JOT
JOURNAL OF
OBJECT TECHNOLOGY

Journal of Object Technology | **RESEARCH ARTICLE**

Variability Exploration for Decision Making: Supporting Domain Experts in Configuring Business Processes

Haitam El Hayani[*†], **Benoit Combemale**[†], **Olivier Barais**[†], **and Steffen Zschaler**[§]
[*]ENSIAS, Morocco
[†]IRISA, Univ Rennes, CNRS, INRIA, France
[§]King's College London, United Kingdom

ABSTRACT

Designing a model with built-in variability that can later be specialized for specific needs has become a common practice. This approach enables the consolidation of company expertise within a single model, extending its applicability beyond a single system, process, or behavior. Such modeling reveals a set of choices that Domain Experts (DEs) must evaluate, collectively forming the variability space—the range of potential decisions available to achieve desired project outcomes. Selecting the optimal decision within this variability space is often challenging for DEs, especially those without a technical background. The abundance of alternatives, each with the potential to significantly influence the capabilities of the final solution, adds complexity to the decision-making process. To assist DEs in exploring their models, we propose a tool-supported method for discovering and visualizing the variability space captured within feature models. This method allows experts to explore and evaluate different options against predefined objectives. By representing the variability space in a format conducive to decision-making, our method helps identify key choices that impact overall business processes, assess the implications of each option, and explore alternative configurations. We validate this method through a simplified case study of the OneWay project of Airbus, a leading international aircraft manufacturer. Applying our method to their feature model and business processes for avionics program development planning demonstrated its effectiveness in supporting decision-making activities and its overall performance.

KEYWORDS Variability management, Software product lines, Evolutionary algorithms

1. Introduction

Model-based variability management is commonly used for managing large variability spaces in the development of modern complex systems such as cyber-physical systems, configurable software systems and business processes. We can cite examples from the open source sector, such as JHipster[1] (Halin et al. 2017), but also from the systems engineering sector at Airbus (Foures et al. 2023) and Thales (Noir et al. 2016). It aims at providing systematic engineering processes and tool support for configuring and composing reusable components for a given objective. However, large variability spaces lead to error-prone and time-consuming manual exploration activities and make hard the decision for the various options. These activities usually involve various stakeholders and apply a multi-stage approach, while trying to reach a global optimization of the resulting product with regard to a given objective.

The exploration of a large variability space requires specific decision-making tools to let Domain Experts (DE) explore different configurations and/or options, and assess them against predefined objectives. The challenge is twofold: i) the variability space and the various decision choices need to be abstracted and visualized for the sake of decision-making, and ii) the involved stakeholder (DE), needs to interact with the exploration process to drive it according to specific preferences.

JOT reference format:
Haitam El Hayani, Benoit Combemale, Olivier Barais, and Steffen Zschaler. *Variability Exploration for Decision Making: Supporting Domain Experts in Configuring Business Processes.* Journal of Object Technology. Vol. 24, No. 2, 2025. Licensed under Attribution - NonCommercial - No Derivatives 4.0 International (CC BY-NC-ND 4.0) http://dx.doi.org/10.5381/jot.2025.24.2.a3

[1] https://www.jhipster.tech/

To navigate this complex variability space effectively, Evolutionary Algorithms (Alain Petrowski 2017) (EA), particularly genetic algorithms (Mitchell 1996), have emerged as relevant solutions due to their ability to explore large search spaces efficiently for both mono and multi-objective problems. As search-based optimization algorithms, EAs can systematically explore the configuration space while simultaneously optimizing multiple objectives, generating an approximation of the Pareto front. However, the Pareto front alone proves insufficient for solving the problem (Deb 2007) (Fernandez et al. 2011), since domain experts still face the challenge of identifying the best compromise among numerous solutions. Therefore, it becomes crucial to incorporate domain experts' preferences into the optimization process, guiding the search toward a specific Region of Interest (ROI), defined by Adra et al. (S.F. Adra 2007) as the set of non-dominated solutions that the DE prefers over the other solutions. This preference-guided method not only streamlines the decision-making process but also ensures that the solutions generated align more closely with the domain experts' objectives and constraints.

In this paper, we propose a tool-supported method to interactively explore a variability space for the sake of decision-making. From a variability model and its associated realization in a given base model, the variability management method combines genetic algorithms with the use of Parallel Coordinates to support decision-making, and interactions to drive the exploration with predefined preferences. In this paper, we report on the experience of applying the method in the context of a case study provided by Airbus, a European aircraft manufacturer, in the OneWay project (Foures et al. 2023). The case study involves the use of feature models to capture the variability space over BPMN models that capture business processes of program development plans. We demonstrate the ability to implement the toolchain involving the derivation and stochastic simulation of BPMN models to support domain experts.

In summary, the contributions of this paper are:

- A tool-supported method to explore a variability space over models for the sake of decision making,
- An implementation to BPMN models and their simulation for decision making,
- An application to a real-world example about program development plans for a leading international aircraft manufacturer.

We validate this method through an industrial case study. The application of our method to business processes that contain variability for the planning of avionics development highlights the advantages and limitations of our method. In particular, by providing domain experts with a clear and comprehensive view of their options and potential outcomes, our method enables them to make more informed choices, ultimately leading to more effective business processes for planning the development of future programs. We also show how the DE is in the loop to point out, in an incremental way, the areas of interest (s)he wants to explore. By finely analyzing the whole process of exploring the configuration space, we also evaluate the phases that allow interactive exploration and those that require pre-processing.

The remainder of the paper is structured as follows. Section 2 introduces the required background and the motivation. Section 3 introduces the overall proposed method. Section 4 presents our implementation and tool support. Section 5 evaluates our tool-supported method through a real-world case study. Section 6 discusses the related works, and Section 7 concludes and outlines future work.

2. Background and motivation

Variability refers to the ability of an artifact to be adapted, configured, or modified for specific contexts (Bachmann & Paul 2005). **Variability Management (VM)** involves representing variability in software artifacts, managing their dependencies, and their instantiation throughout the software life-cycle (Schmid & John 2004). Given the inherent complexity of these tasks, specialized approaches, techniques, and tools are essential (Schmid & John 2004) (Bosch et al. 2001). The systematic identification and management of variability across system families distinguishes Software Product Line Engineering (SPLE) from other reuse-based development approaches (Bosch et al. 2001). Techniques for variability management include amalgamated and separated (aka. orthogonal) approach (Chen et al. 2009) (Haugen et al. 2008). The amalgamated approach suggests extending the base language (i.e., the language used for defining the software artifact) with variability concepts. In contrast, the separated approach maintains independence between the base language and the variability language through a defined mapping (Foures et al. 2023). While the Common Variability Language (CVL) (Haugen et al. 2012) wasn't adopted as a standard for **Orthogonal Variability management (OVM)**, it offers valuable insights into the key concepts:

- **Variability Model (VM)** provides a tree-based, high-level representation of the SPL's features and constraints, inspired by feature models (Czarnecki et al. 2012).
- **Base Models (BMs)** are a set of models, each conforming to a domain-specific language (e.g., UML, BPMN) called base language. In CVL, these models serve as the foundation for product derivation.
- **Variability Realization Model (VRM)** defines how features in the VM map to elements in the BMs, specifying the modifications (addition, removal, substitution) required for a feature selection or deselection.
- **Resolution Models (RMs)** store feature selections for a specific product configuration. These selections are then applied to the base models to derive the final product model.

