Evaluation Across Multiple Views for Variable Automation Systems

Lothar Hotz, Yibo Wang, Matthias Riebisch
Department of Informatics
University of Hamburg
wang@informatik.uni-hamburg.de

Olaf Götz, Josef Lackhove
Lenze Automation GmbH
olaf.goetz@lenze.com

ABSTRACT

Automation systems in industry are often software-intensive systems consisting of software and hardware components. During their development several engineers of different disciplines are involved, such as mechanical, electrical and software engineering. Each engineer focuses on specific system aspects to be developed. To enable an efficient development, product lines especially with feature models for variability modeling are promising technologies. In order to reduce the complexity of both feature models and development process, views on feature models can be applied. The use of views for filtering purposes constitutes an established method. However, views also enable further options missing in current approaches, such as evaluations regarding requirements, including non-functional ones. This paper presents an approach for evaluation across multiple views to enable collaborative development for developers who focus on different system aspects. We validate our approach by applying it in an industrial project for the planning of flying saws.

Categories and Subject Descriptors

D.2.13 [**Software Engineering**]: Reusable Software—*Domain engineering, reusable models*

Keywords

Product lines, Automation systems, Feature model, Configuration, Consistency check, Multi-criteria evaluation

1. INTRODUCTION

In many engineering fields, more and more emphasis is put on product line technology due to the need for customization with low efforts and in short terms. Planned variability allows a smart adaptation of products and services. However, the increasing complexity of both the development processes and the artifacts due to an introduction of variability requires modeling as a means to master this complexity. Models are essential for many engineering disciplines, however, their linkage with variability and their integration into product line processes requires special efforts. Furthermore, developed systems for products and services have to fulfill growing non-functional requirements, such as safety or cost, due to the increased competition on the markets. A continuous evaluation of the development processes even in early phases has to be performed to reduce risks. Modeling facilitates a prediction of properties early in a development process. A prediction of global system properties calls for a consideration of several engineering disciplines. The problem is, different models have been established in these disciplines. For evaluation purposes, these models have to be integrated into one holistic model with several views. Furthermore, variability has to be accounted to each view.

In this paper, we present an approach, in which multiple views have been integrated along with feature models. The approach has been developed primarily for the production automation domain, with a consideration of the relevant views. Customization is performed here as a configuration of pre-fabricated building blocks. These configurations have to be evaluated for validity, for which constraints are used as reasoning technology (see Section 3). This paper concentrates on evaluations in multiple views (Section 4), other aspects arising in the configuration process such as resolving conflicts in invalid configurations, are not considered. Our approach has been evaluated for applicability and feasibility during an industrial project (see Section 2 and the examples throughout the sections). The paper closes with related work (Section 5) and a conclusion (Section 6).

2. RUNNING EXAMPLE

In order to illustrate our approach, we use a running example of a so-called flying saw: A slide with the flying saw is synchronized with the material's speed [5], as shown in Figure 1. Once the material is cut completely, it returns to the starting position as soon as possible to get ready for the next cut. Variability of the flying saw exists for example in its control architecture (centralized or decentralized) and cutting methods (cutting controlled by length or by marks on the material). In our previous work [10], we used a feature model to represent variability and defined integrity constraints between features and their attributes.

As stated above, several disciplines are involved in the development process for the flying saw, such as production, mechanical, electrical, communication, and automation engineering. There are constraints both within one discipline and between different disciplines. An example for a con-

SPLC 2015, July 20 - 24, 2015, Nashville, TN, USA
© 2015 ACM. ISBN 978-1-4503-3613-0/15/07...$15.00
DOI: http://dx.doi.org/10.1145/2791060.2791082

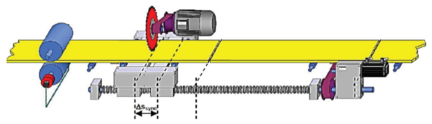


Figure 1: Running example - Flying saw

straint within the discipline production engineering is that the *motion speed* of the saw carriage is constrained by the *motion speed* of the conveyor. An example for constraints between two disciplines is that the *motion speed* of the saw carriage in the discipline production engineering is constrained by the *motor* and *transmission* in the discipline mechanical engineering.

In addition, configurers from different disciplines may have different needs and objectives during configuration. Although all are interested in system attributes such as building costs, machine weight, production performance, and future sustainability, a single configurer is concerned in particular with the properties of his own discipline, such as process accuracies (e.g. position, speed, torque etc.) and maintainable software design. Thus, the need for an evaluation approach based on constraints and criteria across different disciplines is very urgent. Furthermore, the configuration of an automation system as part of a product line requires expertise in the different disciplines, which is rarely available from a single expert. However, experts with expertise in one discipline are available, and they can cooperate according to our approach.

