to a real-world installation for computer-generated art.
– We introduce SPLE formalisms as a computer-generated art technique.
– The approach is explained in detail so it can be employed in other software families where it is relevant to predict user-specific qualities.

This paper is an extension of a 2-page idea article [17] focused on genetic algorithms while this paper is more focused on the potential for assessing variants. The paper is structured as follows: Section 2 introduces related work and Section 3 presents our case study. Section 4 presents the motivation and Section 5 provides the details of our approach. Section 6 presents the results of the case study and its evaluation is presented in Section 7. Section 8 presents the threats to validity and Section 9 concludes and outlines future work.

2 Related work

Leveraging user feedback to improve the results of the creation of software products is a challenging endeavour. This challenge has been already tackled both in the computer-generated art and SPLE community. Evolutionary computing applied to computer-generated art is the main technique used to leverage user feedback [19]. For evolution, the main aspect is the fitness function that represents the requirements to adapt to (it defines what improvement means). In our case, the evaluation function is based on user feedback in opposition to automatically calculated fitness functions. Some works already proposed to learn to predict user aesthetic preferences [12,15] but we focus on real user feedback and an SPL context in our approach. The relevant difference of our work compared with the evolutionary art approaches is that they only contain the evolution phase. Their assumption is that the better adapted products found at the end of the evolution should be the best ones. However, in a normal situation, not all the possible configurations were assessed given the time and resource limitations. In this paper we present a second phase to predict the user feedback of the non-assessed products. This phase was also missing in previous works in SPL-based User Interface (UI) design [18] or in other works dealing with SPL-based video configurations [7]. Regarding SPLE domain, only few works consider user feedback as driver for SPLE processes. In single-system development, user feedback has been already automatically leveraged for dealing with quality attributes related to software design (e.g., [2, 8]). However, for SPLs, its importance was overlooked.

Many existing work can be found on selecting optimal SPL product variants based on some criteria. To achieve this, it is a common practice to enhance the FM by what is referred to as quality attribute annotations, which mainly specify non-functional properties [3, 20, 21, 23]. Quality attribute annotations associated to the FM are then used to reason about the optimal selection of features. For instance, Benavides *et al.* [3] consider the selection problem as a Constraint Satisfaction Problem and solved it using CSP solvers, or Sayyad *et al.* [20] use Search-based algorithms. Also in the SPL testing community it is a well-known issue to select the best configurations to be tested guided by a set of predefined testing objectives [5, 6, 11, 13] which can be considered as quality attributes. In comparison with approaches dealing with quality attributes, our work consider user feedback that presents two main peculiarities: The first one is that user appreciations are always based on a product as a whole while in these approaches quality attribute annotations are directly mapped to concrete features. The second peculiarity is that user feedback subjectivity requires feedback aggregation mechanisms while quality attribute annotations in these approaches are fixed independently of user feedback. For example, *cost* quality attribute annotations in [11, 20] are defined as a fixed value per feature.

3 Introducing SPL-based generative art

We collaborated with Gabriele Rossi, a professional art painter with whom we were able to conduct a large user study on computer-generated art built by composing portions of paintings. We developed this system using SPLE techniques.

Variability within a landscape painting: In the last years, Gabrielle has been drawing abstract representations of landscapes with quite a recognizable style of decomposing the canvas in different parts: a *sky* part, a *middle* part and the *ground.* This variability is further enhanced by the fact that the sky and ground are mandatory while the middle part is optional. For the realization of the computer-generated art system, he created different representative paintings for each part. For the sky part, he painted 10 concrete representative sky paintings hereafter noted S_i, with $i \in [1..10]$. Similarly, he painted 9 paintings of the middle part (noted M_i, where M_{10} means the absence of middle part), and 10 paintings of the ground part (noted G_i).

Another variability dimension identified by the artist concerns the perception of the composition. Indeed, this perception can change if any instance of any part is flipped horizontally, thus adding an optional property for configurations: For example, a given ground part may have more brightness in its left or right side, adding a compositional decision for where to place this brightness in the whole painting. The artist also stated that each of the parts could take more or less space in the canvas. We therefore included an optional property for extra size:

For example, if a painting should have the sky visible in only a small section of the canvas, the middle and ground parts must be of extra size.

The variability in this domain can be expressed through Feature Modeling [14] which exposes the different configurations that can be selected to yield painting variants. It was thus easy to introduce the artist to different FM concepts and to discuss the different elements of paintings in terms of features. This led to the establishment of a FM (shown in Figure 1) for his painting style.

