

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL  
INSTITUTO DE INFORMÁTICA  
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

JACOB DE QUADROS STEIN

**Supporting Feature Model  
Configuration based on  
Multi-Stakeholder Preferences**

Thesis presented in partial fulfillment  
of the requirements for the degree of  
Master of Computer Science

Prof. Ingrid Oliveira de Nunes  
Advisor

Prof. Daltro José Nunes  
Coadvisor

Porto Alegre, January 2015

## CIP – CATALOGING-IN-PUBLICATION

de Quadros Stein, Jacob

Supporting Feature Model Configuration based on Multi-Stakeholder Preferences / Jacob de Quadros Stein. – Porto Alegre: PPGC da UFRGS, 2015.

74 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2015. Advisor: Ingrid Oliveira de Nunes; Coadvisor: Daltro José Nunes.

1. Software product lines. 2. Recommender systems. 3. Feature model. 4. Feature model configuration. 5. Social choice. 6. Group recommendation. I. Nunes, Ingrid Oliveira de. II. Nunes, Daltro José. III. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Reitor: Prof. Carlos Alexandre Netto

Pró-Reitor de Coordenação Acadêmica: Prof. Rui Vicente Oppermann

Pró-Reitor de Pós-Graduação: Prof. Vladimir do Nascimento

Diretor do Instituto de Informática: Prof. Luís da Cunha Lamb

Coordenador do PPGC: Prof. Luigi Carro

Bibliotecário-Chefe do Instituto de Informática: Alexander Borges Ribeiro

# ABSTRACT

Feature model configuration is known to be a hard, error-prone and time-consuming activity. This activity gets even more complicated when it involves multiple stakeholders in the configuration process. Research work has proposed approaches to aid multi-stakeholder feature model configuration, but they rely on systematic processes that constraint decisions of some of the stakeholders. In this dissertation, we propose a novel approach to improve the multi-stakeholder configuration process, considering stakeholders' preferences expressed through both hard and soft constraints. Based on such preferences, we recommend different product configurations using different strategies from the social choice theory. Our approach is implemented in a tool named SACRES, which allows creation of stakeholder groups, specification of stakeholder preferences over a configuration and generation of optimal configuration. We conducted an empirical study to evaluate the effectiveness of our strategies with respect to individual stakeholder satisfaction and fairness among all stakeholders. The obtained results provide evidence that particular strategies perform best with respect to group satisfaction, namely average and multiplicative, considering the scores given by the participants and computational complexity. Our results are relevant not only in the context software product lines, but also in the context of social choice theory, given the instantiation of social choice strategies in a practical problem.

**Keywords:** Software product lines, recommender systems, feature model, feature model configuration, social choice, group recommendation.

## Suporte a Configuração de Feature Model baseada em Preferências de Múltiplos Stakeholders

### RESUMO

Configuração modelo de features é conhecida por ser uma atividade complexa, demorada e propensa a erros. Esta atividade torna-se ainda mais complicada quando envolve múltiplas partes interessadas no processo de configuração. Trabalhos de pesquisa têm proposto abordagens para ajudar na configuração de modelo de features, mas elas dependem de processos sistemáticos que restringem as decisões de alguns dos stakeholders. Neste trabalho, propomos uma nova abordagem para melhorar o processo de configuração multi-stakeholder, considerando as preferências dos stakeholders expressas através de restrições duras e brandas. Com base em tais preferências, recomendamos diferentes configurações de produto utilizando diferentes estratégias da teoria da escolha social. Nossa abordagem é implementada em uma ferramenta chamada SACRES, que permite criar grupos de stakeholders, especificar preferências dos stakeholders sobre uma configuração e gerar as configurações ideais. Realizamos um estudo empírico para avaliar a eficácia de nossas estratégias no que diz respeito à satisfação individual e justiça entre todos os stakeholders. Os resultados obtidos provem evidência de que estratégias em particular possuem melhor performance em relação à satisfação de grupo, chamadas *average* e *multiplicative* considerando as pontuações atribuídas pelos participantes e complexidade computacional. Nossos resultados são relevantes não só no contexto de Linha de Produto de Software, mas também para a Teoria da Escolha Social, dada a instanciação de estratégias de escolha social em um problema prático.

**Palavras-chave:** linha de produto de software, sistemas de recomendação, modelo de features, configuração de modelo de features, teoria da escolha social, recomendação para grupos.

## LIST OF ABBREVIATIONS AND ACRONYMS

<b>AVG</b>	Average
<b>BC</b>	Borda Count
<b>BDD</b>	Binary Decision Diagram
<b>CPC</b>	Collaborative Product Configuration
<b>CSP</b>	Constraint Satisfaction Problem
<b>CR</b>	Copeland Rule
<b>FM</b>	Feature Model
<b>FMP</b>	Feature Model Plugin
<b>FODA</b>	Feature-Oriented Domain Analysis
<b>LM</b>	Least Misery
<b>MP</b>	Most Pleasure
<b>MULTI</b>	Multiplicative
<b>NP</b>	Average Number of Preferences
<b>RFID</b>	Radio-Frequency Identification
<b>SCSP</b>	Soft Constraint Satisfaction Problem
<b>SD</b>	Standard Deviation
<b>SH</b>	Stakeholder
<b>SPL</b>	Software Product Line
<b>SPLE</b>	Software Product Line Engineering

## LIST OF FIGURES

1.1	Supply management feature model . . . . .	14
2.1	Development cost of single system and SPLE (POHL; BÖCKLE; LINDEN, 2005). . . . .	20
2.2	Time to market with and without SPLE (POHL; BÖCKLE; LIN- DEN, 2005). . . . .	21
2.3	The main graphical notations for feature modeling (BAGHERI et al., 2010). . . . .	22
2.4	Example of a feature model of a restaurant system. . . . .	23
2.5	Evolution of notations for feature model (CZARNECKI; HELSEN; EISENECKER, 2004). . . . .	24
2.6	Example of a feature model in editor view (ANTKIEWICZ; CZAR- NECKI, 2004). . . . .	26
2.7	Example of a configuration (ANTKIEWICZ; CZARNECKI, 2004). . . . .	26
2.8	Example of a feature model and the transformation to proposi- tional formulas (CZARNECKI; WASOWSKI, 2007). . . . .	27
4.1	Tablet feature model (partial). . . . .	36
4.2	Preference-based configuration meta-model. . . . .	38
6.1	Stakeholder configuration editor. . . . .	49
6.2	Recommendation example. . . . .	50
7.1	Tablet product line. . . . .	56
7.2	Satisfaction score frequency distribution. . . . .	60
7.3	Fairness score frequency distribution. . . . .	61

9.1	Tablet product line. . . . .	72
-----	------------------------------	----

## LIST OF TABLES

4.1	Stakeholder configuration examples. . . . .	40
5.1	Strategy recommendations. . . . .	46
6.1	Mapping of feature model relations to propositional logic. . . . .	51
6.2	Complete mapping of tablet feature model. . . . .	52
6.3	Mapping of propositional logic to CHOCO. . . . .	52
7.1	Goal definition. . . . .	54
7.2	Participant characteristics. . . . .	57
7.3	Preference analysis. . . . .	58
7.4	Optimal configuration intersection. . . . .	59
7.5	Individual satisfaction and fairness scores. . . . .	59



# CONTENTS

<b>1</b>	<b>INTRODUCTION</b>	<b>12</b>
1.1	Problem Statement	13
1.2	Proposed Solution and Contributions Overview	15
1.3	Outline	17
<b>2</b>	<b>BACKGROUND ON SOFTWARE PRODUCT LINES</b>	<b>18</b>
2.1	Software Product Lines	18
2.2	Feature Modeling	21
2.2.1	Feature Model	21
2.2.2	Feature Model Configuration	25
2.3	Final Remarks	27
<b>3</b>	<b>RELATED WORK</b>	<b>28</b>
3.1	Automated Support for Feature Model Configuration	28
3.2	Collaborative Feature Model Configuration	30
3.3	Discussion	32
3.4	Final Remarks	34
<b>4</b>	<b>PREFERENCE-BASED CONFIGURATION</b>	<b>35</b>
4.1	Motivating Example	35
4.2	Meta-model Description	37
4.3	Final Remarks	40

<b>5</b>	<b>RECOMMENDING PRODUCT CONFIGURATIONS BASED ON STAKEHOLDERS PREFERENCES . . . . .</b>	<b>42</b>
5.1	Stakeholder Satisfaction . . . . .	42
5.2	Social Choice Strategies . . . . .	44
5.3	Final Remarks . . . . .	46
<b>6</b>	<b>SACRES: A TOOL FOR RECOMMENDING OPTIMAL CONFIG- URATIONS . . . . .</b>	<b>48</b>
6.1	Features . . . . .	48
6.2	How Recommendations are Generated . . . . .	50
6.3	Final Remarks . . . . .	53
<b>7</b>	<b>EVALUATING THE EFFECTIVENESS OF SOCIAL CHOICE STRATE- GIES . . . . .</b>	<b>54</b>
7.1	Procedure . . . . .	55
7.2	Participants . . . . .	57
7.3	Results and Analysis . . . . .	57
7.3.1	Provided Preferences . . . . .	57
7.3.2	Strategy Similarity . . . . .	58
7.3.3	Strategy Evaluation . . . . .	59
7.4	Discussion . . . . .	61
7.5	Final Remarks . . . . .	62
<b>8</b>	<b>CONCLUSION . . . . .</b>	<b>63</b>
8.1	Contributions . . . . .	63
8.2	Future Work . . . . .	64
	<b>REFERENCES . . . . .</b>	<b>66</b>
<b>9</b>	<b>QUESTIONNAIRES . . . . .</b>	<b>70</b>

<b>9.1</b>	<b>Tablet Configuration for Group . . . . .</b>	<b>70</b>
<b>9.2</b>	<b>Questionnaire about Tablet Configuration for Group - Instructions . . . . .</b>	<b>73</b>

# 1 INTRODUCTION

Software Product Lines (SPL) came as a new and promising way of producing software with large-scale and a more organized reuse. The emergence of the product line concept reduce the time and costs of production. However, it reduced the possibility of customisation. We had the standard and cheap products or individual products created on a handcrafted basis. This market necessity gave origin to the concept of mass customization, when became possible to produce customized products maintaining a low cost in comparison with a handcrafted product for a specific client. It was possible by introducing the concept of common platforms, creating a base of technologies on which other technologies or processes are built (POHL; BÖCKLE; LINDEN, 2005). The variability in a family of systems that comprise an SPL is described by different features in a Feature Model (FM) (PARNAS, 1976), (CZARNECKI; EISENECKER, 2000), (KANG et al., 1990). Configuring FMs to instantiate specific product thus becomes a key activity, part of the SPL development, which is known to be hard, error-prone and time-consuming. Feature model configuration gets even more complicated if we consider that this activity can be performed not only by a single stakeholder, but also by multiple stakeholders geographically distributed (MENDONÇA; BARTOLOMEI; COWAN, 2008; JUNIOR; CIRILO; LUCENA, 2011; CZARNECKI; HELSEN; EISENECKER, 2005). Consequently, this calls for approaches that support this configuration process.

Much work has been carried out in the context of feature model analysis and configuration (BENAVIDES et al., 2013). The former facilitates the process of configuration by automating the search for possible configurations (BENAVIDES et al., 2013), while the latter provides active assistance during the configuration process (MENDONÇA; BARTOLOMEI; COWAN, 2008; JUNIOR; CIRILO; LUCENA, 2011). Automated support for feature model configuration has focused solely on single stakeholders, and collaborative approaches restrict decisions over a particular feature to be made by only a single stakeholder. They may also cause decisions made during the configuration process by one stakeholder constrain poste-

rior decisions made by another stakeholder, thus occasionally requiring backtracking and making it difficult to reach a valid configuration agreed by all stakeholders. A promising way of addressing this is to allow stakeholders to freely express their preferences, and then help them by suggesting valid configurations that maximise their satisfaction.

We thus propose a novel approach to improve the multi-stakeholder configuration process, by allowing stakeholders to express preferences over features of a feature model and then recommending optimal configurations according to these preferences. In order to do so, we must provide a way to model stakeholders' preferences and to consider them to identify an adequate configuration. Therefore, we propose a meta-model that provides concepts to create a stakeholder configuration, which consists of hard and soft constraints over available features. Given the preferences of different stakeholders, we propose using strategies from the social choice theory (ARROW; SEN; SUZUMURA, 2002) to make group recommendations (MASTHOFF, 2011). We adopt different strategies because different configurations may be optimal depending on the perspective we consider. For example, an optimal configuration may be the one which, on average, better satisfies the stakeholders. But this configuration may let one stakeholder very dissatisfied, so a configuration that does not let any stakeholder very dissatisfied may be better. Based on our meta-model and strategies, we developed the SACRES tool, which was used to evaluate our approach. We conducted an empirical study to evaluate the effectiveness of our strategies with respect to individual stakeholder satisfaction and fairness among all stakeholders. Results indicate that there is no strategy that is best regarding individual stakeholder satisfaction. However, with respect to fairness, some particular strategies perform best. As some of which are computationally hard, and produce equivalent results to others, we indicate the strategies that are potentially more adequate to be adopted when time constraints are given to provide recommendations.

We next better motivate the problem investigated in this dissertation with an example in Section 1.1. We also discuss limitations of related work. Next, in Section 1.2, we present and describe the key objectives and contributions of our work.

## 1.1 Problem Statement

Based on the context presented in the previous section, we define our research focus on this task of product configuration. This activity may seem easy at a first look, but can become very complicated in complex scenarios. First, we could have a large product line where we would need to configure a large-scale feature model

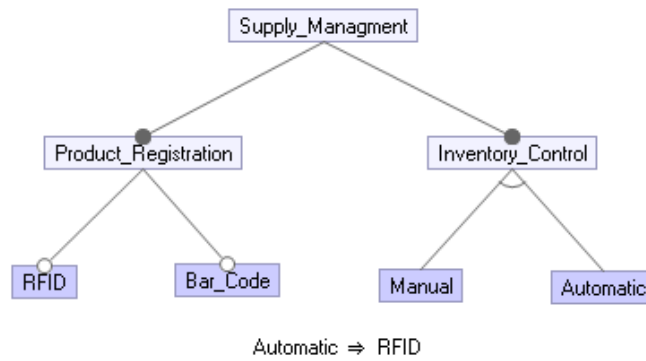


Figure 1.1: Supply management feature model

with hundreds of features, a lot of constraints and complicated relations between features, making the activity of feature model configuration very complex and time consuming. Second, a feature model may be configured by several stakeholders: costumers with their own needs, managers with specific expertise (and, as a consequence, divergent opinions) and people responsible for configuring the software to be produced.

In Figure 1.1, we demonstrate a very simple example of a set of features in a feature model in order to better illustrate our problem. Assume that we have two stakeholders reasoning about the features to be selected. In the left-hand side we have an alternative selection (only one of the two features can be selected). So, one first problem would be if a stakeholder 1 has a very high value of preference for the feature representing manual inventory control and a medium level of preference for automatic control, while a stakeholder 2 gives a high value of importance for automatic and a low level of preference for manual.

Additionally, we can see another kind of conflict. We may consider a scenario where a first stakeholder has a high value of preference for choosing automatic inventory control, but a second stakeholder strongly disagrees about selecting RFID product registration. As we can see, these features are related by a constraint (located in the bottom of Figure 1.1), that is, if the feature automatic is selected, the feature RFID also needs to be selected. Therefore, we would have another conflict of interest that also does not have a simple and direct solution.

Even though this case is very simple, it is already complicated to see whose preferences are going to be satisfied, that is, which features are going to be selected, because preferences for features are expressed imprecisely. Moreover, even if such preferences were expressed precisely, i.e. numerically, it is not trivial how preferences should be combined to select a valid set of features.

In these discussed cases, the stakeholders could have a meeting to decide about the feature model configuration. However, it is not always possible. We would have a much more complex task of product configuration if we consider a large-scale scenario composed by hundreds of features and several stakeholders spread across the globe reasoning about the configuration of the same feature model.

By taking into account the specific needs and preferences for features of each stakeholder, as well as the feature model constraints, we may conclude that there is no valid configuration of the feature model that satisfy all of them. Therefore, this calls for approaches that help in the feature model configuration, and recommend valid configurations that mostly satisfy needs and preferences. Thus, it is important to have tools and supporting research aiming to reach the best product configuration as possible, considering the whole variability of possible scenarios. Therefore, much work has been and must be done on this field.

## 1.2 Proposed Solution and Contributions Overview

Considering the kinds of problem we want to address presented in the previous section, we now introduce our proposed solution to the problem of providing support to the multi-stakeholder configuration problem. As we already briefly introduced before, a lot of distinct approaches have been proposed in order to help the process of selecting features, and consequently, the activity of feature model configuration.

The key idea of our approach is to use the concept of soft constraints applied in the context of product configuration. Using them as a way to describe preferences of stakeholders about a product line in order to create a more flexible and strong process of configuration. We expect that using sets of preferences with levels of importance and stating them as a Soft Constraint Satisfaction Problem (SCSP) (ROSSI; BEEK; WALSH, 2006), we can: reach an agreement between stakeholders' wishes; and, consequently, generate a valid product configuration that maximizes the expected requirements.

