
ON THE SELECTION AND ANALYSIS
OF SOFTWARE PRODUCT LINE
IMPLEMENTATION COMPONENTS
USING INTELLIGENT TECHNIQUES

JORGE LUIS RODAS SILVA



ADVISED BY:
PHD. DAVID BENAVIDES AND PHD. JOSÉ A. GALINDO

INTERNATIONAL DOCTORAL DISSERTATION

First published in July 2019 by
The Department of Computer Languages and Systems
ETSI Informática
Avda. de la Reina Mercedes s/n
Sevilla, 41012. SPAIN

Copyright © MMXIX Jorge Luis Rodas Silva
jrodass@unemi.edu.ec

Classification (ACM 2012):

Categories and subject descriptors:

[100] - Software and its engineering: Software development process management.

[100] - Software and its engineering: Software product lines.

[500] - Information systems: Recommender systems.

General Terms: Design, Theory, Algorithms, Performance

Additional Key Words and Phrases: software product lines, feature models, automated analysis, recommender systems, implementation components.

Support: This work has been partially supported by University of Milagro (UNEMI) with its scholarship program. It has also been partially funded by the EU FEDER program, the MINECO project OPHELIA (RTI2018-101204-B-C22); the Juan de la Cierva postdoctoral program; the TASOVA network (MCIU-AEI TIN2017-90644-REDT); and the Junta de Andalucía METAMORFOSIS project.

Don David Benavides, profesor titular del Área de Lenguajes y Sistemas Informáticos de la Universidad de Sevilla y Don José A. Galindo, investigador Juan de la Cierva.

HACEN CONSTAR

que Don Jorge Luis Rodas Silva , Ingeniero en Sistemas Computacionales por la Universidad Estatal de Milagro, Ecuador; ha realizado bajo nuestra supervisión el trabajo de investigación titulado

*On the Selection and Analysis of Software
Product Line Implementation Components
using Intelligent Techniques*

Una vez revisado, autorizamos el comienzo de los trámites para su presentación como tesis doctoral al tribunal que ha de juzgarlo.

Fdo. PhD. David Benavides y PhD. José A. Galindo
Área de Lenguajes y Sistemas Informáticos
Universidad de Sevilla,
Sevilla, julio de 2019

Yo, Jorge Luis Rodas Silva, con número de DNI 0921633988,

DECLARO

Ser el autor del trabajo que se presenta en la memoria de esta tesis doctoral que tiene por título:

*On the Selection and Analysis of Software
Product Line Implementation Components
using Intelligent Techniques*

Lo cual firmo en Sevilla, julio de 2019.

Fdo. Jorge Luis Rodas Silva

In addition to the committee in charge of evaluating this dissertation and the two supervisors of the thesis, it has been reviewed by the following researchers:

- PhD. Edward Mauricio Alferez Salinas (Université du Luxembourg, Luxembourg)
- PhD. Jaime Chavarriaga (Universidad de los Andes, Colombia)

A mis padres

Contents

Acknowledgements	19
Abstract	21
Resumen	25

I Preface

1 Introduction	31
1.1 Overview	32
1.2 Research method	35
1.3 Contributions	35
1.3.1 Summary of contributions	35
1.3.2 Publications in chronological order	37
1.3.3 Tools	40
1.4 Research internships and collaborations	41
1.5 Structure of this dissertation	42

II Background information

2 Software product lines	47
2.1 Introduction	48
2.2 Feature models	48
2.3 Automated analysis of feature models	52
2.4 Constraint satisfaction problems	53
2.5 Application engineering	55
2.6 Summary	56
3 Recommender systems	57

3.1	Introduction	58
3.2	Classification of recommender systems	59
3.2.1	Collaborative-based recommender systems	60
3.2.2	Content-based recommender systems	64
3.2.3	Hybrid approaches	65
3.3	Evaluation metrics for recommender systems	66
3.4	Summary	68

III Contributions

4	Motivation	71
4.1	Introduction	72
4.2	Analysis of current solutions	72
4.2.1	Product line engineering	73
4.2.2	Feature interaction	73
4.2.3	Model-driven architecture	74
4.2.4	Customization techniques in software product lines	74
4.3	Discussion	75
4.3.1	Automated analysis of two-layered feature models	75
4.3.2	Recommender systems in SPL	77
4.4	Summary	81
5	Automated analysis of two-layered feature models	83
6	Implementation components selection using recommender systems	85

IV Validation

7	MAYA: Putting variability at the application and systems level.	89
8	RESDEC: Using recommender systems for SPL components selection	91

V Final Remarks

<i>Contents</i>	13
-----------------	----

9 Conclusions and future work	95
9.1 Conclusions	95
9.2 Future work	97
9.2.1 Two-layered feature models	98
9.2.2 Implementation components selection	99

VI Appendix

A RESDEC: Online Management Tool for Implementation Components Selection	105
A.1 Introduction	106
A.2 RESDEC Tool Suite	106
A.2.1 Architecture	107
A.2.2 Web Application	108
Bibliography	117

List of Figures

1.1	Websites mass-customization	33
1.2	Overview of this thesis scope	34
1.3	Chronological order of conferences and publications	38
1.4	Stays to cope with the PhD objectives	41
2.1	Customization of an SPL in the mobile phone industry (from [94]) ..	48
2.2	Feature model inspired by the web development industry (from [80])	50
2.3	Features model that describes a list of components for an e-commerce website	51
2.4	Mapping from feature model to CSP (from [15])	54
2.5	SPL engineering process	55
3.1	General classification of recommender systems	60
3.2	Scheme for the design and evaluation of recommender systems(from [27])	66
A.1	RESDEC architecture	107
A.2	RESDEC web application	108
A.3	RESDEC login	109
A.4	RESDEC main screen	110
A.5	Cold start scenario	111
A.6	Recommendations of implementation components based on ratings	112
A.7	Recommendations of implementation components based on features	113
A.8	Case Study applied to e-commerce Website	114
A.9	Screen "You may also like" that show recommendation alternatives	115
A.10	Screen of RESDEC tool demo	116

List of Tables

3.1	Metrics and methods for evaluating the performance of recommender systems (from [27])	68
4.1	MAYA vs. other proposals	75
4.2	RESDEC vs. others proposals	77

Acknowledgements

I remember my mother telling me since I was little “Everything you propose in life you will achieve: persevere and resist

these words marked my life and became my encouragement voice to cope with the hard facets that appeared along the way and to be able to culminate this long process.

When arriving this stage, many things cross my mind, it has been years of ups and downs, but at the same time of a lot of learning both academically and personally. Undoubtedly, perseverance and steadiness have been fundamental pillars to make it.

First I want to thank God for having been my light and my guide; then, to the people who accompanied me during this journey and who were an important part of this puzzle. I would like to express immense gratitude to my parents and brothers for having always been there even when thousands of miles away separated us, at the beginning it was hard, but I can tell that it was worth it. To my friends of studies, of long days, with whom we have shared experiences (good and bad) but that in the end allowed us to enjoy the landscape while we were advancing in this adventure that ends today.

Then, I want to thank the institution that has trusted me since always, my second family, the place where I was formed and which I am part of now, my beloved State University of Milagro. Thanks to its authorities for giving me the opportunity to fulfill one of my great wishes, to be able to study abroad. I have not disappointed them. Also, to INRIA and Professor Benoit Baudry, for opening the doors of this research center to me and having welcomed me in their DIVERSE working group for approximately seven months.

How can I leave aside Mike, Jorge, Juliana and Mathieu, people I met during this doctoral process and whose contributions supported the fulfillment of the objectives outlined in this thesis. To my friends from the development clan, Robert and Samuel, thank you for being there and holding out my hand.

Finally, special thanks to my supervisors David and José, to the first for the patience and for teaching me to overcome each challenge I assumed; to the

second, for having been a counselor and support all the time.

Abstract

In recent years and with increasing technological advancement companies are no longer focused exclusively on designing a product for a customer (e.g. designing a website for Decameron Hotel), but on producing for a domain (e.g. designing websites for hotels.); that is, designing a product that can easily adapt to the different variations that may exist for the same product and that fits the individual tastes of customers.

In software engineering, this can be achieved through the management of Software Product Lines (SPL). A software product line is defined as a set of systems that share a common set of features that satisfy the demand of a specific market. SPL intends to reduce the effort and cost to implement and maintain a set of similar software products over time. However, managing variability in these systems is a difficult task, the greater the number of products, the more complex it is to manage them.

Feature models (FMs) are used to represent the common and variable parts of an SPL. Given the high number of products that can be derived from a feature model (FM), the management of feature models become a challenge task. The Automated Analysis of Feature Models (AAFM) is about the use of automated tools to extract information from feature models. The AAFM deals with feature model complexity. However, with current solutions, there are certain scenarios where the configuration of a product becomes a complex activity given the number of components that could exist to implement a certain feature and that feature models can be more complex in terms of layers, features or implementation components.

In this thesis, we explore intelligent techniques to solve two problems that arise when managing an SPL:

- i. On the one hand, we have identified the problems that arise when a developer wants to keep up their applications with the latest advances in technology. The close relationship between application features and platform components that realize these features is hard to track. Developers need to be aware of the consequences on existing applications when the hardware where it will run changes, e.g., when an application shall be

- ported from a smartphone to a tablet computer or more generally, when a platform shall be upgraded to a new version. The different screen sizes and resolutions, the possible absence of a cellular radio or the increased amount of memory may all have positive or negative impacts on an application. In this context, since the features of the application and platform are conceptually separated, their features can be modeled in two separate models. Consequently the traceability between these two layers and the changes that certain features of one layer may affect the other is a problem to be addressed.
- ii. On the other hand, we have found how complicated it is for the application developer to configure a product when there are variety of implementation components for each feature. For example, a web developer in WordPress manually searches for those components (plugins) that are feasible and most optimal for each web site. This task takes time and does not always guarantee that the selected components are the most suitable (in terms of quality) for the required application. Two scenarios could arise during this configuration: first, the empirical selection of a component, in practice, may not provide the expected results. Furthermore, not having criteria based on other users' experience regarding these components, could induce a bad selection and achieve a bad experience for the end user. In this context, managing the relationship between implementation components and their features is another problem to be solved.

Specifically, the contributions of this thesis are detailed below;

Multi-layer feature models: In this area we introduce a framework for the analysis of multi-layer feature models called MAYA. The objectives we pursue with this solution are: i) modeling the variability of software systems in two layers, i.e. a dependency mapping between two feature models to enable automated analysis application features of a system on a top layer and platform features on a bottom layer; ii) a definition of a set of operations that can be used on such models; iii) a reference implementation for multi-layer analysis based on an Android case study and, finally; iv) two empirical evaluations that demonstrate the feasibility of our proposal in practice.

Implementation Components: A product configuration is one of the most error-prone activities, even more so when for each feature there is more than one component that implements it. To manage these configurations, we introduced a component-based recommender system called RESDEC

that facilitates the selection of implementation components when creating products in an SPL. Specifically, the contributions presented with this proposal are: i) modeling of the implementation component selection problem as a recommendation task using collaborative-based and content-based filtering algorithms; ii) design of a prototype component-based recommender system tool ready to be used and extended to other environments from the selection of implementation components and, finally; iii) an empirical evaluation based on e-commerce websites in WordPress.

Resumen

En los últimos años y con el creciente avance tecnológico, las empresas ya no se centran exclusivamente en diseñar un producto para un cliente (por ejemplo, el diseño de un sitio web para el Hotel Decameron), sino en producir para un dominio (por ejemplo, el diseño de sitios web para hoteles); es decir, el diseño de un producto que pueda adaptarse fácilmente a las diferentes variaciones que puedan existir para un mismo producto y que se adapte a los gustos individuales de los clientes.

En la ingeniería de software, esto puede lograrse a través de la gestión de líneas de productos de software (SPL). Una SPL se define como un conjunto de sistemas que comparten un conjunto común de características que satisfacen la demanda de un mercado específico. Una SPL intenta reducir el esfuerzo y el costo de implementar y mantener en el tiempo un conjunto de productos de software similares; sin embargo, manejar la variabilidad en estos sistemas es una tarea difícil, a mayor número de productos más complejo se hace manejarlos.

Los modelos de características (FMs) se emplean para representar gráficamente las partes comunes y variables de una SPL. Dada la gran cantidad de características que se pueden derivar de un modelo de característica (FM), resulta difícil de gestionarlos. Para hacer frente a estos problemas se ha propuesto el Análisis Automático de Modelos de Características (AAFMM) que mediante el uso de herramientas asistidas por ordenador, se ocupa de la extracción de información de los modelos de características. No obstante, existen ciertos escenarios en los que la configuración de un producto se convierte en una actividad compleja dado el número de componentes que existen para implementar una determinada característica.

En esta tesis, exploramos técnicas inteligentes para resolver dos problemas que surgen al manejar una SPL:

- i. Por un lado, hemos identificado los problemas que surgen cuando un desarrollador desea mantener sus aplicaciones al día con los últimos avances tecnológicos. La estrecha relación entre las características de aplicación y los componentes de plataforma es difícil de rastrear. Los

- desarrolladores deben ser conscientes de las consecuencias que podrían traer a las aplicaciones existentes cuando cambia el hardware donde se va a ejecutar; por ejemplo, cuando una aplicación se traslada de un smartphone a una computadora/tablet, o cuando una plataforma se actualiza a una nueva versión. Los diferentes tamaños y resoluciones de pantalla, la posible ausencia de un radio celular o el aumento de la cantidad de memoria pueden tener impactos positivos o negativos en una aplicación. En este contexto, dado que las características de aplicación y de plataforma están conceptualmente separadas, sus características pueden modelarse en dos modelos distintos. Por consiguiente, manejar la trazabilidad entre estas dos capas y cómo los posibles cambios en ciertas características puedan afectar a la otra capa, es un problema que está por resolver.
- ii. Por otro lado, hemos encontrado lo complicado que es para el desarrollador de aplicaciones configurar un producto cuando hay una variedad de componentes de implementación para cada característica. Por ejemplo, un desarrollador web en WordPress busca manualmente aquellos componentes (plugins) que son factibles y más óptimos para cada sitio web. Esta tarea lleva tiempo y no siempre garantiza que los componentes seleccionados sean los más adecuados (en términos de calidad) para la aplicación requerida. Dos escenarios podrían surgir durante esta configuración: primero, la selección empírica de un componente, en la práctica, puede no proporcionar los resultados esperados; además, no tener criterios basados en la experiencia de otros usuarios con respecto a estos componentes, podría inducir una mala selección y lograr una mala experiencia para el usuario final. En este contexto, el manejo de la relación entre los componentes de implementación y sus características es otro problema a resolver.