Feature models are the defacto standard for representing and managing **variability** in Software Product Line(Kang et al. 1990). Hence, software products can be derived from modeling variability with feature models to meet a specific version and variant of components (Ziadi T. 2006) by choosing the features relevant to a particular configuration. A feature model is a tree-structured representation of system characteristics, where features form system configurations. Features can be mandatory or optional, grouped in or/xor relationships, and connected by

cross-tree constraints (David Benavides 2010). Initially proposed by FODA, the model was extended to include cardinality (Czarnecki et al. 2005) and attributes (Benavides et al. 2005), to add extra-functional information.

Evolutionary Algorithms (EAs) are well-suited for exploring complex configuration spaces (Alain Petrowski 2017), particularly for potential system configurations captured by feature models. To solve complex computational problems, EAs iteratively refine a population of candidate solutions guided by a fitness function that evaluates their quality. Core operators such as selection, crossover, and mutation drive the exploration of the solution space. This involves initializing a random population, evaluating their fitness, selecting individuals for reproduction, applying variation operators, and replacing the population until a termination criterion is met (Bäck 1996)(Alain Petrowski 2017). During the 1960s and 1970s many independent approaches were developed in this area, notably the evolution strategies of Schwefel and Rechenberg (Beyer 2001), the evolutionary programming of Fogel *et al* (Fogel et al. 1966), and the genetic algorithms that were presented in 1975 by Holland (Holland 1992) and stand out as the most popular evolutionary algorithms (Alain Petrowski 2017).

While EAs excel at navigating complex solution spaces, **integrating Domain Experts' (DE) preferences** into the process is critical for **steering** the algorithm toward **Regions Of Interest (ROI)** that align with the DE's priorities. According to Miettinen (Miettinen 1998), Ching-Lai Hwang and Abu Syed Md.Masud (Hwang & Masud 1979), and J.Branke et al (Branke et al. 2008) there are three manners for the participation of the domain experts in the search:

- **A priori**: Preference information is given before the algorithm execution. It's the most efficient method, however, it presents high risks of achieving unsatisfactory solutions, since it assumes that domain experts have enough precise knowledge to be aware of the trade-offs (Meignan et al. 2015) (Piemonti et al. 2017).
- **A posteriori**: Preferences are articulated after the end of calculations. Its main advantage is providing the domain expert with a comprehensive overview of solutions, but its downside is that calculations can be so long, and the DE may not have enough time to wait.
- **Interactive**: Preferences are iteratively refined during the optimization process. This method involves the domain expert in the search process, balancing efficiency and effectiveness, and allowing him/her to learn from intermediate solutions and guide the search toward more desirable regions (Kok 1986).

To ensure effective interaction with domain experts during the optimization process, it is imperative to employ an intuitive representation that facilitates the capture of preferences and the visualization of results in a readily understandable format.

Parallel coordinates (PC) represent N-dimensional data using N parallel axes. Each data point is a polyline that intersects each axis at its corresponding value. PC are widely utilized (Heinrich & Weiskopf 2013) for representing and exploring high-dimensional datasets (Inselberg 2009) making them suitable for widely interactive, multi-objective, optimization visualizations. Due to how dimensions are represented, PC facilitates comparisons, identifies trade-offs, and reveals trends and clusters within the data (Shenfield et al. 2007)(Abi Akle et al. 2017). Several studies have explored their practical usage in multi-objective optimization. Akle et al. (Abi Akle et al. 2017) compared the PC to radar charts and combined tables, finding that the PC is more effective and engaging to explore while requiring less cognitive load than other charts. They are recognized for their intuitive representations, which facilitate the understanding of complex relationships between multiple objectives (Abi Akle et al. 2017) (Packham et al. 2005). Furthermore, parallel coordinates are highly scalable, as they can accommodate many criteria while occupying minimal space on the screen (Fleming et al. 2005). This makes them a valuable tool for visualizing and analyzing high-dimensional spaces such as the variability space.

Considering the human mind's limited capacity to handle large amounts of information (G. A. Miller 2010), and the increasing complexity of business processes in modern enterprises, our motivation is to empower domain experts to make informed strategic decisions. This ambition is driven by our desire to combine the power of feature models, for capturing variability within business processes; evolutionary algorithms, for navigating the variability space; and parallel coordinates, for simplifying the technical complexities of exploration, making them more accessible to domain experts.

3. Variability Space Exploration

We aim to help domain experts (DEs) better manage decisions about complex variability spaces by providing a tool-supported method to explore a variability space over models. Figure 1 provides an overview of our method. The method is structured into three key phases: **Variability space discovery**, **Decision-making support**, and **Variability model refinement**. The last phase bridges the previous two steps by allowing the domain expert to guide the exploration according to their points of interest.

The remainder of this section details each phase and specifies the artifacts used.

3.1. Variability space discovery

The objective of the *discovery phase* is to identify a limited set of diverse configurations to enable the domain expert to form a good overview of the variability space in the next phase. To this end, we employ evolutionary search over configurations, model derivation (we assume an orthogonal approach to variability modeling), and model evaluation to be able to understand the impact of configuration decisions on model properties and objective functions.

Case study illustration: We begin with OVM models, which include business processes represented using BPMN (base model), a feature model capturing variability, and a variability realization model that provides rules for applying changes to the BPMN models during feature selection.

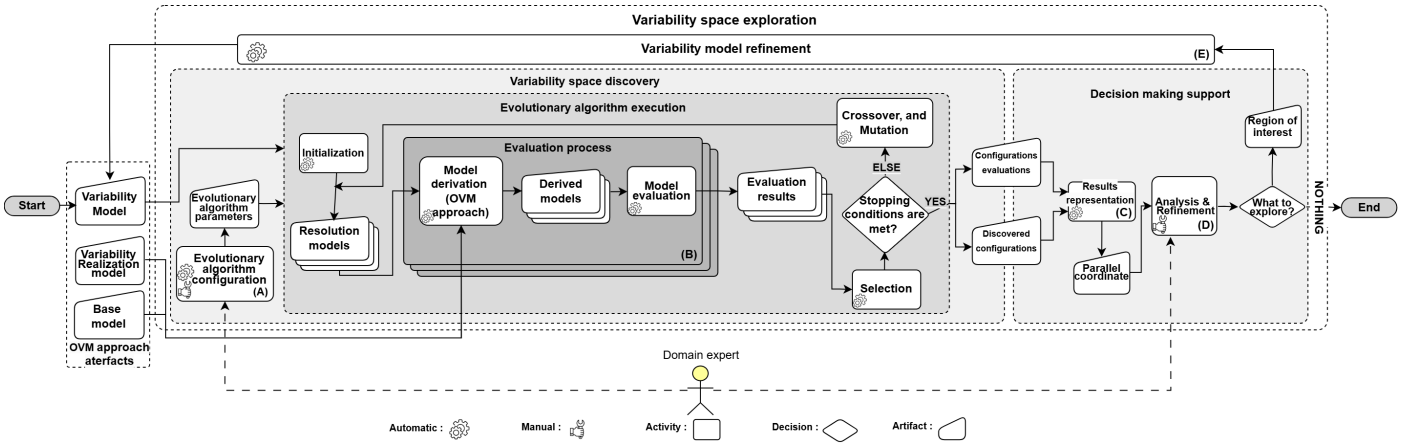

Figure 1 Conceptual pipeline

As shown in Figure 1, the variability space discovery process begins with a **Variability Model (VM)** that formally represents the various features and options available within the business process domain. This model encapsulates all potential options/-choices, thus characterizing the variability space.

Two primary steps follow: (i) the **evolutionary algorithm configuration**, to set the search parameters, (ii) the **evolutionary algorithm execution** to run the evolution discovery activity.