3. BASIC TECHNOLOGIES

In product lines, one can distinguish the development of a product line and the development of products. These engineering tasks are identified as *domain engineering* and *application engineering* [11]. Feature models are often used to express *variability* of the products. Features can be modeled with mandatory, optional, and alternative constraints, as well as attributes (*extended feature models* [1]). Furthermore, relations between features can be expressed such as *exclude* or *require* as well as *algebraic restrictions* (equations, inequalities, including computational functions such as $+$, $-$, $*$, $/$, sum, min etc.) which are all considered here as *integrity constraints* or simply *constraints* (see [6]). Constraints relate features or feature attributes. For example, the constraint $S_{cut} \geq T_{cycle\_time} * V_{conveyor}$ represents that *cut length* of the material is constrained by the *cycle time* of the cutting process and the *motion speed* of the conveyor.

Starting with the feature model including constraints a human *configurer* makes decisions about a specific product in a *product development process* (also called *configuration process* or *decision process*). A *decision* is (a) a selection of a feature, i.e., the decision of an optional or alternative feature, or (b) the selection of a value for a feature attribute, or (c) a change of a previously selected feature or feature attribute value. Through a decision, a new *partial configuration* is created with less variability than the previous partial configuration.

After one decision has been made, a *constraint processor* checks consistency and computes the impacts of the decision on other features or feature attributes. This *impact computation* comprises the addition of mandatory or required features to the partial configuration, the explicit exclusion of excluded features, as well as the computation of feature attribute values based on algebraic constraints. We expect that the impact computation is processed by a so-called *feature tool*, i.e., a configuration system [2], a feature system (such as pure::variants or Gears), or a constraint system [6]. Multiple constraints form a *constraint graph* consisting of features or feature attributes (*constraint variables*) as nodes and constraints as edges. Hence, a constraint processor gets a partial configuration as input and computes another partial configuration that follows from the constraints modeled in the feature model.

The above process is a commonly known process, see e.g. [2, 3]. In this paper, we enhance this technology by views and their evaluation. A *view* is a filter on a feature model. *Multi views* are multiple filters on a feature model. They can be overlapping but don't have to. *Dependencies* between views are modeled by constraints, which relate features or feature attributes from different views. In Figure 2, $View$ 1 and $View$ 2 are two views on a feature model. The features are restricted by *require* constraints as indicated in the figure. If the decision is made that $Feature$ 1 is selected, the constraint processor computes that $Feature$ 2 and $Feature$ 3 in $View$ 2 and $Feature$ 4 in the original $View$ 1 are part of the partial configuration, according to the *require* constraint $require(1)$, $require(2)$ and $require(3)$. We call $require(1)$, and $require(3)$ *view-crossing constraints.* All dependencies can be automatically computed through analysis of the constraint graph.

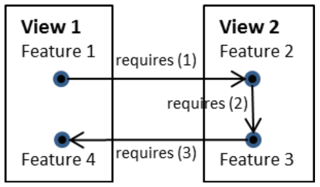


Figure 2: Dependencies of two views through view-crossing constraints in a feature model

Additionally, evaluations are related to views. An *evaluation* is specified by an *evaluation scheme* consisting of an *evaluation criterion*, an *evaluation condition*, and an *evaluation threshold*. An evaluation scheme represents a certain non-functional property (*NFP*) of a system fulfilling a specific non-functional requirement. An example is that the cost of a system (an *evaluation criterion* computed by an *evaluation function*) may not exceed (an *evaluation condition*) a specific value (an *evaluation threshold*) or that the weight of a system (another *evaluation criterion* computed by another *evaluation function*) has to be equal (another *evaluation condition*) a certain value (another *evaluation threshold*). Thus, an evaluation condition in combination with the evaluation threshold determines the condition under which the evaluation scheme holds. The evaluation condition is expressed by an arbitrary logical expression (see Section 4.3 for more details).

4. MULTI-VIEW EVALUATION

The configuration for automation system is a complex task. Lacking complete expertise, each configurer can only concentrate on his own discipline. By using views to separate the feature model, the complexity of configuration will be reduced. Furthermore, it makes configuration changes easier because of a division of responsibilities.