Concretamente, las contribuciones de esta tesis se detallan a continuación;

Modelos de características en múltiples capas: En esta área introducimos un framework para el análisis de modelos de características de múltiples capas, llamado MAYA. Los objetivos que perseguimos con esta solución son: i) modelar la variabilidad de los sistemas software en dos capas, incluyendo sus respectivas interdependencias; ii) definir un conjunto de operaciones que puedan imponerse a dichos modelos; iii) una implementación de referencia para el análisis de múltiples capas basado en un caso de estudio en Android, y finalmente; iv) dos evaluaciones empíricas que demuestran la viabilidad de nuestra propuesta en la práctica.

Componentes de implementación: La configuración de un producto es una

de las actividades más propensas a errores, más aún cuando para cada característica hay más de un componente que la implemente. Para gestionar estas configuraciones, introducimos un sistema de recomendación basado en componentes llamado RESDEC que facilita la selección de componentes de implementación al crear productos en una SPL. Concretamente las contribuciones que se presentan con esta propuesta son: i) modelado del problema de selección de componentes de implementación como una tarea de recomendación utilizando algoritmos de filtrado colaborativo y por contenido; ii) diseño de un prototipo de herramienta de sistema de recomendación basada en componentes lista para ser utilizada y extendida a otros entornos a partir de la selección de componentes de implementación y, finalmente; iii) una evaluación empírica basado en sitios web de comercio electrónico en WordPress.

Part I

Preface

Chapter 1

Introduction

You've got to have an idea, or a problem or a wrong that you want to right that you're passionate about, otherwise you're not going to have the perseverance to stick it through. I think that's half the battle right there.

Steve Jobs

***I**n this dissertation, we report our work on the use of two-layered feature models and selection of implementation components in a software product line. In this chapter, we give an overview of the contributions, the research method and publications related to this document.*

1.1 Overview

After the industrial revolution at the end of the 1980s, and with the appearance of the machines in industry, the production of goods and services completely changed the way of manufacturing a product; introducing a new production paradigm known as “mass production”. According to Zipkin [102] mass production had, as main objective, the production of a large quantity of the same product using standardized processes in a reduced commercialization time.

This type of production mechanism, despite of guaranteeing production in large volumes of products to meet with high demand and immediate delivery, does not completely satisfy the customers since it does not allow the personalization of products according to their particular needs.

In a highly competitive market with constant changes, mass production becomes insufficient, transforming its approach to a “mass customization”. According to McCarthy [57], mass customization consists in the production of (massive) products quantities and services in a personalized way to each client, maintaining quality and delivery time.

The main advantage of this production system is the immediate response to the specific needs of each client without affecting the cost, achieving to maintain prices equivalent to the production in large quantities and complying with the efficiency of mass production. Figure §1.1 shows a motivating scenario of mass customization, where a website designed for an airline can be implemented in different airlines (e.g. KLM, Avianca and American airlines).

Nowadays, with the great technological advances, the industry of the software engineering is not far from this mass customization. In software production, researchers and engineers worked hard to apply techniques of mass customization in software engineering to rise a new development paradigm known as “Software Product Lines (SPL)”. An SPL allows managing of creating reusable software artifacts, as well as to describe variability points and ensure they are reused properly.

Clements and Northrop [28] define an SPL as a set of highly variable systems that share a set of common characteristics that meet the specific needs of a market segment. Managing variability in a product line is a difficult and complex task given the large number of possible software variants that could be derived. Generally, variability is represented through feature models (FMs). An FM [42] is a diagram that graphically describes the features of an SPL and

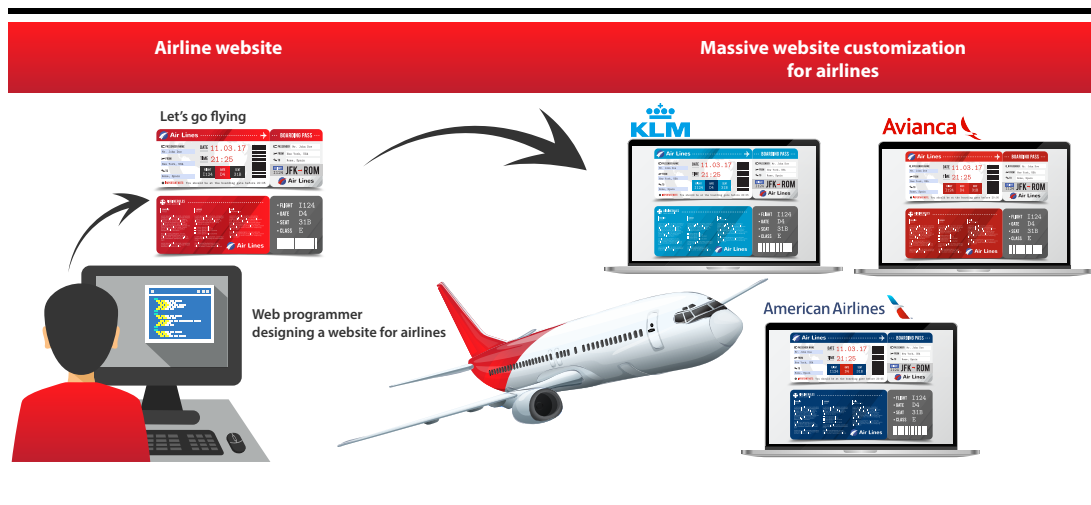


Figure 1.1: Websites mass-customization

its relationships, as well as the possible restrictions that may arise. The features selection with their respective attribute values defines a configuration [9] (a.k.a. software product) that capture a personalization possibility that the system allows and that are used within the ecosystem where the SPL is developed.

The number of configurations resulting from a feature model can grow when adding new features. To manage variability, the literature presents the automated analysis of feature models (AAFMM) [15] that is about extracting information from feature models using computer-assisted mechanisms .

In spite of the efforts made by researchers in the field of AAFMM, there is still a need for solutions that reduce the inconvenience of managing configurations in order to prevent a time-consuming and costly process.

One of the common difficulties is how to link the features that are defined at the application level with the platform level features components. In this context, we have identified two problems:

- i. What happens when we have two feature models and need to connect them to manage the dependencies of the application level with the platform level?
- ii. How to guarantee the optimal selection of components to implement features during the configuration of a product in an SPL?

Figure §1.2 describes the motivation and the main research core presented

in this document.

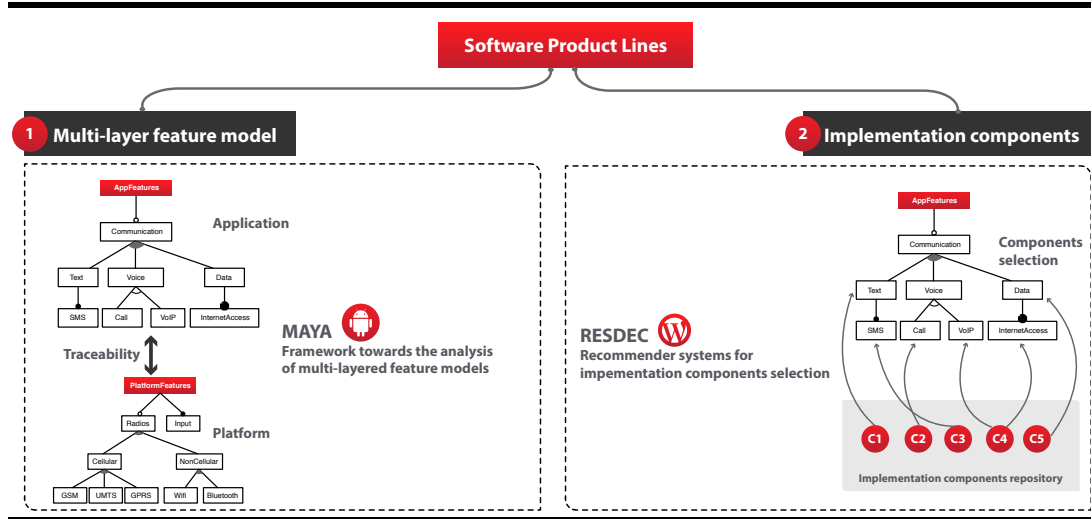


Figure 1.2: Overview of this thesis scope

Specifically, in this thesis we have addressed the following areas:

Multi-layers feature models There are well-established approaches in the literature to analyze variability intensive systems using feature models. However, there is a lack of approaches to analyze the application and platform features in multiple layers. In this first contribution, we present a framework for the analysis of multilayer feature models that is evaluated with an empirical implementation on Android that demonstrate the viability of the approach in real-world scenarios.

Implementation components Given the variety of implementation components that can be used to configure product, identifying the best components set is a challenging task due to the high number of combinations and options that can be selected. In this contribution, we present a proposal that, through information associated with the components, introduces a recommender system that is later exploited in three usage scenarios. An empirical evaluation using WordPress information validates the capacity of our approach to guide users in the implementation components selection.

1.2 Research method

In this work we have used the research method proposed by Ida and Ketil [39]. This method, in addition to solving a problem in a particular context, proposes the design and development of new software or the improvement of existing ones. The main focus is to meet all the requirements from the requirements engineering point of view. Note that this research method is focused on technological research.

The main steps of the technological research are:

- i **Problem statement:** It is focused on identifying the necessary conditions for the new artifacts creation. In other words, the researcher is responsible for seeking the needs for the new technologies development.
- ii **Contribution:** The focus of this step is the solution development through the creation of the artifacts.
- iii **Validation:** In this last step, the artifacts designed to ensure they meet the necessary requirements to solve the problem initially identified are verified and validated.

1.3 Contributions

In this section we summarize the main contributions of our work, which have been published in different conferences and journals.

1.3.1 Summary of contributions

In this thesis we address two problems that originates when managing an SPL. First, we analyze the problem that arises when trying to link features that are defined at the platform level with application level features in multiple layers. To solve this, we introduce a framework called MAYA for the analysis of multi-layered feature models (i.e application layer and platform layer) using features attributes information. Then, we analyze the implementation components selection to configure a product of an SPL at the application level, using recommender systems. For this purpose, we designed a prototype of a tool called RESDEC (*REcommender System that from selecteD fEatures suggest*

implementation Components) that allows implementation components selection using different recommendation techniques. Finally, we show real cases from the industry that allowed us to validate our contributions.

The main objective of this thesis is to provide intelligent techniques that help the software engineer when selecting products that adjust to certain user requirements. That is, when working with multiple layers and when it is necessary to configure features from the implementation components selection.

The main contributions of this thesis are summarized below along with the objective pursued in each one of them:

i) **Automated analysis of two-layered feature models with feature attributes**

Our first contribution proposes a mechanism to manage the dependencies between two feature models (application and platform) through the link of both models.

- *Problem statement:* The proliferation of features and platforms in variability intensive systems, coupled with substantial technological progress, imposes several challenges for software developers and equipment manufacturers. Typically, the features of the application and platform are conceptually separated and can be modeled in two separated models organized in layers. The problem is the difficulty of tracking one layer features to the second features. This requires a means to model each set of features separately to reflect the possible independent evolution of each layer.
- *Contribution:* We introduce MAYA, a two-layered feature model framework that includes application features of a system on a top layer and platform features on a bottom layer. In addition, a detailed dependency mapping between these two layers using features attributes to enable automated analysis is provided.
- *Validation:* To validate our approach, two real-world empirical evaluations are presented regarding application handling and platform evolution in the application development domain for mobile phones. In this case, we demonstrate the applicability and need for working with two-layer models.

See Chapter §5 for more details about this contribution.

ii) **Selection of software product line implementation components using recommender systems**

Once we were able to link two feature models, it was necessary to propose a technique to configure a product features from the selection of implementation components.

- *Problem statement:* Selecting the best set of components to implement features in the product configuration of an SPL is a challenging task. In certain scenarios, given the high number of combinations and component options that can be selected, inappropriate selection could result in bad experience when configuring a product.
- *Contribution:* We propose a component-based recommender system, called RESDEC. RESDEC aims to take advantage from a repository of information generated by users to efficiently search for appropriate components to implement features in a specific context. To do this, we adapt a set of algorithms commonly used by recommender systems.
- *Validation:* We demonstrate the effectiveness of our approach through an empirical evaluation in an e-commerce website scenario using WordPress. The evaluation is based on data from 116,000 users, 680 plugins, and 187,000 ratings.

See Chapter §6 for more details about this contribution.

1.3.2 Publications in chronological order

Results of our research work have been published in different congresses and journals. In Figure §1.3 we shown a complete list of these publications in chronological order.

[2015] In this year, we made a stay at the INRIA research center (Rennes-France) with the DIVERSE research group, which allowed us to carry out formal investigations in a full time schedule. During this period, we began to work on the preliminary core of the dissertation. From the beginning, we were motivated by the advantages of the recommender systems techniques to personalize the user experience in different scenarios; which allowed us to analyze the applicability of these techniques in the configuration of software product lines through user reports. Preliminary results of this work were presented in the conference that is detailed below.

- **10CCC'15.** Jorge Rodas-Silva, José A. Galindo, David Méndez, and David Benavides. Towards testing variability intensive systems using

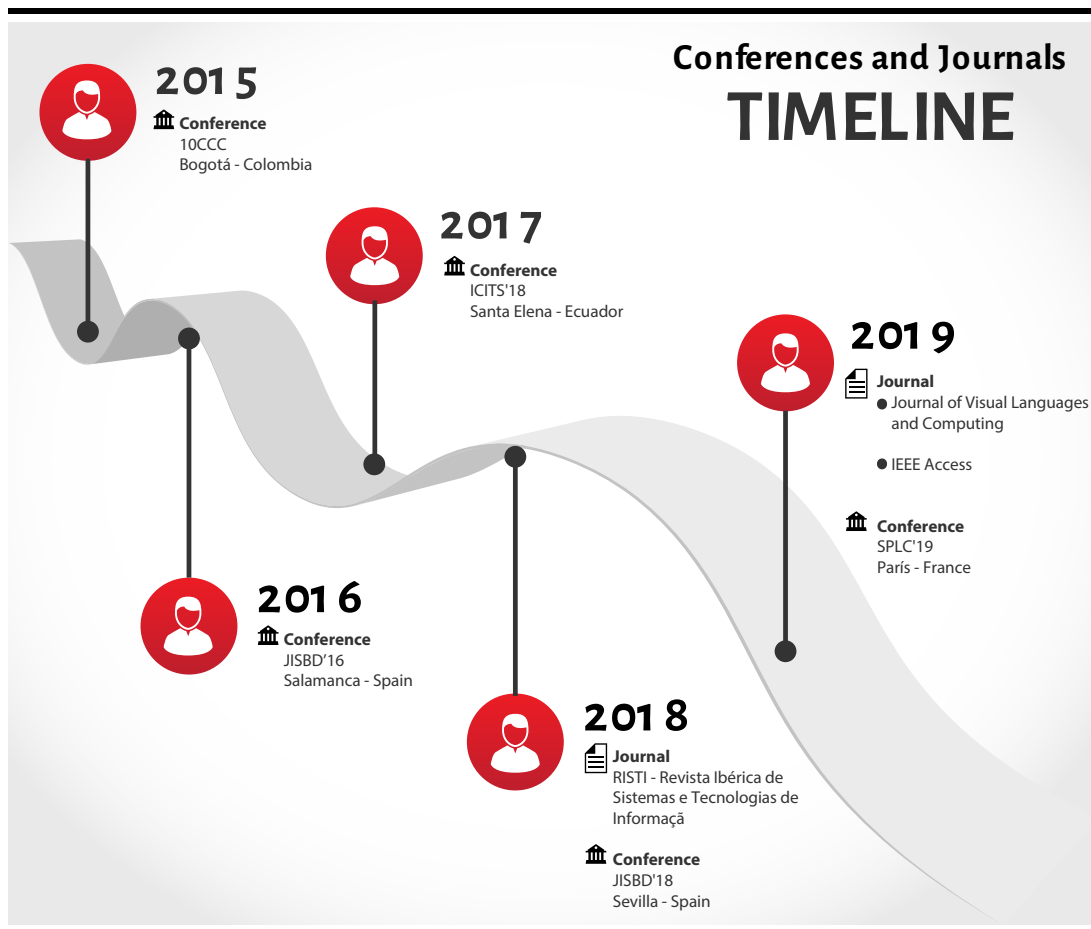


Figure 1.3: Chronological order of conferences and publications

user reviews. Proceedings of the 10CCC - Congreso Colombiano de Computación, Bogotá, Colombia.