Evolutionary algorithm configuration. Before executing the Evolutionary Algorithm (EA), the Domain Expert can manually customize algorithm parameters such as population size, mutation probability, crossover probability, and stopping condition of the EA. Alternatively, the configuration is done automatically by retaining default values. These parameters are then passed to the EA to commence the search process, as illustrated in Figure 1.

Evolutionary algorithm execution. The Evolutionary Algorithm (EA) accepts two primary inputs: the Variability Model and the EA parameters. Subsequently, the EA proceeds through its execution with the following steps.

1. **Initialization:** The solution's space discovery process starts with the initialization phase, where a random configuration is generated. This involves selecting/unselecting various possible feature combinations from the Feature Model, which serves as a base for creating the initial resolution models.

2. **Evaluation process:** Two main steps happen in the evaluation step:

   (a) **Model derivation:** The generated configurations, representing Resolution Models, are used to create specialized models (derived models) through **Orthogonal Variability Management**. This step derives business models by applying the rules defined in the Variability Realization Model.

   Case study illustration: When EA generates a valid configuration, we use the OVM approach to create a specialized BPMN model for this configuration.

   (b) **Model evaluation:** The fitness function, defined by the DE, evaluates the newly generated models against desired criteria, such as performance, cost, delivery time, etc. The result of this evaluation is a quantitative metric for comparing the different generated models that come from different configurations.

   Case study illustration: We use a simulator that computes the lead time of the newly derived model.

3. **Selection :** The goal of this process is to optimize the configurations by converging on the resolution models that have the highest fitness scores.

   Case study illustration: The fitness function takes a resolution model as input and returns the evaluation results by applying the steps outlined in step 2. These results are then used to select individuals with the highest fitness scores for further refinement.

4. **Stopping conditions check:** Stopping conditions determine when to terminate the evolutionary algorithm, as EAs can run indefinitely. Typical stopping criteria include:

   - **Lack of change:** The algorithm terminates if objective values (fitness scores) stagnate. This may indicate convergence or being stuck in a local optimum.
   - **Time limit:** The search stops after a predefined time, which is useful for time-sensitive scenarios.
   - **Number of iterations:** Termination occurs after a fixed number of iterations or fitness evaluations.

   Once the stopping conditions are met, the algorithm concludes the discovery, identifying the configurations that are considered optimal based on the fitness evaluations.

Case study illustration: In the Airbus case study, the EA stops based on either a predefined number of generations or a time limit. For example, the algorithm could stop after running for 100 generations or if 2 hours of computation time elapsed.

5. **Crossover, and Mutation :** These operations are designed to explore the variability space, generating new configurations that are likely to improve upon previous iterations. The resulting configurations are used to create new resolution models.

   Case study illustration:
   In our use case, we used only mutation operators automatically generated by ACAPULCO, more details about these operators can be found here (Horcas et al. 2023).

Through the generation of different configurations, the evolutionary algorithm is responsible for discovering and exploring the variability space. The orthogonal variability management system allows these configurations to be evaluated and the evaluation results are used in the fitness score to drive the variability discovery. Note that only this first phase may require adjustments related to the type of base model, such as adaptations for model evaluation or specialization of derivation semantics, among other considerations.

3.2. Decision making support:

By the end of the Variability space discovery phase, we obtain two key outputs: discovered configurations and their evaluations. To assist non-technical stakeholders, these results are represented as a parallel coordinate (PC) plot, as shown in Figure 1. This requires **abstracting the feature model (FM) into a parallel coordinate plot (PC)** which is ensured by the **activity C** of Figure 1. The DE then analyzes the PC for further refinement in **activity D** of Figure 1. If satisfied, the process ends. Otherwise, the DE's region of interest (ROI), identified in the PC, is passed to the next phase to steer the search towards their ROI.

The key challenge of this step was to investigate the translation of feature models, which capture the system's variability, into a visually intuitive representation. Given the substantial number of features, the commonalities captured through various variation points, and the high-dimensional data generated by the evolutionary algorithm, we selected parallel coordinate visualizations as a suitable representation technique. Parallel coordinate plots, as discussed in the background section, are well-suited for high-dimensional data as they are recognized for their intuitive representations. In this context, each dimension represents a set of choices available to the domain expert. Using parallel coordinates, we aim to provide domain experts with a clear and comprehensive overview of the solution space, facilitating informed decision-making.

Nevertheless, this approach posed challenges, such as determining the appropriate level of **abstraction**, deciding which features to include or exclude, and effectively choosing the dimensions of our parallel coordinate plot. To do so, we introduced a new term called **Deepest Variation Point (DVP)**, representing the internal nodes with at least **one optional leaf**. Only

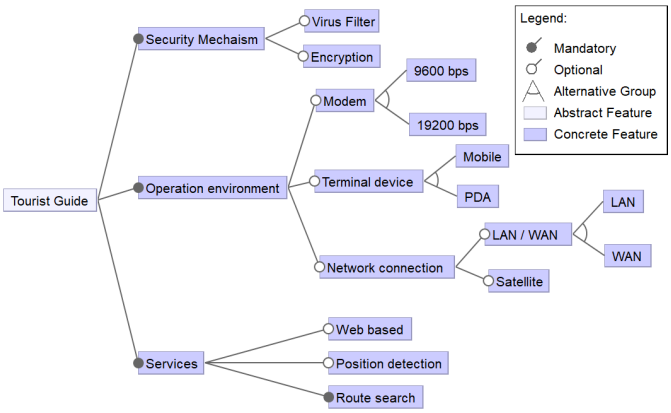

Figure 2 Sample feature model

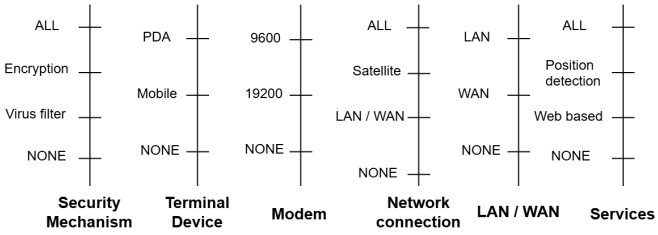

Figure 3 Sample parallel coordinate plot matching figure 2

optional leaves were considered to be the dimension's values since mandatory ones do not represent any choices. **Cross-tree constraints (CTCs) weren't considered in the construction of the parallel coordinate plot. Based on the work of (Knüppel et al. 2017), it's possible to translate a FM with CTCs to a FM without CTCs.**

We adopted the following strategy: **n features/dimension**. Dimensions are DVP, with a maximum of 2^n ***values*** per dimension, where **n = number of optional leaves in the corresponding DVP + number of other optional DVPs under the same DVP** considering the type of grouping (OR, XOR). The worst-case scenario is the OR group case, whereas for XOR group or a combination of XOR and OR groups, the number will be smaller since in the XOR group at most one feature can be selected.

This strategy reflects the structure of the feature model, and the number of dimensions is exactly the number of DVPs of the FM.

For example, let's consider the feature model of figure 2. We can identify 6 DVPs, **Security mechanism, Terminal device, Modem, Network connection, LAN / WAN, and Services**. By applying the rules defined by our strategy, we get the equivalent parallel coordinate plot presented in figure 3.

The first dimension has four values due to the *OR* group. Dimensions two and three, being XOR groups with optional DVPs, have two values plus a NONE option corresponding to scenarios where the dimensions themselves weren't selected. Dimension four has four values presenting the different combinations of Satellite and LAN/WAN group. The fifth

dimension is an optional XOR group, hence the three values and the final dimension is an OR group with two optional leaves, leading to four values. So, the resulting parallel coordinate visualization effectively represents the various configurations of the feature model of Figure 2, focusing on key variation points and their possible choices.

Following the mapping of the feature model to the parallel coordinate visualization, we elucidate the process of translating the domain expert preferences, expressed within the visualization, into a refined feature model.