Our hypothesis is that the use of the definition of an SCSP and associated ways of solving it are helpful to model our addressed scenario and finding optimal configurations. SCSP is a generalization of the classical Constraint Satisfaction Problem (CSP), used for hard constraint solving. A classical constraint network is a triple  $\langle X, D, C \rangle$ , where  $X$  is a finite set of  $n$  variables;  $D$  is the set of the domains corresponding to the variables in  $X$ ; and  $C$  is a set of constraints, which define relations in sequences of variables. In a classical constraint network (CN) all constraints must be mandatorily satisfied in a solution, and that is why they

are called hard constraints. The task of finding a solution in a classical CN is the CSP, which is known to be NP-complete. Therefore, a SCSP seeks a way to satisfy all hard requirements while violating the desires as little as possible. When a problem is mapped as a CSP, it is possible to use of-the-shelf CSP solvers, providing the solutions to the problem. These solvers implement algorithms and heuristics that have been studied during several decades (BENAVIDES; TRINIDAD; RUIZ-CORTES, 2005).

Finally, our approach is supported by a tool, namely Sacres which allows creation of groups of stakeholders and binding each one of them to their respective preferences of configuration, in order to achieve the best level of agreement between all stakeholders.

In this work we thus aim to help developers, analysts, managers, costumers and any other stakeholders involved in the process of product configuration. Thus our main objectives are:

1. Model individual stakeholder's preferences over features in a new way, working with both soft and hard constraints, in opposition of the traditional approaches that consider only mandatory choices for the features of a FM. Providing, therefore, a wider set of possible valid configurations and a more flexible process of configuration.
2. Identify optimal configurations based on such preferences of multiple stakeholders, seeking in Social Choice Theory, the best alternatives to reach an agreement between all interested parts involved in the process of configuration.
3. Provide tool support to our proposed approach. A tool that fulfills the requirements needed to establish stakeholders groups and preferences, providing database support for handling stakeholder's configurations and finding the best recommendations.
4. Evaluate the effectiveness of the social choice strategies, through an empirical study, in order to find the strategies with best performance considering individual satisfaction and fairness to the group of stakeholders.

Considering our objectives we list our main contribution below:

1. A new approach for the task of product configuration of SPLs by dealing with stakeholders preferences;



2. A meta-model that allows to model stakeholder configurations that capture preferences over features;
3. The instantiation of social choice strategies to recommend feature model configurations given multi-stakeholder preferences;
4. A validated approach using soft constraints as way to produce a more flexible configuration;
5. A way to facilitate the configuration task by expanding possibilities of generating valid configurations;
6. A tool that provides group recommendation of configurations; and
7. An empirical study that evaluated the different social choice strategies.

### 1.3 Outline

In this Section, we present the structure of this dissertation. It is as follows.

In Chapter 2, we present the current scenario of the two main areas related to this dissertation, software product lines and feature modeling, detailing their contributions to software engineering and providing foundation to our proposed approach.

Chapter 3 presents related work. In order to better understand and discuss their strengths and limitations, we divided the related work by the type of configuration support.

Chapter 5 describes the procedure proposed to recommend the configurations. We establish and describe the meta-model, strategies of evaluation and the comparison of the recommended configurations.

In Chapter 6, we provide details about the tool developed to support the approach and evaluate the proposed strategies of recommendation. We present the implemented features and how the recommendations are generated.

Chapter 7 details the procedure and participants used to evaluate the recommender strategies, and also, the results achieved by the empirical study.

Finally, in Chapter 8, we present the conclusions derived from the work developed. Additionally, future work related to our conclusions are presented.

## 2 BACKGROUND ON SOFTWARE PRODUCT LINES

Software Product Line Engineering (SPLE) is a paradigm to develop software applications using mass customization and platforms (POHL; BÖCKLE; LINDEN, 2005). Thus, this chapter provides an overview of SPLE, detailing concepts and benefits of Software Product Lines in Section 2.1. Additionally, we describe more specifically Feature Modeling, which is a key activity inside SPL and the major focus of this work.

### 2.1 Software Product Lines

Software Product Lines (SPLs) came as a new and promising way of producing software with large-scale and more organized reuse. The concept brought by this approach was crafted along many years by a sum of other concepts and approaches, in order to reach what we consider nowadays as the Software Product Line Engineering (SPLE).

One of the first ideas taken as basis for SPLE draw us back 110 years, when Henry Ford founded his company bringing to the world the concept of large-scale production with lower costs. Even considering that Ford's approach is one of the most recognized by our time as the birth of product lines, we may acknowledge the reuse idea since the Ancient Egypt, on the construction of the famous pyramids.

The emergence of the product line concept reduce the time and costs of production. However, it reduced the possibility of customisation. Initially, everything was very good. Lower costs and time were just enough to make people happy, but costumers only stayed satisfied with only one kind of product, a standardized one, for a certain period. So, we had the standard and cheap products or the individual product created on a handcrafted basis. This market necessity gave origin to the concept of mass customization, when became possible to produce customized prod-

ucts maintaining a low cost in comparison with a handcrafted product for a specific client. It was possible by introducing the concept of common platforms, creating a base of technologies on which other technologies or processes are built. The use of common platforms, in the car industry increased the sales by 35% (POHL; BöCKLE; LINDEN, 2005).

Another concept related to SPL was introduced in by PARNAS (1976), where he described program families. He described it as a set of programs whose common properties are so extensive that would be worthwhile to first study their common properties and then to determine the special properties of the individual family members. He proposes a new way for creating programs, where instead of modifying a complete program to get a new one, the developer always begins with one of the intermediate stages and continues from that point with design decisions, ignoring the decisions made after that point in the development of the previous versions. In that way it became possible to say that two different versions of programs have the same ancestor.

The combination of the presented concept of Product Families, Platform-Based Development and Mass Customization brings us the possibility of reusing a common base of technology without forgetting the costumers' wishes. The systematic use of this concepts for software-intensive systems and software products built the basis of what we call Software Product Line Engineering paradigm.

In order to work with SPLE, we need first to create the platform on which the future projects will work with, allowing the use of mass customization. Therefore, this platform has to be suited to many different projects. The common parts between the applications are established first, as the basis, providing the artifacts that can be used for all products. After that, the differences that distinguish the applications are determined in order to accommodate costumer's specific desires. Considering the car industry example, the creation of the platform means to have a car basis with several customisable features (POHL; BöCKLE; LINDEN, 2005).

To facilitate mass customization, the artifacts used in different products need to be sufficiently adaptable, allowing them to fit in the different products that will be produced in the product line. However, restrictions need to be applied, because, for example, if you are buying a convertible car, you do not want a rear window water splash. Therefore, this flexibility of the artifacts will often come with a set of constraints. In the Software Product Lines, this flexibility concept is called *variability* (POHL; BöCKLE; LINDEN, 2005).

As we may see, we have a change of paradigm. The products derived from platform artifacts can no longer be treated as being independent. They are related

through the underlying technology. Thus, the companies need to reorganize their structure, adapting the organization to work with a platform approach. Although the company has an increasing of work in the beginning, the SPLE has good motivations for developing software under its paradigm (LINDEN; SCHMID; ROMMES, 2007).

The first and main motivation for introducing SPLs is the reduction of cost provided by the high level of reuse. However, the artifacts to be reused need to be created and a way to manage their reuse also need to be planned beforehand (LINDEN; SCHMID; ROMMES, 2007). Thus, it is needed an initial investment to create the platform, which sometimes, does not seem very good for the companies.

In spite of this first higher investment, after some projects the use of SPLE may accomplished a high reduction of costs. Figure 2.1 shows the cost to develop  $n$  different systems using the traditional way (solid line) and the SPL (dashed line)(POHL; BÖCKLE; LINDEN, 2005). We may notice that after about three projects, we reach the break-even point, where the company starts to profit from the use of SPLE.

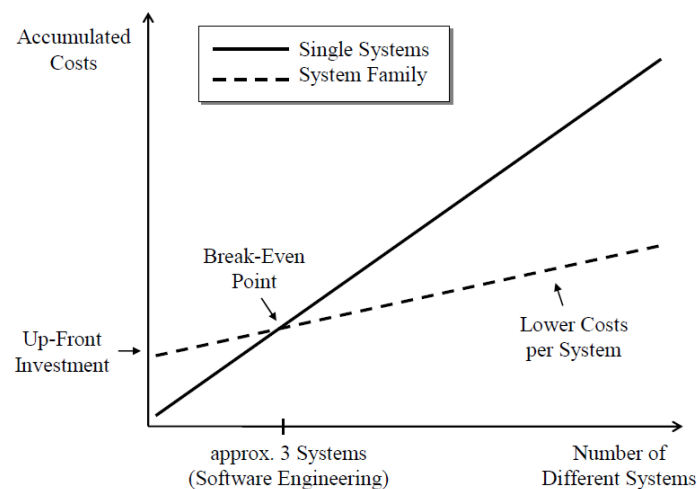


Figure 2.1: Development cost of single system and SPLE (POHL; BÖCKLE; LINDEN, 2005).

A second important motivation for SPLE is the reduction of the time to market. Similar to the costs, the initial time to develop a project is higher, but after the platform and common artifacts have been built, this time is highly decreased, as Figure 2.2 shows.

Another motivation that needs to be pointed is the enhancement of quality. The artifacts produced in the platform end up being reviewed and tested in many products, detecting faults and correcting them, thereby increasing the quality of all products. The SPLE brings also other motivations as the reduction of maintenance

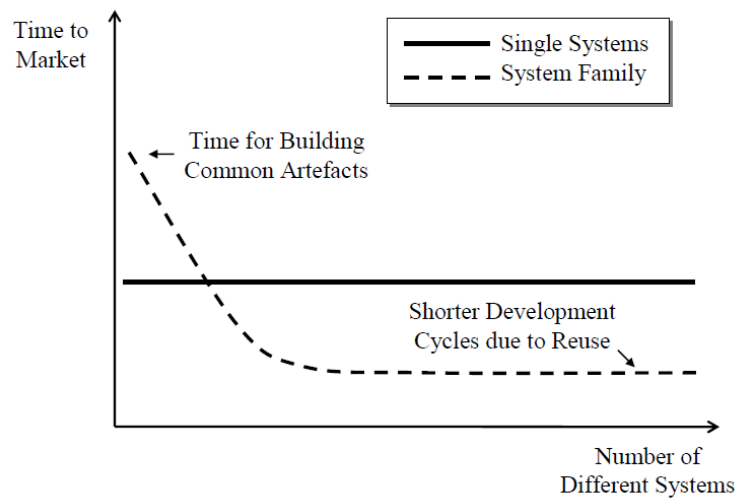


Figure 2.2: Time to market with and without SPLE (POHL; BÖCKLE; LINDEN, 2005).

effort, improved cost estimation and, for the costumers, products with higher quality at lower prices (POHL; BÖCKLE; LINDEN, 2005).

Considering the concepts presented, it is possible to see that new methodologies are needed to work with the SPLE paradigm. Many have been proposed along the years(COPA, FAST, FORM, Kobra, QADA) (AMERICA et al., 2000) (WEISS; LAI, 1999) (KANG et al., 1998) (ATKINSON et al., 2002) (MATINLASSI; NIEMELÄ; DOBRICA, 2002) (MATINLASSI, 2004). Within this approaches, the use of Feature Models as notation for expressing the variability inside a Software Product Lines stands out. Therefore, in the next section, we present the activity of Feature Modeling.

## 2.2 Feature Modeling

Feature Modeling is one of the most popular ways to represent variability in Software Product Lines. Thus, feature modeling is a key activity in product line development (ANTKIEWICZ; CZARNECKI, 2004). Therefore, in this section we present the main definitions about the Feature Model itself (Subsection 2.2.1) and we also detail the task of FM Configuration (Subsection 2.2.2), responsible for the selection of features from the product to be.

### 2.2.1 Feature Model

Features may be defined as the representation of any property that is relevant to a stakeholder. A feature also represents an increment in program functionality.

The use of features came as a key technical innovation to distinguish product-line members, where each one of them is defined by a unique combination of features (BATORY, 2005).

Kang et al. (1990) published one of the main contributions for the area of Software Product Lines by describing the Feature-Oriented Domain Analysis (FODA) method. The FODA brought an identification method of prominent and distinctive features of software systems in a domain, where features represent common aspects and differences between related systems. Thus, as part of product line engineering, we have the key activity of Feature Modeling, proposed as part of FODA method. In this activity, it is created the Feature Model, a graphical representation of common and variable features, their dependencies and constraints, in a product line (CZARNECKI; HELSEN; EISENECKER, 2004).

Features also show possible decisions about the product on time of configuration. They can be classified as mandatory or optional when individual or can be characterized as or-features or alternative features when grouped. A representation of this notation is shown in Figure 2.3.

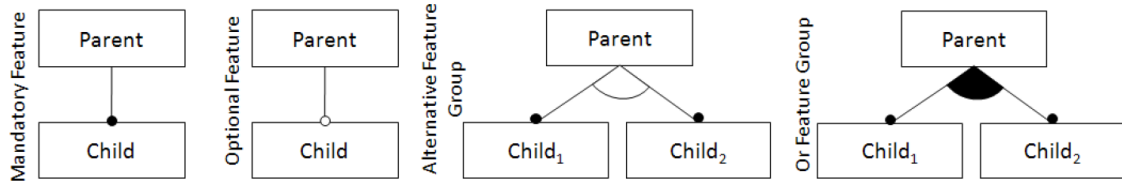


Figure 2.3: The main graphical notations for feature modeling (BAGHERI et al., 2010).

For this work we adopted the representation shown in Figure 2.3 for the graphical Feature Model created using the FeatureIDE tool (KASTNER et al., 2009). One example of that representation in a Feature Model is shown in Figure 2.4. In this case the Feature Model gives us the representation of the variability in a Software Product Line of a Restaurant. Looking at the graphical model, we can see in a better way the objective of the notation shown previously and what is the point of having alternative groups, or-groups, and other kinds of relations. As example, we can take the Food Module, where we have the possibility of choosing between the kinds of ordering modules, addressing the client business needs. It is also important to notice the mandatory features. They represent the basis of the SPL. Thus, this selection of features must be present in all cases, so we can generate a correct product.

The FODA method brought the use of Feature Models. However, we also have some work done on improving this representation, like the extension proposed by Czarnecki (2000) showed in Figure 2.5. Another key work is the cardinality-based

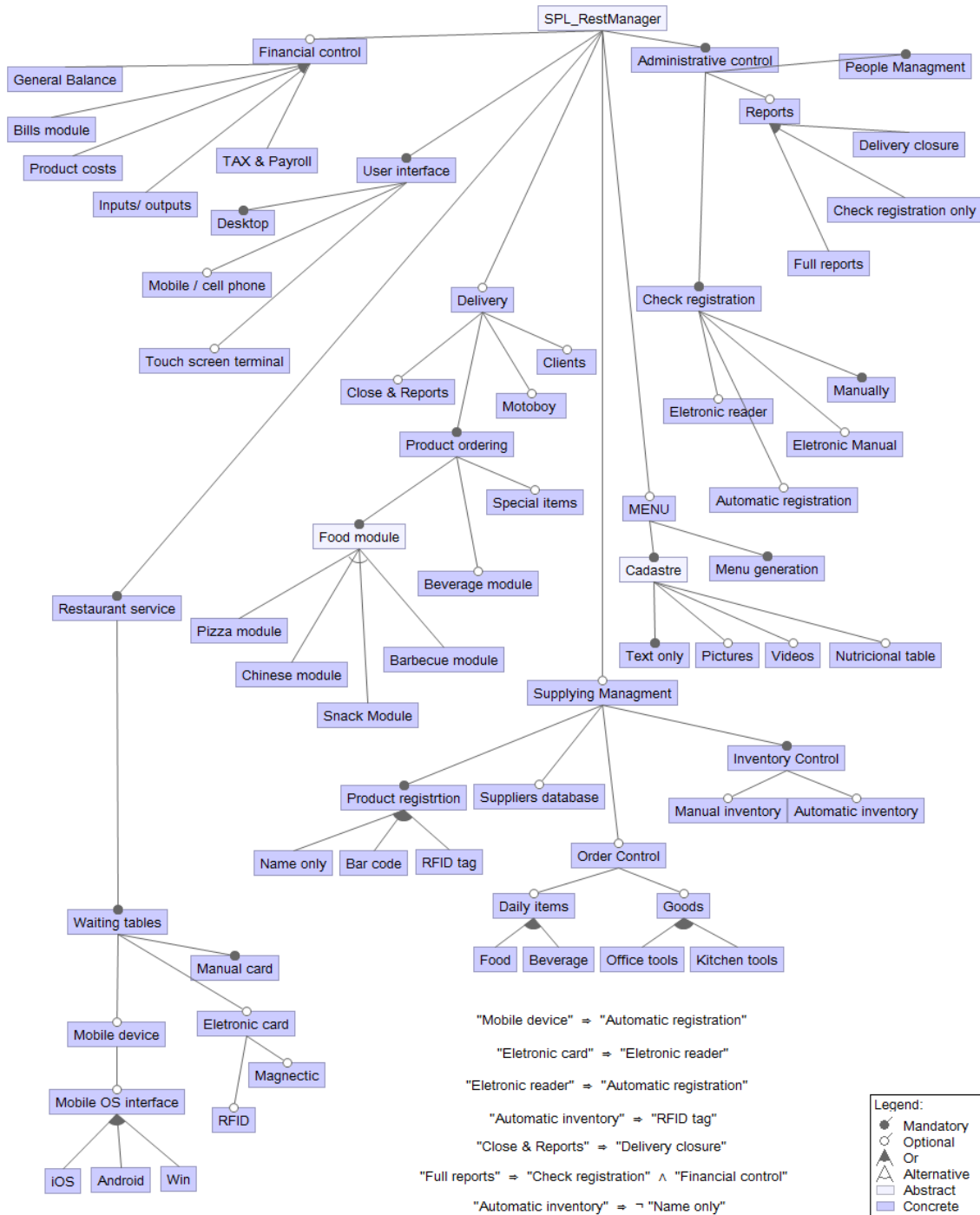


Figure 2.4: Example of a feature model of a restaurant system.

method also proposed by Czarnecki (2004). In this method, the alternative features (only one of the set can be selected) are represented by a [1..1] cardinality and the or-features are represented as [1..N].