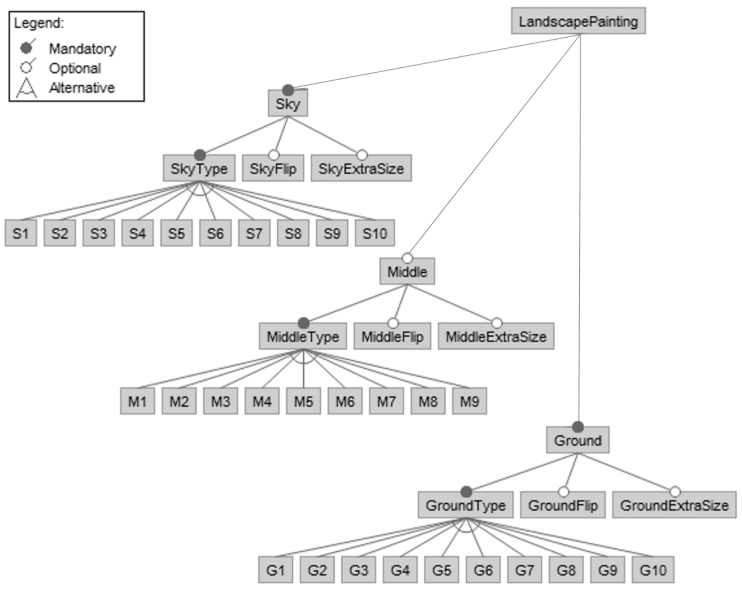


Fig. 1: Feature model of the landscape paintings

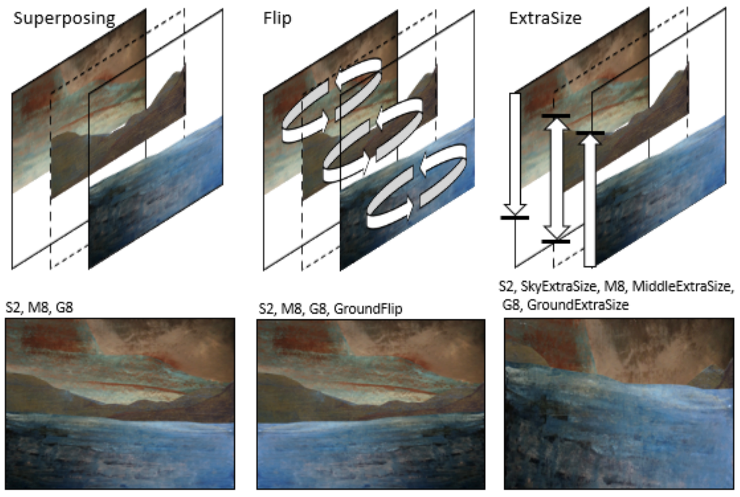


Fig. 2: Painting derivation process and examples

Deriving landscape paintings: Figure 2 illustrates the derivation process where a specific configuration, i.e., a given selection for sky, middle and ground, is assembled by superposition of the different visual components. We further illustrate how different paintings in the style of the artist can be obtained by flipping one or more painting parts or/and increasing the size of elements. We have implemented a tool which, given a configuration from the FM of Figure 1, we can generate a landscape painting based on the reusable assets of painting parts provided by the artist.

4 Motivations and Problem Statement

Computer-generated art and SPLE yield product variants that users may like it or not. For computer-generated art, it is the beauty of the product that is appreciated while in SPLE, it can be the aesthetic quality of the user interface as well, but it can be other user-specific quality attributes such as usability aspects or, more broadly, human-computer interaction concerns [18]. In both cases, the user feedback is important for producing optimal products that are aligned with user expectations. The motivation behind our work is to explore how user feedback can be automatically leveraged in practice to rank SPL product variants. This ranking contain the estimations and predictions that serves as input for domain experts decision-making process to select the products that have more guarantees of collective acceptance.

The problem faced in this paper is **ranking computer-generated art product variants based on user feedback**. Specifically we deal with SPL-based computer-generated art where deriving all products and collecting user feedback on each of them is virtually impossible: with the relatively small FM presented in Figure 1, the total number of possible configurations amounts to 59,200. Therefore, the first research question in this paper is:

– **RQ1**: Given the combinatorial explosion of configurations and the limited resources, how can one identify and select the optimal subset of products that are relevant for user assessment?

As we only consider a subset of products for user assessment, most of the products have not yet been assessed. In addition, user feedback are subjective by nature (different persons could assess the same product differently). Hence, the second research question for ranking product variants is:

– **RQ2**: Given a subset of assessed product variants, how can one infer the user feedback of the non assessed products? How can one aggregate user feedback to calculate new predictions even for the already assessed products?

5 Approach

We propose an enhanced method of collecting user feedback to create a relevant data set (first phase) which is processed to create a ranking (second phase).

5.1 Phase 1: Data set creation through evolution

The first phase of the approach aims at overcoming the challenge of selecting a relevant set of FM configurations to be presented to users for assessment. Indeed, to address the combinatorial explosion of possible configurations, we rely on an interactive genetic algorithm [22] which explore the possible configuration space trying to reach optimal or suboptimal solutions.

The genetic algorithm permits to create more data set instances in the regions that are more adapted to the fitness function. In our case study, the fitness function for the genetic algorithm is based on the user feedback captured using the device shown at the bottom of Figure 3. This device implements a physical 5-point scale with values ranging from 1 (*strong dislike*) to 5 (*strong like*). When a user votes, the displayed painting (top of Figure 3) vanishes and the next painting of the genetic algorithm population is displayed. When all paintings from the population have been assessed, a new population is yielded based on the calculations of the genetic algorithm, and the exploration towards optimal paintings continues, until it is stopped manually at the end of the session.