The goal of our approach is to support a collaborative configuration process for a system development executed by different configurers. Each configurer focuses on one view, i.e., each configurer makes decisions related to his discipline. A constraint processor checks for consistency and impacts of such decisions (see Section 3). More importantly, an evaluation based on predefined evaluation criteria is computed for each view as well as system-wide. After a configurer makes a decision, the influences on the system evaluation can directly be computed and presented to the configurer. Thus, multiple views support an *informed collaborative product development process*, which has the following characteristics:

- All configurers work towards one system development, jointly but not necessarily synchronously.
- Each configurer focuses on his discipline, i.e., on his view in the feature model, and makes decisions related to this filtered feature model. Impacts are computed by the constraint processor.
- Each configurer gets evaluations of his view.
- Each configurer gets information about influencing decisions of other configurers created by view-crossing constraints.
- Each configurer gets information about evaluations of other views on demand.
- Each configurer gets information about evaluations of the system as a whole.
- Each configurer is able to get a complete view of the current partial configuration on demand.

Thus, during a product development process each configurer is informed about the current state of the development in relation to own decisions, other configurers decisions, and the impacts of the decisions on the evaluation criteria. The following sections detail how this process can be achieved.

4.1 Activities in Domain and Application Engineering

For accomplishing such an informed collaborative product development process, the activities in domain engineering and application engineering have to be defined accordingly.

4.1.1 Activities in Domain Engineering

In domain engineering, in addition to a feature model, multiple views on the feature model have to be defined. As mentioned above, views are considered to be filters on the feature model (see Figure 3 left, and Section 4.2). In addition to the feature models, an *evaluation scheme* has to be specified (Figure 3 right). This activity includes the *establishment of the evaluation schemes for the system* as a whole, i.e., the decision which criteria are of interest for all products of the product line. Examples are cost or construction weight (see Section 2). Second, an evaluation scheme per view has to be specified (*view evaluation*) which includes the selection of the criteria of interest for the view. Third, the selection of features and feature attributes has to be specified which contribute to each criterion (see Subsection 4.3). Finally, the specification of the *system evaluation* based on these view evaluations has to be defined.

4.1.2 Activities in Application Engineering

When developing a specific product in the application engineering process, the configurers use the models developed in domain engineering (see Figure 4). During requirements acquisition, functional and non-functional properties

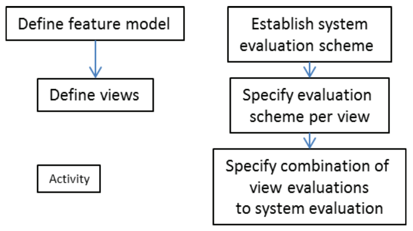

Define feature model
Establish system evaluation scheme
Define views
Specify evaluation scheme per view
Activity
Specify combination of view evaluations to system evaluation

Figure 3: Domain engineering: Feature modeling with views and evaluations

are specified, which are later used as system evaluation criteria. For example, a NFP indicates that cost of a product should not exceed a certain amount of money ($cost < m$). These requirements are mapped to features by the configurers. In an iterative process, each configurer selects features and feature attribute values in his particular view, i.e., makes decisions about the product to be developed. After every single decision, the constraint processor keeps track of consistency and impacts of the decision, which usually cross view borders (see Section 3). Besides the constraint processor, the evaluator computes the evaluation criteria of each single view and the system's overall evaluation criteria. For example, the evaluator keeps track of the current cost of the system as well as the restricting evaluation condition. The evaluator is implemented as a constraint system, similarly to the constraint processor.

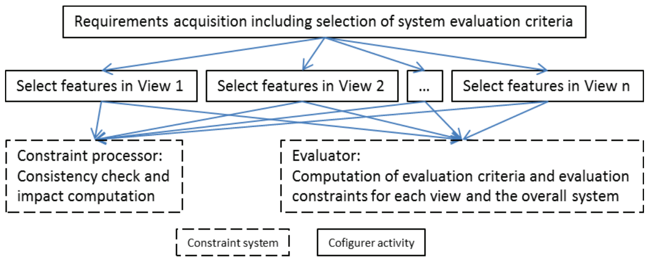

Requirements acquisition including selection of system evaluation criteria
Select features in View 1
Select features in View 2
...
Select features in View n
Constraint processor: Consistency check and impact computation
Evaluator: Computation of evaluation criteria and evaluation constraints for each view and the overall system
Constraint system
Cofigurer activity

Figure 4: Application engineering: Feature selection based on requirements and continuous tool-based computation of constraints and evaluation