Going further, we can have *Integrity Constraints* in a Feature Model, which are (most widely used): *Includes*, representing that the presence of a given feature requires the existence of another; and *Excludes*, representing that the presence of a given feature requires the elimination of another. In the shown case we can see some

FODA notation	Extended notation	Cardinality-based notation
<p><i>mandatory and optional subfeatures</i></p>	<p><i>mandatory and optional subfeatures</i></p>	<p><i>mandatory and optional subfeatures</i></p>
<p><i>alternative subfeatures</i></p>	<p><i>exclusive-or group</i></p>	<p><i>group with cardinality (1-1)</i></p>
<p>n/a</p>	<p><i>inclusive-or group</i></p>	<p><i>group with cardinality (1-k)</i></p>
<p>n/a</p>	<p><i>exclusive-or group with optional subfeatures</i></p>	<p><i>group with cardinality (0-1)</i></p>

Figure 2.5: Evolution of notations for feature model (CZARNECKI; HELSEN; EISENECKER, 2004).

restrictions between the features, showing, for example, that we need the *RFID tag* in order to be able to apply *Automatic Inventory*.

Considering the same case cited above of the *Product Ordering Module*, we may see a different representation of feature, in a brighter color. This kind of feature was introduced by Thum (2011) and it is called abstract. This type of feature adds more meaning in the Feature Model representation, but it does not have influence on the final product. It helps represent variability, but the selection of a abstract feature does not represent a direct change in the SPL.

The SPL of this Restaurant is an illustrative example, thought to test techniques of configuration. A real feature model can be very large, having hundreds of



features and a lot of complex relations and constraints. Therefore, we can imagine that the task of choosing the features that will form the future product is very hard and a lot of time will be spent until it is accomplished, when a valid product addressing the requirements of all stakeholders will be derived. Therefore, as part of the SPLE, we have the key activity of configuring a Feature Model, which will be presented in the next section.

### 2.2.2 Feature Model Configuration

Feature Model configuration or Product Configuration, as previously said, is a key activity within SPLE. To achieve a valid product, developers need to find a selection of features from the model that satisfies requirements and rules associated with the Feature Model. This configuration process involves reasoning over a complex set of constraints to meet an end goal (WHITE et al., 2009).

This task is hard and error prone. To reach a valid configuration we need to derive a product following as many requirements and concerns as possible, considering, in many times, multiple stakeholders with distinct wishes and opinions about the product to be. As stakeholder we can have: costumers with their own needs, managers with specific expertise (and, as a consequence, divergent opinions) and people responsible for configure the software to be produced.

Product configuration is an activity that needs to be done carefully. If we consider only a small set of optional features in a feature model, we can already have a large number of choices, and consecutively, a large number of possible valid configurations. For example, in case of having five optional features, we will have 32 possible configurations. Therefore, each addition to a feature model makes its configuration even harder, each property or functionality of a system added to a product line as a new feature may create a new set of possible configurations. In the last example, if we only add one optional feature, we will exponentially increase the number of possible configurations to 64.

Configuration is the process of choosing the features in order to derive a valid concrete configuration conforming to the Feature Model (ANTKIEWICZ; CZARNECKI, 2004). In Figure 2.6, we have the representation of a Feature Model of an Electronic Shop system in the editor view of the Feature Model Plugin (FMP). Figure 2.7 shows a configuration of the FM from Figure 2.6. The process of configuration permits for the stakeholder to choose the features needed for the product to be produced.

A key work on the area of Feature Modeling introduces the use of grammar and propositional formulas to represent the relations, structure and constraints in

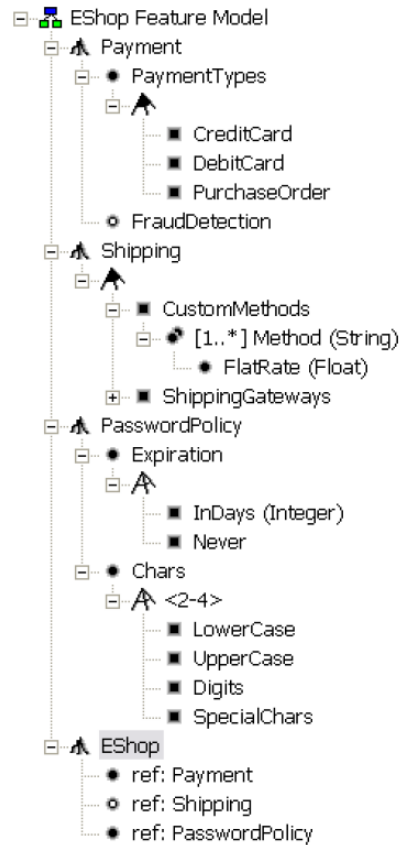


Figure 2.6: Example of a feature model in editor view (ANTKIEWICZ; CZARNECKI, 2004).

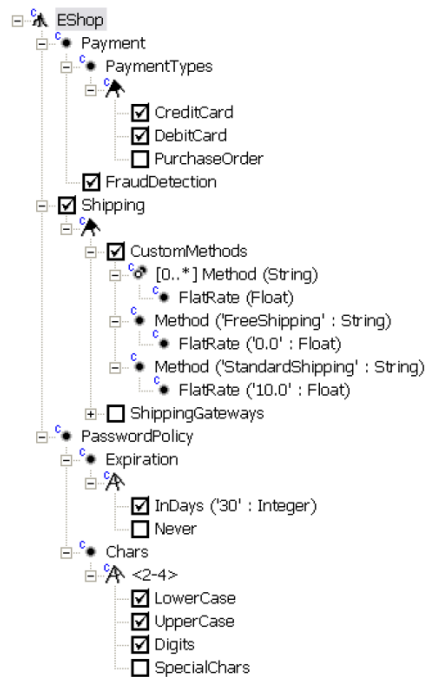


Figure 2.7: Example of a configuration (ANTKIEWICZ; CZARNECKI, 2004).

a Feature Model. This technique was introduced by Batory (2005). The main contribution brought by this paper was to enable the possibility of using of-the-shelf solvers for reasoning and analyzing possible configurations. In Figure 2.8, we show this transformation.

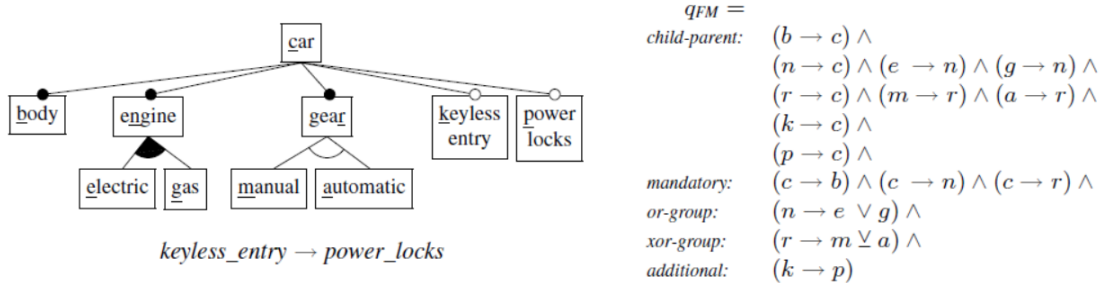


Figure 2.8: Example of a feature model and the transformation to propositional formulas (CZARNECKI; WASOWSKI, 2007).

Given the importance and challenges of the presented task, many approaches, tools and contributions to help in this process have been done, each one trying to achieve a better product, with less effort and a smaller number of errors.

## 2.3 Final Remarks

This chapter presented an overview of software product lines, describing the main ideas of this paradigm and the motivations behind the application of product line engineering compared to regular software development, where we are able to see how SPL can improve software engineering. We also detailed the activity of feature modeling, describing the objectives, notations and difficulties around this task. It is notable the importance of feature model configuration that, spite of being a subject already addressed by many researches, is still a topic that needs attention and improvement in order to facilitate the task of configuring a FM. Next, we present the main approaches related to feature model configuration in order to better understand the current scenario of research.

## 3 RELATED WORK

The goal of this chapter is to provide understating of how feature model configuration has been supported to date. In order to facilitate the feature model configuration activity, a group of approaches proposes the automation of the search for possible configurations (Section 3.1), while others provide active assistance during the collaborative configuration process (Section 3.2). In Section 3.3, a discussion about the presented approaches is made.

### 3.1 Automated Support for Feature Model Configuration

Considering the importance and complexity of the task of Feature Model configuration. Much work has been proposed trying to help in this task. Currently, various approaches (BENAVIDES; TRINIDAD; RUIZ-CORTES, 2005; PILLAT et al., 2013; BENAVIDES et al., 2013) have been developed to help reduce the complexity of a single-stakeholder configuration process by automating parts of the feature model configuration activity. Most of them trying to facilitate the search for possible valid configurations. The main approaches are based on the use of propositional formulas proposed by Batory (2005). In his work, not only he proposed the previous presented transformation of the FM into propositional logic, but also the use o Satisfiability problem solver (SAT solvers). SAT solvers attempt to decide whether a given propositional formula is satisfiable or not, that is, a set of logical values can be assigned to its variables in such a way that makes the formula true.

Following a similar path, we have many approaches using Constraint Satisfaction Problem Solvers (CSP solvers), which have been initially proposed by Benavides et al. (2005). The basic idea in CSP solvers is to find states (value assignments for variables) where all constraints are satisfied. CSP and SAT solvers are a good option for manipulating feature models since they are becoming more and more efficient despite the NP-completeness of the problem itself. Benavides et al. (2005) provide a constraint satisfaction problem interpretation of the feature model configuration,

in which the relations between one or more attributes of a feature are considered, in addition to the model integrity constraints. Therefore, based on such an interpretation, their proposed method not only is able to check whether a given feature model configuration is correct, but also to recommend the best product according to a criterion.

An alternative work proposes the use of Binary Decision Diagrams (BDD). BDD is a data structure for representing the possible configuration space of a boolean function, which can be useful for mapping a feature model configuration space. The weakness of BDD is that the data structure size can grow exponentially in certain cases (MENDONÇA et al., 2008).

In contrast, Pillat et al. (2013) proposed an approach to ensure the consistency of the feature model configuration activity in which a Business Rules Management System (BRMS) is used as a decision support mechanism. The application of BRMS helps the stakeholders to resolve decision conflicts with respect to the model integrity constraints. For example, the system is able to recommend which steps must be carried out to achieve a new valid configuration, and also to propagate feature selection. The main benefit of using business rules to express how features should be selected is that rules tend to be simple. Such rules might seamlessly work together, thus providing a user-friendly way for stakeholders to specify and manage their own configuration rules.

In a continuous pursuit for making the process of feature selection easier, we have other approaches where one of the main objectives is automatic selection based on dependencies and restrictions between features. Considering it, we present next, the key works proposed in the area.

The previous approaches consider only hard constraints for solving the configurations and that can make the configuration task very complicated by establishing strong restrictions to configure a specific product. Based on that (Barreiros, 2011), proposed the use of soft constraints, in place of the traditional Integrity Constraints, in situations where an exception needs to be made, and also, this type of constraints can be used to capture useful information in a feature model. The paper makes a categorization of soft constraint semantics, the formalization of the impact of these constraints on the logic representation of the feature model and the description of possible automated analysis procedures.

White et al. (2009) proposes an automated approach for multi-step feature model configuration and presents a tool called MUSCLE (Multi-step Software Configuration Problem Solver). The authors develop an automated method for deriving a set of configurations following a series of requirements over a span of configuration

steps using CSP solvers. An important contribution of the paper is the fact of its working on multi-step requirements, such as limits of cost on each step along time. It is also provided a formal model of multi-step configuration and how it can be mapped to a Constraint Solving Problem.

Felfernig et al. (2001) presents an approach for intelligent support for interactive reconfiguration of mass-customized products. Their approach aims to help the user to recover from invalid configurations generated by the need of modifications in the current selection of features. The reconfiguration problems are modelled as a CSP in order to calculate a minimal subset of elements to be relaxed for reaching a satisfiable configuration. The approach generates reconfigurations alternatives for the user to choose from.

Bagheri et al. (2010) proposes the use of soft constraints as a way to model concerns related to the features in a feature model, where concerns are a representation of business quality attributes related to strategic objectives and the target product audience. To accomplish it, they use fuzzy logic to describe stakeholders' soft constraints. Their approach brings a new concept and differently from the last introduced approaches. They use their own algorithm, instead of using CSP solvers to test the configuration. A two step configuration is proposed, where in the first step the hard constraints are evaluated and in a second step the soft constraints are considered in order to reach a final configuration. However, this approach needs manual expert annotations about the feature's concerns in order to create the feature utility knowledge base. Moreover, it considers preferences as a single set of concerns shared by all stakeholders, and does not consider conflicting preferences among multiple stakeholders involved in a collaborative product configuration.

Nevertheless, the selection of features should be based not only on the integrity constraints (which cannot be violated) but also should take into account preferences about strategic goals, non-functional limitations and so on — which, in some cases, can only be partially supported (BAGHERI et al., 2010).

## 3.2 Collaborative Feature Model Configuration

A trivial approach to perform the activity of Feature Model Configuration is to try to configure a product in a random way, selecting or deselecting features based on the system's knowledge of a single person in an unguided process. However, this is a time consuming and error prone approach if we are aiming to produce a valid, correct and robust product, mainly when we are dealing from medium to large-scale feature models, which have an extremely large number of possible configurations.

An alternative to configure feature models randomly is to perform this task in a stepwise fashion, leading us to the concept of choosing features separated in stages, where each stage eliminates some configuration choices. This process is referred by CZARNECKI et al. (2005) as *Staged Configuration*. In this process, it is introduced the concept of specialization, and multi-level configuration, where we can take a Feature Model and yield as specialized one, where the set of systems described by the last is a subset of systems described by the main Feature Model. The configuration spaces can be defined in terms of three different dimensions: (i) time: separating spaces in terms of different phases in a product lifecycle; (ii) roles: creating configuration spaces and assigning tasks of configuration based on expertise; and (iii) targets: defining configuration stages based on a target system for a given software that needs to be configured. The authors claimed that a product configuration is derived from one stage to another, by eliminating configuration options via specialisations progressively after a certain number of stages. The configuration reflects preferences of the different groups and different people involved in the feature model configuration activity. Nevertheless, as illustrated before, feature selection made by one stakeholder can indirectly govern the selection of other stakeholders or influence which features are still available. While the staged configuration approach allows stakeholders to select features collaboratively, it lacks guidance mechanisms to assist stakeholders in consistently expressing their preferences about features selection. In this case, there is high risk that feature selection will be made in an inconsistent manner. This process enables the possibility of collaborative product configuration where a different stakeholder can configure each specialized model. At the end, a general product configuration is derived, reflecting the decision made by all stakeholders (CZARNECKI; HELSEN; EISENECKER, 2005).

An interesting work is proposed by Czarnecki et al.(2008), adding the concept of probabilities of selection between features and giving support to an interactive configuration based on probabilistic feature models. Lin et al. (2010) proposed an approach to optimize the configuration process in order to make it more efficient. This approach introduces two measures. Positive and negative coverage of variant (variants automatically included or excluded of the product when a variant is selected or deselected) and Configuration Coverage (CC, which is the union of PC and NC). The CC determines how important a variation point is and, therefore, should be considered earlier in the configuration task.

Extending the approach of Staged Configuration, MENDONÇA et al. (2008) proposed an approach to Collaborative Product Configuration (CPC) that aims to provide support for the coordination of teamwork on the decision making process of product configuration. He proposes the creation of a CPC plan, splitting the fea-

ture model in configuration sessions, separated by the expertise of each configuration actor; establishing merging sessions to resolve decisions conflicts. The merging session provides a good way for stakeholders to reason together about some decisions. Therefore they suggested organising the collaborative product configuration process into a workflow as a solution to circumvent the previously mentioned issue, idea that was latter formalised (HUBAUX; CLASSEN; HEYMANS, 2009; LOCHAU et al., 2013). The general solution is to split the feature model into decision sets or views. In short, a decision set is a portion of the feature model that is assigned to a particular stakeholder based on an eligible criterion such as expertise. Configuration workflow is a valuable mechanism to ensure the effectiveness of the configuration activity. It provides a valid arrangement where independent choices (disjoint decision sets) can be made in parallel and dependent ones (dependent decision sets) must be performed in sequential stages. Thus, at the end of the configuration process, a correct product respecting the configuration constraints is always achieved.