[2016] In this year, we continued with the stay in the INRIA research center. In this period, we went deeper into the different techniques and algorithms that a recommender system uses and its application in common problems faced by the software engineer when configuring or deploying a product of SPL. Specifically, we analyzed how these algorithms could help in the selection of configurations for the testing and deployment of applications. Results of this approach are published in the conference described below.

- **JISBD'16 Jorge Rodas-Silva**, Javier Olivaro, José A. Galindo, and David Benavides. Hacia el uso de sistemas de recomendación en sistemas de alta variabilidad. XXI Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2016), Salamanca, España.

[2017] In this year, our research was resumed at the University of Seville. In this period, the main research focus was the selection of deployment configurations to implement an SPL product using recommender systems. Parallel to this research, we began to collaborate on the proposal to manage the variability in a two-layer model and the traceability operations to connect these models. Some of the work done in this year, is published in the conference described below.

- **ICITS'17 Jorge Rodas-Silva, José A. Galindo, David Benavides and Robert Soriano.** Selección de configuraciones usando sistemas de recomendación en Android. The 2018 International Conference on Information Technology & Systems (ICITS), Santa Elena, Ecuador.


[2018] In this year, the main contribution for the dissertation of this thesis was the publication of the results obtained when applying recommender systems in the selection of deployment configurations. Later, we designed a prototype tool called RESDEC based on collaborative filtering recommender systems that was implemented using a knowledge base built from Android data. Results of this work are published in the journal and conference described below.

-  **RISTI'18 Jorge Rodas-Silva, José A. Galindo, David Benavides, Robert Soriano.** Selection of deployment configurations using recommender systems on Android. RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação.


- **JISBD'18 Jorge Rodas-Silva, José A. Galindo and David Benavides.** RESDEC: Un prototipo de herramienta para la selección de configuraciones de despliegue basado en Sistemas de Recomendación. XXIII Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2018), Sevilla, España.

[2019] In this year, results were obtained which contributed to the development and culmination of the doctoral work. In this context, the proposal to connect layers of the application level with the level of platform that began to work in 2017 was published in a journal. Also, the proposals for recommender systems shown above were reinforced, this time focusing on the selection of implementation components to configure features. For this purpose, an improved version of RESDEC was designed in which new algorithms

of recommender systems collaborative-based and content-based filtering were incorporated. In addition, the proposal was validated using real-world data from WordPress, which allowed us to validate the scope of our approach in the development of a product line based on websites built in this platform. Results of this work are published in the following report and conference.

- 

JCL'19 Michael Letter, **Jorge Rodas-Silva**, José A. Galindo, David Benavides. Automated analysis of two-layered feature models with feature attributes. *Journal of Computer Languages*; 0.971 Impact Factor (Q3).

- 

IEEE'19 **Jorge Rodas-Silva**, José A. Galindo, Jorge García-Gutiérrez, David Benavides. Selection of software product line implementation components using recommender systems: An application to WordPress. *IEEE Access*; 3.557 Impact Factor (Q1)

- SPLC'19** **Jorge Rodas-Silva**, José A. Galindo, Jorge García-Gutiérrez and David Benavides. RESDEC: Online Management Tool for implementation components selection in a Software Product Lines using Recommender Systems. *Software Product Lines - 23rd International Conference, SPLC*, París, France.

1.3.3 Tools

During this PhD, two prototype tools were developed:

- MAYA^{†1} a framework towards the analysis of multi-layered feature models (see Chapter §7 for more details about this tool); and
- RESDEC^{†2} a component-based recommender system designed to select implementation components to configure features in an SPL (see Appendix §A for more details about this tool).

^{†1}<https://github.com/FaMaFW/FaMA/tree/branches/fama-two-layers>

^{†2}<http://resdec.com>

1.4 Research internships and collaborations

The research results presented in this document and which are part of the contributions developed in the course of the doctorate, were carried out in three different countries under the supervision of different people. Figure §1.4 shows the universities and institutes in which they carried out research, along with each institution hosts and advisors in each country.

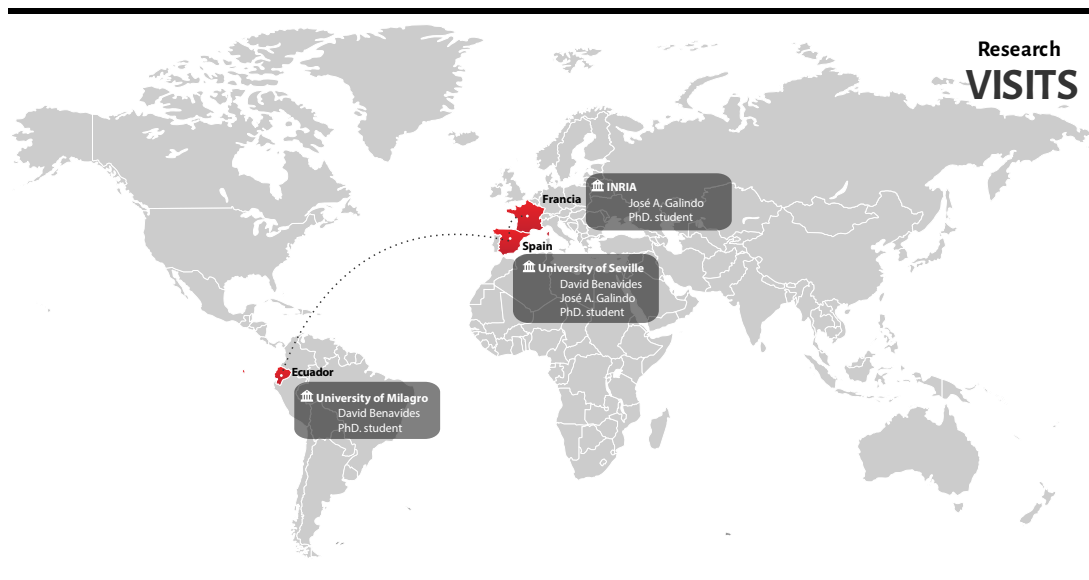


Figure 1.4: Stays to cope with the PhD objectives

During the development of this thesis, we worked on stays in France (INRIA), complementing the work done in Sevilla (University of Seville) and Ecuador (University of Milagro).

In France, the research stays was at the *Institut National de Recherche en Informatique et en Automatique (INRIA)*. This research center is located in the city of Rennes and is considered one of the most important centers in the field of science and technology development in Brittany. In this center we stayed for a period of seven months between 2015 and 2016, the hosts were PhD. José A. Galindo and PhD. Benoit Baudry, this last Senior Researcher of INRIA DIVERSE, department of which I was part. DIVERSE is a group of researchers focused on research in four areas: modeling and languages engineering, advanced testing, DevOps for distributed and heterogeneous system and variability engineering.

During this period, the main research topic was the application of recommender systems in the context of software product lines. The results of this

visit were materialized with the writing of two articles that presented preliminary advances of our research in two congresses, one international [81] and one national [82] as stated previously.

At the end of this stay, in 2016 we returned to University of Seville to reinforce the work carried out at INRIA and to continue improving the research proposal. In Seville, I was under the supervision and direction of PhD. José A. Galindo and PhD. David Benavides.

Finally, we returned to Ecuador to design and build the RESDEC tool that we introduced in this thesis.

1.5 Structure of this dissertation

This document is organized as follows:

Part I: Preface. In the first part of this thesis, Chapter §1 is presented, which addresses the background that have driven this work. Also, the main contributions of this research.

Part II: Background information. In the second part of the thesis we present basic information necessary to understand the objectives of this work, it is organized in two chapters. In Chapter §2, we explore the concepts related to software product lines and examine feature models through some examples. Then, we present a brief review on application engineering. Later, in Chapter §3, we address the field of recommender systems and review the types of systems, their classification and finally we review the evaluation techniques used to validate them.

Part III: Our contribution This part constitutes the core of our thesis, it consists of three chapters and is organized as follows. In Chapter §5, we introduce as the first proposal MAYA, which incorporates the concept of multilayers from a traceability approach between two feature models studying the consequences of the changes of one layer on the other layer. Subsequently, in Chapter §6 we present RESDEC, our approach to configuring product features through the optimal selection of implementation components using recommender systems.

Part V: Validation. In this part we present two chapters that describe the evaluation for our contributions presented in this thesis. For this purpose, we shown empirical evaluations that were carried out to justify the validity of the contributions. In Chapter §7, we validate MAYA through a

case study on Android, in which we model two feature models and analyze the consequences of the changes that may occur in the application layer, and how these could affect the platform layer. In Chapter §8, we present the validation made to RESDEC using a case study in WordPress. In this case, we use WordPress historical information (plugins, tags and ratings) to test the recommendation algorithms that were implemented to select implementation components in the configuration of a software products line. The results obtained using different evaluation metrics showed promising data that corroborated the validity of our proposal.

Part IV: Final remarks. In this part, we show the conclusions and propose future work to address the new research problems arising from the contributions made in this thesis.

Part V: Appendixes. A brief description of the RESDEC web application work environment based on an example is described in Appendix §A.

Part II

Background information

Chapter 2

Software product lines

The way get started is to quit talking and begin doing.

Walt Disney

A software product line (SPL) is defined as a type of variability intensive systems in which a set of common features can be customized according to the specific needs of the stakeholders in a particular context. For the management of an SPL, models that allow to represent all the possible products that can be derived from it are used. The most popular models used for this purpose are called feature models [42]. In this chapter we present the definitions of these models as well as examples that describe their applicability in software engineering. In addition, we define the process of automated analysis and the current techniques for extracting information.

2.1 Introduction

Software product lines (SPL) have become a new important paradigm in software development [31]. In software engineering, it represents a new and intensive area [85]. An SPL is a set of variability intensive systems that share a common, manageable set of features that meet the specific needs of a particular market segment [9, 31, 63, 85]. Its engineering is based on families that produce similar systems instead of the production of individual systems [15].

An SPL aim to reduce the time of commercialization of products and increase software quality through planned reuse of artifacts to meet customer needs and decrease customization effort [12, 34]. In addition, SPL seek to provide a common platform that allows the derivation of particular systems that are capable of adapting to the different needs required [76].

Specifically, an SPL aligns engineering resources with business objectives to ensure that efforts are focused on the most cost-effective features and functions of a product. The most characteristic element of an SPL is its variability, and its management is fundamental to define it; therefore, SPL paradigm is a solution for managing variability in a family of products [11].

Figure §2.1 shows an example of how the customization of a software product line is offered to mobile telephones consumers through web configurators in which users can select the functions that best adapt to their needs.

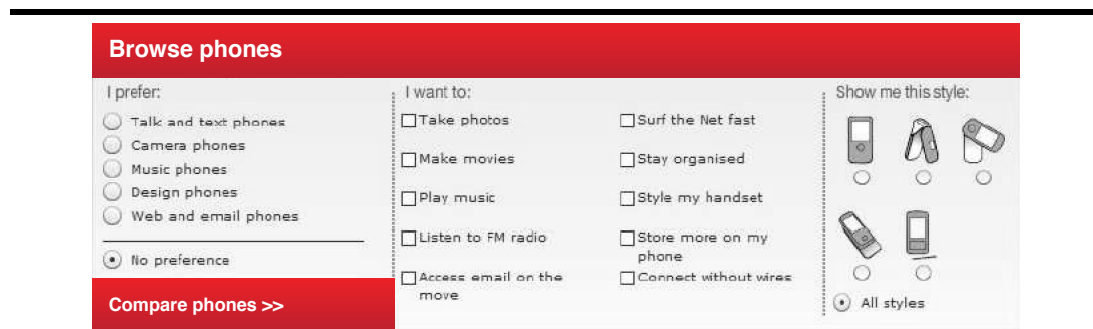


Figure 2.1: Customization of an SPL in the mobile phone industry (from [94])

2.2 Feature models

One of the main artefacts of software product lines are features models (FMs) used to represent or model the common and variable features and the

relationship between them [34, 42, 85]. In industry, FMs have been widely adopted to capture, organize and reuse the requirements of a set of similar applications in a software domain [100].

Since the first proposal of Kang *et al.* [42] FMs have become the de facto standard to represent variability and are used to model SPL in terms of features and relations among them [16, 81]. A feature represents a functionality operation in a system to meet a product requirement and provides a potential configuration element to enrichment an SPL [7]. Also, it is an increment in the product functionality [14]. An FM is a compact representation of the set of products in an SPL in terms of features and relations between them [87].

In general, FMs are tree-shaped hierarchical structures that describe the successive refinement of variability in a product line [40]; and where, restrictions between trees can be used to connect features [33].

FM can be seen as a “general landscape” of the functionalities of a software system, and their integration plays a fundamental role in software engineering tasks [22]. To use FMs, first is to build them, which generally involves a systematic analysis of common features and variability with the software domains in focus [100].

A feature can be abstract or concrete [97]. A feature is **abstract**, if it is not mapped to any implementation artifact, and is **non-abstract** or **concrete**, when at least one implementation artifact is assigned. The combinatorial features derived from the model are known as configurations and can be used to generate a product.