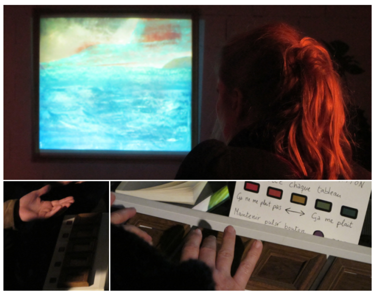


Fig. 3: Displayed painting and voting device with a five points scale

Algorithm 1 Interactive genetic algorithm for data set creation

```
input: Population = 20 members, Genetic representation of a member = 9 positions: SkyType,
SkyFlip, SkyExtraSize; MiddlePart, MiddleFlip, MiddleExtraSize, GroundType, GroundFlip,
GroundExtraSize; Type value from 0 to 9, Flip and ExtraSize values 0 or 1; Example: 801510210
output: data set of user assessments
1: population ← initializePopulation()
2: while ManualStopNotPerformed do
3:     for $member_i \in \{population\}$ do
4:         $member_i.fitness$ ← getScore($member_i$)
5:         registerDataInstance($member_i$)
6:     end for
7:     parents ← parentSelection(population)
8:     offspring ← crossover(parents)
9:     offspring ← mutate(offspring)
10:    population ← offspring
11: end while
```

Algorithm 1 shows the panmictic, non-elitist, generational and interactive genetic algorithm that we implemented. The *input* section of the algorithm shows how the genotype of a landscape painting phenotype was designed. Sky type, middle and ground positions are assigned with values representing all possible parts. In the case of the middle part, which is optional, value 9 represents its absence in the composed painting. Further, for this special case, the flip and extra size features are not valid, and thus a special treatment is performed in the operators of the genetic algorithm.

We now describe the characteristics of the algorithm. At line 1, the *initialization operator* creates a pseudo-random initial population: actually, we force all sky, middle and ground parts to appear in the initial population. The evolution starts at line 2 until it is manually stopped. During the evolution, from line 3 to line 6 each member of the population is evaluated using an *evaluation operator* based on user feedback. In comparison with generic implementations where the fitness function is normally automatically computed, in our process we need the participation of users to set the fitness value. The *parent selection operator* is then based on a fitness proportionate selection (line 7). At line 8, the *crossover operator* is based on one-point crossover with the peculiarity that it is not possible to select the last two positions to force to crossover the sky, middle and ground parts. Then the *mutation operator* used at line 9 is uniform with $p = 0.1$. Such a high mutation factor is meant to prevent a loss of motivation from users by reducing the likelihood that they will keep assessing similar products from the population, while thus enabling us to explore new regions. Finally, at line 10, the *survivor selection operator* is based on a complete replacement of the previous generation with the new generation.

5.2 Phase 2: Ranking computation

We make the assumption that when two products are similar in the way they were assembled they will be appreciated similarly. We describe how we compute the ranking of all products, including those which have not been directly assessed by users. First, we define a similarity measure for computing a *similarity distance* between two configurations in the domain (composed landscape paintings in our example). We then retrieve neighbors based on this similarity distance to compute a ranking of the configurations as well as confidence metrics.

Similarity distance: Given two configurations C_i and C_j, we aim at formally computing a value for the similarity distance between them. The notion of similarity distance between configurations was already studied in the software engineering literature, specially by the SPL testing community [1, 9, 10]. Apart from the use of generic similarity distances (e.g., Jaccard distance in [10] or Hamming distance in [1]), other approaches can be based on ad hoc domain-specific similarity functions. In this work, we defined our own similarity function for the configurations of the FM presented in Figure 1 as we considered that the results will be better than generic similarity distances. That means that the measure is ad hoc and defined based on artist's comments and our own expert judgement. To compare two configurations, we start by assuming that they are the same ($distance = 0$). The distance between them increases by 1 point for each painting part which is different. If a part is the same in both configurations, we check the flip feature and increase the distance by 0.2 in case of dissimilarities. Finally, and whether the parts are identical or not, we increase the distance by 0.2 if a part has an extra size in one configuration and not in the other: extra size is independent to the part as it has an impact on the whole composition. To account for the fact that the optional middle part may not be present, we always check that the part P is not null. The maximum distance between two configurations is 3.6 when the three parts are different and every part has extra size in one configuration and not in the other. Our intuition was that the fact that parts are different (1 point) is more important than the flip or extra size (we estimated it with 0.2 points). This definition of distance allows us to reason about the neighborhood of a configuration in the space.