The workflow approach has a significant disadvantage; namely collaborative product configuration is inflexible. Such a method poses several obstacles for interactive negotiation of customer requirements because dependent choices must be performed in sequential stages. In many cases, configurations made in earlier stages overlay choices in later phases (WHITE et al., 2009). Moreover, the priori identification of all possible reconfiguration paths is computationally hard; even a product line with few points of variability can result in an exponential reconfiguration space (TSANG, 1993). In order to deal with these issues, Marques et al. (2011) presented a product configuration method based on the *personal assistant* metaphor. The proposed method consists of a dynamic and distributed product configuration with personal assistance provided by software agents. Such agents recommend features to be selected given a set of previously selected features in order to keep the product being configured valid. But still stakeholders are constrained by prior decisions when making new choices, and can make decisions only over features that remain undecided.

### 3.3 Discussion

After presenting the main concepts of Software Product Lines and SPLE, we can realize the importance of this new paradigm of software development. It is notable how it may decrease the costs of production and the time to market of the product. In spite of being based on brand new concepts, it brings a high improvement on the idea of software reuse, which has being done for many years, but not in a systematic way.



The activity of Feature Modeling created a new way of treating the variability inside SPL. The Feature Model brought a great contribution not only for SPLE, but also for Software Engineering in general. This model has been used for representing many things, i.e. ontologies, not only software artifacts of a product line. The extensions proposed also allowed a better representation of the FM, providing easier and more flexible way to work with it.

Considering the activity of product configuration we can realize how important is to have tools and supporting researches aiming to reach the best product configuration as possible, considering the whole variability of possible scenarios.

Beginning by one of the main contributions for the task of product configuration we have the Czarnecki's Staged Configuration, which brought substantial contribution to the process of Feature Model Configuration. However, it does not predict the influence that could exist between decisions made by stakeholders in separated stages. It lacks of a method to guide stakeholders to collaborate. For example, if a stakeholder chooses a feature in a given stage, his choice can affect posterior decisions to be made about the product to be. A selected feature in a given moment may constraint that another feature in a future time will have to be selected or that this will not be possible to be selected by a restriction, which may not lead to the best option, according to the stakeholders wishes.

Trying to help fulfilling the gaps of the Staged Configuration we have CPC proposed by Mendonça et al.(2008), which allows only some of the configuration spaces to be configured in parallel, but there is no recommendation system in configuration-time to prevent decision conflicts before merging sessions. And, as the approach is based on a workflow arrangement, it results in an inflexible configuration activity.

Entering the semi-automatic approaches for product configuration, it is important to notice the great contribution brought by Batory's work on propositional formulas for Feature Models. Most of the current approaches use his techniques to work and find solutions for the problem of configuration.

Considering this last presented semi-automatic approaches, we see that most of them are based on finding and recommending a solution for the FM considering the hard constraints it imposes. However, in the first approaches, the concept of soft constraints could be applied on the context of product configuration. Using them as a way to describe preferences of stakeholders about a product line in order to create a more flexible and strong process of configuration.

Soft constrains provide a way to model preferences formalizing desired properties rather than requirements that cannot be violated, which would be model as

hard constraints. Soft constraints describe desires about a system that should be followed as much as possible, but could be violated in order to reach a valid product. By implementing these soft constraints, we could reach a wider set of possibilities, recommending feature selections by also considering different levels of preference of the stakeholders (ROSSI; BEEK; WALSH, 2006).

Therefore, we may see the improvement provided by the use of soft constraints. We described previously, three approaches that use soft constraints. The first is Czarnecki's Probabilistic Feature Model, which inserts the concept of soft constraints by giving probabilities of choice for the features. The second approach proposes a flexibilization of the Integrity Constraints, creating restrictions working more as recommendations that can be broken. The last of them, Bagheri's work, uses fuzzy logic to create a more flexible process, where the features are annotated concerning quality attributes and the stakeholder establishes, for example, if performance is more valuable than speed.

All these approaches bring their own point-of-view and they are all valid. Maybe a good tool could unite these soft approaches and, in addition, consider a multi-stakeholder scenario, which is the case addressed by Marques's work. Then we could have a much more flexible and open process of configuration, where many stakeholders could give their opinions about what is more important or what needs to be selected. This flexibility is the main idea of our approach, where multiple stakeholders can give different levels of preference for the features of the future product, expressing how much they want, or not, a specific feature of the FM.

After all presented ideas, we may see that the SPLE and the task of Feature Model Configuration can still be improved and there are still problems to be addressed, but in spite of that, the work done by the companies and researchers already possibilities a very good application of this paradigm.

### 3.4 Final Remarks

In this chapter, we introduced the main approaches developed to improve the product configuration process. Most of the work help to improve the task of configuration mainly in two ways. By organizing the manual process of configuration or by finding the possible configuration in an automated way. Analyzing the approaches, feature model configuration still is a field of study that needs improvement. Thus, our approach seeks to improve the process of product configuration by helping the stakeholders to express their preferences with soft and hard constraints and, also, by recommending optimal configurations. Therefore, next, we present our model for recommending configurations based on stakeholders preferences.

## 4 PREFERENCE-BASED CONFIGURATION

In the previous chapter, we discussed many approaches which aim to facilitate the feature model configuration process. Approaches range from helping organizing the process of configuration in a systematic way to automating most of the work related to this task. For a better understanding of what kinds of problems we want to solve, we next (Section 4.1) present a motivating example, describing the kinds of conflicts that our approach addresses. And in Section 4.2 we specify our meta-model, providing means for stakeholders to specify their preferences over features, which indicate preferred configurations.

### 4.1 Motivating Example

In the previous chapters, we introduced the key idea of our approach, background about software product lines and the related work. In this section, we describe a simple example that illustrates the need for our approach, demonstrating the kinds of conflicts that our approach addresses. This example will also be used in the following sections as a running example.

Our example involves the configuration of a tablet, which has optional and alternative features associated with both software and hardware. We selected a small portion of variable features to be part of our example. The tablet feature model, indicating available features and domain-specific constraints over them, is presented in Figure 4.1.

In this feature model, there are three optional features (Network, Cell Phone and Frontal Camera), and each of which is associated with other optional features or alternatives that, when selected, should respect a specified cardinality. For example, if the Frontal Camera is selected for a configuration, either the feature 1MP or 2MP should be selected, but not both. Constraints (shown below the feature model tree) indicate, for instance, that if the SMS feature is selected, either 3G or 4G should also be selected. The tablet feature model, with its constraints, leads to 92 possible

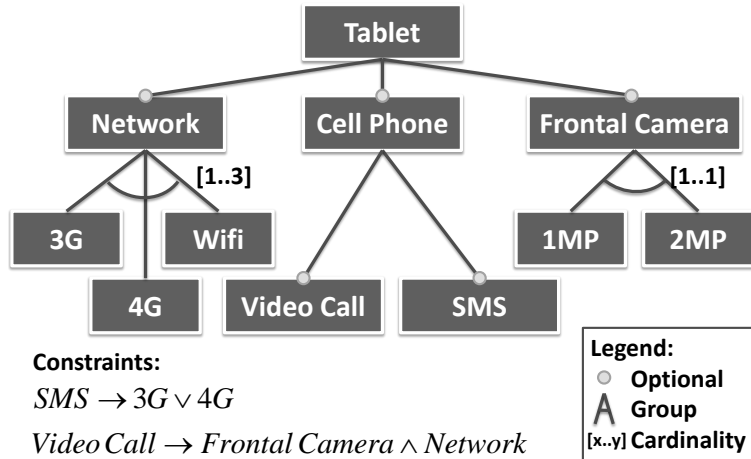


Figure 4.1: Tablet feature model (partial).

valid configurations, which is a number that can be easily treated computationally, but is already high enough to prevent stakeholders to evaluate all of the possibilities.

As stated in the introduction, a feature model may be configured not only by a single stakeholder, but by multiple stakeholders with different needs and preferences. Sometimes, configuring a product may involve the CEO of a company, employees from the financial department, technical experts, domain experts, users and so on. In this case, reaching a consensus is not straightforward. First, because of the divergent opinions. Second, because such stakeholders may be geographically distributed, making discussions among them more difficult to happen. In the remainder of this section, we will illustrate three possible categories of conflicts that can emerge considering our small feature model and two stakeholders. Such conflict categories are also applicable to large feature models. The first (trivial) type of conflict is when two stakeholders (Alice and Bob) disagree about a single feature, i.e. one of them desires it to be part of the product, and the other does not. For example, Alice wants a tablet with a frontal camera, and Bob believes that it increases too much the cost of the tablet, and thus should not be selected.

Second, consider a conflict associated with alternative features from which solely one should be selected. Assume that Alice convinced Bob that a frontal camera is important. But now, she wants a 2MP camera, and Bob — still concerned with cost — prefers a camera with 1MP. So the second type of conflict thus emerges due to feature model constraints (as opposed to domain-specific constraints). Another example of this kind of conflict is the following. Alice expresses a desire for a tablet with SMS, and Bob states that he does not want the Cell Phone feature. But, satisfying Alice’s desire means unsatisfying Bob’s preference.

Finally, the third type of conflict is associated with domain-specific constraints, which are those explicitly stated for a particular feature model indicating

relationships among features. Assume that Alice does not express any preferences for network and its children, but states that she wants Video Call. Bob, in turn, says he does not want Network. Although they have preferences over features of different branches of the feature model tree, due to a domain-specific constraint ( $VideoCall \rightarrow FrontalCamera \wedge Network$ ) it is impossible to satisfy both Alice and Bob.

This scenario exposes two different problems that should be dealt with the provision of an approach that will help Alice and Bob to configure their tablet. First, how can we *model* Alice’s and Bob’s *individual preferences over features*? Second, how can we use such preferences and resolve conflicts to *recommend an optimal configuration*?

In order to provide recommendations of feature model configurations for a group of stakeholders, we must provide means for them to specify their individual preferences. We thus, in next section, describe a meta-model that specifies concepts that allow to capture such preferences. There are two main ways of preference expression: (i) *qualitative preferences*, such as I prefer A to B, i.e. the preference is given as an order relation; and (ii) *quantitative preferences*, such as I like this very much, in which a degree of preference is given. In this work, stakeholders are able to express quantitative preferences over available features, using a *numeric* value to specify their degree of preference for a feature.

## 4.2 Meta-model Description

In the previous section we described the kinds of conflicts that may arise considering a multiple stakeholders scenario. Some of this problems may happen even in a single stakeholder process of configuration if we consider that previous decision could constrain future ones. In order to address this issues, we need to find a consensus between the stakeholders wishes. In addition, we also need to provide a way for the stakeholders to express their preferences in a proper manner. Thus, given the stakeholders wishes expressed trough both hard and soft preferences, we can search for possible configuration in a more flexible scenario, considering that we are working not only with mandatory choices, but also with recommendations based on the stakeholders levels of preferences about the features of the product to be configured. Thus, we created a meta-model that provides support for the discussed issues.

Therefore, our meta-model is presented informally in Figure 4.2, which presents concepts to instantiate feature models, product configurations, and stakeholder configurations. As it is possible to see, we kept the traditional structure of a feature

model, with the constraints specified by propositional formulas, the kinds of features and a configuration being a valid selection of features. However, in order to support our approach, we add the concept of a stakeholder group related to a feature model and representing the group of stakeholders involved in the process of feature model configuration. A stakeholder group can have many stakeholder configurations which represent the wishes of the stakeholders established using hard constraints (preferences that cannot be violated) and soft constraints (degree of preference showing how much a we want or not a feature and used as a recommendation that, therefore, can be violated if needed).

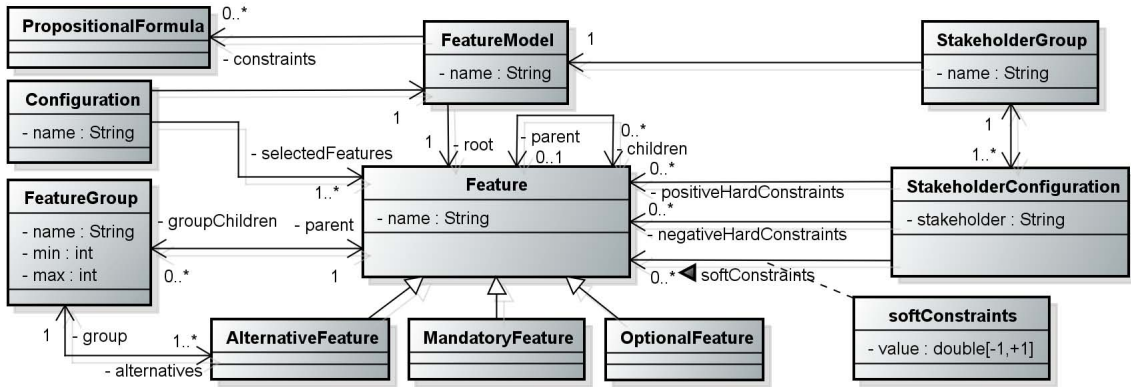


Figure 4.2: Preference-based configuration meta-model.

We next describe formally our meta-model in detail.

The possible configurations of a software or a product line are specified with a feature model, which is represented as a *tree* of features, which can be mandatory, optional, or alternative. In addition, the feature model is associated with a set of constraints, which indicate constraints over the possible configurations, such as features that are mutually exclusive. Such constraints include default constraints, such as the specification that a feature may be part of a configuration only if its parent is also part of the configuration. Formally, the feature model is defined as follows.

$$featureModel = \langle name, Features, root, Constraints \rangle$$

where *name* is the feature model identifier, *Features* are the features available in the feature model,  $root \in Features$  is the root feature of the feature model, and *Constraints* is the set of constraints over the possible configurations of the feature model. Such constraints are expressed with propositional formulas.

A feature has a type — as said above, mandatory, optional, or alternative — and may have sub-features. Mandatory and optional features are direct children of a particular feature, but alternative features are grouped into feature groups, which

are in turn children of the particular feature. Feature groups have an associated cardinality (CZARNECKI; EISENECKER, 2000), which indicate the minimum and maximum possible number of selected alternatives for a configuration. Therefore, feature and feature group are defined as shown below.

$$feature = \langle name, type, Children, GroupChildren \rangle$$

where  $name$  is the feature identifier,  $type \in \{mandatory, optional, alternative\}$  is the type of the feature,  $Children$  is the set of features  $f$ , such that  $f[type] = mandatory \vee optional$ , that are children of the feature, and  $GroupChildren$  is the set of feature groups that are children of the feature.

$$featureGroup = \langle name, min, max, Alternatives \rangle$$

where  $name$  is the feature group identifier,  $min, max \in \mathbb{Z}$  are the feature group cardinality, and  $Alternatives$  is the set of alternative features. Moreover,  $1 \leq min \leq max \leq |Alternatives|$ .

The feature model has also some restrictions related to its tree structure. This restrictions are defined as follows.

$$parent(f) = (f_p | \exists f \in f_p[Children]) \vee (f_p | \exists fg \in f_p[Alternatives] \wedge fg \in f_p[GroupChildren])$$

where  $fg$  is a featureGroup and  $f_p$  is a feature that is parent of other feature  $f$ .

$$parents(f) = \{f_p\} \cup parents(f_p)$$

$$\nexists f_p, f'_p. (f_p \neq f'_p \wedge f_p = parent(f) \wedge f'_p = parent(f))$$

$$\forall f \in f_p[Children]. (f \notin parents(f))$$

therefore, there is no feature with more than one parent, and also, there is no cycles.

A product configuration  $cfg = \langle fm, F \rangle$  specifies a set  $F$  of selected features of a feature model  $fm$ , such that  $F \subseteq fm[Features]$ . When such selected features respect the feature model constraints, the configuration is said *valid*.

Based on a feature model, stakeholders can specify their preferences over features. First, they may indicate *hard constraints*, indicating features that *must* be part or excluded of the configuration. Therefore, such hard constraints can be either positive or negative. Second, they may indicate *soft constraints* in the form of a function, indicating features that *should (or not)* be part of the configuration. Such indication is made with a real number in  $[-1, 1]$ . Positive numbers indicate positive preferences (features that should be part of the configuration), negative numbers

Feature	SH A	SH B	SH C
Network			
3G	-0.5		0.3
4G	-0.5	0.2	0.1
Wifi	+		
Cell Phone			
Video Call	0.75		
SMS		0.9	
Frontal Camera			-0.6
1MP		×	
2MP			

Table 4.1: Stakeholder configuration examples.

indicate negative preferences (features that should not be part of the configuration), and zero indicates indifference. Formally, a stakeholder configuration is defined as follows.