3.3. Variability model refinement

Once DE's preferences are specified within the Parallel Coordinate (PC) plot during **activity D** of Figure 1, by selecting specific features or a range of features in the PC, they are incorporated in the form of configurations. Hence the challenge is based on the original FM, we should generate a new sub-FM that narrows the solution space based on the domain expert preferences.

To produce the new sub-FM (**Activity E** of figure 1) we apply the following process dimension by dimension:

1. Determine the propositional formula representing the DE's preferences for the selected dimension.
2. Remove features that do not appear in the formula.
3. Translate, if possible, AND ($\wedge$) groups into equivalent constraints ($\iff$) (useful only for valid configurations).

To find the propositional formula of the DE's preferences, the different choices are linked with OR ($\vee$), and the combinations of base features, which are part of the **same configuration**, are presented with AND ($\wedge$) relationship. Once we get the formula, and remove all features that do not figure on it, we have to translate, potentially, AND ($\wedge$) occurrences into equivalent($\iff$) constraints. To do so we distinguish two cases, if all features of the AND ($\wedge$) relation do not appear anywhere on the formula, we apply the equivalent($\iff$) constraint to them two by two, otherwise we apply the same logic only to those that appear uniquely pairwise.

For example, let's consider an OR group composed of features A, B, C as presented in figure 4. The equivalent dimension will have all possible combinations of A, B, C (8 values). The combination of base characteristics is presented with AND ($\wedge$), and the different configurations (preferences) per dimension are linked with OR ($\vee$). The DE can select different configurations for this dimension, like:

1. If A, B, C are selected, then preference = A $\vee$ B $\vee$ C
2. If AB is selected, then preference = A $\wedge$ B
3. If AB, C are selected then preference = (A $\wedge$ B) $\vee$ C
4. If AB, B are selected then preference = (A $\wedge$ B) $\vee$ B

For the first example, no feature will be removed (case 1 of figure 4), in the second, C is removed, and the A $\iff$ B constraint is added (case 2 of figure 4), for the third example, no feature is removed, but the A $\iff$ B constraint is added, even if we will keep the same group structure, the added constraint does indeed reduce the search space provided that the resulting configurations are valid (case 3 of figure 4), and for the last example C is removed, but no constraint is added since B appears twice in the preference (case 4 of figure 4).

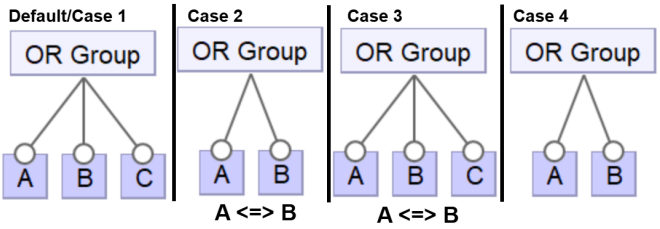

Figure 4 Refined Variability Models

Note that selecting "AB" (together) means you're considering one single configuration where both A and B are selected simultaneously. In contrast, writing "A, B" (separated by a comma) means you're referring to two separate configurations: one involving A and another involving B independently.

4. Implementation

To translate the conceptual pipeline into practice, we integrated three main tools. **Acapulco** (Martinez et al. 2022) is used for variability space discovery, **Greal** (Foures et al. 2023) ensures model derivation, and **PragmaDev** was used for model evaluation. The base model provided by Airbus for our implemented pipeline was specified in a BPMN format.

Acapulco (Martinez et al. 2022) is a search-based optimization tool designed for feature models and Software Product Lines (SPLs). It addresses the challenge of finding optimal configurations within variability-rich systems. Compared to other existing tools Acapulco has the advantage of generating valid configurations during the optimization process while being efficient. This result is achieved by implementing the Indicator-Based Evolutionary Algorithm (IBEA) (Henard et al. 2015) enhanced with Consistency-Preserving Configuration Operators (CPCOs) (Horcas et al. 2023). Built on FeatureIDE, Acapulco is user-friendly and easily extensible, allowing customization for domain-specific objectives.

Using the **extensibility** of Acapulco, we customized our implementation for **Activities A and B** of Figure 1. The configuration step was extended to incorporate the necessary artifacts for model derivation (Base model and Variability Realization Model). The **B Activity** representing the evaluation step of the EA, was implemented by the integration of Greal and PragmaDev. To evaluate the generated Resolution models, encompassing business process configurations, Greal (Foures et al. 2023) derives specific products from the base model by providing a declarative realization language that can be combined with various base meta-models to create domain-specific realization languages. Additionally, Greal includes a product derivation

algorithm that applies the realization model to a base model, producing a derived product. To return the fitness score of this evaluation process, the derived BPMN models are evaluated through PragmaDev, a simulation tool used to evaluate the derived models. The evaluation method returns the simulation result quantitatively comparing the derived models.

Once the Variability space discovery process is finished, its utility resides in providing **decision-making support**. Hence, we have implemented the strategy detailed in section 3.2 for visually representing the discovery results during **Activity C**. The dimensions of the generated Parallel Coordinate plot represent the feature model's abstraction, and the last dimension reflects possible values of the fitness function. The polylines represent the different discovered configurations and their fitness score. Then as specified in Figure 1, the **D Activity** consists of analyzing discovered configurations represented within the parallel coordinate plot by the DE. If satisfied with the results, the process concludes. However, if the domain expert desires further refinement or exploration of the solution space, they can directly specify their preferences within the parallel coordinate, either as ranges or precise points. Based on these specified preferences, a new sub-feature model is generated **(Activity E)**, resulting in a reduced and refined variability space. The optimization process then restarts, iteratively refining the solution space until the DE ends the exploration.

5. Evaluation

The evaluation focuses on three dimensions. The first two, assessed qualitatively, examine the system's ability to (1) represent the variability space using a parallel coordinate and (2) incorporate a feedback loop that enables users to guide the exploration of the variability space by identifying areas of interest. The third dimension evaluates the system's capability to support **interactive** exploration of the variability space by domain experts.

To assess our implementation, we conducted a case study based on a subset of Airbus's program development plans. These plans encapsulate the company's expertise in designing new aircraft and modeling industrial alternatives for manufacturing various aircraft components. This knowledge results from years of modeling and systems engineering research, allowing program architects to analyze design and manufacturing alternatives for new aircraft. For confidentiality reasons, we focused on an excerpt from this business product line. The corresponding feature model is shown in Figure 5. Additionally, we use a set of Business Process Modeling Notation (BPMN) models to represent the company's workflows (base model). A Variability Realization Model (VRM) was employed to map the variability model (VM) to the base model, facilitating the derivation of customized business processes based on specific feature selections.

The evaluation was performed on a **population size of 100**, we have considered only a single objective optimization for simplicity, the goal of the optimization was to minimize the lead time of aircraft fabrication.