4.2 Defining Feature Models and Views regarding Disciplines

In addition to commonly used extended feature models (see Section 3), we introduce a view indicator v for each feature and each feature attribute to mark it. A feature attribute *belongs to* one view (e.g. v) and a feature can belong to multiple views (e.g. v and w) (see Figure 5). Each view represents a discipline, and each feature and feature attribute which belongs to v are handled by one configurer C_v who is familiar with that discipline. If a feature belongs to v and w, the existence of the feature in the partial configuration can be decided by C_v or C_w, i.e., the decision if an optional or alternative feature has to be selected is made by C_v or C_w. Note that mandatory, required, and excluded features are computed automatically by the feature tool (*computed features*), as described in Section 3.

For feature attributes, we introduce three not necessarily disjoint attribute types: a feature attribute is (a) a *computed feature attribute* if it is computed by a constraint (e.g. the *weight* of a flying saw is the sum of weights of all its

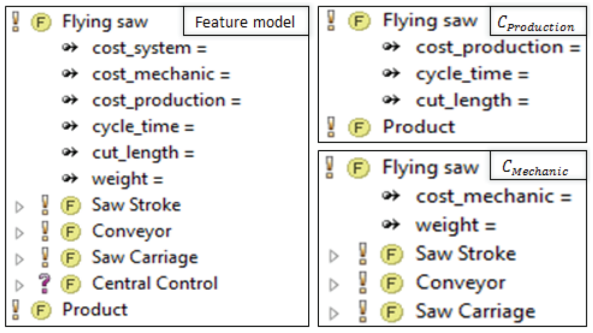


Figure 5: Example for feature modeling with views

mechanical parts), (b) an *evaluation feature attribute* (see Section 4.3) if it is used for computing an evaluation criterion (e.g. the *cost* of a flying saw), or (c) a *configurable feature attribute* if the attribute value shall be selected by a configurer[1] (e.g. the production requirements *cycle time* and *cut length*). Thus, if a configurable feature attribute belongs to a view v, the attribute's value has to be selected by C_v. See Figure 5 for an example for feature modeling with two views for $C_{Production}$ and $C_{Mechanic}$ with *cost_production* and *cost_mechanic* as evaluation feature attributes.

4.3 Evaluation Across Multiple Views

As one of our central contributions, we now introduce the evaluation schemes for the development of product lines (see Section 4.3.1). These system evaluation schemes are specified through single view evaluations (Section 4.3.2), which are then combined to system evaluations (Section 4.3.3).

4.3.1 *Establishing System Evaluation Schemes*

During domain engineering, all evaluation schemes ES for a product line are established. In our example, these are cost for the product and construction weight. As usual in domain engineering, the models, here the evaluation schemes, are pre-thought for the whole product line. However, for the evaluation schemes it is not yet specified how they are computed, this is done in the activity *single view evaluation*.

During application engineering, in particular requirements acquisition, the customer, assisted by the configurer, selects from ES a subset of evaluation schemes that is important for the product to be developed. Furthermore, the customer specifies the evaluation thresholds for the selected evaluation schemes. This is an optional specification and only needed if the configurers should be informed if the product to be developed does not fulfill a certain criterion.

4.3.2 *Single View Evaluation*

For each evaluation scheme established in the above described activity, a *single view evaluation* (see Figure 6) defines how an evaluation criterion is computed for a single view, i.e., *evaluation feature attributes* contribute to the evaluation criterion are selected. This specification is done during domain engineering by (a) selecting evaluation feature attributes of the view and (b) specifying an evaluation function of the attribute values that combines these values (*attribute evaluation function*). For example, for the cost evaluation scheme in view V, the cost feature attributes are selected that constitute the cost in V. In this case, the evaluation function is the sum of those costs. Typical evaluation functions are *sum*, *minimum*, *maximum*, *mean value*. Beside the computation of an evaluation criterion, an *evaluation condition* for the scheme has to be specified, e.g., the fact that cost should not exceed an evaluation threshold. In principle one can define here suitable power functions for a specific domain such as step functions. In total, for each established evaluation scheme in each view, the underlying feature attributes and their combination (i.e. the evaluation criterion), as well as the evaluation condition are specified in this activity. Figure 6 shows a constraint graph which represents feature attributes as constraint variables and evaluation functions as well as evaluation conditions as constraints. Such a constraint graph can be implemented with a constraint system (see Section 3).