**Definition 1 (Stakeholder Configuration).** *A stakeholder configuration is a tuple*

$$sh\_cfg = \langle fm, HC^+, HC^-, sc \rangle$$

where  $fm$  is the feature model associated with the stakeholder configuration,  $HC^+$  is the set of features that are positive constraints,  $HC^-$  is the set of features that are negative constraints, and  $sc: Feature \rightarrow [-1, 1]$  is a soft constraint function that specifies the degree of preference for features. Moreover,  $HC^+ \cap HC^- = \emptyset$ ,  $HC^+ \cup HC^- \cup \text{dom}sc \subseteq fm[Features]$ , and  $HC^+ \cup HC^-$  must respect the feature model constraints  $fm[Constraints]$ .

In Figure 4.2, we also introduce the concept of stakeholder group that groups a set of stakeholder configurations, which must be associated with the same feature model. Table 4.1 shows examples of three stakeholder configurations (+ are positive hard constraints, and  $\times$  are negative hard constraints), associated with the feature model presented in Figure 4.1. These examples are used in next chapter to better illustrate the strategies instantiated for recommending configurations.

### 4.3 Final Remarks

In this chapter, we provided a description of our meta-model, detailing each relevant point added to the context of feature models and their structure. We also presented a motivating example in order to give a better illustration of what conflicts we intended to attack. Considering the conflicts presented, aligned with the the provided meta-model, we allow the stakeholders a way to express their preferences



in order to find an agreement between them. Given that, we present in the next chapter possible alternatives to find this consensus considering all possible valid configurations.

## 5 RECOMMENDING PRODUCT CONFIGURATIONS BASED ON STAKEHOLDERS PREFERENCES

In the previous chapter, we described the possible conflicts that may arise in the context of a multiple stakeholder scenario. Seeking an alternative to address the presented problems we explain informally and described formally our metal model for expression of preference over configurations considering multiple stakeholders. Given that, we now should combine such preferences in such a way that it is possible to identify an optimal configuration with respect to the preferences of a group of stakeholders. In order to do so, we must first define how a particular configuration satisfies an individual stakeholder according to his preferences (Section 5.1). Second, as previously said, different configurations may be optimal considering different perspectives. Therefore, in Section 5.2 we describe different social choice strategies used to provide group recommendations in the context of feature model configuration, considering the introduced stakeholder satisfaction.

### 5.1 Stakeholder Satisfaction

Given the presented meta-model in the last chapter, now we describe the concept of a Stakeholder Satisfaction which represents how much a stakeholder is satisfied by a single configuration. Preferences expressed by stakeholders according to our meta-model are associated with particular features. To understand how a configuration satisfies a stakeholder, we first consider hard constraints and then soft constraints. Hard constraints are used to express required or unacceptable features, consequently, they are used to exclude valid configurations. Configurations are excluded due to two possibilities: (i) it has at least one feature associated with a positive hard constraint that is not in the configuration selected features, and/or (ii) it has at least one feature associated with a negative hard constraint that is in the configuration selected features. This is formalised below.

**Definition 2 (Excluded Configuration).** A configuration  $cfg$  is said excluded  $exc(cfg, sh\_cfg)$  by a stakeholder, when exists a feature  $f^+ \notin cfg[F]$  such that  $f^+ \in HC^+$ , or a feature  $f^- \in cfg[F]$  such that  $f^- \in HC^-$ .

By excluding configurations, we obtain a *consideration set*, which is the set of all valid configurations excluding those that for at least one stakeholder configuration  $sh\_cfg$ , hard constraints are unsatisfied, i.e.  $exc(cfg, sh\_cfg)$  holds. Each configuration part of the consideration set is a candidate for being recommended as an optimal configuration. An example of excluded configuration would be one that does not have wi-fi as part of the selected features. As we can see, wi-fi was selected as a mandatory feature by SH A, thus, this feature must be present in the final configuration. Any configuration that does not have wi-fi will be an excluded one.

Given that we have considered hard constraints, we will now focus on soft constraints. To understand how a configuration satisfies a stakeholder, we must identify which soft constraints are satisfied by a configuration. Two cases indicate whether a soft constraint  $s$  is satisfied: (i) if the degree of preference is a positive number,  $s$  is satisfied if the associated feature is in the configuration selected features; and (ii) if the degree of preference is a negative number,  $s$  is satisfied if the associated feature is not in the configuration selected features. The set of satisfied soft constraints  $SC_{Sat}(cfg, sh\_cfg)$  is formalised next.

**Definition 3 (Satisfied Soft Constraints).** A set of features  $SC_{Sat}(cfg, sh\_cfg) \subseteq sh\_cfg[fm[Features]]$  is said the set of satisfied soft constraints of a stakeholder, when for all  $f \in SC_{Sat}$ , either  $f \in cfg[F]$  and  $sc(f) > 0$  or  $f \notin cfg[F]$  and  $sc(f) < 0$ .

Assume the configuration with the following set of selected feature:

$$F = \langle Network, 4G, Wifi, CellPhone, VideoCall \rangle$$

Considering this configuration, the presence of video call and the absence 3G constitute the set of satisfied soft constraints.

Finally, given the set of satisfied soft constraints, the stakeholder satisfaction is a function  $sh\_sat: Cfg \times SH\_Cfg \rightarrow \mathbb{R}$  that gives *how much* a particular configuration  $cfg \in Cfg$  satisfies a stakeholder configuration  $sh\_cfg \in SH\_Cfg$ . This satisfaction is the *sum* of all satisfied soft constraints, as presented next.

$$sh\_sat(sh\_cfg, cfg) = \sum_{f \in SC_{Sat}(sh\_cfg, cfg)} |sc(f)|$$

Considering the same configuration from the previous example, the Stakeholder Satisfaction would be 1.25. 0.5 from the absence of 3G and 0.75 from the presence of Video Call.

## 5.2 Social Choice Strategies

We now know how each stakeholder evaluates each of the configurations of the consideration set. This evaluation is given as a real number between zero and the number of features available in the feature model. Therefore, these numbers can be interpreted in two different ways: first, as scores given to the configuration and, second, as a (partial) ordering relation among configurations. In order to choose the optimal configurations considering a group of stakeholders, we used seven different social choice strategies (MASTHOFF, 2011) that are single-winner voting systems. Therefore, they choose one winner (i.e. optimal configuration) given a set of candidates (i.e. configurations of the consideration set). Next, we introduce each of them and indicate the mathematical property satisfied by the optimal configuration according to each strategy.

*Average (AVG)*. The AVG strategy focuses on finding the maximum overall satisfaction of stakeholders. It averages the individual stakeholder satisfactions, and selects the maximum average.

$$\max_{sh\_cfg \in SH\_Cfg} \sum \frac{sh\_sat(cfg, sh\_cfg)}{|SH\_Cfg|}$$

*Multiplicative (MULT)*. The MULT strategy also focuses on finding the maximum overall satisfaction of stakeholders, but it aggregates them in a different way: it multiplies the individual stakeholder satisfactions, and selects the maximum multiplication.

$$\max_{sh\_cfg \in SH\_Cfg} \prod sh\_sat(cfg, sh\_cfg)$$

*Most Pleasure (MP)*. The MP strategy is concerned with achieving the best possible stakeholder satisfaction, so it takes the maximum of the maximum individual stakeholder satisfaction.

$$\max \max sh\_sat(cfg, sh\_cfg)$$

*Least Misery (LM)*. The LM strategy is concerned with not letting stakeholders very dissatisfied, which may occur with the above strategies. Therefore, this strategy takes the maximum of the minimum individual stakeholder satisfaction.

$$\max \min sh\_sat(cfg, sh\_cfg)$$

*Borda Count (BC)*. The BC strategy uses the stakeholder satisfaction as an ordering relation among configurations. This gives a rank of configurations from highest satisfaction (best) to lowest configuration (worst) and, based on this rank, points are given to each configuration: the worst configuration gets 0 points, the second worst gets 1 point, and so on. Then, such points of each individual stakeholder configuration is summed up, and the maximum is selected.

$$\max \sum_{sh\_cfg \in SH\_Cfg} rank(cfg, sh\_cfg)$$

*Copeland Rule (CR)*. The CR strategy also considers the ordering relation among configurations given by the stakeholder satisfaction. It selects the configuration that has the maximum difference between how often a configuration beats other configurations (using majority vote) and how often it loses. This is expressed in the formula below, where  $Win(cfg)$  is the set of other possible configurations  $cfg'$  such that  $cfg$  beats  $cfg'$ , according to a majority rule considering all stakeholders, i.e. the majority of stakeholders agree that  $cfg$  beats  $cfg'$ . A  $cfg$  beats  $cfg'$  according to a stakeholder configuration  $sh\_cfg$  if  $sh\_sat(sh\_cfg, cfg) > sh\_sat(sh\_cfg, cfg')$ .  $Lose(cfg)$ , in turn, is the set of other possible configurations  $cfg'$  such that  $cfg'$  beats  $cfg$ .

$$\max |Win(cfg)| - |Lose(cfg)|$$

This two strategies, BC and CR, are interesting because they try to overcome a possible cheating from the stakeholders, given that the recommendations made by this strategies are not sensitive to specific degrees of preference. Thus, if a stakeholder establish a lot of high values of preference, trying to get some kind of advantage, it would not make a difference in the final result since this strategies use the degree of preference just as a way to order the configurations.

*Average Number of Preferences (NP)*. The NP strategy ignores the degrees of preference informed by stakeholders and considers just the satisfaction of soft constraints. It selects the configuration that has the maximum number of satisfied soft constraints.

$$\max \sum_{sh\_cfg \in SH\_Cfg} \frac{SC_{Sat}(cfg, sh\_cfg)}{|SH\_Cfg|}$$

Feature	AVG	MULT	MP	LM	BC	CR	NP
Network	+	+	+	+	+	+	+
3G	+	+			+	+	+
4G				+	+	+	
Wi-fi	+	+	+	+	+	+	+
Cell Phone	+	+	+	+	+	+	+
Video Call	+		+		+		+
SMS	+	+		+	+	+	+
F. Camera	+		+		+		+
1MP							
2MP	+		+		+		+
SH 1	1.25(2)	0.5(1)	1.75(3)	0.5(1)	0.75(1)	0.0(0)	0.0(0)
SH 2	1.1(2)	0.9(1)	0.0(0)	1.1(2)	1.1(2)	1.1(2)	1.1(2)
SH 3	0.1(1)	0.9(2)	0.0(0)	0.7(2)	0.4(2)	1.0(3)	1.0(3)
AVG	0.81	0.76	0.58	0.76	0.75	0.70	0.77
SD	0.39	0.18	0.82	0.25	0.29	0.50	0.25

Table 5.1: Strategy recommendations.

In some situations, there may be ties between two configurations according to a strategy, leading us to more than one optimal strategy. In this case, we select that with the lowest number of selected features. As stakeholders may be indifferent to some features, we interpret that the absence of preference indicates that a feature is not needed. If there is still a tie, we select one of the configurations randomly.

Each strategy has a particular way of selecting the optimal configuration, but the same configuration may be optimal according to different strategies. As shown in Table 5.1, this occurs with the AVG and NP strategies considering the feature model presented in Figure 4.1 and the stakeholder configurations detailed in Table 4.1. Table 5.1 shows the optimal configuration according to each of our seven possible strategies. Besides showing which features are present in each optimal configuration (using the ‘+’ symbol), we also detail the satisfaction of each stakeholder and the number of satisfied soft constraints (in parentheses). Moreover, we give the average and standard deviation of the stakeholder satisfactions.

### 5.3 Final Remarks

In this chapter, we provided a description of our meta-model, detailing each relevant point added to the context of feature models and their structure. We also presented a motivating example in order to give a better illustration of what conflicts we intended to attack. Finally, we described the strategies used to generate the possible configurations to be recommended. Given that, we present in the next

chapter a tool we developed that allows expression of preferences according to our meta-model and provides recommendations according to the presented strategies.

## 6 SACRES: A TOOL FOR RECOMMENDING OPTIMAL CONFIGURATIONS

Based on our meta-model and social choice strategies, we implemented a tool in Java — named Sacres — that is able to provide recommendations of configurations for a group of stakeholders. This tool was also used to evaluate our proposed approach, providing the features needed for the empirical study and mainly, database support, allowing us a easier way to work with the stakeholders preferences and the creation of groups of stakeholders for, finally, recommend the optimal configurations accordingly to the chosen social choice strategies.

The Sacres tool also provides relevant data about the recommended configurations. It gives us information like standard deviation, stakeholders satisfaction, number of satisfied preferences, features selected and number of possible configurations. Therefore, in Section 6.1, we describe the main features provided by Sacres and how they work. In Section 6.2, we explain how the process of searching the optimal configuration is performed.

### 6.1 Features

In the previous chapters we presented our meta-model and how it works on supporting our preference-based approach. Here we describe the three main functionalities provided by SACRES tool, which are key features in order to generate the optimal configurations and analyse them to reach the best recommendation.

*Creating a Stakeholder Group.* Our tool persists stakeholder configurations in a database, so it allows to create groups to assign such configurations to groups. When creating a group, users must provide a group name and select an associated feature model. Feature models are stored in the database. However, currently it is not possible to create or edit feature models through the SACRES interface.



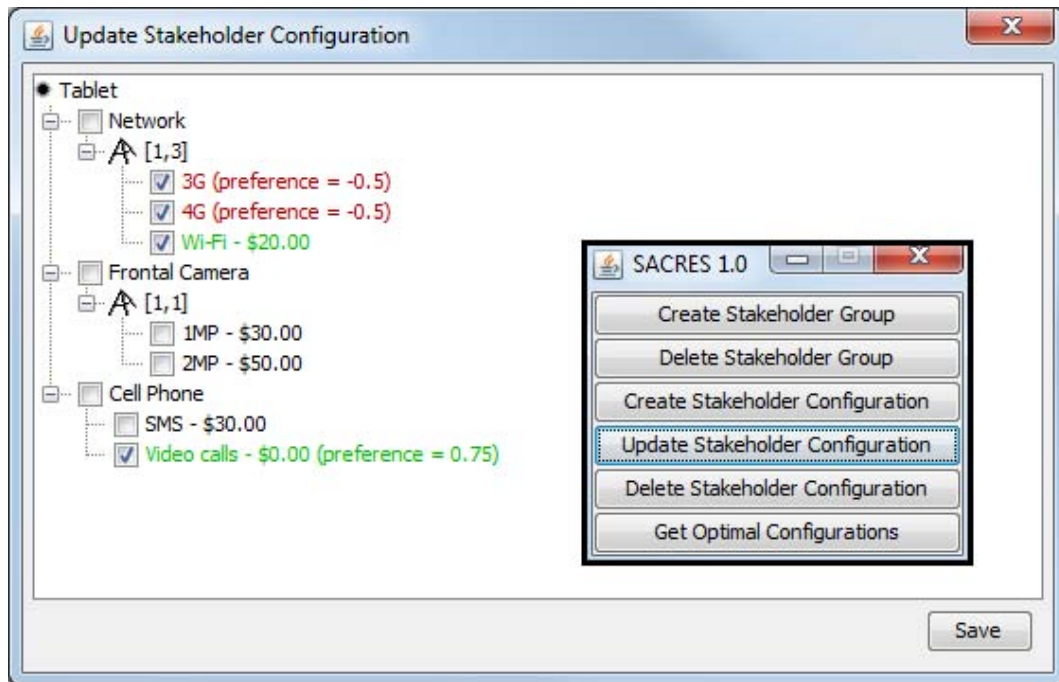


Figure 6.1: Stakeholder configuration editor.

*Creating a Stakeholder Configuration.* In order to create a stakeholder configuration, users must first provide a name for it and choose a group. Then, an instance of the feature model is created and displayed to the user. When selecting a feature, users select one of three options: (i) positive hard constraint (the selected feature will be shown in green); (ii) negative hard constraint (the selected feature will be shown in red); and (iii) soft constraint, which requires users to inform a number between -1 and 1. In this case, the selected feature will be shown in green for positive numbers and red for negative numbers, and the degree of preference will be displayed beside the feature name. A screenshot of this functionality is shown in Figure 6.1.

*Getting Optimal Configurations.* Users get a recommendation by choosing a group of stakeholders. Then, our tool performs the following steps. First, it checks whether there is intersection between all stakeholders' positive hard constraints and all stakeholder's negative constraints. If so, there is no possible solution, and stakeholders must review their hard constraints. Second, it generates the consideration set of configurations using a CSP solver. Third, it evaluates the configurations obtained with the different strategies. Finally, it displays the results as shown in Figure 6.2.