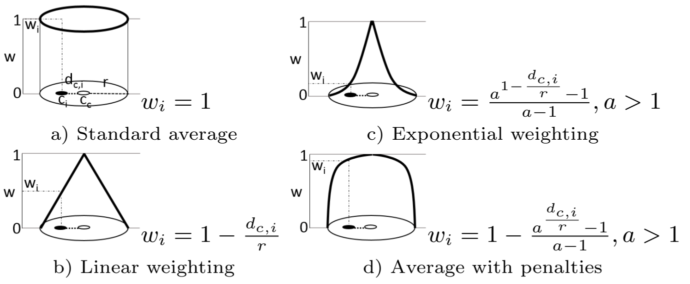
Similarity radius: To be part of the neighborhood of a given configuration, any configuration must be inside a *similarity radius*. Such a radius will allow to restrict the products that will be considered similar enough for inferring information from one to the other. For example, if we consider that two paintings are similar when only one of the parts is different (+1) and two transformations at most (flip +0.2 or extra size +0.2) should be performed to make them equal, then the radius value will amount to 1.4.

Weighted mean score computation: To assess the collective agreement on the score of a given configuration, we consider its neighborhood and compute the mean score in a similarity radius. We also consider this computed score as the expected level of appreciation by users. For computing reliable mean values, a weight is assigned for each instance of the data set. This weight depends on their proximity with the configuration under study, C_c.

Equation 1 provides the formula for computing the weighted mean score $\bar{s}_c$ for configuration C_c where s_i and w_i represent respectively the score and the associated weight of each of the N configurations C_i, in the similarity radius, that were in the data set.

$$\bar{s}_c = \frac{\sum_{i=1}^{N} w_i \cdot s_i}{\sum_{i=1}^{N} w_i} \tag{1}$$

Figure 4 describes four approaches for assigning a weight to the scores of configurations in the similarity radius. In each weighting approach, we will consider that the weight of the score for any configuration outside the radius is null ($w_i = 0$ when $d_{c,i} > r$, where $d_{c,i}$ is the similarity distance between configurations C_c and C_i and r the value of the radius).



1
w
0
w_i
$d_{c,i}$
r
C_i
C_c
$w_i = 1$
a) Standard average
$w_i = \frac{a^{1-\frac{d_{c,i}}{r}}-1}{a-1}, a > 1$
c) Exponential weighting
$w_i = 1 - \frac{d_{c,i}}{r}$
b) Linear weighting
$w_i = 1 - \frac{a^{\frac{d_{c,i}}{r}}-1}{a-1}, a > 1$
d) Average with penalties

Fig. 4: Different options to calculate the weight

Empirical selection of the approach settings: To select the radius and the weighting approach, we explore different combinations to identify the one that minimizes the error rate. The ideal experimental scenario would require to reconvene with all people who participated in the user study and ask them to assess some missing configurations for which we had previously inferred a score, and then evaluate the accuracy of the inference. However, this was not practically feasible, leading us to rely on a classical 10-fold cross-validation scenario.

The error rate is computed based on the difference between the expected value (the computed weighted mean score) and the actual user feedback score for each instance of the test set. The evaluation is a numeric prediction so we selected the *mean absolute error* (mae) as error rate metric as shown in Equation 2, where T is the number of instances in the test set, $\bar{s}_i$ is the weighted mean score of C_i computed with the training set (which represents the expected score) and s_i is the score of C_i in this instance of the test set (the actual score).

$$mae = \frac{\sum_{i=1}^{T} |\bar{s}_i - s_i|}{T} \tag{2}$$

Figure 5 shows the performances of different combinations of radius values and weighting approaches. Besides the mean absolute error values, we also represent how the choice of radius values impacts the coverage of the test set: if the radius is small there is a possibility that for some configurations the neighboring will be empty, thus preventing the computation of a mean score. Because such instances are not taken into account for the computation of the mean absolute

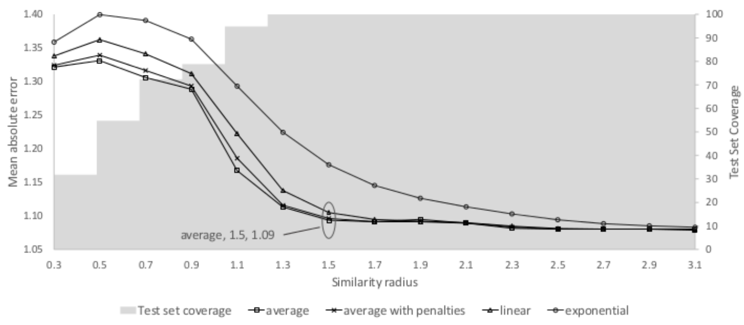


Fig. 5: Mean absolute error for different radius values and weight calculation approaches

error, it is important to know the average proportion (i.e., percentage) of the test set that is covered. The graph reveals that we start covering all instances of the test set with a radius of 1.3.

Ranking creation: After identifying empirically the most reliable radius value and weighting approach, we exhaustively compute the weighted mean for all possible configurations. In our case study, an exhaustive calculation for the 59,200 configurations is computationally feasible. In other cases where exhaustive calculation is not feasible, non-euclidean centroid calculations could be used to limit the configurations to analyze and make the approach scalable. In the possible cases where a configuration C_c has no neighbors within its similarity radius, no score is computed and it is ignored in the final ranking.