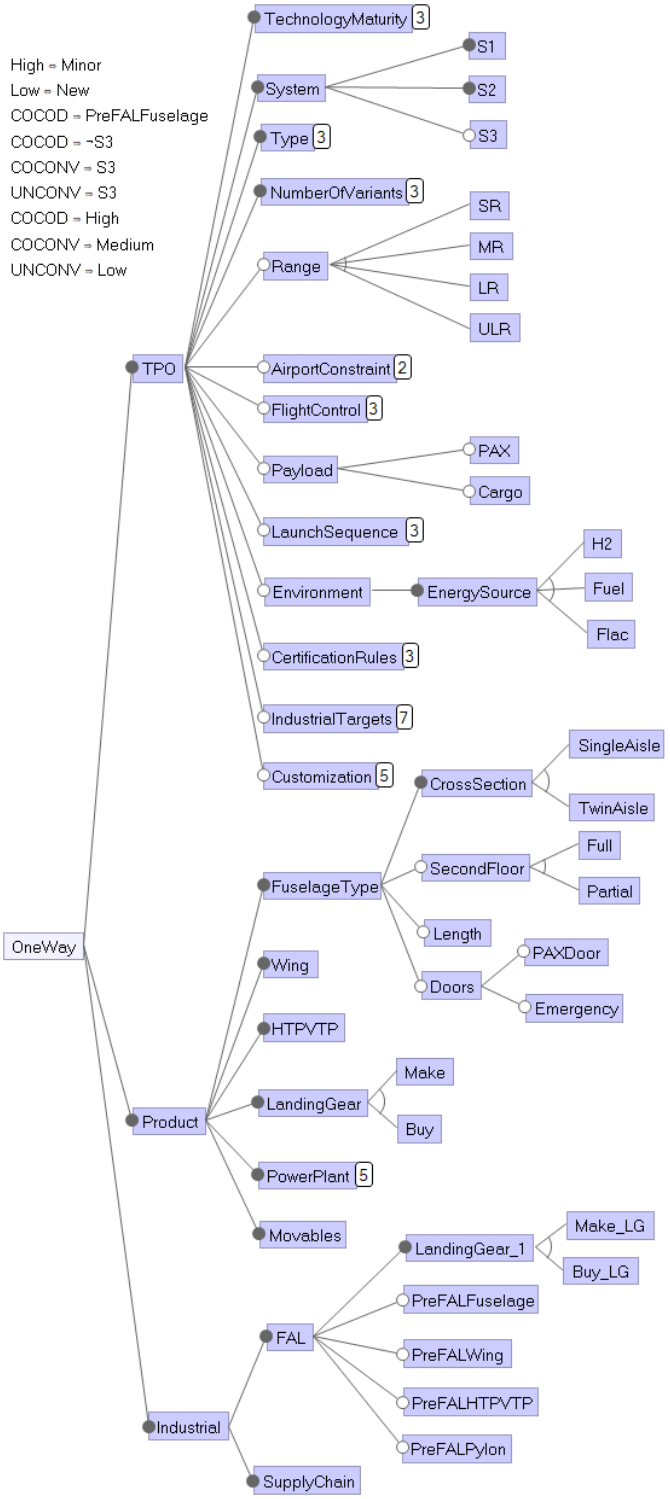

Figure 5 Simplified excerpt of a feature model for program development plan

5.1. Representing the variability model using parallel coordinate

We compared different strategies for representing the parallel coordinate before sticking to the one presented in section 3.2. The strategies are summarized as follows:

– **1 feature/dimension :** This approach is a binary parallel coordinate, the dimensions are the optional leaves, and the values are 1 for selected and 0 for unselected features.
  - **Pros :** Straightforward representation.
  - **Cons :** Number of columns(dimensions) explodes for large feature models.

– **2 features/dimension :** We aimed to reduce dimensions by focusing on DVPs with two leaves. Dimensions would represent DVPs, and values would be their possible leaf combinations with OR or XOR links. For DVPs with more than two leaves, we grouped them pairwise, creating new dimensions.
  - **Pros :** Significant dimension reduction compared with the first suggestion.
  - **Cons :** Ineffective for DVPs with more than two leaves due to the introduction of intermediate dimensions not present in the base feature model.

– **n features/dimension :** This is the more generic approach, it is a generalization of the previous ones. The dimensions are the DVPs, and values would be their possible leaf combinations.
  - **Pros :** Directly represents the FM structure without intermediate dimensions.
  - **Cons :** The number of values per dimension explodes with the rise of optional leaves or DVPs.

The presented strategies do not only show the different manners of abstracting the variability space incorporated in the feature model but also the different levels of abstractions towards the same variability space.

In the context of the Airbus case study, Figure 6 shows a zoomed-in version of the parallel coordinate generated by the first variability space discovery. The dimensions, representing the DVPs identified in Airbus FM of Figure 5, are the different points where the DEs should/can specify their choices , and the last dimension presents the lead time, enabling DE to evaluate the impact of each choice(a feature selection/deselection) or a configuration(a set of choices) on the lead time. This abstraction of the FM in the form of a PC allows, as a result, DEs to focus only on key features(decisions) that have an impact on their business without the need to understand all the complexity of the variability space presented in the FM.

5.2. A human in the loop to drive the space exploration

After the presentation of the results, a human, and notably the domain expert, can conduct a more in-depth analysis to gain a comprehensive understanding of the diverse potential configurations of the business process. This may involve optimizing lead time, evaluating trade-offs between competing

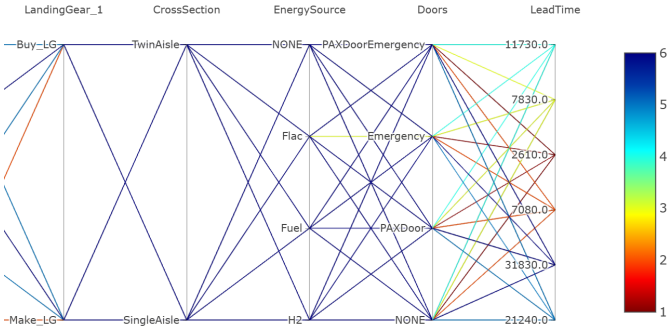

Figure 6 Zoomed in parallel coordinate of the 1st discovery of the variability space

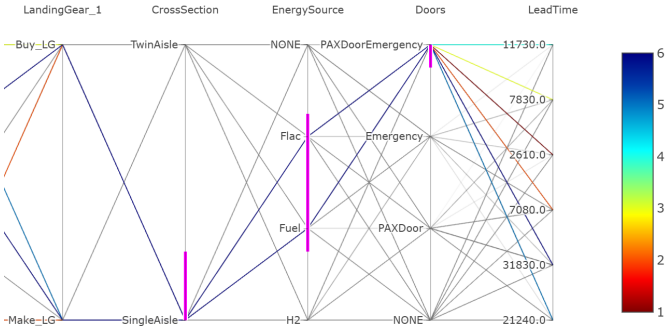

Figure 7 DE's preferences within the PC

decisions, or identifying configurations that align with some strategic goals.To facilitate this exploration, a human can interact with the parallel coordinate, specifying their preferences in the form of ranges or specific points. By refining their preferences, the search space can be narrowed, focusing on regions of interest. This iterative process of exploration and refinement allows anyone to make informed decisions and identify optimal configurations.

In the example of Figure 7, the DE decided to focus only on options where the Energy Sources are of type Flac or Fuel, the Door should be Emergency and PAXDoor, and the CrossSection is SingleAsile. Once the domain expert's preferences are validated, a new feature model is generated, refining the variability space and concentrating on DE's regions of interest. Red lines of Figure 8 highlight the changes introduced in the refined FM and that distinguish it from original FM. No feature is removed from the Doors node, but the PAXDoor $\iff$ Emergency constraint is added, and for other nodes, features that weren't selected on the PC were removed. The remaining features of the original FM remain unchanged, subsequently aren't shown again here. Thus, as shown in Figure 8, based on the domain expert's preferences, representing the role of humans in the loop, a sub-FM was generated, driving the search for upcoming iterations towards their ROI.