An example used to illustrate a feature model is the one inspired by the web development industry, presented by Rodas *et al.* [80]. In the feature diagram, the features are represented by boxes and relationships by lines. As shown in Figure §2.2, the model illustrates how the features are related to create an e-commerce website.

There are different proposals to represent FMs. For the most part, they have the common elements of Czarnecki *et al.* [30], the most used. Czarnecki’s notation for FM, represented in Figure §2.2 for the case of e-commerce website, proposes four types of relationships: mandatory, optional, alternative and or-relation.

Relations.- Batory *et al.* [13] classifies the relationships into three groups: “and-group”, “or-group”, and “alternative-group”. *And-group* can be mandatory or optional. A relationship is *mandatory* when a child feature has a mandatory relationship with its father and is included in all the products in which its parent characteristic appears; and it is *optional*, when a child feature

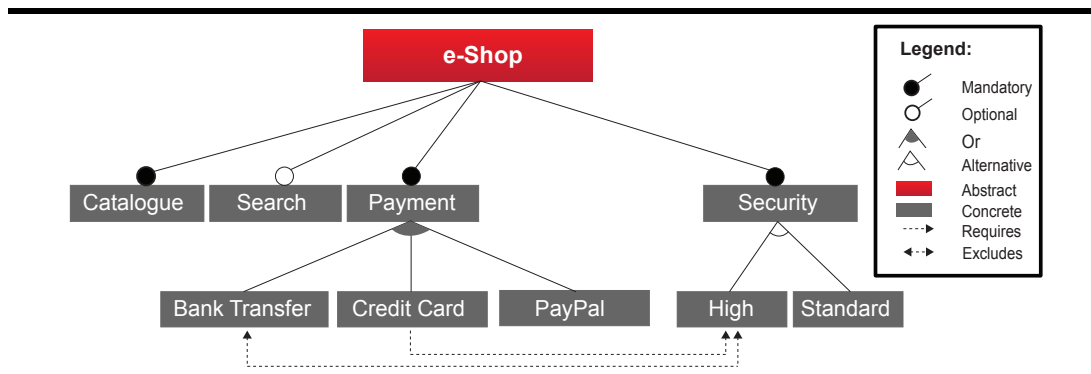


Figure 2.2: Feature model inspired by the web development industry (from [80])

has an optional relationship with its parent and can optionally be included in all the products in which its parent characteristic appears.

Following the model presented in Figure §2.2 an e-commerce website should implement a product catalog, a payment module, security policies and could optionally include a search tool.

In addition to the parental relationships between features, a feature model may also contain constraints between unconnected features. These restrictions can be of two types: *requires* and *excludes*. A restriction is *requires* when an A feature requires a B feature, that is, the inclusion of A also implies the inclusion of B within the configuration of a product. For example, for the e-commerce website in Figure §2.2, the implementation of the credit card payment function requires high security policies. On the other hand, it is *excludes* when a feature A excludes a feature B, that is, both features cannot be part of the same product. In the case of the e-commerce website, *high security* policies and payments by *bank transfer* are incompatible, since the activation of high security policies requires the use of a payment via *credit card* or *PayPal*.

Configurations.- A *configuration* specifies a particular instantiation of the product line. It is characterized by specifying a set of *selected* and *eliminated* features.

The configurations can be classified into different categories, for example, a *valid* configuration adheres to the defined relationships and constraints. A *complete* configuration contains all the features of the selected or deleted lists. When not all the features are contained in the selected/removed sets, it is called *partial* configuration. A complete configuration that only contains the set of selected features, while all other characteristics are implicitly eliminated, is known as *product* [15]. In the context of extended feature models, config-

urations can also include the configuration of attributes (that is, determine a value for each attribute).

Finally, for a product to be properly configured, a set of necessary components is required to implement each feature. This set of components is known as *implementation components* [61].

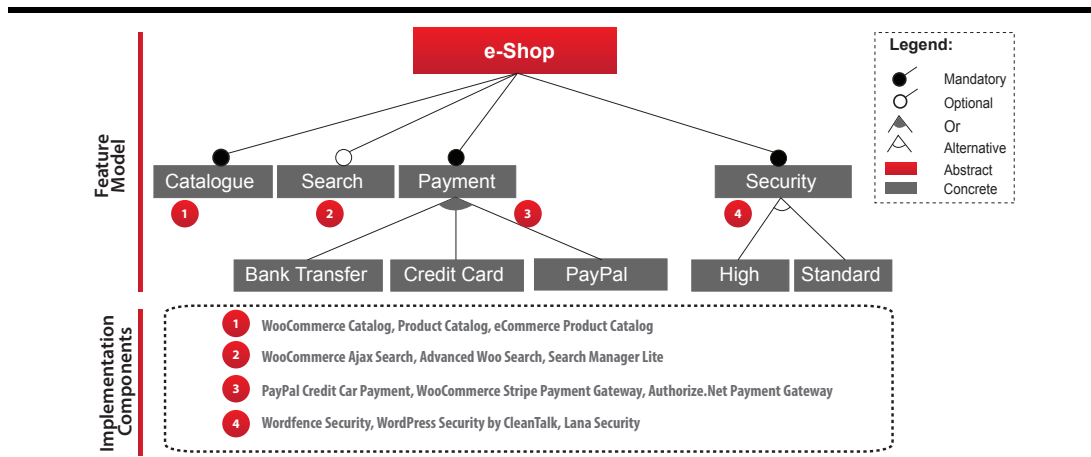


Figure 2.3: Features model that describes a list of components for an e-commerce website

As an example, *the catalog, search, payment and security* features on the e-commerce website presented in Figure §2.3 can be implemented using the list of components shown at the bottom of the figure to comply with the functionality required on the website. Keep in mind that different features can implement the same component. For instance, if we want to implement the *catalog* feature (identified with 1 in the graph), we could have more than one selection alternative to implement it. In Figure §2.3 we present a list of three components from which the user can choose one. In practice, we can find a wide variety of components available for selection and that could implement the *catalog* feature.

Types of Extended Feature Models.- In addition to the basic feature models described above, a couple of extensions has been proposed. Schobbens [91] summarizes the general semantics of common feature diagrams in the literature.

For example, an *extended* feature model is characterized by features that may contain *attributes*. This allows to model specific properties of a feature, such as costs or information in the version. At the same time, it allows complex constraints between trees that depend on the value of a given attribute, for

example, if the resolution of the camera is greater than a certain threshold X , then we need to have a high screen resolution. The alternative terms for extended feature models are “attributed” or “advanced” feature models [16].

2.3 Automated analysis of feature models

The Automated Analysis of Feature Models (AAFMM) is a research topic that has attracted the researchers and professionals’ attention over the past two decades, time in which, the amount of tools and techniques that allow the analysis of feature models has increased and also its complexity [93]. The research carried out so far proposes the use of AAFMM to cope with the variability management in software product lines [81].

The AAFMM has recently been identified as one of the most important areas in the development of software product lines [34]. The AAFMM consists of the computer-assisted extraction of information from models [15] and can be summarized in three steps. First, FMs are translated into a logical representation. Secondly, a specific algorithm or solution is used to perform a certain analysis operation (e.g. counting the number of products or verifying the consistency of an FM). Finally, the result is obtained and used in a specific domain such as the configuration or product derivation [34].

Recently, Galindo *et al.* [34], identify six different facets of variability where the AAFMM is being applied:

- i. Product configuration and derivation. It is the most widely used automated analysis mechanism and is used to support the derivation and product configuration process.
- ii. Testing and evolution. It consists of using automation mechanisms that guide the process of selecting configurations for testing purposes.
- iii. Reverse engineering. The AAFMM in this area from descriptive information of products and using logical formulas is in charge of the extraction of feature models.
- iv. Multi-model variability analysis. In certain scenarios it is necessary to analyze more than one model, in this case, AAFMM has proposed analysis operations to manage the multilayer configuration process.
- v. Variability modeling. The analysis of models using basic constructions, in some cases are not enough. For this reason, it is necessary to use

additional information from the models for a improvements in analysis. For example, AAFM uses attributes to model different situations that help decision-making during the product configuration process.

- vi. Variability intensive systems. AAFM is not directly associated with SPL, its field of application has been extended and it is possible to use it in other areas of application.

Among the activities that are carried out automatically using different approaches are: Discovering whether a product is valid, obtaining all the products, calculating the number of products, detecting errors, explaining errors, among other operations [34]. As specified by Benavides *et al.* [17], using operations, different aspects of the models are analyzed. More than thirty different operations can be found in the literature to date. For example, a *Valid product* operation will check if a specific product represents a valid combination of features belonging to a given feature model. Also, the *Number of products* operation determines the number of valid products. Other operations are aimed at comparing feature models, for example, in terms of similarity.

Feature models analysis is a tedious and error-prone task. In order to help feature modeling professionals extract information, different computer-aided mechanisms have been proposed that allow different analysis of SPL in different contexts and domains. There are proposals based on specific algorithms, Binary Diagrams (BDD), SAT and CSP. Some available tools are FaMa [19], FaMiLiAr [3] and FeatureIDE [98], among others.

Currently, there are efficient solutions for errors diagnosis in automated analysis of feature models, such as the proposal of Segura *et al.* [93], where they propose BeTTy, a framework that supports random feature models generation and its set of products allows, among other things, to generate expected inputs and outputs for a series of analysis operations in feature models that accelerate failure detection. On the other hand, Galindo *et al.* [36], propose to use automated analysis of feature models to automate the configuration selection and prioritization for testing in a products line; to this end, TESALIA (TESTing vAriAbiLity Intensive Systems) is introduced, a method that seeks to analyze the set of coded products and the information associated with the features (costs) to prioritize the products that will be used for testing.

2.4 Constraint satisfaction problems

As mentioned in the previous section, there is a great variety of tools that can be used in the AAFM as backends. One of them are CSPs (Constraint

Satisfaction Problems)

A CSP [15] allows to create models formed by a set of variables in which it should indicated which possible values could be taken and the constraints that should be fulfilled between those variables. Given the model, the system is able to find those solutions that comply with the specifications established in the constraints [18]. Unlike propositional formulas that use binary values (true or false), a CSP solver also uses numeric values (e.g. integers or intervals).



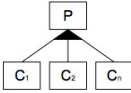
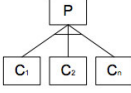
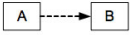

	Relationship	CSP mapping	Mobile phone example
MANDATORY		$P = C$	Mobilephone = Calls Mobilephone = Screen
OPTIONAL		if ($P = 0$) $C = 0$	if (Mobilephone = 0) GPS = 0 if (Mobilephone = 0) Media = 0
OR		if ($P > 0$) Sum(C_1, C_2, \dots, C_n) in $\{1..n\}$ else $C_1 = 0, C_2 = 0, \dots, C_n = 0$	if (Media > 0) Sum(Camera, MP3) in $\{1..2\}$ else Camera = 0, MP3 = 0
ALTERNATIVE		if ($P > 0$) Sum(C_1, C_2, \dots, C_n) in $\{1..1\}$ else $C_1 = 0, C_2 = 0, \dots, C_n = 0$	if (Screen > 0) Sum(Basic, Colour, High resolution) in $\{1..1\}$ else Basic = 0, Colour = 0, High resolution = 0
REQUIRES		if ($A > 0$) $B > 0$	if (Camera > 0) High resolution > 0
EXCLUDES		if ($A > 0$) $B = 0$	if (GPS > 0) Basic = 0

Figure 2.4: Mapping from feature model to CSP (from [15])

Figure §2.4 shows the process to map a feature model to a CSP using a CSP solver. According to Benavides et. al [15, 16] the steps to carry out this mapping process are the following:

- i. According to the type of variable the solver supports, a CSP is mapped to each feature model with values between 0...1 or TRUE - FALSE, as the case may be.
- ii. Depending on the type of relationship, restrictions are established for each relationship of the model giving rise to possible auxiliary variables.

- iii. Finally, CSP is composed of the result of steps i and ii, that is, with values of the variables and the set of restrictions for each relation of the model. Also, an additional constraint is assigned for the root variable (i.e $\text{root} \Leftrightarrow \text{true}$ or $\text{root} == 1$) according to the variables domain.

In the literature there are several proposals that propose the use of constraints programming for the AAFM. Some of these proposals can be found in [16, 18, 19].

2.5 Application engineering

The set of features and dependencies defined in an FM in the domain engineering phase specifies the complete set of valid features combinations of an SPL. Each resulting combination defines a different product that is part of the application engineering phase [8]. Application engineering is a systematic process for creating a product member from the main assets created at the domain engineering stage [41].

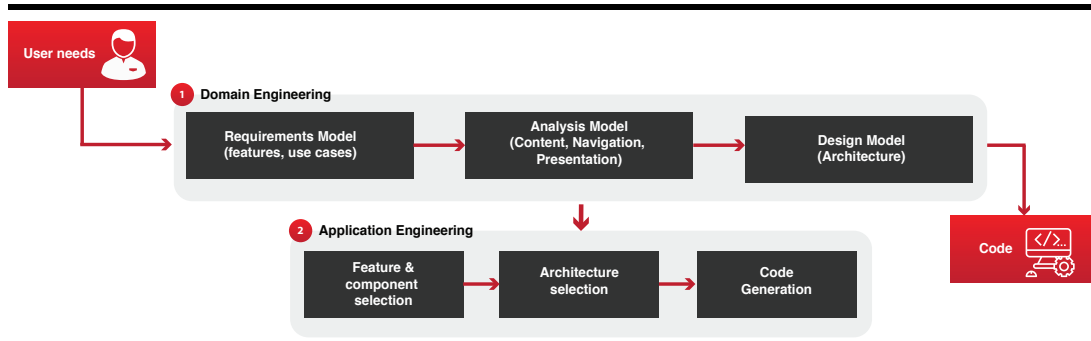


Figure 2.5: SPL engineering process

Thus, there are two distinct phases in the definition of SPL: domain engineering and application engineering as illustrated in Figure §2.5. The process begins by identifying the user's needs. The domain engineering phase includes determining model requirements, model analysis and design; application engineering includes, features and components selection, architecture selection and codes generation.

The domain engineering phase begins with domain analysis, where knowledge of the domain is used to identify common and variable features, and these are made during the design and implementation of the domain, while, applications engineering focuses on the products creation, first identifying the

needs of the client, which are then used to guide the derivation of the product [54].

The application engineering is responsible for selecting a set of features according to the requirements of stakeholders to obtain a software product that meets their needs [45]. To carry out this process, the application engineering phase meets the requirements for a product, sets out the features that meet those requirements and finally obtains a product configuration with the implemented features [75].