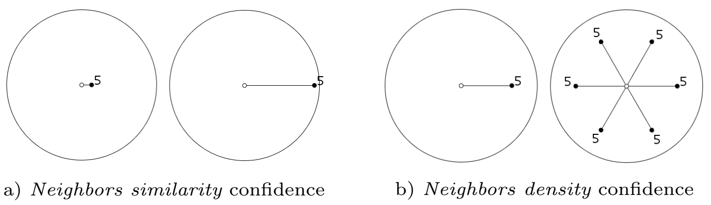
Confidence levels for ranking items: We define three main metrics for measuring the confidence levels.

Neighbors similarity confidence: The first metric explores the average distance with the neighbors. Figure 6a illustrates the importance of this metric. In both cases, the weighted mean is 5, however, intuitively, one is more confident about the accuracy of the mean score for the case on the left because the instance actually assessed is more similar.

We define a neighbors similarity confidence ($nsimc$) using Equation 3 where N is the number of data instances (i.e., neighbors) within the radius of C_c. When $N = 0$, this metric is not applicable and $nsimc_c = 1$ when all data instances are in the center, i.e., all configurations in the radius are identical to the configuration for which the score is inferred. For the calculations of $nsimc$ in this paper we will use the linear weighting approach.

$$nsimc_c = \frac{\sum_{i=1}^{N} w_i}{N} \tag{3}$$

Neighbors density confidence: The second metric explores the density of neighbors in the similarity radius. Figure 6b illustrates the importance of this



a) *Neighbors similarity* confidence b) *Neighbors density* confidence

Fig. 6: Illustration of the importance of confidence levels for mean scores

metric. In both cases, the weighted mean has a value of 5, but intuitively again, the confidence is greater for the second case. Equation 4 provides the formula for computing the neighbors density confidence where num represents the number of data instances within the radius of C_c, max is the highest value of num found in all the possible configurations and the function $instancesWithNeighbors(i)$ returns the number of instances from the data set that contain exactly i neighbors within their radius. We make a design decision to have a neighbors density confidence of 0.5 when the number of data instances within the radius of the configuration corresponds to the median of $instancesWithNeighbors(i)$.

$$ndenc_c = \frac{\sum_{i=1}^{num} instancesWithNeighbors(i)}{\sum_{i=1}^{max} instancesWithNeighbors(i)} \tag{4}$$

We describe in Figure 7 the graph of neighbors density confidence for all possible configurations of our case study. The sum of all accumulated values is the total number of 59,200 possible configurations. For example, we can see that around 500 configurations in the data set have each a total of exactly 8 neighbors within their radius that corresponds to a $ndenc$ of the 8%. We also observe that the maximum of data instances within a similarity radius is 78 and was recorded for only one possible configuration. We highlight the Q1, Q2 and Q3 quartiles which divide the data set into four equal groups where Q2 corresponds to the median. In the example of our case study, Q1=16, Q2=23 and Q3=29 correspond to a neighbors density confidence values of 25%, 50% and 75% respectively.

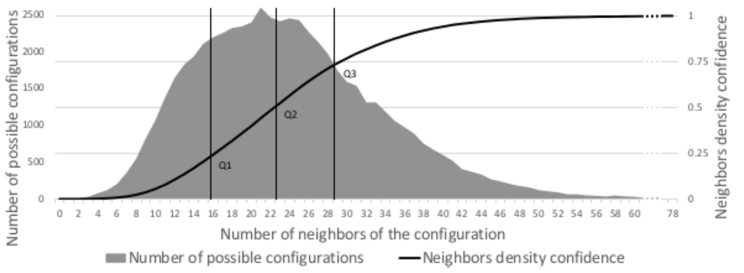


Fig. 7: Graph of neighbors density confidence

Global confidence: Finally we define a global confidence metric that takes into account the previous metrics. Equation 5 provides the formula for computing the global confidence. For our case study we decided to put more weight to $ndenc$. The rationale was to put an emphasis on the number of instances of user assessments used to compute the mean. We made the design decision of setting w_{ndenc} to 0.75 and w_{nsimc} to 0.25 in the computations presented in this paper.

$$gconf_c = w_{ndenc} \cdot ndenc + w_{nsimc} \cdot nsimc \tag{5}$$

6 Case study results

Phase 1 results: Data set creation. The installation was available to the public as part of an art festival at Théâtre de Verre at Paris. The installation was operative for 4 hours and 42 minutes and 1,620 votes were collected. The genetic algorithm and its exploration process towards optimal products led to

1,490 paintings being voted once, 62 paintings being voted twice and only 2 paintings being voted three times. No data was gathered to make distinctions about different user profiles nor any control mechanism was used to limit the number of votes per person. On average we were able to register 5.74 votes per minute from around 150 people of different ages and sociocultural backgrounds who voted for one or more paintings. Figure 8 shows the scores' distribution.