5.3. An interactive assistant for domain experts

To preserve the interactive nature of the relationship between a digital system and a human user, it is important to consider three

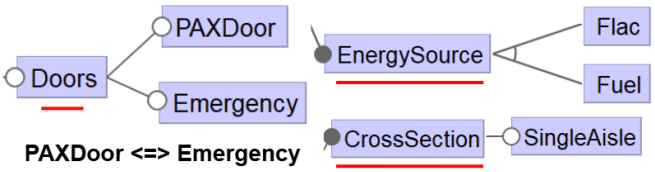

Figure 8 Changes on the refined FM

thresholds—determined by human perceptual abilities—when optimizing application performance. These thresholds, which have been consistent for over thirty years (Card et al. 1991; R. B. Miller 1968), provide guidance on acceptable response times:

- 0.1 second: This is the upper limit for the user to perceive the system as reacting instantaneously, eliminating the need for special feedback other than displaying the result.
- 1.0 second: This is the threshold for maintaining the user's flow of thought. While users will notice the delay, they can still feel engaged. No special feedback is typically needed for delays between 0.1 and 1.0 seconds, though the sense of directly interacting with the data diminishes.
- 10 seconds: This marks the upper limit for keeping the user's attention focused on the interaction. For delays exceeding this threshold, users are likely to shift their attention to other tasks. In such cases, feedback indicating the expected completion time is essential. If response times are highly variable, continuous feedback becomes even more critical to manage user expectations.

For situations where immediate responses are not feasible, continuous feedback, such as a progress indicator, should be provided to inform users of the status of the system and the estimated completion time (Myers 1985).

For our system, we evaluate the time taken to explore the variability space, the reactivity of the parallel coordinates and the time to produce a refined variability model when the domain expert refines areas of interest. It should be noted that the time taken to derive the model is linked to the complexity of the VRM, and similarly the time taken to simulate the BPMN model obtained is also bound to the complexity of the BPMN models (resources), but the idea of applying it to the simplified case of Airbus's OneWay project allows us to have an evaluation on a realistic model.

We conduct a performance analysis, examining the derivation time, simulation time, and execution time for the discovered configurations. We conduct all these experiments on a laptop with 32GB of RAM and a CPU: i5-12500H (12 cores, 16 threads, base speed 2.5GHz). All programs were written in JAVA and executed using a JRE 17.0.10 on a Windows 11 OS.

The simulation time, which represents the time required to simulate the derived BPMN diagrams, remained relatively constant throughout the experiment, owing to the simplicity of the base models. In contrast, the derivation time, which encompasses the time required to generate new BPMN models based on the configurations of the feature model, and the execution time, which represents the combined duration of derivation and simulation, increased linearly.

5.3.1. Results

We do several experiments in varying the size of the population (20, 30, 40, 50, 100).

For exploration with a population equal to 20, the whole discovery took **5min**, **161 configurations** were discovered.

For exploration with a population equal to 30, the whole discovery took **6min**, **201 configurations** were discovered.

For exploration with a population equal to 40, the whole discovery took **16min**, **373 configurations** were discovered.

For exploration with a population equal to 50, the whole discovery took **33min**, **670 configurations** were discovered.

For exploration with a population equal to 100, the whole discovery took **2hrs 17min**, **1431 configurations** were discovered.

We got an average of **0.98 sec** for each BPMN simulation and an average of **2.12** seconds for the derivation process.

Loading the parallel coordinate, selecting a feature within the parallel coordinate, and generating a new variability model based on DE regions of interest also takes less than 100ms.

Based on this configuration, we can assess that the *decision-making support* steps are interactive. The variability discovery steps could not be considered interactive. Although the entire experiment lasted 2 hours and 17 minutes in the worst case (population = 100), most of the initial 600 configurations were processed in under 3 seconds each, demonstrating significant efficiency compared to human capabilities. Thus, when considering the large number of configurations explored, the diverse BPMN files simulated, and the interactive capabilities provided by the parallel coordinate visualization, the proposed solution remains a valuable tool for domain experts, enabling them to gain deeper insights into various decision possibilities and make informed choices in less time than they would normally need.

5.4. Limitations

The current implementation and evaluation suffer from certain limitations and biases. In the implementation, no parallelism has been employed to leverage natural speedups during the exploration phase. Incorporating parallelism could allow the derivation and evaluation of each discovered solution to occur in separate processes or threads, enhancing efficiency. Additionally, no caching mechanism has been implemented to accelerate the derivation or simulation processes. For instance, it would be possible to identify derivation models that merely add rules to previously executed derivations, potentially enabling incremental evaluation of the generated BPMN processes.

Although the evaluation model is derived from an industrial case study and offers practical relevance, it may not fully represent the complexity of all possible derivations or BPMN models. Consequently, the observed derivation and simulation costs, which are relatively stable with a low standard deviation in this study, are likely specific to our case study and should not be generalized to other contexts.

6. Related work

Interactive optimization methods are especially well suited for helping human domain experts understand complex problems while allowing a computer to learn about their preferences to generate relevant solutions.

This challenge involves three primary types of work that aim to assist domain experts in navigating complex problems characterized by many inputs and requiring optimization across one or more criteria:

1. Capturing Input Variability: The first objective is to effectively model the input space, including their constraints and possible values. This involves techniques from the product line community(Rabiser et al. 2010; Berger et al. 2013), which are commonly used by companies to capture the variability and constraints associated with various parameters of a product.

2. Guided Exploration Based on User Preferences: The second objective focuses on guiding an automated exploration process to identify regions of interest based on user-defined criteria. For instance, Cruz-Reyes *et al*. (Cruz-Reyes et al. 2017) propose a method for incorporating implicit preferences in multi-objective evolutionary optimization, creating a selection bias towards specific Regions of Interest (ROIs) rather than the entire Pareto front.

3. Interactive Representation for Exploration: The third objective is to create interactive tools that allow domain experts to explore configuration spaces, understand the impacts of different choices on system output, and specify their preferences. Some approaches employ interactive visualizations, such as parallel coordinate plots, to help domain experts explore variability spaces and understand the effects of specific choices on various outputs. Examples include the PARASOL library (Raseman et al. 2019), which supports the development of web applications for multi-objective decision-making, and the SAGESSE (Cajot et al. 2019) approach.

Our work builds on these methods by integrating aspects from each approach: we retain a feature model-based notation to represent variability, incorporate an automated mechanism to explore configuration spaces and analyze the impact on business processes, and offer an interactive view for domain experts to assess how specific choices affect process-related resources (such as total lead time or other resource consumption). In our iterative approach, users receive a simplified feature diagram that reflects the remaining variability space after setting certain configuration parameters, enabling them to explore areas of interest in greater depth.

7. Conclusion and Future work

In this study, we proposed a method that leverages feature models, evolutionary algorithms, and orthogonal variability management to aid domain experts in exploring the variability space of business processes. By discovering and visually representing possible configurations, our approach addresses the complexity inherent in decision-making, especially in environments with a vast array of choices. The introduction of interactive parallel coordinates allowed domain experts to interact with high-dimensional data, representing different possible choices for their business processes, and incorporate their preferences to discover configurations that align with their interests. Consequently, the understanding of trade-offs, decision impacts, and alternatives becomes more easy. We validated our solution using the case study provided by Airbus in the OneWay project, demonstrating the method's effectiveness in discovering diverse decision outcomes while offering insights into potential configurations. The implementation of evolutionary algorithms combined with domain experts' input suggests our solution's promising potential in supporting complex decision-making processes. Building on the positive feedback received from Airbus' domain experts, we recognize the importance of further evaluating the comprehensibility and usability of our generated visualizations. Future work will expand user studies to include a wider range of domain experts, allowing us to more comprehensively assess how effectively these visualizations support decision-making and facilitate intuitive exploration of the variability space.

About the authors

El Hayani Haitam is a PhD student at the University of Rennes, supervised by Dr. Benoit Combemale and Dr. Olivier Barais. His research focuses on the intersection of Cloud computing, DevOps methodologies, and Infrastructure as Code (IaC). You can contact the author at haitam.el-hayani@inria.fr.