We are aware that generating the consideration set of configurations using a SAT solver is a hard problem, even though heuristics are used and results are often obtained in reasonable time. However, our main concern in this work is to *explore and compare different social choice strategies* — and each strategy requires

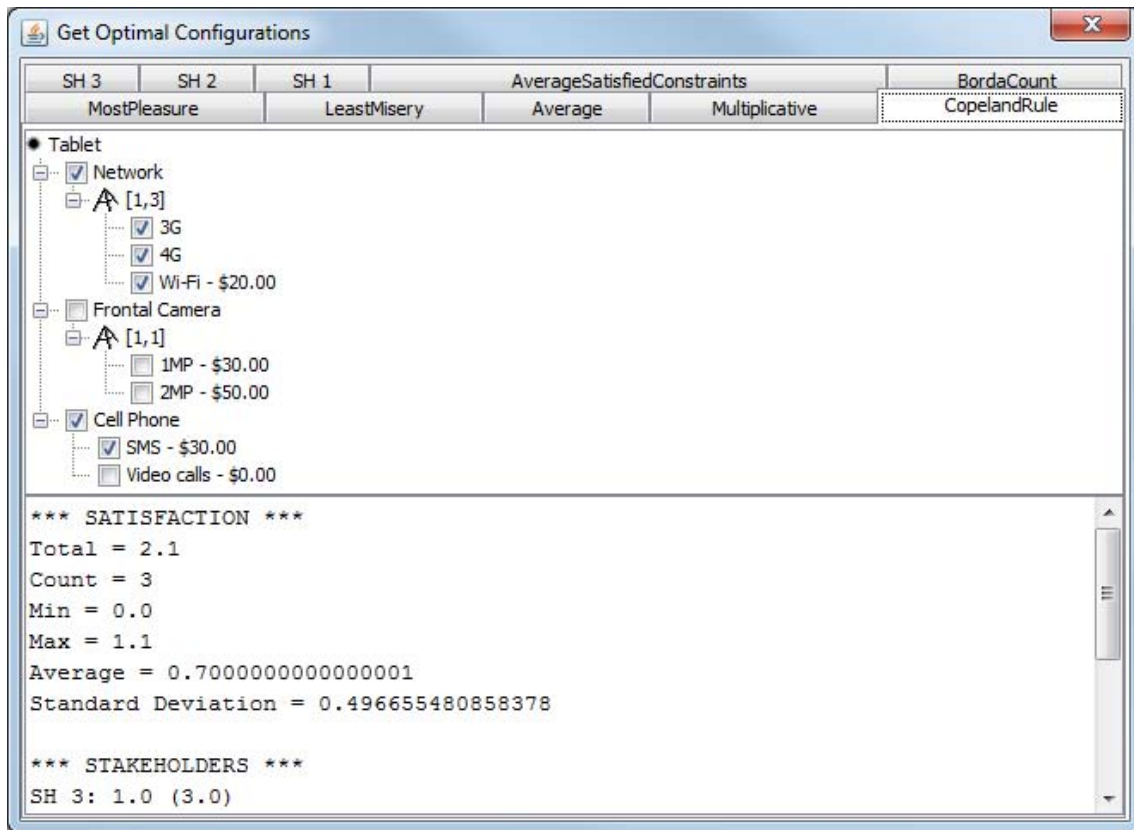


Figure 6.2: Recommendation example.

a different algorithm to get faster solutions. In spite of that, in order to improve the consideration set generation we used CHOCO <sup>1</sup>, a well known CSP solver used in many fields of study. Our main goal with the development of Sacres is to use it to evaluate the effectiveness of the different social choice strategies for product configuration, which will be shown in next chapter. This study gives indication of which strategies are better from a user perspective and should be further explored with respect to performance.

## 6.2 How Recommendations are Generated

In order to generate the optimal configurations according to the chosen strategies of social choice theory, we first seek what we called as consideration set. This consideration set is the set of all possible valid configurations (set of selected features) after considering the restrictions provides by the feature model itself, the possible constraints between features (includes and excludes relations) and the positive and negative hard constraints provided by stakeholders involved in the process of configuration.



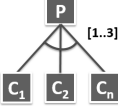
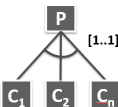
<sup>1</sup><http://choco-solver.org/?q=Choco3>

To achieve this first step, we used a CSP solver — named CHOCO — which provide us the necessary tools to improve our search of possible valid configurations, the consideration sets to be evaluated by the group recommender strategies. In order to do so, we first map the feature model, its constraints and the hard constrains given by the stakeholders to propositional formulas, similarly to the notation shown in chapter 2. Next we describe the main used rules for mapping the problem.

As previously shown, the idea of representing feature models using propositional logic is not new. However, some of the initial relations for feature model have changed along the years. The addition of cardinality to the relations between feature is one example. Therefore, we present next, the adopted rules for mapping the feature model as propositional formulas in the present work.

Based on the Feature Model presented in Figure 4.1, in Table 6.1 we separate the kinds of relations used in our running example. For each relation, we establish the adopted formalization using propositional logic and we also show the mapping between the logic and the graphical example. Some of the mapped relations are exemplified just as an example to be used. These are the “exclude” and “mandatory” relations, which are very common, but do not appear in the present example of FM. In the generic graphical representation,  $P$  represents a parent node,  $C$  represent a child node and  $A$  and  $B$  are features related by a cross-tree constraint.

Table 6.1: Mapping of feature model relations to propositional logic.

Relation	Graphical	Propositional Logic	Mapping Example
Mandatory		$P \Leftrightarrow C$	$Tablet \Leftrightarrow Network$
Optional		$C \Rightarrow P$	$FrontalCamera \Rightarrow Tablet$ $CellPhone \Rightarrow Tablet$ $VideoCall \Rightarrow CellPhone$ $SMS \Rightarrow CellPhone$
Or [1..N]		$P \Leftrightarrow (C_1 \vee C_2 \vee C_N)$	$(Network \Leftrightarrow (3G \vee 4G \vee Wifi))$
Alternative [1..1]		$(C_1 \Leftrightarrow (\neg C_2 \wedge \neg C_N \wedge P)) \wedge$ $(C_2 \Leftrightarrow (\neg C_1 \wedge \neg C_N \wedge P)) \wedge$ $(C_N \Leftrightarrow (\neg C_1 \wedge \neg C_2 \wedge P))$	$((1MP \Leftrightarrow (\neg 2MP \wedge FrontalCamera)) \wedge (2MP \Leftrightarrow (\neg 1MP \wedge FrontalCamera)))$
Implies	$A \rightarrow B$	$A \Rightarrow B$	$SMS \Rightarrow 3G \vee 4G$ $VideoCall \Rightarrow FrontalCamera \wedge Network$
Excludes	$A \rightarrow \neg B$	$\neg(A \wedge B)$	$\neg(VideoCall \wedge 2MP)$

Given the described mapping into propositional logic, we can construct the complete mapping for the example shown in Figure 4.1 simply by making the conjunction of all formulas described in Table 6.1. The complete mapping is presented in Table 6.2.

Table 6.2: Complete mapping of tablet feature model.

$Tablet \Rightarrow Network \wedge$
$FrontalCamera \Rightarrow Tablet \wedge$
$CellPhone \Rightarrow Tablet \wedge$
$VideoCall \Rightarrow CellPhone \wedge$
$SMS \Rightarrow CellPhone \wedge$
$(Network \Leftrightarrow (3G \vee 4G \vee Wifi)) \wedge$
$((1MP \Leftrightarrow (\neg 2MP \wedge FrontalCamera)) \wedge$
$(2MP \Leftrightarrow (\neg 1MP \wedge FrontalCamera))) \wedge$
$(SMS \Rightarrow 3G \vee 4G) \wedge$
$(VideoCall \Rightarrow FrontalCamera \wedge Network)$

After mapping the problem to propositional formulas we are able to construct the same feature model using CHOCO notation, adding also the hard constraints given by the stakeholders. Reaching, therefore, the formalization of the problem as a constraint programming problem, allowing us to use CHOCO Solver. In Table 6.3, we show the three main examples of mapping from propositional logic to CSP. The cross-tree constraints were also constructed by the same kind of logical relations.

Table 6.3: Mapping of propositional logic to CHOCO.

Relation	Prop. Logic	CHOCO Mapping Example
Mandatory	$P \Leftrightarrow C$	<code>solver.post(IntConstraintFactory.arithm(tablet, "=", 1));</code>
Optional	$C \Rightarrow P$	<code>solver.post(LogicalConstraintFactory.ifThen( IntConstraintFactory.arithm(net, "&gt;", 0), IntConstraintFactory.arithm(tablet, "&gt;", 0)));</code>
Group-features [1..N]	$P \Leftrightarrow (C_1 \vee C_2 \vee C_N)$	<code>solver.post(LogicalConstraintFactory.ifThenElse( IntConstraintFactory.arithm(net, "&gt;", 0), IntConstraintFactory.sum(new IntVar[] {G3, G4, wifi, or3}, IntConstraintFactory.sum(new IntVar[] {G3, G4, wifi, zero})));</code>

Following the process of generating the optimal configurations, after the generation of the consideration set, the tool can search the remain valid configurations in order to find the best one following the criteria of each strategy of social choice. Creating then, up to seven different recommended configurations.

### 6.3 Final Remarks

In this chapter, we presented our tool to support the configuration process and generation of optimal configurations, explaining its functionalities and how they work. We also explained how the recommendations are made by SACRES. In the next chapter, we describe our empirical study, where Sacres features were used, providing the necessary support for our approach.

## 7 EVALUATING THE EFFECTIVENESS OF SOCIAL CHOICE STRATEGIES

In previous sections, we presented a meta-model for establishing stakeholder preferences with both hard and soft constraints, and described social choice strategies to recommend configurations based on such constraints obtained from the stakeholders. As can be seen in Table 5.1, strategies may select different optimal configurations. Each strategy has a rationale behind the choice for an optimal configuration, and the selected optimal configurations are all correct with respect to the property specified by the strategy. However, it is not possible to tell which strategy is best, because this depends on personal judgement. Therefore, we conducted an empirical study in order to understand how people evaluate the configurations recommended by the different strategies from two points of view: *individual satisfaction* and *fairness*. Our experiment goal is described in Table 7.1, using the GQM template (BASILI; SELBY; HUTCHENS, 1986). We first describe the procedure of the study in Section 7.1, then we detail the study participants in Section 7.2, and next we present the obtained results and its analysis in Section 7.3. Finally we present a discussion about the obtained results in Section 7.4.

<b>Element</b>	<b>Our experiment goal</b>
Motivation	To understand how people evaluate recommender strategies,
Purpose	characterize and evaluate
Object	recommended product configurations based on preferences
Perspective	from a perspective of the researcher
Scope	in the context of graduate and undergraduate students in Computer Science.

Table 7.1: Goal definition.

## 7.1 Procedure

In order to evaluate the effectiveness of our strategies, we performed a three-step study. In a nutshell, we first collected participants' preferences, then used the SACRES tool to get recommendations of optimal configurations, which were then given to participants so that they could evaluate the recommendations. These three steps are detailed next.

*Step 1: Participants Preferences.* Participants were requested to imagine the following scenario. “*Imagine you were going to buy a tablet, which would be shared by a group of friends. Therefore, a tablet configuration should be adequately selected for the group. In order to do so, you should provide your individual preferences with respect to the desired configuration, and based on such preferences, you would receive recommendations of adequate configurations for the group.*” The participant received: (i) a Tablet feature model, shown in Figure 7.1, which has 41 available features; (ii) an explanation of the notation used in the feature model; and (iii) an explanation of how to express preferences (hard and soft constraints). We highlighted that the tablet is for a group of friends, and therefore if they and their friends stated many hard constraints, it would not exist a possible tablet configuration for them. Participants took around 15 minutes to complete this step of the study.

*Step 2: Recommendation Generation.* Given the provided preferences, we randomly assigned participants into groups of four. Then, we stored their stakeholder configurations in a database using the SACRES tool. With these stored stakeholder configurations, we generated the optimal configurations for each group according to the different strategies. If the informed preferences had conflicting hard constraints, we used the following approach: we analysed the conflicting hard constraints, and changed them to soft constraints using the maximum degree of preference. We next prepared the following material for each participant: (i) the stakeholder configuration of each participant of the group; and (ii) the optimal recommended configurations.

*Step 3: Configuration Evaluation.* The material we prepared in the previous step was given to the participants so that they could evaluate the optimal recommended configurations. Together with the prepared material, they received a questionnaire with the following questions. First, we asked them for their personal data to characterise our study participants. Second, they had to answer a set of three questions with respect to their *individual satisfaction* to be answered: (i) how do you evaluate each of the received configurations — from 1 (worst) to 7 (best)? (ii) which is the best configuration and why? (iii) which is the worst configuration and why? Third, the same set of questions were asked with respect to *fairness* among

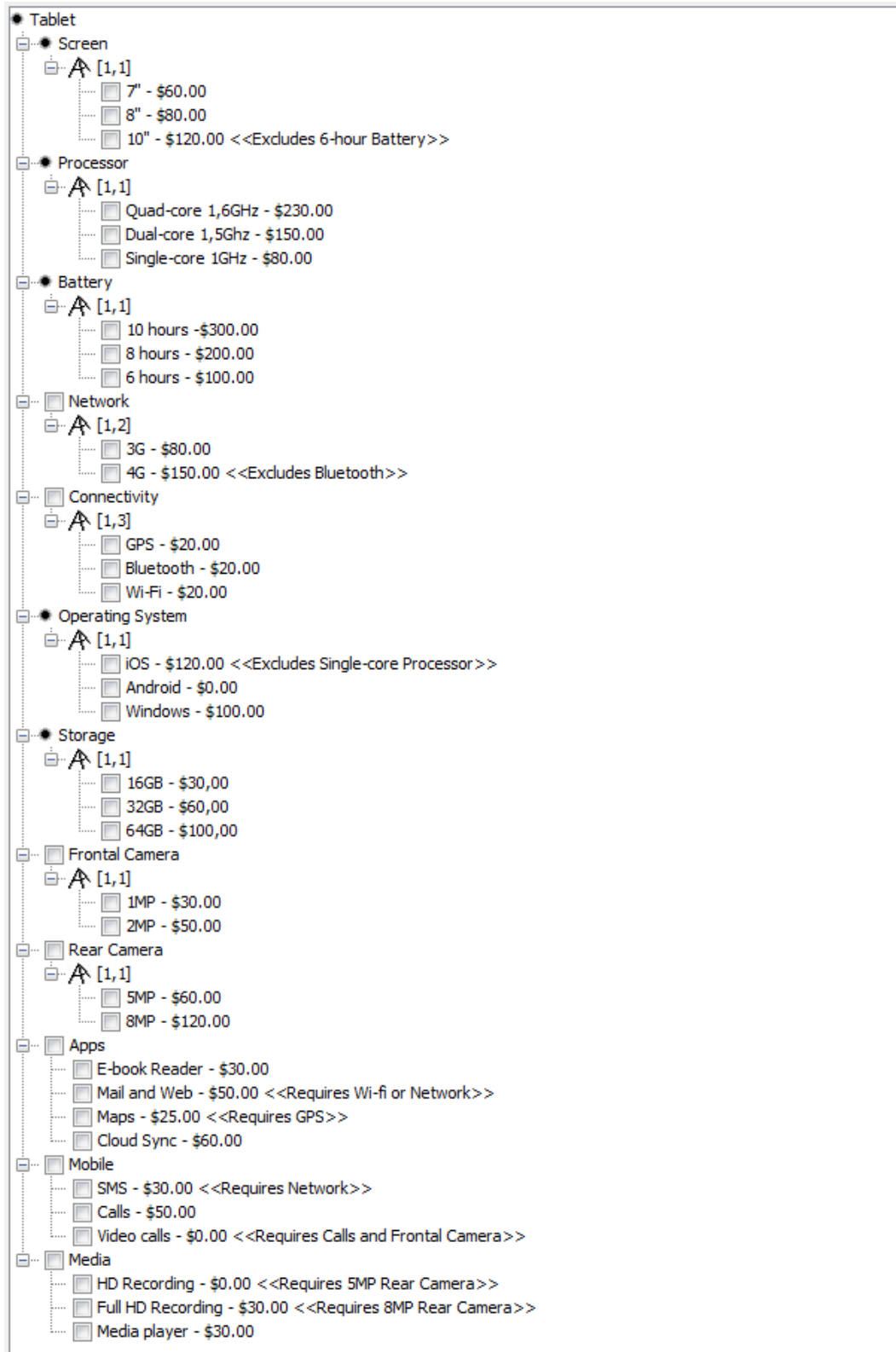


Figure 7.1: Tablet product line.

the group members. And, forth, they could provide general comments. Participants took around 20 minutes to complete this step of the study.



## 7.2 Participants

The participants of our study were selected using convenience sampling, with Computer Science graduate and undergraduate students from the university of the lead researcher of this study. Consequently, our participants are all Brazilians. We have a total of 76 participants, from which 8 were not present for step 3 of the study, therefore only 68 participants answered our configuration evaluation questionnaire. The demographic characteristics of the participants that completed the study are described in Table 7.2.

Age (years)		Knowledge about Tablets	
17–20	(15) 22.06%	None	(1) 1.49%
21–24	(21) 30.88%	Little	(4) 5.97%
25–28	(23) 33.82%	Moderate	(22) 32.84%
29–32	(4) 5.88%	Good	(31) 45.59%
33–36	(5) 7.35%	Expert	(10) 14.93%

Table 7.2: Participant characteristics.

## 7.3 Results and Analysis

Our study allowed us to collect data with respect to: (i) how participants expressed their preferences; (ii) the similarity between the strategies considering recommended optimal configuration; and (iii) how participants evaluated the different strategies. We in this section present and analyse our obtained results according to these three aspects.