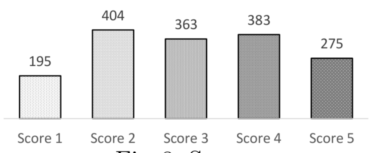


Fig. 8: Scores

We applied data mining attribute selection algorithms in an attempt to discriminate between relevant and irrelevant features. Concretely, we evaluated the worth of each feature by computing the value of the chi-squared statistic with respect to the class (i.e., a score in the range of 1 to 5). The ranked features showed that ground, sky and middle parts were more relevant features than middleFlip, groundExtraSize, groundFlip, skyFlip, middleExtraSize, skyExtraSize. However, *a priori* association discovery algorithms and decision trees performed poorly (incorrect classification of 80% of instances in the 10-fold test sets).

Phase 2 results: Ranking creation. Based on the empirical investigation of the different radius and weighting approaches, we have tuned our ranking computation parameters to optimal values. We aimed at high coverage of configurations while minimizing mean absolute error rates. In Figure 5, we marked with a circle our selected parameters. We set the radius to 1.5 and selected the standard average weighting approach for the weighted mean score computation. A radius of 1.5 further fits the artist's intuition of the minimum differences in paintings to be considered "similar".

We computed the weighted mean score for all 59,200 possible configurations and ranked them accordingly. Table 1 depicts the paintings that were derived from the top 10 configurations with highest weighted mean scores (wmean). The highest wmean was 4.75 for only one configuration. None of the configurations in the top 10 were part of the data set created in Phase 1 when collecting user feedback. For example, the third best configuration score was obtained based on the wmean of 11 different configurations with scores computed from actual user feedback. Table 2 depicts the bottom 5 configurations of the ranking. We observed that bottom configurations had in general less confidence given the effect of the genetic algorithm that tried to avoid these regions.

We also used the global confidence metric to filter and reorder at will the ranking items. The following examples aim to show the kind of analyses that are possible thanks to enriching the ranking with confidence metrics. Figure 9a shows the product that was derived for the configuration with the highest global confidence (82%). This configuration got a wmean of 3.27 and holds the rank 14,532. The configuration with the highest confidence level for configurations that are liked (i.e., $score > 4$) holds the position 102 within our ranking. The corresponding product is illustrated in Figure 9b. Similarly, we identify the configuration with the highest confidence level for configurations that are disliked (i.e., $score < 2$). It holds the rank 59,145 and it is shown in Figure 9c.

Table 1: Top 10. Each configuration is defined by: wmean {gconf% (*ndenc%, nsimc%*)}

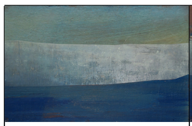
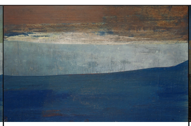
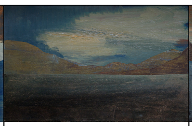
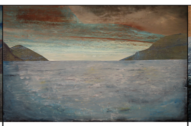
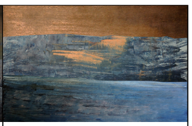
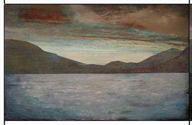
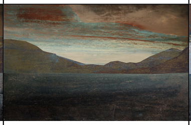
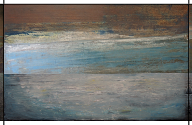
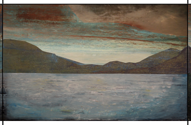
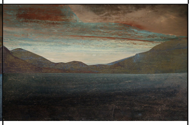
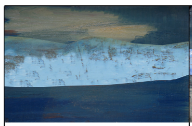
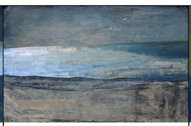
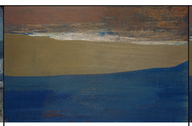
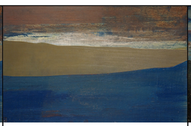
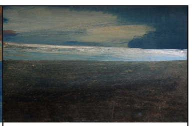
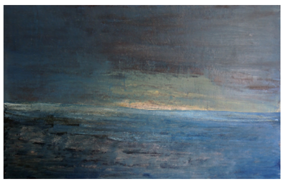
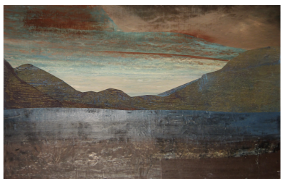
				
4.75 {4% (≈ *0%,17%*)}	4.67 {5% (≈ *0%,20%*)}	4.64 {8% (*8%,10%*)}	4.54 {16% (*14%,23%*)}	4.5 {5% (*2%,13%*)}
				
4.42 {33% (*36%,32%*)}	4.41 {43% (*48%,27%*)}	4.4 {6% (≈ *0%,23%*)}	4.36 {48% (*56%,23%*)}	4.35 {37% (*40%,30%*)}

Table 2: Bottom 5 defined by: wmean {gconf% (*ndenc%, nsimc%*)}

				
1.5 {4% (*2%,8%*)}	1.5 {3% (*1%,11%*)}	1.33 {5% (≈ *0%,20%*)}	1.33 {3% (≈ *0%,11%*)}	1 {3% (≈ *0%,13%*)}


a) $gconf = 82\%$
Rank 15,532


b) $gconf = 63\%$
Rank 102 - liked

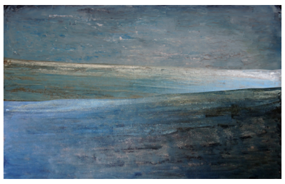

c) $gconf = 21\%$
Rank 59,145 - disliked

Fig. 9: Key paintings with high confidence levels (overall, liked only, disliked only)