The selection of features of an FM and its possible combinations define a product configuration which must comply with the restrictions derived from the model and the requirements for the definition of the product [8].

Specifically, domain engineering ensures that the activities of analysis, design and implementation of a products family are carried out thoroughly for all members of the product, while application engineering ensures reuse of the main components of the product family for the product members creation [41].

2.6 Summary

In this chapter the theoretical aspects supporting the doctoral thesis were presented. First, we have introduced the paradigm of SPL, their definition, objective, benefits, and importance in software engineering. Given the importance of the concept of feature models, another section expanded on this key concept in the software product lines development; we also introduced automated feature model analysis, recently identified as one of the most important areas of software engineering.

Domain engineering and application engineering represent the main phases of the development life cycle of SPL engineering; the first, focused on establishing a reuse platform, while the second is related to the effective reuse of assets in different products. In this review, a section outlining application engineering is included, given its application in this doctoral thesis.

Chapter 3

Recommender systems

We generate fears while we sit. We overcome them by action.

Dr. Henry Link

A recommender system is defined as a system that provides users with a series of suggestions in a personalized way according to their tastes or preferences. In this chapter, the recommender systems are presented as a strategy for the information management by users in software engineering.

3.1 Introduction

Today, a great deal of information (e.g., data, images, videos and documents) is shared in social media. This considerable growth gives rise to information overload making the search for products or services by users complex. This problem for users with the growing evolution of the Internet is present in several areas, becoming a challenge both for researchers and software developers [99]. To mitigate this problem, websites have chosen to use recommender systems (RS) to suggest specific information to users based on their interests [6].

The study of the RS is at the crossroads of scientific and socioeconomic life and its great potential was observed for the first time on the web during of the information revolution. Although, originally it was a field dominated by informatics scientists, recommendations requires contributions from diferents areas and is now of interest also for mathematicians, physicists and psychologists [53].

Recommender systems are a set of tools and techniques to provide useful and relevant recommendations to users to help them in the decision-making process. Thus to choose the right products or services [4, 99]; they seek to forecast the “qualification” or “choice” that a user would grant to an element [79].

Historically, RS was part of the data mining and information filtering. Later, in the 1990s, it was recognized as a full-fledged research area due to a variety of practical applications and complex problems to solve [20]. The term now has a broader approach and groups together any system that produces recommendations or has the effect of guiding the user in a personalized way towards useful products or services within a variety of possible selection options [26].

The purpose of the RS is to provide customized models by collecting user activities to display results in accordance with their expressed preferences, explicitly or implicitly [20]. Also, RS uses the input data to predict possible additional tastes and interests of its users [62].

Recommender systems suggest to each user in a personalized manner through previous preferences, current interests or a combination of methods [74]. RS has become the background application of many electronic commerce applications and information service providers; they help to resolve information overload and provide adequate information in the era of today’s information explosion. In its simple form, a RS adopts a unique approach and

recommends elements that have the greatest global significance.

The investigation of RS is increasingly important in e-commerce environments. Currently, companies such as Amazon, Netflix, Launch, Google, YouTube and Facebook are using and relying on RS to sell their products and services through the recommendation of articles to users to obtain an increase in revenue in short, medium and long term[20]. In a different context, RS is also used in other application areas; for example, for the classification of medical images into diseases. This scenario highlights the importance of RS by opening up opportunities for the development of tools in new fields of application.

The main basis for the RS operation are the algorithms implemented. The algorithms for the RS have attracted the attention of the researchers due to their practical application, most of these algorithms are based on the symmetry measurement to filter information and recommendations for users. Another new trend is the use of statistical implications analysis in the RS, which addresses the asymmetric influence of the user the problem, and solves the evaluating occurrence or the functional relation the problem. The libraries of known recommendation algorithms include: Apache Mahout^{†1}, LensKit^{†2}, MyMediaLite^{†3} and RecSys^{†4}.

Finally, among the most recent literature reviews that have had the RS, applications on microblogs [96], television [99], and mobile telephony [74], stand out among others. In [24, 25] several applications that use recommender systems are described in detail.

3.2 Classification of recommender systems

The most common types of algorithms for the RS are those based on content and collaborative filtering [8]. Figure §3.1 illustrates the two main groups of RS.

In addition to those mentioned above; other classifications include hybrid approaches in the classification [4, 27, 53, 101]; while others, include, in addition, the approaches based on demographic data [25]. The choice of the model depends on the explicit use of data for a “memory-based” recommendation,

^{†1}Apache Mahout WebSite: <https://mahout.apache.org/>

^{†2}LensKit WebSite: <https://lenskit.org/>

^{†3}MyMediaLite WebSite: <http://www.mymedialite.net/>

^{†4}RecSys WebSite: <https://pypi.org/project/recsys/>

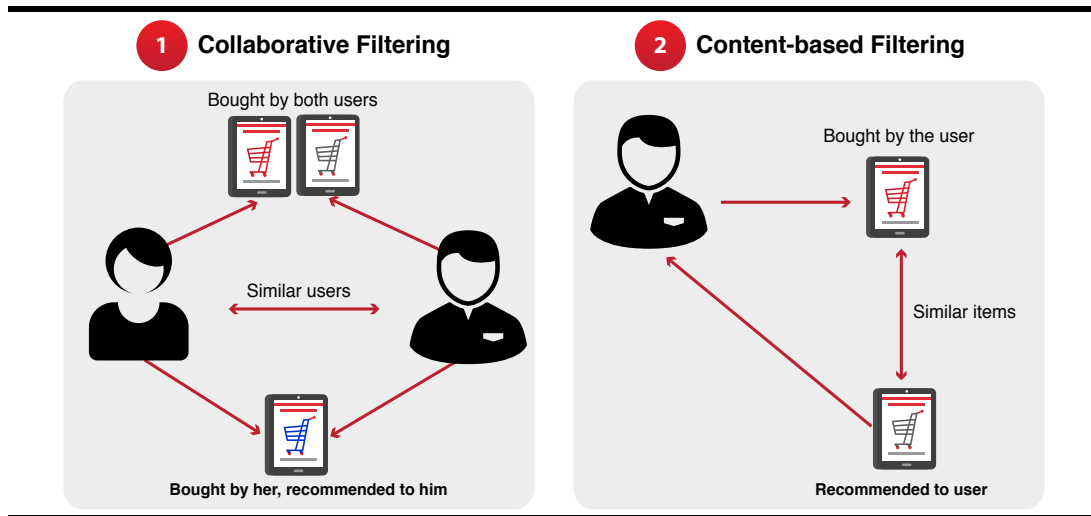


Figure 3.1: General classification of recommender systems

or implicit application of the data through a generative model learned from them “based on the model” [101].

3.2.1 Collaborative-based recommender systems

The systems based on collaborative filtering techniques [46, 51], also known as personalized recommender systems, are based on the analysis of user profiles, where recommendations are generated according to the tastes of users with similar preferences. For example, on *Netflix*, a user who has rated a series of movies could receive recommendations from other users who have also rated the same movies in a similar way or at least a large part of them, which we call a group of users with similar interests.

One of the main challenges of collaborative filtering systems is how to recommend to new or inexperienced users, which means, users who have never used the system, so there is no record of their interests. The lack of experience of these users makes it difficult to find relevant results (similar users or items) that fit their profiles. This case is defined in RS as *Cold Start*. Non-personalized recommender systems have been introduced in the literature to solve cold-start problems [90]. This recommendation technique calculates the average rating (\bar{r}_{p_j}) of each item (p_j) from the users who have rated it (U_{p_j}) (see Equation §3.1). Then, the best rated items are selected from Equation §3.2

to build the recommendation of the new user.

$$\bar{r}_{p_j} \leftarrow \frac{\sum_{u_i \in U_{p_j}} r_{u_i}}{|U_{p_j}|}; j = 1..n \quad (3.1)$$

$$P_{u_k} \leftarrow \max_t(\bar{r}_{p_j}) \quad (3.2)$$

A most interesting case in the collaborative filtering scenario is to make recommendations to users who have experience in the system, which means that have issued some kind of ratings for used items, so the system uses the records to find other users with similar interests. Next, we will describe two collaborative-filtering techniques used to build recommendations to experienced users.

3.2.1.1 Memory-based collaborative-filtering

Memory-based collaborative filtering algorithms [89] are characterized by employing the entire matrix of ratings to generate predictions (*i.e.*, estimate how a user would rate each item). In these algorithms, each user is part of a group of people with similar interests that is known as “neighborhood”. From the identified neighborhoods, preferences can be combined to make predictions. The most commonly used approaches in this category according to the literature are the so-called *neighbourhood-based collaborative filtering (kNN-CF)*. KNN-CF algorithms [58] use statistical techniques to find neighbors with a ratings record similar to the active user ratings (*i.e.*, user for whom recommendations are done). When the nearest neighbors are found, their preferences are combined to create a list of recommendations for an active user. Two well known KNN-CF algorithms are: *user-user KNN* and *item-item KNN*.

- **User-user KNN:** This algorithm [49] uses the experience of other users to build recommendations to an active user. The input of the system is a matrix of ratings (M). Ratings are collected in advance by measuring the relevance of the items by users. The similarity between users is established by the *Pearson Correlation Coefficient (PCC)* [78]. In Equation §3.3, we compute the similarity between an active user u_k and any other user of the system u_i .

$$s_{u_k, u_i} = \frac{\sum_{p \in P} (r_{u_k, p_j} - \bar{r}_{u_k})(r_{u_i, p_j} - \bar{r}_{u_i})}{\sqrt{\sum_{p \in P} (r_{u_k, p_j} - \bar{r}_{u_k})^2} \sqrt{\sum_{p \in P} (r_{u_i, p} - \bar{r}_{u_i})^2}} \quad (3.3)$$

Where:

- s_{u_k, u_i} represents the similarity between the user k and the user i for $k \neq i$.
- P is the set of items rated by users u_k and u_i .
- r_{u_k, p_j} and r_{u_i, p_j} is the rating on the item p_j issued by the user u_k and u_i respectively.
- \bar{r}_{u_k} and \bar{r}_{u_i} is the average rating on all items also rated by u_k and u_i .

Once the similarity between the users is established, the algorithm uses these results in Equation §3.4 to predict the relevance that the user u_k would give to those items p not yet rated.

$$r_{u_k, p} = \bar{r}_{u_k} + \frac{\sum_{u \in U} (r_{u, p} - \bar{r}_{u_i}) S_{u_k, u_i}}{\sum_{u \in U} S_{u_k, u_i}} \quad (3.4)$$

Where:

- $r_{u_k, p}$ is the possible rating that would give u_k to the item p .
 - S_{u_k, u_i} is the similarity between users k and i (result of Equation §3.3).
 - U is the set of users (more) similar to u_k . This set of users varies depending on the user population. In this case the top-10 of the best ratings has been used.
- **Item-item KNN:** This algorithm, unlike the previous one, generates the recommendations based on the similarities between the items [89] rated by an active user. The similarity between items is also calculated through the PCC (i.e., similar to Equation §3.3). Equation §3.5 describes this process:

$$s_{p_i, p_j} = \frac{\sum_{u \in U} (r_{u, p_i} - \bar{r}_{p_i})(r_{u, p_j} - \bar{r}_{p_j})}{\sqrt{\sum_{u \in U} (r_{u, p_i} - \bar{r}_{p_i})^2} \sqrt{\sum_{u \in U} (r_{u, p_j} - \bar{r}_{p_j})^2}} \quad (3.5)$$

where:

- s_{p_i, p_j} determines the similarity between the items p_i and p_j .
- U is the set of all the users who have rated both the item p_i and p_j .
- r_{u, p_i} is the rating of user u on the item p_i .
- \bar{r}_{p_i} is the average rating on the item p_i .

Once the similarity between the items is established, the algorithm predicts the rating of the user u_k for an item p_i not yet rated, using the Equation §3.6.

$$r_{u_k, p_i} = \frac{\sum_{p_j \in P} r_{u_k, p_j} S_{p_i, p_j}}{\sum_{p_j \in P} |S_{p_i, p_j}|} \quad (3.6)$$

Where:

- S_{p_i, p_j} is the similarity between the items p_i and p_j (result of Equation §3.4).
- P represents the set of items more similar to the item p_i .

3.2.1.2 Model-based collaborative filtering

The ratings matrix is often very large and sparse. For this reason, model-based collaborative filtering algorithms [89] use knowledge base reduction techniques which aim to decompose the matrix into smaller ones, which reflect the common characteristics of the original matrix. These algorithms create a model through which matrices of smaller dimensions are built. These matrices represent the affinity degree between users and items. Thus, they may allow the system to recognize patterns that may be hidden in the dataset.

Unlike the algorithms based on memory, model-based collaborative filtering does not use the whole set of items and users to make predictions. Previously, it performs a pre-filtering process to create groups or user segments based on their common interests. From these segments, it establishes the recommendations.

A representative algorithm in this group is the *Matrix factorization (MF)* algorithm [48], characterized by decomposing the matrix of ratings in n sub-matrices. This algorithm, according to the literature, is designed to process large volumes of data, achieving good scalability, more accurate predictions and flexibility in the model. An example of this type of algorithm is SVD.

- **Singular Value Decomposition (SVD)** This algorithm [88] decomposes the rating matrix (M) into three matrices (U, V, S). The first one is an orthogonal matrix $U_{n \times n}$ that represents the relationships between users. The second one is an orthogonal matrix $V_{m \times m}^t$ that determines the relationships between the features. The third one is a diagonal matrix $S_{n \times m}$ that establishes the relationship between both matrices.

To improve efficiency, the algorithm applies the Eckart-Young theorem [21] that obtains an approximation with only k factors ($k < n$), so that the matrices would remain as $U_{n \times k}$, $S_{k \times k}$ and $V_{k \times m}^t$, as it is shown in Equation §3.7.

$$\text{SVD}(M_{1m \times n}) \simeq U_{m \times k} \times S_{k \times k} \times V_{k \times n}^T \quad (3.7)$$

After this process, calculating the prediction of the user u_k on the item p is reduced by multiplying the k -th row vector of the matrix U (i.e., $U(u_k)$) with the matrix S and the p – th vector column of the matrix V^t , ($V^t(p)$) as shown in the Equation §3.8.

$$ru_{k,p} = \bar{r}_{u_k} + U(u_k) \times S \times V^T(p) \quad (3.8)$$

Finally, although the collaborative filtering algorithms described use different equations to estimate the ratings ($r_{u_k,p}$) of an active user in a set of target items (see Equations §3.1, §3.4, §3.6, and §3.8), in all cases the best-rated k components are selected to build the list of recommendations.