### 7.3.1 Provided Preferences

We first discuss the preferences provided by participants, including those who did not participate of step 3 of the study. Although we explicitly stated to participants that if they provided too many hard constraints, a valid tablet configuration for their group would not exist, 84% of our participants provided at least one hard constraint. Usually, they provided more positive hard constraints ( $AVG = 5.59$ ) than negative hard constraints ( $AVG = 2.13$ ). Therefore, in this particular domain, participants wanted to be sure that a set of their desired features would be selected. The complete description of how participants used hard constraints is presented in Table 7.3, together with soft constraint usage.

Participants provided many soft constraints, sometimes for almost all features. This can be seen in the average number of expressed soft constraints,  $AVG =$

<b>Preference</b>	<b>AVG</b>	<b>SD</b>	<b>MIN</b>	<b>MAX</b>
Positive Hard Constraints	5.59	4.63	0	20
Negative Hard Constraints	2.13	2.82	0	15
<b>Hard Constraints</b>	7.72	6.74	0	29
Positive Soft Constraints	14.63	7.03	2	33
Positive Soft Constraint Values	0.71	0.14	0.38	1.00
Negative Soft Constraints	3.85	4.18	0	19
Negative Soft Constraint Values	0.73	0.23	0.30	1.00
<b>Soft Constraints</b>	18.48	8.10	2	39
<b>Soft Constraint Values</b>	0.71	0.14	0.48	1.00
<b>All Preferences</b>	26.20	7.99	7	40

Table 7.3: Preference analysis.

18.48, with high standard deviation,  $SD = 8.10$ . Similarly to hard constraints, participants provided (much) more positive soft constraints than negative soft constraints. We also analysed the expressed degree of preference of soft constraints. We calculated the average of soft constraint values per participant, and then the average among participants, which is  $AVG = 0.71$ . We observed that participants tend to provide high values (i.e. minimum 0.3) to express their preferences.

Finally, note that participants provided 26.22 preferences, on average, meaning that they provided constraints for more than half of the available features (on average).

### 7.3.2 Strategy Similarity

Based on the participant preferences, we used SACRES to recommend configurations. Because participants provided many hard constraints, we had to turn conflicting hard constraints into soft constraints (as said in our procedure description). This occurred for 11 (of the 19) groups.

Different strategies may recommend the same configuration, and for all groups there was at least one configuration that was recommended by more than one strategy. Running the seven strategies resulted on 4.47 different configurations, on average ( $SD = 1.12$ ,  $min = 3$ ,  $max = 6$ ). By analysing the intersection of the recommendations made by the strategies, we observed that some of them often provided the same recommendations, as shown in Table 7.4. Frequently, the AVG, MULT and CR strategies provided the same recommendation (for more than half of the groups).

Strategy	AVG	MULT	MP	LM	BC	CR	NP
AVG		12	1	2	3	13	5
MULT	12			2	5	11	4
MP	1				1	2	2
LM	2	2				2	1
BC	3	5	1			5	3
CR	13	11	2	2	5		3
NP	5	4	2	1	3	3	

Table 7.4: Optimal configuration intersection.

Strategy	Individual Satisfaction				Fairness			
	AVG	SD	MIN	MAX	AVG	SD	MIN	MAX
AVG	<i>4.69</i>	1.37	2	7	5.11	1.17	2	7
MULT	4.80	1.41	1	7	5.12	1.21	2	7
MP	4.82	1.49	1	7	4.98	1.32	1	7
LM	4.71	1.38	2	7	<i>4.74</i>	1.18	2	7
BC	<b>4.92</b>	1.24	2	7	5.05	1.22	2	7
CR	4.75	1.43	2	7	<b>5.15</b>	1.16	2	7
NP	4.91	1.34	1	7	4.88	1.38	1	7

Table 7.5: Individual satisfaction and fairness scores.

### 7.3.3 Strategy Evaluation

After providing participants with recommendations, we analysed how they evaluated strategies with respect to individual satisfaction and fairness. Note that participants were not aware of the strategies, they just received a set of configurations that were recommended to them. As we had to consider some of the hard constraints as soft constraints, we first compared the scores given by groups that required this intervention and by the remaining groups. By conducting an independent 2-group Mann-Whitney U test, we concluded that the difference between scores provided by the two sets of groups is not statistically significant, so we treated the scores given by all groups equally. Table 7.5 summarises the obtained scores for both measurements (highest values are in bold and lowest values are in italics), and Figures 7.2 and 7.3 show the frequency distributions of scores given by participants.

By analysing individual satisfaction, as expected, the results obtained for the different strategies are very similar. This was expected because a strategy that recommended a configuration that better satisfied one particular member of the group, thus being well evaluated by this member, would receive a low score by other members with preferences that conflict with those of this particular member. Therefore strategies received different scores by different group members, and this distribution of scores can be seen in Figure 7.2. A Friedman’s test indicated that the

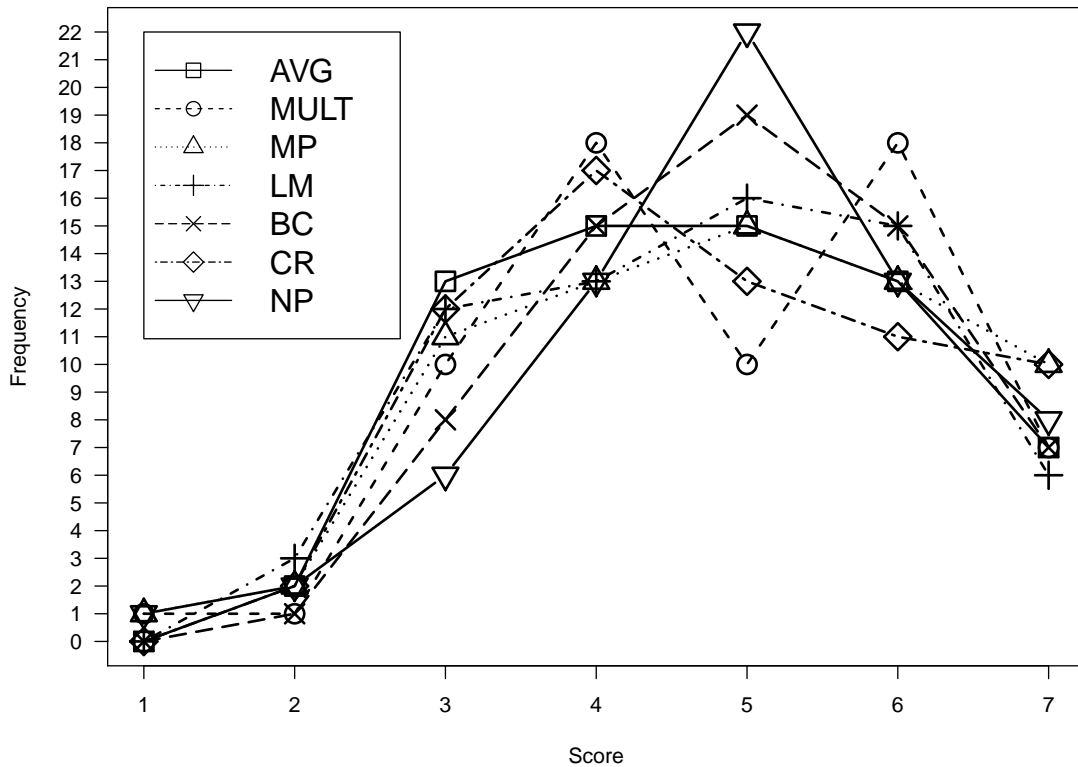


Figure 7.2: Satisfaction score frequency distribution.

difference among the average scores of the different strategies are indeed insignificant (p-value= 0.7704).

With respect to fairness, the results are different. The strategy that received the worst average score for individual satisfaction, AVG, was the third best strategy regarding fairness, and had an average score similar to those of the first and second best strategies (CR and MULT, respectively). By conducting a Friedman's test, we observed that there is a significant difference among the different strategies (p-value= 0.03263), so we further conducted the post-hoc tests of Wilcoxon-Nemenyi-McDonald-Thompson, which showed that the differences are due to LM and CR, strategies with worst and best scores, respectively. Other differences are not statistically significant, but considering that the strategies CR, AVG, and MULT often recommended the same configurations, they received the same scores by many participants, justifying the small difference on their averages.

Regardless the strategies, participants pointed out that some participants within their group made an inadequate use of hard constraints. This caused some configurations that would better satisfy the majority of the participants of the group be excluded, and consequently recommended configurations matched many preferences of the participant that used hard constraints, and not those from the majority.

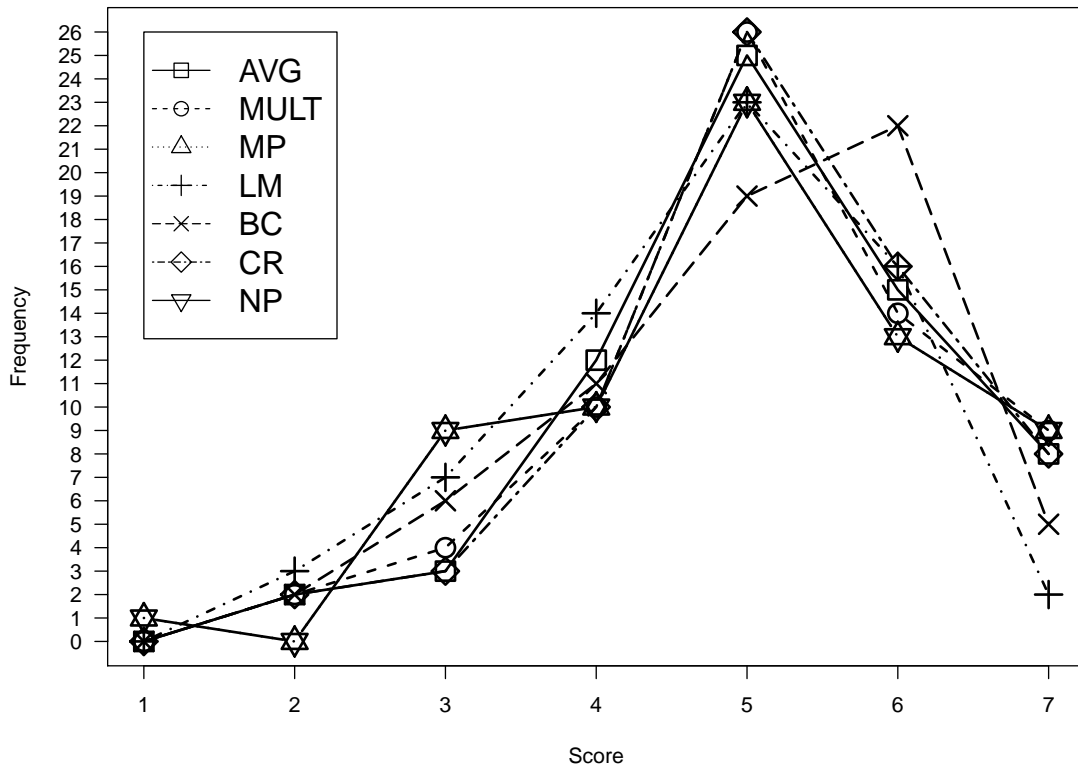


Figure 7.3: Fairness score frequency distribution.

## 7.4 Discussion

Our study results indicated that there is no best strategy to increase individual stakeholder satisfaction, which is reasonable because satisfying more one stakeholder means satisfying less another, when they have conflicting preferences. Regarding fairness, four strategies had an average score higher than 5.0. Computing the winner with two of them (CR and BC) is computationally hard, while the others (MULT and AVG) can be expressed as an optimisation problem, which with tools and heuristics can provide faster results, with similar performance. This similarity is interesting because CR and BC are not sensitive to the specific degrees of preferences provided by participants, while MULT and AVG are.

The results of our empirical evaluation of social choice strategies and our approach can be extended to other scenarios. First, although our approach focus on multiple stakeholders, part of it is also *applicable to aid a single stakeholder* to configure a feature model. Feature models usually have constraints over features, typically with expressions that state dependencies between features or features that are mutually exclusive. Therefore, a stakeholder should be aware of such constraints to select a valid product configuration. With our approach, a (single) stakeholder may express her preferences over features and the *sh\_cfg* function can be used to

select an optimal valid configuration according to the provided preferences. This can be solved by modelling an optimisation problem whose goal is to maximise *sh\_cfg*.

Second, we emphasise that our empirical evaluation provides contribution not only for feature model configuration, but also for the area of social choice. By comparing different strategies, we provided understanding of how users evaluate the different strategies, and perhaps our results can be *generalised to other domains*.

Finally, we discuss some threats to the validity of our empirical study. First, a threat to the internal valid of the study is the participants who were not present in the third step of it, and the outcomes are thus unknown for these individuals. In response, we used a relatively large sample to account for dropouts. Second, another threat to the internal valid is that participants could communicate with each other between the first and third steps of the study. We asked them to not discuss about their preferences and, because the participants were randomly assigned into groups, the probability of having two participants who discussed their preferences in the same group is low. Finally, an external validity threat is, because of the narrow characteristics of participants in the experiment, perhaps the results could be not generalisable to individuals who do not have the characteristics of participants. We selected a relatively large sample of participants to address this threat. We also had the concern that participants' preferences would be similar as their age difference is not so high, but this was not observed in the study. However, similarly to other empirical studies, it is important to replicate the study for increasing its statistical power.

## 7.5 Final Remarks

In this chapter we described an empirical study conducted in order to find which strategies perform best accordingly to individual satisfaction and fairness. We conducted the study with large number of participants in order to evaluate the relevance of the features selected in the recommended configurations. Finally, in the next chapter we provide a conclusion describing our main contributions and possible future work.

## 8 CONCLUSION

Feature model configuration is an important activity in the product derivation process of software product lines, which is known to be a hard, error-prone and time-consuming activity. When there are multiple stakeholders involved in this activity, it can be very complicated, due to conflicting stakeholder preferences over features.

In this work, we proposed a novel technique that improves the multi-stakeholder configuration process, by allowing stakeholders to express preferences over features of a feature model and recommending optimal configurations according to these preferences. This prevents stakeholders to constrain their decisions to prior decisions made by other stakeholders. Our approach exploited single-winner voting strategies from the social choice theory, in order to provide group recommendations. Our key contributions are a meta-model that captures stakeholder configurations with their preferences, the contextualisation of such strategies for our problem, and a tool named *Sacres*, which implements our approach. We conducted an empirical study to evaluate the effectiveness of our strategies from two points of view: individual stakeholder satisfaction and fairness among all stakeholders. Results indicate that the best strategies to provide fair recommendations are **Multiplicative** and **Average**, as they were better evaluated by our study participants and are computationally less expensive than others with similar participant evaluation (Borda Count and Copeland Rule). We next present the main contributions contributions of our approach (Section 8.1) and possible future work (Section 8.2).

### 8.1 Contributions

Given the results presented in this dissertation, we list below our main contributions.

**Meta-model for Stakeholders Preferences** Most of the work related to feature model configuration uses only hard constraints as way to express the stakeholders preferences. In our presented meta-model we provide an alternative for the stakeholders, allowing them to provide different levels of preference, facilitating the process of configuration by making it more flexible. The meta-model compatible with our novel approach using soft constraints also overcomes some issues related to feature model configuration. They ease the activity of configuration not only by the expression of preferences, but also because the decisions of one stakeholder do not constrain further choices from other stakeholders. Therefore, the use of this approach contributes for the generation of configuration trying to satisfy the most all the parts involved in the task of FM configuration.

**Social Choice Strategies** The instantiation of social choice strategies for our approach is a valuable contribution for both SPLs and social choice theory. The application of many strategies for recommending optimal configuration together with the empirical study evaluating their effectiveness provided good results and analysis. We were able to determine which strategies performed best in the task of recommending a tablet for a group of people. It is important to notice that this results may also be extended to other fields of study, since the proposed strategies are not strictly to be used in the present scenario.

**SACRES Recommender Tool** In order create a stronger support to our meta-model and preference-based approach, we developed the SACRES tool, which provide the necessary features to apply our approach e test it with the many chosen social choice strategies. The tool brings good functionalities for the creation and edition of configurations and groups of stakeholders. Finally, considering the obtained results, this tool may be of great value as a recommender system, becoming even more valuable, if we consider a continuous improvement and possible extensions to it.

## 8.2 Future Work

The developed approach improved the process of configuration by helping stakeholders to reach an agreement between them. The preferences are expressed using positive and negative numbers. However, one alternative that could be tested would be to consider pre-established discrete levels of preference instead of rational numbers between -1 and 1. We can also identify other limitations in our approach, which leads to possible extensions to our approach. We limit stakeholders to provide preferences solely associated with individual features, and we do not allow them to



specify preferences over *groups of features* or conditional preferences. For example, stakeholders may need to express preferences like “*if feature X is part of the configuration, then I would like feature Y to also be present.*”

Another issue that we do not take into account is *power* of stakeholders. Some stakeholders’ preferences may be more important than others, for instance, a technical expert opinion may be more important than that of a regular user of the product being configured. In our approach, all stakeholders are treated equally. Therefore, it is interesting to consider weights for stakeholder configurations, or even using a lexicographic approach. A possible extension could be the analysis and use of rationales, in order to create an even more realistic scenario, maybe considering an study where the participants were or pretend to be members of a company with distinct needs about a future configuration or recommendation.