Benoit Combemale is a Research Director at Inria and a Full Professor of Software Engineering at the University of Rennes. His research interests in Software Engineering include Software Language Engineering, Model-Driven Engineering, and Software Validation & Verification. You can contact the author at benoit.combemale@inria.fr or visit http://combemale.fr.

Olivier Barais is a full professor of software engineering at the University of Rennes 1 and heads the INRIA DiverSE team in Rennes. With a career dedicated to advancing the field of software engineering, Dr. Barais has made significant contributions to model-driven engineering, software architecture, adaptive systems and open-source software supply chain security. In his role at INRIA, Dr. Barais leads several high-impact research projects and collaborates with industry partners to bring cutting-edge solutions to real-world challenges. Outside of his professional endeavors, he is passionate about mentoring students and fostering collaborative environments that encourage innovation and excellence in software engineering. You can contact the author at olivier.barais@irisa.fr or visit https://olivier.barais.fr.

Steffen Zschaler is a Reader in Software Engineering at King's College London. His research interests are in model-driven engineering, focussing on modularity, optimisation, and the principled engineering of simulations. He is the director of

MDENet, the expert network in model-driven engineering, co-director of the Centre for Doctoral Training on Digital Twins for Healthcare, and theme lead for digital twins in the Centre for Doctoral Training on Data-Driven Healthcare. You can contact the author at steffen.zschaler@kcl.ac.uk or visit www.steffen-zschaler.de.

Acknowledgments

The authors gratefully acknowledge Pragmadev for providing a free license to their tool, offering essential simulation capabilities for evaluating the BPMN models.

References

Abi Akle, A., Minel, S., & Yannou, B. (2017, January). Information visualization for selection in Design by Shopping. *Research in Engineering Design*, *28*(1), 99-117. Retrieved from https://hal.science/hal-01438790 doi: 10.1007/s00163-016-0235-2

Alain Petrowski, S. B. H. (2017). *Evolutionary algorithms.* John Wiley & Sons, Ltd. (https://hal.science/hal-02091413)

Bachmann, C., Felix, & Paul. (2005). *Variability in software product lines* (Tech. Rep. No. CMU/SEI-2005-TR-012). Carnegie Mellon University, Software Engineering Institute's Digital Library. Retrieved from https://insights.sei.cmu.edu/library/variability-in-software-product-lines/ (Accessed: 2024-Nov-4)

Benavides, D., Trinidad, P., & Ruiz-Cortés, A. (2005). Automated reasoning on feature models. In *Proceedings of the 17th international conference on advanced information systems engineering* (p. 491–503). Berlin, Heidelberg: Springer-Verlag. Retrieved from https://doi.org/10.1007/11431855_34 doi: 10.1007/11431855_34

Berger, T., Rublack, R., Nair, D., Atlee, J. M., Becker, M., Czarnecki, K., & Wąsowski, A. (2013). A survey of variability modeling in industrial practice. In *Proceedings of the 7th international workshop on variability modelling of software-intensive systems.* New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/2430502.2430513 doi: 10.1145/2430502.2430513

Beyer, H.-G. (2001). *The theory of evolution strategies*. Springer Berlin, Heidelberg. Retrieved from https://link.springer.com/book/10.1007/978-3-662-04378-3 doi: 10.1007/978-3-662-04378-3

Bosch, J., Florijn, G., Greefhorst, D., Kuusela, J., Obbink, H., & Pohl, K. (2001, 10). Variability issues in software product lines. In *Revised papers from the 4th international workshop on software product-family engineering* (p. 13-21). doi: 10.1007/3-540-47833-7_3

Branke, J., Deb, K., Miettinen, K., & Słowiński, R. (2008). *Multiobjective optimization*. Springer Berlin, Heidelberg. Retrieved from https://link.springer.com/book/10.1007/978-3-540-88908-3 doi: 10.1007/978-3-540-88908-3

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press. Retrieved from https://doi.org/10.1093/oso/9780195099713.001.0001 doi: 10.1093/oso/9780195099713.001.0001

Cajot, S., Schüler, N., Peter, M., Koch, A., & Maréchal, F. (2019). Interactive optimization with parallel coordinates: Exploring multidimensional spaces for decision support. *Frontiers in ICT*, *5*. Retrieved from https://www.frontiersin.org/journals/ict/articles/10.3389/fict.2018.00032 doi: 10.3389/fict.2018.00032

Card, S. K., Robertson, G. G., & Mackinlay, J. D. (1991). The information visualizer, an information workspace. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 181–186).

Chen, L., Ali Babar, M., & Ali, N. (2009). Variability management in software product lines: a systematic review. In *Proceedings of the 13th international software product line conference* (p. 81–90). USA: Carnegie Mellon University.

Cruz-Reyes, L., Fernandez, E., Sanchez, P., Coello Coello, C. A., & Gomez, C. (2017). Incorporation of implicit decision-maker preferences in multi-objective evolutionary optimization using a multi-criteria classification method. *Applied Soft Computing*, *50*, 48-57. Retrieved from https://www.sciencedirect.com/science/article/pii/S1568494616305610 doi: https://doi.org/10.1016/j.asoc.2016.10.037

Czarnecki, K., Grünbacher, P., Rabiser, R., Schmid, K., & Wąsowski, A. (2012). Cool features and tough decisions: a comparison of variability modeling approaches. In *Proceedings of the 6th international workshop on variability modeling of software-intensive systems* (p. 173–182). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/2110147.2110167 doi: 10.1145/2110147.2110167

Czarnecki, K., Helsen, S., & Eisenecker, U. (2005). Formalizing cardinality-based feature models and their specialization,. *Software Process: Improvement and Practice,*, *10*. (https://www.researchgate.net/publication/220542223_Formalizing_cardinality-based_feature_models_and_their_specialization) doi: 10.1002/spip.213

David Benavides, A. R.-C., Sergio Segura. (2010). Automated analysis of feature models 20 years later: A literature review,. *Information Systems,*, *35*. (https://www.sciencedirect.com/science/article/pii/S0306437910000025) doi: 10.1016/j.is.2010.01.001

Deb, K. (2007). Current trends in evolutionary multiobjective optimization. *Int. J. Simul. Multidisci. Des. Optim.*, *1*. (https://www.ijsmdo.org/articles/smdo/abs/2007/01/smdo0703/smdo0703.html)

Fernandez, E., Lopez, E., Lopez, F., & Coello, C. A. (2011). Increasing selective pressure towards the best compromise in evolutionary multiobjective optimization: The extended nosga method. *Information Sciences*, *181*. doi: 10.1016/j.ins.2010.09.007

Fleming, P. J., Purshouse, R. C., & Lygoe, R. J. (2005). Many-objective optimization: an engineering design perspective. In *Proceedings of the third international conference on evolutionary multi-criterion optimization* (p. 14–32). Berlin, Heidelberg: Springer-Verlag. Retrieved from https://doi.org/

10.1007/978-3-540-31880-4_2 doi: 10.1007/978-3-540-31880-4_2

Fogel, L. J., Owens, A. J., & Walsh, M. J. (1966). *Artificial intelligence through simulated evolution*. John Wiley and Sons Inc. Retrieved from https://api.semanticscholar.org/CorpusID:262309796

Foures, D., Acher, M., Barais, O., Combemale, B., Jézéquel, J.-M., & Kienzle, J. (2023, October). Experience in Specializing a Generic Realization Language for SPL Engineering at Airbus. In *MODELS 2023 - 26th International Conference on Model-Driven Engineering Languages and Systems* (p. 1-12). Västerås, Sweden: IEEE. Retrieved from https://inria.hal.science/hal-04216627