7 Evaluation

To respond to RQ1 and RQ2 we evaluated our approach from different angles:

Controlled assessment: The objective of the controlled assessment is to quantitatively study the error rates in how the scores were estimated in the presented case study. In Section 5.2, we had already performed a 10-fold cross-validation to empirically select an optimal value for the similarity radius and the weighting approach. Now we also estimate the error rates using the whole data set with real user feedback to draw both training and test sets. The goal is to predict the score of a configuration and assess against the actual scores provided by users. The computed *resubstitution error* is an optimistic case for evaluating classification approaches. The mean absolute error in the case of resubstitution is 1.0523 and the mean absolute error using 10-Folds average is 1.0913. These results suggest that any prediction has a margin of error around 1. This means that if the actual score of the painting is 4 (*like*), the margin of error is between 3 (*normal*) and 5 (*strong like*). We consider the mean absolute error values, in conjunction with the confidence metrics of the rankings, to be a good performance when attempting to capture collective understanding of beauty within the boundaries of our landscape paintings SPL.

Artist perspective: The artist obtained and analyzed the ranking and the confidence metrics using the approach providing the following qualitative discussion. He claimed that the collective as a whole stated their scores in a very coherent fashion according to the parameters of traditional and classical painting principles of perspective and contrast. For example in Figure 9b, the brightness in the sky found its counterpart in the brightness of the sea but only in the left side because of the mountain on the right. People understood that they were dealing with landscapes and they disliked the ones that tended to be flat or that they did not respect some of these principles. The objective of the installation was to explore his painting style in a feasible way to leverage user feedback. The resulting ranking was very interesting to understand people sensibility about the possible configurations. The ranking showed him liked configurations with high global confidence that he never considered and that he liked them too. The results of the approach exposed him to novelty that, as added value, he considered that they have some guaranties of success when exposing them to the public. For example, before this exercise, whether he put mountains or sea but never together in the same composition. He considered that he has learned about his own painting style as well as about the perception of the people about it.

Individualized evaluation: Using the previous controlled assessment it was not feasible to bring the whole collective back for evaluating the created ranking. However, by conducting the experiment with only one person we can create a ranking with his or her own feedback and then evaluate the validity of our estimations with the person on site. The objective was to evaluate if the ranking successfully discriminate between the liked and disliked for the perception of each user. We selected 10 persons for this evaluation which had not seen the paintings before. Each user was voting in a session of 20 minutes. In average this duration corresponded to a data set of 16 populations (320 paintings). Once the ranking was created we took 10 liked and 10 disliked that they were not shown during the evolutionary phase. In concrete, we took the first 10 paintings with a weighted mean score from 4.5 to 5 that had the highest global confidence. In the same way, we took the first 10 with a weighted mean score from 1 to 1.5 that had the highest global confidence. After randomly shuffling these 20 paintings, we obtained an average of 91% accuracy in the prediction between like and dislike. The results suggest that the predictions in the extremes of the ranking are accurate in the case of individualized estimation.

8 Threats to validity

Despite of the promising findings of our study, the approach presents threats to validity as we have made several design choices, which may introduce biases in our results. In Phase 1 we should investigate different genetic algorithm operators for the same case study to try to find the optimal operators and compare it to other approaches not relying on genetic algorithms. In Phase 2, the main threat in validity lies in the approach for computing the similarity distance between two paintings. We chose to implement the presented distance function algorithm while we could have relied upon more complex algorithms based on image difference metrics or on distance matrices for each of the features. In addition, the methods for empirically selecting the similarity radius as well as the approaches for weighting the scores could be further improved.

The principles and techniques of the presented approach are repeatable for any case study dealing with user feedback on SPL product variants. However, as discussed during the paper, this approach will not scale in phase 2 for FMs that can produce large number of possible configurations. In this case it is not feasible to compute and create ranking for all configurations. To solve this, instead of calculating the weighted mean score for all the possible configurations as we have done in the case study, we will investigate on non-euclidean centroid-based approaches or filtering mechanisms to restrict the calculation to a feasible amount of configurations.

Given that we deal with subjective assessments, the inherent subjectivity of ratings is an important threat. Also, our approach considers the rates as numbers in order to summarize the variant assessments for calculating the score means. In rating scales, for example from one to five, depending on the person, the distance from one to two may not be the same as the distance from two to three. This non-linearity of the scores scale is related to personal and cultural factors. The conversion of numerical values to nominal values [16] is an alternative for the ranking creation phase that is worthy to explore and compare.