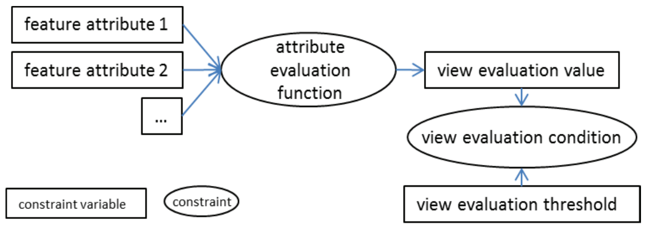


Figure 6: Constraint graph for a single view evaluation criterion

During application engineering, the evaluator computes the view evaluation and the results are presented to the configurer of the view, i.e, the computed evaluation criterion value and the checked evalution condition. Figure 8 illustrates this aspect. For each configurer $C_{Mechanic}$, $C_{Electric}$ (i.e. hardware engineers), and $C_{Automation}$(i.e. software engineers), a specific view is presented where the influence of the view-specific partial configuration on the evaluation criteria costs and weight are presented in respect to the overall system evaluation, e.g., in view Mechanic *cost_mechanic* ='115 (of 131)'.

4.3.3 *System Evaluation*

For computing a *system evaluation criterion* for an established evaluation scheme, the single view criteria are combined through a *system evaluation function* (see Figure 7). In the example, the costs of each view are combined through addition. Furthermore, a *system evaluation condition* determines if the combined value, the *system evaluation value*, holds a certain condition in respect to the *system evaluation threshold*. In Figure 8 the system evaluation criterion *weight* is fulfilled whereas the criterion *cost_system* is not.

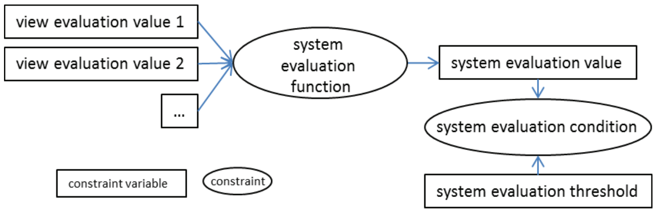
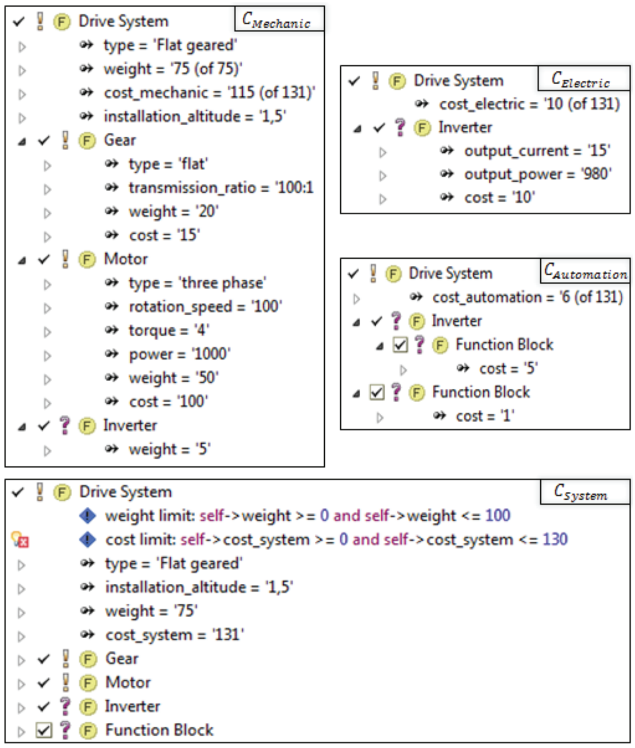


Figure 7: Constraint graph for one system evaluation criterion

[1]If a configurable feature attribute occurs in a constraint, it is also a computed feature attribute.



```
Drive System                                   C_Mechanic
    type = 'Flat geared'
    weight = '75 (of 75)'
    cost_mechanic = '115 (of 131)'
    installation_altitude = '1,5'
  Gear
    type = 'flat'
    transmission_ratio = '100:1
    weight = '20'
    cost = '15'
  Motor
    type = 'three phase'
    rotation_speed = '100'
    torque = '4'
    power = '1000'
    weight = '50'
    cost = '100'
  Inverter
    weight = '5'
```

```
Drive System                                   C_Electric
    cost_electric = '10 (of 131)
  Inverter
    output_current = '15'
    output_power = '980'
    cost = '10'
```

```
Drive System                                   C_Automation
    cost_automation = '6 (of 131)
  Inverter
    Function Block
      cost = '5'
  Function Block
    cost = '1'
```

```
Drive System                                   C_System
    weight limit: self->weight >= 0 and self->weight <= 100
    cost limit: self->cost_system >= 0 and self->cost_system <= 130
    type = 'Flat geared'
    installation_altitude = '1,5'
    weight = '75'
    cost_system = '131'
  Gear
  Motor
  Inverter
  Function Block
```

Figure 8: Example for configurers' screens. Each configurer C_x sees his view and the system view C_{System}.