3.2.2 Content-based recommender systems

Content-based recommender systems [66] make recommendations based on the characteristics of the items. Without the need to use information from other users. They are generally used for information retrieval, such as search engines. In this thesis, we use TF-IDF (Term frequency – Inverse document frequency) algorithm to the SPL configuration domain.

- **TF-IDF:** This algorithm [67] is commonly used to perform customized searches by internet search engines. It is characterized by being able to find the *local weight* and the *global weight* in a collection of documents that are being analyzed. The *local weight* is known as TF (*Term Frequency*) and specifies the number of times a word is repeated within a document; while the *global weight*, known as IDF (*Inverse Document Frequency*), indicates the number of documents in which that word appears at least once. The number of TF and IDF occurrences for each document determines the elements to recommend.

This algorithm use a user rating matrix M and a binary matrix N that relates the items with their features. The vector v_u describes the ratings of each user and a binary vector v_p determines the profile of each item p . The vector v_u represents the frequency in which each feature appears in

the item rated by the user u , and the vector v_p determines the presence or not of each characteristic in the items p (obtained by each row of the matrix N). Vectors v_u and v_p have the same dimension, determined by the number of features f in the matrix N .

Once we attributed the values to the vectors, the algorithm uses the TF-IDF strategy to obtain a weight for each characteristic, penalizing those that are not very similar and rewarding the most distinctive ones. Equation §3.9 presents the way to calculate the weighting of each feature f_i .

$$w_{f_i} = F(f_i, u_k) \cdot \log \left(\frac{n}{IF(f_i)} \right) \quad (3.9)$$

Where $F(f_i, u_k)$ represents the frequency which the characteristic f_i appears in the items rated by the user u_k and represents $IF(f_i)$ the inverse frequency or number of times the same characteristic appears but in the items that have not yet been rated by u_k .

Then, to recommend items to the user u_k , the algorithm calculates the cosine similarity (see Equation §3.10) between the user profile (v_{u_k}) and the profile of the items not rated by the user ($v_p, p \in \bar{P}$). In this computation, we use the weight vector calculated according to Equation §3.9.

$$S_{\cos}(v_{u_k}, v_p, w) = \frac{\sum_{i=1}^t v_{u_k}(i) * v_p(i) * w(i)}{\sqrt{\sum_{i=1}^t (v_{u_k}(i))^2} * \sqrt{\sum_{i=1}^t (v_p(i))^2}} \quad (3.10)$$

Finally, the algorithm recommends the k items with higher similarity with the user.

3.2.3 Hybrid approaches

This type of algorithm combines collaboration methods with those based on the content or with different variants of other collaboration methods. Hybrid filtering is commonly used in combination with demographic filtering or content-based filtering to exploit of each of these techniques; specifically, it is based on probabilistic methods such as genetic algorithms, diffuse genetics, neural networks, Bayesian networks, clustering and latent characteristics [25].

3.2.3.1 Approach based on demographic data

This algorithm is characterized by establishing recommendations based on the criterion that people with certain common personal attributes (e.g. sex, age, country, among others.) will also have common preferences [25].

3.3 Evaluation metrics for recommender systems

To assess the performance of the recommendations, there are many indicators and their choice depends on the system objectives, and it is determined by the judgment of their users [53]. Initially, most of the RS have been evaluated and classified according to their predictive power, their ability to accurately predict the user's options; however, it is now widely accepted that, precise predictions are crucial but insufficient to implement a good recommender engine [95]. Figure §3.2, shows the importance of the evaluation process within the RS.

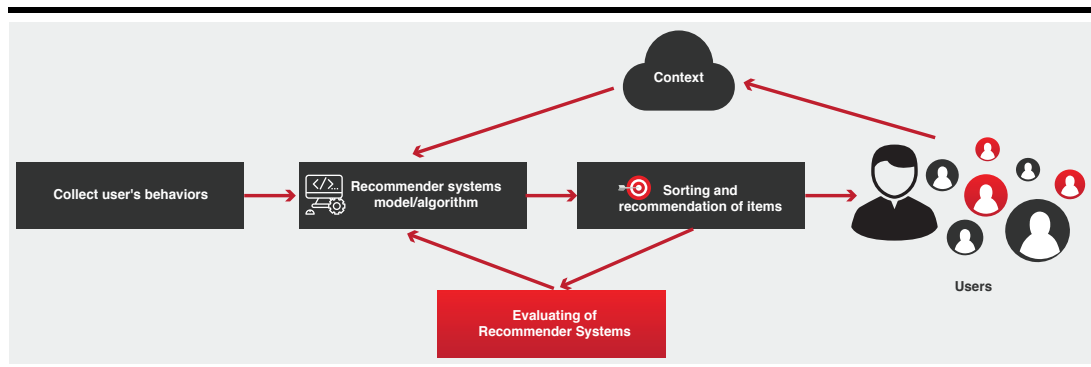


Figure 3.2: Scheme for the design and evaluation of recommender systems(from [27])

Although the core aspect of the RS is the algorithm, the evaluation of the performance of said algorithm is basic for its selection. In general, a new recommender system must complete the performance evaluation of three stages before its final release: off-line analysis, user study and online experiment [27].

To determine RS performance and compare results, commonly used indicators, metrics, or statistics used in other contexts are used that are useful in software engineering. These indicators include the following: *Mean Absolute Error* (MAE), and *Root Mean Squared Error* (RMSE), precision metrics that measure the closeness of predicted ratings to actual ratings; the *Pearson product-moment correlation* (PCC), the *Spearman Correlation Coefficient*

(SCC), the *Kendall's Tau coefficient*, the *Normalized distance-based performance measure* (NDPM), the *Area Under ROC Curve* (AUC), among others.

Root Mean Squared Error (RMSE), is a very used indicator to qualify an algorithm (Rating accuracy metrics). RMSE is ideal for the prediction task because it measures inaccuracies in all grades whether positive or negative. Low values of RMSE correspond to better predictions in precision. Other metrics derived from this group are: *Mean Square Error* (MSE), *Mean Average Error* (MAE), and *Normalized Mean Error* (NMAE). Unlike NMAE that normalizes the MAE according to the range of ratings to facilitate comparison of errors between domains, RMSE penalizes errors much more than other evaluation metrics.

Another way to assess precision of recommendations is to calculate the correlation between true ratings and system predictions (Rating and ranking correlations). For this purpose, Pearson's product-moment correlation (PCC), Spearman's correlation coefficient (SCC), and Kendall's Tau coefficient are used, which measure the degree to which a linear relationship is present between the two sets of qualifications.

Metrics have also been proposed for cases where only implicit ratings are available. In this case, the (*Classification accuracy metrics*) specially created for tasks such as "Finding good objects" appear. A well-known metric for evaluating this type of system is Area Under the Curve (AUC), which aims to measure the ability of an RS to distinguish relevant objects or objects of interest to users from irrelevant ones.

For the RS performance evaluation, the metrics can also be classified according to different perspectives [27]:

- i. Perspective of machine learning, which includes MAE, MSE and RMSE.
- ii. Perspective of information retrieval, which includes the Precision measurements, the average precision (Mean Average Precision, MAP), the ROC curve (Receiver Operating Characteristic), the average Reciprocal Range (Mean Reciprocal Rank, MRR), the Spearman rank correlation coefficient (SRCC), the standardized discounted cumulative gain (nDCG), and, the Coverage.
- iii. Perspective of human-computer interaction and user experiences, which includes Diversity, Confidence, Novelty, and, Euphoria (Serendipity).
- iv. Perspective of software engineering, which includes Real Time, Robustness and Scalability.

Perspectives	Indicators	Evaluation Content	Methods
Machine Learning	MAE MSE RMSE	Prediction Accuracy	Offline analytics
Information Retrieval	Precision Recall F-measure ROC	Precision of Recommendations	Offline analytics
	MAP MRR SRCC nDCG	Ranking precision of recommended items	Offline analytics
	Coverage	Coverage of users or items	Offline analytics
	Human-computer interaction and user experience	Diversity	Diversity of recommended items
Trust		User trust on the recommender systems	User study
Novelty		Novelty of recommended items	
Serendipity		Serendipity of recommended items	
Software Engineering	Real-time	Real-time performance of recommender systems	User study or online experiment
	Robustness	Robustness of recommender systems	Online experiment
	Scalability	Scalability of recommender systems	

Table 3.1: Metrics and methods for evaluating the performance of recommender systems (from [27])

Table §3.1 presents a summary of the perspectives, metrics and methods to evaluate the recommender systems performance.

3.4 Summary

In this chapter we have presented the main concepts and classifications of the RS deepened in this thesis. Specifically, we explore the techniques employed by collaborative-based and content-based RS. In addition, the metrics used to evaluate the validity of the recommendations.

Part III

Contributions

Chapter 4

Motivation

Difficult and meaningful will always bring more satisfaction than easy and meaningless.

Maxime Lagacé

Recently, products configuration using automation mechanisms has been one of the most relevant research areas software products line field. However, current solutions are not practical enough to deal with the common problems the software engineer faces when models grow exponentially.

In this chapter, we present some of the problems that motivate the search for new solutions and sustain our contributions. Section §4.1 describes the problems that develop around the configuration of products in an SPL; later in Section §4.2 we analyze the current proposals in the automated analysis of features models field based on two layers and the recommender systems applied to an SPL. Then, in Section §4.3, we summarize all the proposals and compare them with our contributions, emphasizing the advantages they provide and the need to implement them. Finally, we summarize the chapter in Section §4.4.

4.1 Introduction

Software product lines has had a strong impact in recent years, which has led to different investigations and practical experiences from the industry.

One of the main challenges the software product lines is the AAFM, where there is an important weakness that has not yet been completely resolved. For example, we find few solutions that allow linking two feature models and cover the problems that arise when working with models in multiple layers; that is, when we have features that interrelate with more than one model. Also, we find the lack of automation techniques for the selection of implementation components for the correct products configuration given the variability of components that may exist in an ecosystem to implement a certain feature.

In this context, as we mentioned in Chapter §2, there are several proposals aimed at supporting automation processes with different approaches; however, none of them seems to be adequate enough to solve the automation problems described above, due to the following reasons:

- i. None of the proposals analyzes the features interaction through layers of different granularity (platform layer and application layer).
- ii. There are no tools that facilitate interaction in multilayer models to prevent the changes of certain features from affecting the related layer.
- iii. So far, none of the proposals have presented an approach to facilitate the implementation of features given the amount of components that may exist to implement them.
- iv. None of the proposals present configurators for the implementation components selection in an SPL.

Therefore, we focused on developing new proposals and automation tools that cover the demand in this field of the SPL and which constitute the main motivation of this thesis. In the literature we review some of the proposals that motivate this work and support the contributions that presented in Chapters §5, §6.

4.2 Analysis of current solutions

In this section we present different approaches and analyze the research gaps.

4.2.1 Product line engineering

Back in 1990, the FODA (Feature-Oriented Domain Analysis) feasibility study by Kang *et al.* [42] introduced the *feature modeling* concept, which remained as one of the major research areas in product line engineering since then.

Feature models were used in a variety of scenarios [15], including model-driven development, feature-oriented programming, software factories or generative programming [29]. However, there are no proposals to explicitly model the interrelationships between the features in different layers, and to draw conclusions from top to bottom and from bottom to top (for example, what is the impact of a change of platform in an existing application).

An extended work by Kang *et al.* was the Feature-Oriented Reuse Method (FORM) [43], which already recognized the importance of different granularity levels. Each feature therein pertains to one of four layers (capability layer, operating environment layer, domain technology layer, implementation technique layer), and features across layers could be connected using *implemented by* relations.

The contribution we present in Chapter §5 is similar to this categorization given its different layers of FMs and focuses mainly on the relationships between the platform and implementation layers, to the point that, the useful consequences analysis of the features selection of the application layer or the platform (in both directions) is possible.

Galindo *et al.* [35] propose a framework to configure multi product lines (i.e., multiple product lines from possibly different suppliers and potentially different notations). To connect these models, they define similar inter-model dependencies. While this highlights the importance of dependencies across models, their focus is quite different as the goal is to support the end-user product configuration (spanning multiple product lines), while our approach aims at analyzing the feature interaction across layers of different granularity (which for instance requires sophisticated relations for meaningful results).

4.2.2 Feature interaction

In the early 1990s, *feature interaction* was acknowledged as a problem in the telecommunication domain, characterizing positive and, more importantly, negative side effects when introducing new features into an existing base system. It was then discovered that the basic problem of feature interac-

tion spans across a lot more disciplines, including software engineering [23].

Previous work has studied this problem in mobile phones. A case study on Nokia phones [52] tried to model feature interaction using Coloured Petri Nets, but focused on interactions with the *user interface* of the phones. Opposed to that, including the platform layer is central in our contribution.

A simple feature model of a mobile phone was depicted in [15]—although it demonstrated an example of feature interaction (a *camera requiring a high resolution screen*), its sole purpose (unlike ours) was to illustrate a sample feature model, but not to investigate the dependencies of the mobile phone’s features.

4.2.3 Model-driven architecture

Lack of portability due to technical revolution and changing requirements is one of the main problems of software development for consumer devices. MDA [64] has been proposed by OMG as a means to tackle these issues. MDA relates to our work in a way that our proposed two-layered models from Chapter §5 show a certain similarity to MDA’s separation of concerns—note the resemblance of our top layer for application functionality to the concept of Platform-independent models (PIM) in MDA. Analogously, the bottom layer for platform components resembles a Platform-specific model (PSM). While the aim of applying MDA is often to generate code, we want to reason about software and platform evolution impacts on both layers—which could be beneficial knowledge when it comes to targeting platforms in MDA.

To analyze problems from a bottom-up perspective (e.g., what is the impact on application features due to a platform change), Architecture-Driven Modernization (ADM) is a related field [65], as it is concerned with modernization of existing software solutions.

4.2.4 Customization techniques in software product lines

There are several works in the literature that uses recommender techniques to support the SPL configuration process. These studies address configuration from different perspectives and contexts, such as, feature model structural properties [8, 37, 68, 70, 86] in which the use of recommendation techniques and variability testing techniques for the selection and prioritization of configurations are proposed, and also could be used on different use cases in a particular scenario; and quality attributes [36, 69, 72, 73] that aim to use con-

textual information of the features to select optimal configurations and offer a more personalized product to the user.

These studies do not investigate the components selection specific scenario to implement features in an SPL, as shown in our second contribution in Chapter §6. As a consequence, it can be concluded that, in product configuration environments, there are no effective solutions to efficiently assist in the task of finding suitable components to implement features using recommender systems.

4.3 Discussion

In this section, we summarize some proposals that relate directly to the problems presented in the previous section and motivate the contributions presented in this thesis.

4.3.1 Automated analysis of two-layered feature models

To complement this chapter, we present some related proposals our first contribution called MAYA shown in Chapter §5. Table §4.2 summarizes some of the main characteristics of our two-layered FMs proposal and compare them with others.