9 Conclusion

Leveraging user feedbacks in the context of SPLE is an emerging problem that addresses an important aspect of products success: user expectations on product variants. This paper presents an approach for estimating and predicting user perception through the creation of a ranking of all the possible configurations based on user feedbacks. We use computer-generated art as an extreme case of subjective perception of a software product. This ranking, that we enhance with confidence metrics for each ranking item, has the objective to serve as input for the decision making process to select the products that have more guarantees of collective acceptance by users. The approach contains two phases at which we use 1) an interactive genetic algorithm for the initial data set creation and 2) a tailored data mining interpolation technique for reasoning about the data set to infer the ranking and the confidence metrics. We apply and validate the approach on an SPL-based computer-generated art system dealing with landscape paintings. This case study exhibits the same challenges than other assessment scenarios on user-specific qualities of a family of products: combinatorial explosion of possible configurations and limitation of resources for getting user feedback. The results of the case study are promising and enhanced versions of the approach will be applied to other case studies on SPLs where exploiting user feedbacks is of special importance.

Acknowledgments. Martinez and Tewfik's work is supported by the ITEA3 15010 REVaMP^2 project: FUI the Île-de-France region and BPI in France.

References

1. Al-Hajjaji, M.: Scalable sampling and prioritization for product-line testing. In: Software Engineering & Management 2015, Dresden, Germany. pp. 295–298 (2015)

2. Bavota, G., Carnevale, F., Lucia, A.D., Penta, M.D., Oliveto, R.: Putting the developer in-the-loop: An interactive GA for software re-modularization. In: SSBSE (2012)
3. Benavides, D., Martín-Arroyo, P.T., Cortés, A.R.: Automated reasoning on feature models. In: CAiSE. pp. 491–503 (2005)
4. Boden, M.A., Edmonds, E.A.: What is generative art? Digital Creativity 20(1-2), 21–46 (2009)
5. do Carmo Machado, I., McGregor, J.D., de Almeida, E.S.: Strategies for testing products in software product lines. ACM SIGSOFT SE Notes 37(6) (2012)
6. Cohen, D.M., Dalal, S.R., Fredman, M.L., Patton, G.C.: The aetg system: An approach to testing based on combinatiorial design. IEEE TSE 23(7) (1997)
7. Galindo, J.A., Alférez, M., Acher, M., Baudry, B., Benavides, D.: A variability-based testing approach for synthesizing video sequences. In: ISSTA (2014)
8. Ghannem, A., El-Boussaidi, G., Kessentini, M.: Model refactoring using interactive genetic algorithm. In: SSBSE (2013)
9. Hemmati, H., Arcuri, A., Briand, L.C.: Achieving scalable model-based testing through test case diversity. ACM Trans. Softw. Eng. Methodol. 22(1), 6 (2013)
10. Henard, C., Papadakis, M., Perrouin, G., Klein, J., Heymans, P., Traon, Y.L.: Bypassing the combinatorial explosion: Using similarity to generate and prioritize t-wise test configurations for software product lines. IEEE TSE 40(7) (2014)
11. Henard, C., Papadakis, M., Perrouin, G., Klein, J., Traon, Y.L.: Multi-objective test generation for software product lines. In: SPLC. pp. 62–71 (2013)
12. Hornby, G.S., Bongard, J.C.: Accelerating human-computer collaborative search through learning comparative and predictive user models. In: GECCO (2012)
13. Johansen, M.F., Haugen, Ø., Fleurey, F., Eldegard, A.G., Syversen, T.: Generating better partial covering arrays by modeling weights on sub-product lines. In: MoDELS. pp. 269–284 (2012)
14. Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S.: Feature-oriented domain analysis (foda) feasibility study. Tech. rep., DTIC Document (1990)
15. Li, Y.: Adaptive learning evaluation model for evolutionary art. In: CEC (2012)
16. Martínez, H.P., Yannakakis, G.N., Hallam, J.: Don't classify ratings of affect; rank them! T. Affective Computing 5(3), 314–326 (2014)
17. Martinez, J., Rossi, G., Ziadi, T., Bissyandé, T.F.D.A., Klein, J., Traon, Y.L.: Estimating and predicting average likability on computer-generated artwork variants. In: GECCO (Companion). pp. 1431–1432. ACM (2015)
18. Martinez, J., Sottet, J.S., Frey, A.G., Ziadi, T., Bissyandé, T., Vanderdonckt, J., Klein, J., Le Traon, Y.: Variability Management and Assessment for User Interface Design, pp. 81–106. Springer International Publishing (2017)
19. Romero, J., Machado, P. (eds.): The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music. Natural Computing Series, Springer (2008)
20. Sayyad, A.S., Menzies, T., Ammar, H.: On the value of user preferences in search-based software engineering: a case study in software product lines. In: ICSE (2013)
21. Siegmund, N., Rosenmüller, M., Kästner, C., Giarrusso, P.G., Apel, S., Kolesnikov, S.S.: Scalable prediction of non-functional properties in software product lines: Footprint and memory consumption. IST 55(3) (2013)
22. Takagi, H.: Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. IEEE 89(9), 1275–1296 (2001)
23. White, J., Benavides, D., Schmidt, D.C., Trinidad, P., Dougherty, B., Cortés, A.R.: Automated diagnosis of feature model configurations. JSS 83(7) (2010)