Halin, A., Nuttinck, A., Acher, M., Devroey, X., Perrouin, G., & Heymans, P. (2017, February). Yo Variability! JHipster: A Playground for Web-Apps Analyses. In *11th International Workshop on Variability Modelling of Software-intensive Systems* (p. 44 - 51). Eindhoven, Netherlands. Retrieved from https://inria.hal.science/hal-01468084 doi: 10.1145/3023956.3023963

Haugen, O., Moller-Pedersen, B., Oldevik, J., Olsen, G. K., & Svendsen, A. (2008). Adding standardized variability to domain specific languages. In *2008 12th international software product line conference* (p. 139-148). doi: 10.1109/SPLC.2008.25

Haugen, O., Wąsowski, A., & Czarnecki, K. (2012). Cvl: common variability language. In *Proceedings of the 16th international software product line conference - volume 2* (p. 266–267). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/2364412.2364462 doi: 10.1145/2364412.2364462

Heinrich, J., & Weiskopf, D. (2013). State of the art of parallel coordinates. In *Eurographics*. Retrieved from https://api.semanticscholar.org/CorpusID:6629981

Henard, C., Papadakis, M., Harman, M., & Le Traon, Y. (2015). Combining multi-objective search and constraint solving for configuring large software product lines. In *2015 ieee/acm 37th ieee international conference on software engineering* (Vol. 1, p. 517-528). doi: 10.1109/ICSE.2015.69

Holland, J. H. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

Horcas, J.-M., Strüber, D., Burdusel, A., Martinez, J., & Zschaler, S. (2023). We're not gonna break it! consistency-preserving operators for efficient product line configuration. *IEEE Transactions on Software Engineering*, *49*(3), 1102-1117. doi: 10.1109/TSE.2022.3171404

Hwang, C.-L., & Masud, A. S. M. (1979). *Multiple objective decision making — methods and applications*. Springer Berlin, Heidelberg. Retrieved from https://link.springer.com/book/10.1007/978-3-642-45511-7 doi: 10.1007/978-3-642-45511-7

Inselberg, A. (2009). *Parallel coordinates: Visual multidimensional geometry and its applications*. Springer-Verlag, New York. Retrieved from https://link.springer.com/book/10.1007/978-0-387-68628-8 doi: 10.1007/978-0-387-68628-8

Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., & Peterson, A. S. (1990). *Feature-oriented domain analysis (foda) feasibility study*. Technical Report. Carnegie-Mellon University Software Engineering Institute. (https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=34c2c089ca2f921d90faba064ece3a15291f31b1)

Knüppel, A., Thüm, T., Mennicke, S., Meinicke, J., & Schaefer, I. (2017, 08). Is there a mismatch between real-world feature models and product-line research? In (p. 291-302). doi: 10.1145/3106237.3106252

Kok, M. (1986). The interface with decision makers and some experimental results in interactive multiple objective programming methods. *European Journal of Operational Research*, *26*(1), 96-107. Retrieved from https://www.sciencedirect.com/science/article/pii/0377221786901621 (Second EURO Summer Institute) doi: https://doi.org/10.1016/0377-2217(86)90162-1

Martinez, J., Strüber, D., Horcas, J. M., Burdusel, A., & Zschaler, S. (2022). Acapulco: an extensible tool for identifying optimal and consistent feature model configurations. In *Proceedings of the 26th acm international systems and software product line conference - volume b* (p. 50–53). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3503229.3547067

Meignan, D., Knust, S., Frayret, J.-M., Pesant, G., & Gaud, N. (2015, 10). A review and taxonomy of interactive optimization methods in operations research. *Transactions on Interactive Intelligent Systems*, *5*, 1-43. doi: 10.1145/2808234

Miettinen, K. (1998). *Nonlinear multiobjective optimization*. Springer New York, NY. Retrieved from https://link.springer.com/book/10.1007/978-1-4615-5563-6

Miller, G. A. (2010). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review,*, *63*. (https://www.sciencedirect.com/science/article/pii/S0306437910000025) doi: 10.1016/j.is.2010.01.001

Miller, R. B. (1968). Response time in man-computer conversational transactions. In *Proceedings of the december 9-11, 1968, fall joint computer conference, part i* (pp. 267–277).

Mitchell, M. (1996). *An introduction to genetic algorithms*. Cambridge, MA: MIT Press.

Myers, B. A. (1985). The importance of percent-done progress indicators for computer-human interfaces. *ACM SIGCHI Bulletin*, *16*(4), 11–17.

Noir, J. L., Madelénat, S., Gailliard, G., Labreuche, C., Acher, M., Barais, O., & Constant, O. (2016). A decision-making process for exploring architectural variants in systems engineering. In H. Mei (Ed.), *Proceedings of the 20th international systems and software product line conference, SPLC 2016, beijing, china, september 16-23, 2016* (pp. 277–286). ACM. Retrieved from https://doi.org/10.1145/2934466.2946045 doi: 10.1145/2934466.2946045

Packham, I., Rafiq, M., Borthwick, M., & Denham, S. (2005). Interactive visualisation for decision support and evaluation of robustness—in theory and in practice. *Advanced Engineering Informatics*, *19*(4), 263-280. Retrieved from https://www.sciencedirect.com/science/article/pii/S1474034605000613 (Computing in Civil Engineering) doi: https://doi.org/10.1016/j.aei.2005.07.006

Piemonti, A. D., Macuga, K. L., & Babbar-Sebens, M. (2017). Usability evaluation of an interactive decision support system for user-guided design of scenarios of watershed conservation practices. *Journal of Hydroinformatics*, *19*, 701-718. Retrieved from https://api.semanticscholar.org/CorpusID:64725671

Rabiser, R., Grünbacher, P., & Dhungana, D. (2010). Requirements for product derivation support: Results from a systematic literature review and an expert survey. *Information and Software Technology*, *52*(3), 324–346.

Raseman, W. J., Jacobson, J., & Kasprzyk, J. R. (2019). Parasol: an open source, interactive parallel coordinates library for multi-objective decision making. *Environmental Modelling & Software*, *116*, 153-163. Retrieved from https://www.sciencedirect.com/science/article/pii/S1364815219301148 doi: https://doi.org/10.1016/j.envsoft.2019.03.005

Schmid, K., & John, I. (2004). A customizable approach to full lifecycle variability management. *Science of Computer Programming*, *53*(3), 259-284. Retrieved from https://www.sciencedirect.com/science/article/pii/S0167642304000929 (Software Variability Management) doi: https://doi.org/10.1016/j.scico.2003.04.002

S.F. Adra, P. F., I. Griffin. (2007). A comparative study of progressivepreference articulation techniques for multiobjective optimisation, evolutionary multi-criterion optimization,. *Springer, Berlin, Heidelberg*, *4403*. (https://link.springer.com/chapter/10.1007/978-3-540-70928-2_67) doi: 10.1007/978-3-540-70928-2_67

Shenfield, A., Fleming, P. J., & Alkarouri, M. (2007). Computational steering of a multi-objective evolutionary algorithm for engineering design. *Engineering Applications of Artificial Intelligence*, *20*(8), 1047-1057. Retrieved from https://www.sciencedirect.com/science/article/pii/S0952197607000085 doi: https://doi.org/10.1016/j.engappai.2007.01.005

Ziadi T., J. J. (2006). *Software product line engineering with the uml: Deriving products.* Springer, Berlin, Heidelberg. (https://link.springer.com/chapter/10.1007/978-3-540-33253-4_15) doi: 10.1007/978-3-540-33253-4_15