Papers	Models	Layers	Attributes	O1	O2	O3	O4	Other operations	Tool	Evaluation
VFD+[59]	FM/OVM	✓						✓	✓	✓
<i>Feature Trees</i> [77]	FM	✓	✓	✓				✓	✓	✓
CVM[1]	FM	✓						✓	✓	✓
VELVET[83]	FM	✓		✓				✓	✓	
<i>SPL Workflow</i> [2]	FM	✓						✓	✓	✓
FAMILIAR[3]	FM	✓	✓	✓				✓	✓	✓
VeAnalyzer[92]	FM	✓		✓				✓	✓	✓
Invar[35]	FM	✓		✓	✓			✓	✓	✓
<i>Clafer</i> [10]	FM/CM	✓	✓					✓	✓	
<i>SPLAnE</i> [61]	FM	✓		✓	✓			✓	✓	✓
MAYA	FM	✓	✓	✓	✓	✓	✓		✓	✓

Table 4.1: MAYA vs. other proposals

The *Papers* column in Table §4.2 shows some of the works that are directly related to our first proposal.

The *Models* column in Table §4.2 indicates the type of model used by the proposals, unlike [59] that uses FMs together with orthogonal variability models, and [10] that unifies FMs with class models; all of the proposals use FMs as the main basis of the work.

The *Layers* column in Table §4.2 indicates the if the proposal support layers or not. Only [3] and [2] propose an architecture that, although it is not formally defined in the work, we have considered as two layers as it clearly distinguish two levels: the upper level destined to the requirements of the users and the lower level corresponding to the logic of operations. In any case, only MAYA uses a well-defined architecture based on two layers (see Section §??).

The *Attributes* column in Table §4.2 indicates whether the proposal uses quality attributes. We note that in addition to MAYA, [77], [3] and [10] fulfill this characteristic. Although it is true that the aforementioned proposals use attributes at a certain moment, only MAYA takes full advantage of the attributes to perform the integration of feature models between the layers.

Columns [O1 - O4] of Table §4.2 show the analysis operations implemented in our proposal for managing two layers feature models.

The column *O1* refers to the “*Platform Capability Analysis*” operation included in MAYA. Analyzing each of the proposals, we can say that all of them make use of analysis operations that, in addition to allowing the integration of large-scale models. This enables new variability management operations not previously defined.

The column *O2* refers to the operation of MAYA “*Platform Compatibility Analysis*”. Here, we find that the proposal presented in [35] allows easy adaptation with modeling techniques and notations of other existing tools. On the other hand, in the proposal [61], an analysis is done on the models to ensure that they are compatible with all the specifications given by the user, with which we would say that both proposals include analysis operations for platform compatibility. The other proposals do not include operations that allow this analysis process.

The column *O3* and the column *O4* refer to the operations of MAYA “*Application Functionality Potential Analysis*” and “*Platform Migration Analysis*” respectively. In this group, none of the proposals include analysis operations that allow easy integration of models or the ability to cope with the changes that a feature may have without affecting the related models. On the other hand, it is also not evident that the proposals presented include operations that allow the handling of conflicts that arise due to changes in the configurations of the models, and how to control that the change of one does

not affect the other or vice versa. In this sense, MAYA is a pioneer in including this type of analysis operations.

The *Tool* column in Table §4.2 indicates whether the proposal implements a tool as part of the contribution. All the proposals present a tool for the analysis of feature models in different layers. However, MAYA is the only tool that was implemented using a well-defined and structured scheme based on layers.

Finally, the *Evaluation* column in Table §4.2 indicates whether the proposal has been validated using case studies. Almost all the proposals include an evaluation, some of them carried out in the business sector such as [3], [2], [60], [77] and [1]. Although it is true that the proposals mentioned took real data provided by companies, none had direct participation during the evaluation process. MAYA, on the other hand, was evaluated in a real environment as one of the authors worked directly on projects with the companies where the case studies were carried out.

4.3.2 Recommender systems in SPL

In this section we compare our second proposal called RESDEC shown in Chapter §6, with existing literature. In Table §4.2 we summarize the main characteristics of our contribution and compare them with other proposals

Papers	Features	Implementation Components	Collaborative	Content	Algorithms	Tool
Galindo <i>et al.</i> [36]	✓				3	✓
Al-Hajjaji <i>et al.</i> [5]	✓				3	✓
Mazo <i>et al.</i> [56]					6	✓
Martinez <i>et al.</i> [55]			✓		1	✓
Pereira <i>et al.</i> [8]	✓		✓		3	✓
Pereira <i>et al.</i> [73]	✓		✓		4	
Pereira <i>et al.</i> [69]	✓		✓		4	✓
Pereira <i>et al.</i> [72]	✓		✓		1	
RESDEC	✓	✓	✓	✓	5	✓

Table 4.2: RESDEC vs. others proposals

The column *Papers* of Table §4.2 shows some works directly related to our proposal. First, we will review in the literature proposals based on the selection of configurations, then, address works related to recommender systems applied in an SPL. All proposals presented use feature models.

The *Features* and *Implementation Components* columns in Table §4.2 indicate whether the cited works employ features and implementation compo-

nents in their proposals. As it is observed, only the proposals presented by Galindo *et al.* [36], Al-Hajjaji *et al.* [5] and Pereira *et al.* [8, 69, 72, 73] make use of features. On the other hand, none of the proposals use implementation components.

In the *Collaborative* and *Content* columns of Table §4.2 we refer to the recommendation algorithms we have used in RESDEC, with which we determine whether the proposals presented use some of them in the configuration selection process; while the *Algorithms* column indicates the number of algorithms implemented.

To learn more about the proposals presented in this section, we briefly analyze each one of them.

In the work presented by Galindo *et al.* [36] the authors present a solution to prioritize configurations for testing based on value attributes, in this case the cost of testing. The proposal incorporates a prototype tool that processes a set of configuration rules for an SPL, given by the developer, through the use of cost and value functions. The proposal does not use any recommendation technique nor does it have a dataset with historical data, since the information it processes is generated manually and is not constantly updated.

Al-Hajjaji *et al.* [5] presents a proposal to prioritize configurations based on the similarity between one or more configurations. The hypothesis handled by the proposal supposes that, if a configuration presents some failure, it is probable that the similar ones also have it, having as a result more possibilities to quickly detect errors.

The techniques used to prioritize configurations according to similarity are not based on algorithms commonly used in recommender systems. On the other hand, the data used in the solution is derived from a model that is previously generated manually; that is to say, like the previous proposal, there is no mechanism that generates and automatically updates the information that is processed.

The work of Mazo *et al.* [56] presents a proposal that through a collection of heuristics and using programming with restrictions, seeks to improve the process of configuring a product to reduce the number of steps and the time to prove the validity of the product line. This work is the first one that introduces short-term recommendation techniques since it uses six algorithms to solve each of the presented heuristics. However, it is not determined whether the information shown to the user for the configuration of a product is actually known or useful, or based on the experience of other users in similar configuration processes, for example, which questions whether the final configuration

of the product will satisfy the user's requirements.

In the proposal of Martinez *et al.* [55] the authors propose the use of an interactive genetic algorithm for the selection of a relevant set of configurations for users. The proposal uses a dataset with information of configurations valued by users that is exploited by data mining techniques. Although it is true that the proposal contains all the elements that a recommendation system needs (users, items and ratings), the way how the algorithm operates in its entirety is not presented in detail. By the use of the genetic algorithm and the information that is used we could say that we are employing recommendation techniques based on collaborative filtering.

In the work of Pereira *et al.* [8] the authors present the first proposal in which a well-structured recommender system is used to configure products in an SPL. The proposed solution involves the user throughout the product configuration stage guiding the selection of features that best suit their requirements. However, the information presented to users for the configuration of the product does not come directly from criteria given by users in the past; that is, the features shown for the configuration do not have an indicator that determines if those features have been evaluated positively or negatively by the users, which makes it difficult to determine if a selected feature has been implemented successfully in past configurations.

Following the same approach as in the previous proposal, Pereira *et al.* [69] show an extension of the work presented in [70] in which it is present a tool that improves the visualization aspects for the configuration of products in an SPL by means of a recommender system based on non-functional properties (NFPs) of the features. The objective of the proposal is to ensure the consistency of the configured products and reduce the effort of those responsible for the configuration thereof.

The same authors in the work [73] present a solution that uses a recommender system to predict the features during the configuration of products using contextual information of the users, specifically, the requirements that the users define for a product. In the first proposal the user defines the requirements of the product to be configured, then the system performs a search to include historical data based on the specified requirements. Later, it creates a list of features that help the user to identify those features relevant for the configuration of the product and finally, the system checks the integrity of the configuration verifying if there are some undefined features. In case the system finds a partial configuration, it predicts the features for said configuration, complements it and shows them to the user, thus improving the general quality of the recommendation.

Finally, in the work [72] a new proposal for the configuration of products is presented based on a recommender system that uses contextual information of the users. In this case, new analysis dimensions are introduced, which go beyond the two typical dimensions, users and items, which are usually used by a recommender system. For this purpose, the authors introduce the technique of collaborative filtering Tensor Factorization (TF) [44] to automatically prioritize the features and auto-configure the SPL at execution time according to the contextual information that it originates around users and improve performance in the process of configuring a product.

All the proposals of Pereira *et al.* which have been mentioned above, employ a well-structured recommender system whose main elements are users, items and ratings. In addition, they make use of a historical knowledge base that is constantly updated. However, it can be perceived that the obtained ratings do not come from opinions that common users would give to a configuration of a product (such as errors during operation, poor design, among others). On the contrary, the information that is used is probably collected from the opinions of expert users, since common users could not accurately evaluate technical aspects related to the configuration of a product. On the other hand, the proposals only use collaborative-based recommender systems and focus exclusively on the configuration of products, leaving aside the components that implement the features during the configuration of an SPL.

Comparing the works described above with RESDEC, based on the characteristics of columns [4-7] of Table §4.2, we can say that one of the main advantages of our proposal is the use of a knowledge base that feeds of information that comes from the experience that ordinary users have had when using products in an SPL domain. Specifically, in our proposal this information is represented by the ratings that users have made about the components used to implement features in the configuration of a product. We believe that it is more feasible and real, to obtain user evaluations about the implementation components, rather than the features of the product; since as mentioned before, a common user would not have the experience to evaluate technical aspects related to a product.

The column “*Tool*” in Table §4.2 indicates whether the proposals include some prototype tool as part of contributions. All the proposals, except those presented by Pereira *et al.* [72, 73], propose a tool for configuring products in an SPL. However, RESDEC is the only tool that has been implemented using a well-defined scheme of recommender systems; and besides, it is the only one that incorporates a set of algorithms that are executed in three different scenarios that validate the platform’s capacity in terms of operationalization and scalability.

Finally, all proposals include an evaluation with information obtained from various business sectors.

4.4 Summary

In this chapter we have reviewed some of the proposals that address the problems during the products configuration in an SPL. According to the results of analysis carried out in Section §4.3.1, we conclude that our first contribution is the only one that incorporates a set of options that validate analysis capacity in terms of operationalization, and considers aspects such as migration, support and the potential to solve problems when there are changes in the definition of models of two-layered features.

In addition, from the analysis results we have presented in Section §4.3.2 about the automation of configuration processes in an SPL, we conclude that our second contribution is the first to introduce the concept of component-based recommender systems, in the features configuration of a product based on the selection of optimal components, using collaborative-based and content-based filtering techniques.

As seen, none of the proposals found in the literature manages to address all the problems at once. This justifies the originality of our contributions and contributes with the state of the art for future research.

Chapter 5

Automated analysis of two-layered feature models

If I don't know one thing, I'll look into it.

Louis Pasteur

The proliferation of features and platforms in variability intensive systems, coupled with substantial technological progress, imposes several challenges for software developers and equipment manufacturers—in some cases referred as technical sustainability. For instance, in the mobile application domain, developers often need to know the requirements and limitations of their applications to be supported on a specific platform. Conversely, an equipment manufacturer is interested in knowing what additional features become accessible on the application layer when the hardware or platform is being upgraded. To date, analyzing such interdependencies between specific feature and platform combinations is a tough problem, but important to solve. There are well-established approaches in the literature to analyze variability-intensive systems using feature models. However, there is a lack of approaches to analyze application and platform features in multiple layers.^{†1}

^{†1}Part of this chapter is published in the Journal of Visual Languages and Computing [50]. We have the agreement of all the authors to include the corresponding text as part of this manuscript.

Chapter 6

Implementation components selection using recommender systems

Everything seems impossible until it's done.

Nelson Mandela

In a Software Products Line (SPL) there may be features that can be implemented by different components, which means, that we have different implementations for the same feature. In this context, the selection of the best set of components to implement a given configuration is a challenging task given the high number of combinations and options that could be selected. In certain scenarios, it is possible to find information associated with the components that could help in this selection task, such as user ratings and performance. We introduce a component-based recommender system, called RES-DEC (REcommender System that from selecteD fEatures suggest implementation Components), that uses information associated with the implementation components to make recommendations in the domain of SPL configuration. ^{†1}

^{†1}Part of this chapter is in press in the IEEE Access Journal [80]. We have the agreement of all the authors to include the corresponding text as part of this manuscript.

Part IV

Validation

Chapter 7

MAYA: Putting variability at the application and systems level.

The greater the effort, the greater the glory.

Pierre Corneille

In a world where hardware frequently outpaces software in terms of innovation and speed up (which holds especially true for mobile phones and consumer electronics in general), a mechanism to understand the impact of these frequent technology changes on existing software is desirable. At the same time, new types of applications are enabled due to new or improved hardware components. It is equally interesting to see whether certain types of applications can be implemented on existing hardware, and to understand the limitations why some applications may not be realizable (e.g., the platform version in use doesn't support a feature yet). In this chapter, we present an empirical evaluation of MAYA approach using case studies on two real world subjects. Finally, we show how MAYA could improve the state of the art in managing the evolution of applications and platforms in the field of mobile phone application development.^{†1}

^{†1}Part of this chapter is in press in the Journal of Visual Languages and Computing [50]. We have the agreement of all the authors to include the corresponding text as part of this manuscript.

Chapter 8

RESDEC: Using recommender systems for SPL components selection

We may encounter many defeats but we must not be defeated.

Maya Angelou

Developers and configurators that creates WordPress sites face the challenge of choosing the plugins that are best suites for their needs. In this chapter, we present an empirical evaluation of RESDEC approach in an scenario of e-commerce website using WordPress. The evaluation is based on data from 116,000 users, 680 plugins, and 187,000 ratings, with a total of n independent runs of the experiment. Our experiments show that a content-based recommender algorithm produces more accurate predictions than collaborative-based recommender algorithms to support the SPL component-based configuration. Results show promising values with a margin of error of less than 15% according to our evaluation.