Given the Sacres tool, it could be extended to work with other tools like FeatureIDE<sup>1</sup> or pure::variants<sup>2</sup> as a plugin or something similar. And considering the results obtained in the study, future work may be to develop specific algorithms for the strategies that obtained the best results.

Despite all the provided contributions by our approach, tool and experiment, there is still lots of work to be done and areas to be explored.

---

<sup>1</sup>[http://wwiti.cs.uni-magdeburg.de/iti\\_db/research/featureide/](http://wwiti.cs.uni-magdeburg.de/iti_db/research/featureide/)

<sup>2</sup><http://www.pure-systems.com/>

## REFERENCES

AMERICA, P. et al. COPA: a component-oriented platform architecting method for families of software intensive electronic products. In: THE FIRST CONFERENCE ON SOFTWARE PRODUCT LINE ENGINEERING. **Proceedings...** [S.l.: s.n.], 2000.

ANTKIEWICZ, M.; CZARNECKI, K. FeaturePlugin: feature modeling plug-in for eclipse. In: OOPSLA WORKSHOP ON ECLIPSE TECHNOLOGY EXCHANGE, 2004., New York, NY, USA. **Proceedings...** ACM, 2004. p.67–72. (eclipse '04).

ARROW, K. J.; SEN, A. K.; SUZUMURA, K. (Ed.). **Handbook of social choice and welfare**. [S.l.]: Elsevier, 2002.

ATKINSON, C. et al. **Component-based Product Line Engineering with UML**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

BAGHERI, E. et al. Configuring Software Product Line Feature Models Based on Stakeholders' Soft and Hard Requirements. In: INTERNATIONAL CONFERENCE ON SOFTWARE PRODUCT LINES: GOING BEYOND, 14., Berlin, Heidelberg. **Proceedings...** Springer-Verlag, 2010. p.16–31. (SPLC'10).

BASIL, V. R.; SELBY, R. W.; HUTCHENS, D. H. Experimentation in software engineering. **IEEE Trans. Softw. Eng.**, Piscataway, NJ, USA, v.12, n.7, p.733–743, 1986.

BATORY, D. Feature Models, Grammars, and Propositional Formulas. In: INTERNATIONAL CONFERENCE ON SOFTWARE PRODUCT LINES, 9., Berlin, Heidelberg. **Proceedings...** Springer-Verlag, 2005. p.7–20. (SPLC'05).

BENAVIDES, D. et al. Automated Analysis in Feature Modelling and Product Configuration. In: **Safe and Secure Software Reuse**. [S.l.]: Springer, 2013. p.160–175.

BENAVIDES, D.; TRINIDAD, P.; RUIZ-CORTES, A. Automated Reasoning on Feature Models. In: INTERNATIONAL CONFERENCE ON ADVANCED INFOR-

MATION SYSTEMS ENGINEERING, 17., Berlin, Heidelberg. **Proceedings...** Springer-Verlag, 2005. p.491–503. (CAiSE'05).

CZARNECKI, K.; EISENECKER, U. W. **Generative Programming: methods, tools, and applications.** New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000.

CZARNECKI, K.; HELSEN, S.; EISENECKER, U. Staged configuration using feature models. In: SOFTWARE PRODUCT LINES: THIRD INTERNATIONAL CONFERENCE, SPLC 2004. **Proceedings...** Springer-Verlag, 2004. p.266–283.

CZARNECKI, K.; HELSEN, S.; EISENECKER, U. W. Staged configuration through specialization and multilevel configuration of feature models. **Software Process: Improvement and Practice**, [S.l.], v.10, n.2, p.143–169, 2005.

CZARNECKI, K.; SHE, S.; WASOWSKI, A. Sample Spaces and Feature Models: there and back again. In: INTERNATIONAL SOFTWARE PRODUCT LINE CONFERENCE, 2008. SPLC '08., 12. **Proceedings...** [S.l.: s.n.], 2008. p.22–31.

CZARNECKI, K.; WASOWSKI, A. Feature Diagrams and Logics: there and back again. In: INTERNATIONAL SOFTWARE PRODUCT LINE CONFERENCE, 11., Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2007. p.23–34. (SPLC '07).

FELFERNIG, A. et al. Intelligent Support for Interactive Configuration of Mass-Customized Products. In: MONOSTORI, L.; VÁNCZA, J.; ALI, M. (Ed.). **Engineering of Intelligent Systems.** [S.l.]: Springer Berlin Heidelberg, 2001. p.746–756. (Lecture Notes in Computer Science, v.2070).

HUBAUX, A.; CLASSEN, A.; HEYMANS, P. Formal modelling of feature configuration workflows. In: SOFTWARE PRODUCT LINE CONFERENCE, SPLC '2009. **Proceedings...** [S.l.: s.n.], 2009. p.221–230.

JUNIOR, C. M.; CIRILO, E.; LUCENA, C. Assisted User-Guidance in Collaborative and Dynamic Software Product Line Configuration. In: CIBSE11. **Proceedings...** [S.l.: s.n.], 2011.

KANG, K. C. et al. FORM: a feature-oriented reuse method with domain-specific reference architectures. **Ann. Softw. Eng.**, Red Bank, NJ, USA, v.5, p.143–168, Jan. 1998.

KANG, K. et al. **Feature-oriented domain analysis (FODA) feasibility study.** [S.l.: s.n.], 1990. (CMU/SEI-90-TR-021).

KASTNER, C. et al. FeatureIDE: a tool framework for feature-oriented software development. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 31., Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2009. p.611–614. (ICSE '09).

LIN, Y.; YE, H.; TANG, J. An Approach to Efficient Product Configuration in Software Product Lines. In: INTERNATIONAL CONFERENCE ON SOFTWARE PRODUCT LINES: GOING BEYOND, 14., Berlin, Heidelberg. **Proceedings...** Springer-Verlag, 2010. p.16–31. (SPLC'10).

LINDEN, F. J. v. d.; SCHMID, K.; ROMMES, E. **Software Product Lines in Action: the best industrial practice in product line engineering.** Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

LOCHAU, M. et al. **Extended Version of Automated Verification of Feature Model Configuration Processes based on Workflow Petri Nets.** [S.l.]: TU Braunschweig, 2013. (Informatik-Bericht 2013-01).

MASTHOFF, J. Group Recommender Systems: combining individual models. In: **Recommender Systems Handbook.** [S.l.]: Springer, 2011. p.677–702.

MATINLASSI, M. Comparison of software product line architecture design methods: copa, fast, form, kobra and qada. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 2004. ICSE 2004., 26. **Proceedings...** [S.l.: s.n.], 2004. p.127–136.

MATINLASSI, M.; NIEMELÄ, E.; DOBRICA, L. Quality-driven architecture design and quality analysis method, A revolutionary initiation approach to a product line architecture. **Technical Research Centre of Finland,** Esbo, Finland, 2002.

MENDONÇA, M.; BARTOLOMEI, T. T.; COWAN, D. Decision-making Coordination in Collaborative Product Configuration. In: ACM SYMPOSIUM ON APPLIED COMPUTING, 2008., New York, NY, USA. **Proceedings...** ACM, 2008. p.108–113. (SAC '08).

MENDONÇA, M. et al. Efficient Compilation Techniques for Large Scale Feature Models. In: INTERNATIONAL CONFERENCE ON GENERATIVE PROGRAMMING AND COMPONENT ENGINEERING, 7., New York, NY, USA. **Proceedings...** ACM, 2008. p.13–22. (GPCE '08).

PARNAS, D. L. Software Fundamentals. In: HOFFMAN, D. M.; WEISS, D. M. (Ed.). . Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1976. p.193–213.

PILLAT, R. et al. Ensuring Consistency of Feature-based Decisions with a Business Rule System. In: VAMOS '13. **Proceedings...** [S.l.: s.n.], 2013. p.1–8.

POHL, K.; BÖCKLE, G.; LINDEN, F. J. v. d. **Software Product Line Engineering: foundations, principles and techniques**. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

ROSSI, F.; BEEK, P. v.; WALSH, T. **Handbook of Constraint Programming (Foundations of Artificial Intelligence)**. New York, NY, USA: Elsevier Science Inc., 2006.

THUM, T. et al. Abstract Features in Feature Modeling. In: INTERNATIONAL SOFTWARE PRODUCT LINE CONFERENCE, 2011., Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2011. p.191–200. (SPLC '11).

TSANG, E. **Foundations of Constraint Satisfaction**. [S.l.]: Academic Pr, 1993.

WEISS, D. M.; LAI, C. T. R. **Software Product-line Engineering: a family-based software development process**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

WHITE, J. et al. Automated Reasoning for Multi-step Feature Model Configuration Problems. In: INTERNATIONAL SOFTWARE PRODUCT LINE CONFERENCE, 13., Pittsburgh, PA, USA. **Proceedings...** Carnegie Mellon University, 2009. p.11–20. (SPLC '09).

## 9 QUESTIONNAIRES

### 9.1 Tablet Configuration for Group

#### Scenario

Imagine you will buy a tablet that will be of shared use between you and your colleagues. Thus, you should find a tablet configuration that is suitable for the group.

For this, you (and your colleagues) will provide your preferences individually about the desired configuration. Based on these preferences, you will receive appropriate recommendations of configurations for the group.

#### Tablet Features

In the end of this document, you find a model that describes all the options available for the tablet. You should interpret the model as follows:

Symbol	Name	Description
<input checked="" type="radio"/> Screen	Mandatory Feature	Feature that must be present in the tablet.
<input type="checkbox"/> GPS - R\$50,00	Optional or Alternative Feature	Feature that may or may not be present in the tablet. The alternative feature , in particular, is part of a group of which the number of selected features must comply with a restriction (see below) .
<b>A</b> [1,3]	Group of Feature with Cardinality	The group indicates a cluster of features that should be selected respecting a cardinality . The first number indicates the minimum number of characteristics that must be selected and the second represent the maximum number.
«Excludes 6-hour battery»	Constraint	The selection involves selecting some features (or not) other , and this is described in the constraint that appears on some features.

## Preferences over tablet features

You should express your preferences about the tablet's features. You must indicate:

- ✓ For features you want MANDATORILY. That is, you do not accept to buy a tablet with colleagues if the tablet has not this feature.
- × For the features you DO NOT want MANDATORILY. That is, you do not accept to buy a tablet with colleagues if the tablet has this feature.

Value between 0.0 (excluded) and 1.0 at the side of the feature: for the features you WISH (very much or little) that the tablet has. The greater the value, the greater the desire of the characteristic.

Value between 0.0 (excluded) and -1.0 at the side of the feature: for the features you DO NOT WISH (very much or little) that the tablet has. The lower the reported value, the lower the desire on the feature.

No indication or value 0.0 at side of the feature: you are indifferent to whether or not the feature is selected.

## Observations

Remember that if you select characteristics in mandatory form that do not respect the possible configurations, you will not receive any configuration. For example, it is not possible to have a Screen Product in 7 inches and 8 inches at the same time.

Remember that if you select characteristics in mandatory form and your colleagues too, you will not receive any configuration, because your mandatory preferences may not be satisfied.

When entering your preferences about the features, picture a real scenario, taking into account the value of each one of the features. The value of the tablet will be the sum of the features present in it.

The cardinality of the group indicates the characteristics that must be present in the tablet configuration, but not in the preferences. For example, you can indicate a high preference for one of the options, medium of a second option, and negative for a third option.

### Identification - Name:

Your name is requested because in the next meeting you will receive feedback on your configuration.

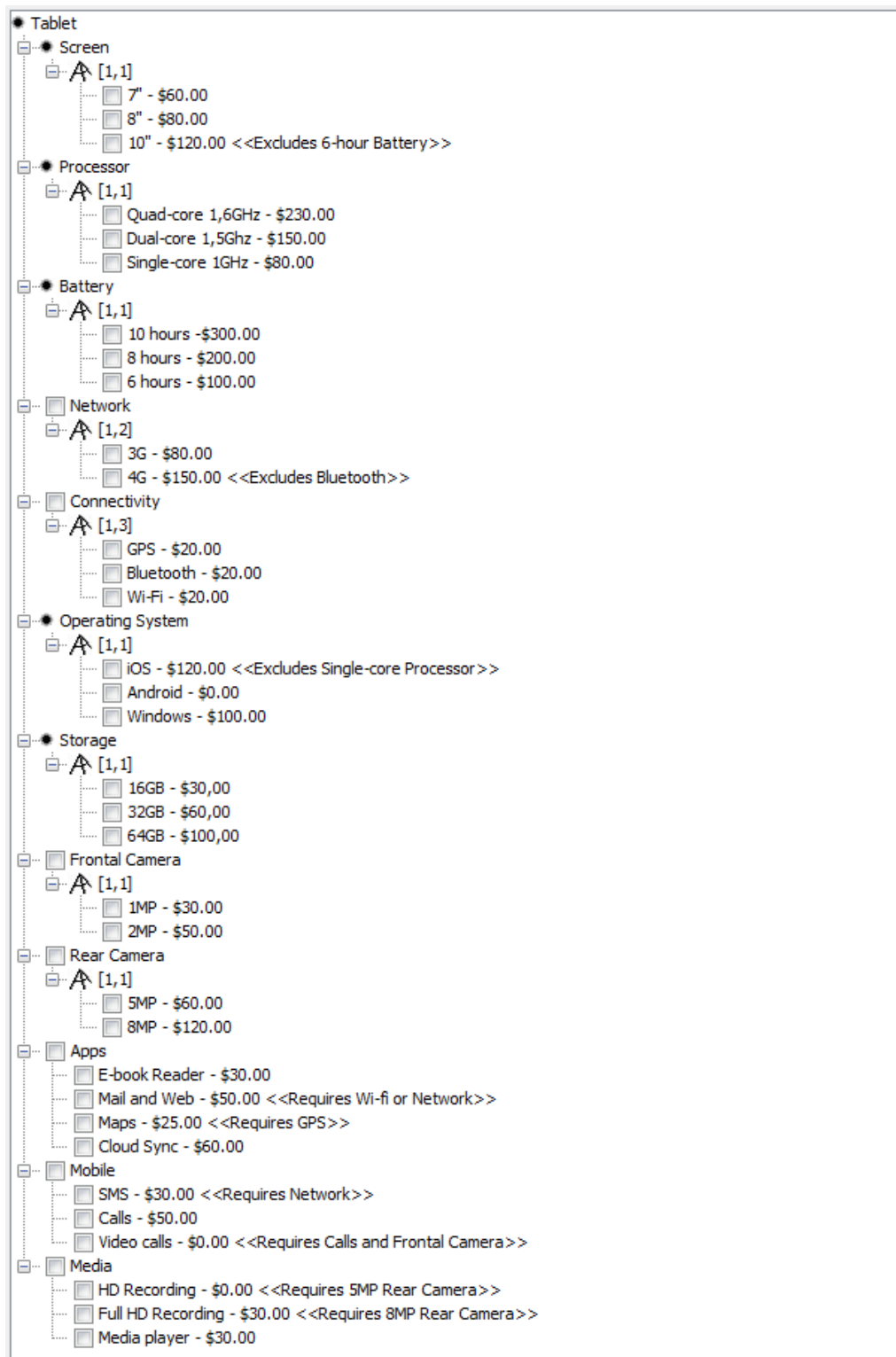


Figure 9.1: Tablet product line.



## 9.2 Questionnaire about Tablet Configuration for Group - Instructions

*Consider the suggestions of configurations generated from the preferences previous selected by the participants of the group. This suggestions seek to evaluate the relation between personal and group satisfaction.*

### 1. Personal Data Group:

Name:

Age:

City:

Profession:

Knowledge about tablets:

( )None ( )Little ( )Moderate ( )Good ( )Expert

### 2. Evaluation of the recommended configurations in relation to your preferences.

2.1. From the analysis of the suggested configurations , what is your level of **personal satisfaction** with regard to these configurations? (1 very dissatisfied and 7 very satisfied).

	1	2	3	4	5	6	7
Configuration 1							
Configuration 2							
Configuration 3							
Configuration 4							
Configuration 5							
Configuration 6							
Configuration 7							

2.2. With respect to **your** preference, which of these configurations has **best** adapted to you?

2.3. What determined your choice (with regard to the previous question)?

2.4. With respect to **your** preference, which of these configurations has **worst** adapted to you?

2.5. What determined your choice (with regard to the previous question)?

**3. Evaluation of the recommended configurations in relation to fairness considering the group preferences.**

3.1. From the analysis of the suggested configurations, what is the level of **group satisfaction (fairness)** with regard to these configurations? (1 very dissatisfied and 7 very satisfied).

	1	2	3	4	5	6	7
Configuration 1							
Configuration 2							
Configuration 3							
Configuration 4							
Configuration 5							
Configuration 6							
Configuration 7							

3.2. With respect to the preference of the **group**, which of these configurations has **best** adapted to you?

3.3. What determined your choice (with regard to the previous question)?

3.4. With respect to the preference of the **group**, which of these configurations has **worst** adapted to you?

3.5. What determined your choice (with regard to the previous question)?

**4. Other comments.**

If you consider necessary, cite suggestions or make comments about the recommended configurations.