5. RELATED WORK

We are not aware of any research or industrial application that focuses on multi-criteria evaluation across multiple views for product line configuration scenarios. However, views similar to our notion are used for the separation of concerns in diverse use cases. [4] proposes three alternative visualizations to generate concern-specific configuration views in feature models. [7] introduces formalization for specification of views and customized perspectives including consistency checks with respect to feature model semantics. Although constraints between different views and perspectives are well discussed in these works, the question is not answered of how to satisfy different stakeholders' objectives and preferences during the configuration process.

[9] addresses a multi-stakeholder configuration process considering stakeholders' preferences by using strategies from the social choice theory. However, the final decision is strongly influenced by the subjective opinions of stakeholders. Using the task planning technique HTN (Hierarchical Task Network), [8] proposes a framework to automatically select suitable features that satisfy both the stakeholders' functional and non-functional requirements. For design optimization, they aggregate qualitative and quantitative properties into a single object value. However, it is hard to find a proper utility function to change a multi-criteria problem to a single-criterion problem.

6. CONCLUSION AND FUTURE WORK

We presented a novel method for evaluating functional and non-functional properties of an automation system across multiple views during the configuration by different configurers. We extend the usual product line activities in domain engineering and application engineering by view evaluation and system evaluation to consider the integrity constraints and evaluation criteria for non-functional properties. The approach is partly implemented as an extension to the product line tool pure::variants. A validation was performed in a first industrial project for designing a flying saw that showed the feasibility of the approach. As future works, we plan to explore more industrial use cases and to integrate a visualization of optimization results such as the Pareto front for competing priorities (e.g. cost vs. security).

7. ACKNOWLEDGMENTS

This research has been partly funded by the German Federal Ministry of Education and Research BMBF under grant 01M3204C in the joint project "Design Methods for Automation Systems using Model Integration and Automated Variant Evaluation" and by the Hamburg Computer Science Technology Center, HITeC e.V. c/o University of Hamburg.

8. REFERENCES

[1] K. Czarnecki, S. Helsen, and U. Eisenecker. Formalizing Cardinality-based Feature Models and their Specialization. *Software Process: Improvement and Practice*, 10(1):7–29, 2005.

[2] A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen. *Knowledge-Based Configuration: From Research to Business Cases*. Morgan Kaufmann Publishers, Massachusetts, US, 2014.

[3] L. Hotz, K. Wolter, T. Krebs, S. Deelstra, M. Sinnema, J. Nijhuis, and J. MacGregor. *Configuration in Industrial Product Families - The ConIPF Methodology*. IOS Press, Berlin, 2006.

[4] A. Hubaux, P. Heymans, P.-Y. Schobbens, D. Deridder, and E. K. Abbasi. Supporting multiple perspectives in feature-based configuration. *Software & Systems Modeling*, 12(3):641–663, Nov. 2011.

[5] Lenze Automation GmbH. *Manual Standardised Application L-force FlyingSaw V1.0*. Lenze Automation GmbH, 2010.

[6] F. Rossi, P. van Beek, and T. Walsh, editors. *Handbook of Constraint Programming*. Elsevier, 2006.

[7] J. Schroeter, M. Lochau, and T. Winkelmann. Multi-perspectives on feature models. In *Model Driven Engineering Languages and Systems*, pages 252–268. 2012.

[8] S. Soltani, M. Asadi, and D. Gašević. Automated planning for feature model configuration based on functional and non-functional requirements. In *SPLC'12*, pages 56–65, 2012.

[9] J. Stein, I. Nunes, and E. Cirilo. Preference-based feature model configuration with multiple stakeholders. In *SPLC'14*, pages 132–141, 2014.

[10] Y. Wang and M. Riebisch. Feature and Constraint Mapping for Configuration and Evolution of Variable Manufacturing Automation Systems. In *Tagungsband des Dagstuhl-Workshop MBEES*, pages 63–73. fortiss GmbH, 2015.

[11] D. Weiss and C. T. R. Lai. *Software Product-Line Engineering*. Addison-Wesley, 1999.