In addition, we introduced a prototype of a component-based recommender system tool ready to use and extend to other environments where it is necessary to configure the features of a product from the selection of implementation components.^{†1}

^{†1}This chapter is based in [80] and part of this material is published in IEEE Access journal. We have the agreement of all the authors to include the corresponding text as part of this manuscript.

Part V

Final Remarks

Chapter 9

Conclusions and future work

You get your wind back, remember the finish line, and keep going.

Steve Jobs

9.1 Conclusions

In this dissertation, we have presented a set of algorithms, techniques and tools to support products configuration in an SPL. These contributions are the result of the application of intelligent techniques explored in new implementation scenarios, such as in two-layer feature model to study the dependence of both; and, in the selection of implementation components to configure the features of a product.

Our main results were the development of two prototypes of tools to address the contributions described above. MAYA that through a set of operations allows to manage the dependence and the changes that can arise in the features of a two-layer model (application layer and platform layer), and RESDEC that was designed to help the software engineer in the selection of components.

Specifically, in this dissertation we have shown that:

Connecting two-layer feature models for the top layer (comprising application functionality) and bottom layer (including platform components) using relationships between trees allows to study the consequences of changes in one layer on the other and determine possible problems that may occur when working with different SPL products.

When it comes to platform and design decisions, both developers and handset manufacturers could profit from such a model, because potential problems and incompatibilities can be identified at a very early development stage. Furthermore, such an instrument can be used to discuss design decisions with marketing and other involved stakeholders who might not have detailed technical knowledge.

The first objective of this dissertation was to provide two complex feature models for the respective layers, as a prerequisite for further research. We identified the required type of inter-tree relationships and how they are compared to well-known cross-tree relationships.

Specifically the operations: (O1) platform capability analysis, that focus on identifies the minimally required platform features; (O2) platform compatibility analysis, that is focus on compare the required platform features for a specific application and determine their (in-)compatibility level; (O3) application functionality potential analysis, that focus on find the corresponding application features that are enabled by a certain platform, and; (O4) platform migration analysis potential, that focus on conflicts of a platform migration and how are existing application features affected by exchanging the underlying platform; set the stage for further analysis.

A prototype implementation in Section ?? on Chapter §7 was used to demonstrate the validity of the proposed approach.

The selection of optimal implementation components for the configuration of an SPL through the use of recommender systems will help the software engineer to make decisions in a more timely manner, saving time and available resources.

Normally, a software developer searches manually those components that are feasible and most suitable to a certain application. This task takes time and does not always guarantee that the selected components are the most adequate (in terms of quality) for the required application. To the best of our knowledge in configurable development environments, there are no effective solutions to efficiently assist the developer in the task of finding suitable implementation components for its application.

Consequently, the impact of implementation components selection in an SPL configuration is a costly and error-prone activity due to the large number possible components that could be selected to implement a feature.

The second objective of this dissertation was the use of explicit user-generated information about implementation components, specifically ratings made to the components according to their experience of use. For this pur-

pose, we have used collaborative-based and content-based recommender systems, which are executed in three scenarios where the software engineer usually faces problems when products configuration.

The first scenario called *Cold start* that recommends components when there is no information associated with the user profile, that is, when the user has not had experience and for the first time is going to configure a product. The second scenario called *Recommendations of implementation components based on ratings*, which, based on the components linked to the user profile, recommends components that other users have used in past configurations. Finally, the scenario *Recommendations of implementation components based on features*, which recommends implementation components based on the features of the components associated with the user profile, that is, in the descriptive information of the components.

The modeling of the problem for the implementation components selection using collaborative-based and content-based recommender systems algorithms and the design of a prototype tool for RESDEC are the new contributions of this dissertation. The results obtained in the evaluation (see Chapter §8) carried out using a WordPress dataset show that RESDEC is capable of making recommendations on implementation components with an error lower than 13%.

This dissertation present results to support the configuration of implementation components in the domain of SPL. Moreover, this contribution could be also applied to other environments that face similar problems, such as, the selection of deployment environments for mobile applications.

9.2 Future work

In this section we show the future work that arises from the contributions presented in this dissertation and that are necessary to cover all the research related to the current trends in automated analysis applied to the configuration of an SPL. It is divided into two areas: first shows the future work derived from the use of multiple layers in feature models; and the second, in the implementation components selection to configure an SPL.

9.2.1 Two-layered feature models

In this dissertation we have addressed the use of two-layered feature models through the use of computer-assisted mechanisms to manage them. However, there are some unresolved problems that we intend to address in the future. Here are some of the proposed challenges:

Extending the operation catalogue.- While we propose four operations in the contributions shown in Chapter §5 of this dissertation, there are others open for extension. For instance, a possible new operation would be to ask for the minimally required platform version to support a particular application. However, this would require to take into account model versioning that is a feature not supported in our current proposal. Also, we can imagine operations that instead of retrieving a configuration work over the inter-tree relationships and analyze them looking for inconsistencies or errors.

Graphical frontend for analysis input/output.- As mentioned in Chapter §7 of this dissertation, for our prototypical implementation we do not employ tool support for product configuration of the input, nor is there a graphical representation of the analysis output. There is ongoing work^{†1} to integrate FaMa toolsuite with various modeling tools (e.g., MOSKitt, pure::variants). While these tools help to model each layer individually, we plan an extension that supports configuration on both layers concurrently, e.g. by combining them in a graphical frontend. Providing support for the complex task of defining mappings is very important [38]. At the same time, such assistance also ensures the integrity of the inter-tree relations, as only valid features and relation types can be selected.

Presenting the output in a visual format, similar to the figures in Section §??, will be another beneficial extension. The different colors to reflect the meanings of the different output categories (e.g., a conflicting feature), will help the user to easily understand the impact of his operation. It is envisioned to be implemented in a way that by just hovering over single features, the respective feature(s) on the connected layer can be highlighted, giving the user a real-time feedback upon (de-)selection of a feature.

Analysis with a unified feature model.- As mentioned in Section §??, proposal shown in Chapter §5 of this dissertation was designed to work exclusively with two-layer models since in a real environment there are different practitioners that make hardware and software, where each layer evolves in a different way[35]. Having the platform and application layers separate enables us to control the changes that may occur in the applications when they

^{†1}http://www.isa.us.es/fama/?FaMa_Current_Projects

update or when they migrate to different platforms without affecting their performance.

However, in the future we could evaluate the integration of the two layers in a single feature model where the hardware and software layers would be represented by two mandatory sub-features. In this way, we would show how a change in one feature in the software can affect the hardware or vice versa and how these variations could be managed in both scenarios to determine which responds better to the transitions that feature models can experience.

More-than-two layers.- In this dissertation we have proposed a framework relying on two levels abstraction. However, we envision that there might be scenarios where having more than two levels is interesting. In future work, we plan to extend this framework to support as many levels of feature models required in the same spirit as in other domains such as in DSLs [84].

Operation formalization.- In the past, we have already formalized several FMs operations [32] for single model scenarios. We plan to perform the same formalization for the operations shown in Chapter §5 considering more than one model in future work.

9.2.2 Implementation components selection

In this dissertation we have approached computer-assisted automation techniques based on recommender algorithms for implementation components selection that help the customization and optimization of SPL products. However, there are still some challenges to be solved:

Recommendations using implicit information.- In the contribution presented in Chapter §6 shown in this dissertation, we have only considered recommender techniques that take into account explicit information of the users (*i.e.*, ratings). As future work, we aim to include implicit information from events defined indirectly by the user, such as, number of clicks, number of views, etc.

For example, in addition to including user ratings on the plugins, we could also include implicit information as the number of downloads, number of views and versions of the plugins. Information that would allow the recommender system to work in a more personalized way. In the literature, there are several recommender algorithms that use implicit information to make the recommendations [47, 79]. Our future aim is to adapt this algorithms to enrich the recommendations of implementation components in the domain of SPL configuration.

Recommendations using of contextual information.- We aim at recommending components based on contextual information derived from the users, features and configurations by adapting the algorithms proposed in Pereira *et al.* [71] and which can be processed according to the RESDEC components presented in Figure §?? of Chapter §6.

The objective that we seek with the implementation of this scenario is to guide the user during the process of configuring a product. This way, as the user progresses in the configuration of a product, RESDEC is able to automatically suggest which feature can be selected to complement the partial configuration based on associated descriptive information to the features, users and configurations.

To face the handling of contextual information and offer a personalized product to the user of better quality, we intend to extend the benefits offered by the techniques of Factoring Matrix by introducing the technique of collaborative filtering Tensor Factorization (TF) [72] that allows an integration of contextual data that does not focus only on information from matrices of user and items. In our case, this technique will allow us to explore beyond the contextual information of the components of implementation of the features, and will facilitate us to involve other dimensions of study such as, for example, contextual information of the features and configurations not considered in this dissertation.

An approximation to this type of recommendation could be the following, suppose that the user who set up a website for *tourism promotion* also set up a website for *travel*; when new users set up a *tourism promotion* website, it is likely for the system to recommend the configuration of the website for *travel*. Note that in this case we make use of contextual information that is developed around a valid configuration of a product.

Recommendation of configurations for testing.- The RESDEC objective, shown in Chapter §6, is to be able to recommend configurations more susceptible to errors and therefore could be candidates for testing. The aim is to provide to the person in charge of supervising the quality of the SPL an automated mechanism that allows him to select the configurations more error-prone. For this, we are based the hypothesis that, the configurations with the lowest rating by the users are those which tend to contain more errors. For example, if we have designed a mobile application for *tourist promotion*, the system should be able to recommend which configurations of mobile devices testing should be performed. For this purpose, mobile devices in which similar applications have had unfavorable ratings will be recommended for testing. To make this type of recommendations, we will use the RESDEC recommendation elements and algorithms shown in Figure §?? of Chapter §6.

As mentioned above, results obtained provide a solid basis for our doctoral thesis and leaves the way open for future research contributing to the progress of software engineering in software product lines area.

Part VI

Appendix

Appendix A

RESDEC: Online Management Tool for Implementation Components Selection

In this appendix we describe a prototype component-based recommender system called RESDEC (REcommender System that suggest implementation Components from selected fEatures) designed to generate implementation component recommendations in the configuration of a product line based on WordPress e-commerce websites. In addition, we present demo video publish on YouTube, that introduce RESDEC tool.

A.1 Introduction

Software product lines (SPL) management is one of the most important activities for the software engineer and it represents one of the key pieces of software product line engineering. When a software system grows fast, configuring a product becomes a costly and error-prone activity due to the amount of features available for configuration. This process becomes more complex when for each feature, there is more than one component that implements it. Currently the tools available for configuration management do not have automated mechanisms to facilitate the optimal components selection that meet the functions required by a given product. In this appendix, we introduce a prototype component-based recommender system called RESDEC (REcommender System that suggests implementation Components from selected features) designed to manage the best implementation components alternatives. Our tool is validated using WordPress-based websites where the implementation components are represented by plugins and the recommendations generated by RESDEC help interested parties in the search and efficient plugins selection to configure websites.

A.2 RESDEC Tool Suite

RESDEC offers two main functionalities: component repository management and automated analysis in the implementation components selection through recommender systems. The following are some advantages of RESDEC:

- It is easy to adapt to any SPL configuration environment. To do this, the knowledge base has been designed based on three attributes commonly used by a recommender system (i.e. users, items and ratings). This allowed us, that algorithms implemented in RESDEC receive these parameters as input and run without problems in any SPL scenario, for example: WordPress, Android, Mozilla, among others.
- It offers information about the implementation components and the ratings history made by the user.
- It provides a set of recommender algorithms that can be extended to provide better recommendation results in the three scenarios presented in this paper.

- It offers on screen, an updated history of the last components of implementation that have been of interest to the user.
- It allows obtaining recommendations, in execution time, of the most appropriate implementation components according to the feature selected by the user.
- It incorporates a case study based on a website software product lines that validates the scope of our tool.

A.2.1 Architecture

RESDEC Tool has three components: a repository manager, a recommender manager and an output manager. The *repository manager* responds

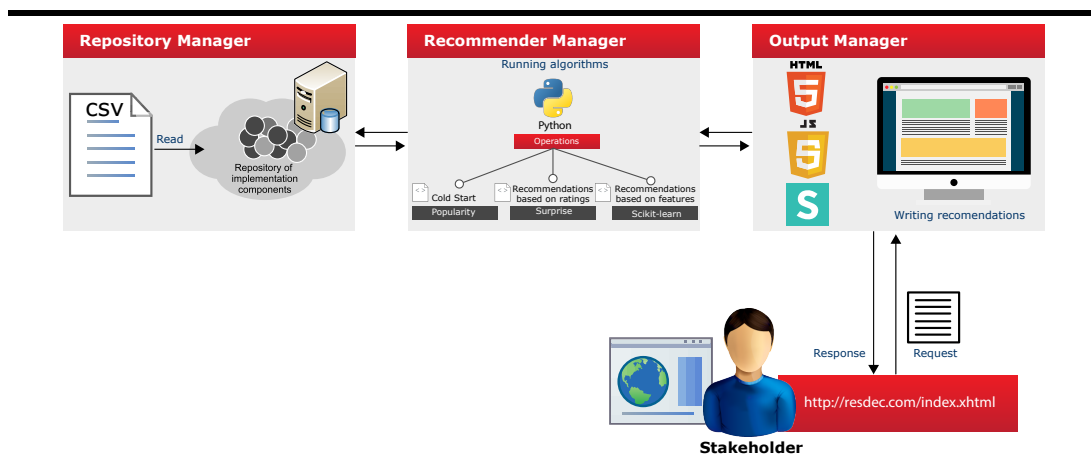


Figure A.1: RESDEC architecture

to the requests of the stakeholders and structures the matrices $M1$ and $M2$ of the *Knowledge base* presented in Section §?? through CSV's.

The *recommender manager* is in charge of processing the recommendations based on the three scenarios presented in Section §??. It is developed in *Python* with a package of libraries that contain the algorithms that the recommender manager runs according to the scenario selected by the stakeholder.

For the *Cold Start* scenario presented in Section §?? RESDEC uses a classical popularity algorithm. While for the algorithms that run in the scenario *Recommendation of implementation components based on ratings* presented in Section §??., employs the Scikit-surprise library^{†1}; and for the *Recommendation*

^{†1}Surprise website: <http://surprise.readthedocs.io/en/stable/>

tion of implementation components based on features scenario presented in Section §??, it uses the Scikit-learn library^{†2}.

The recommender manager is scalable and offers the possibility of implementing new similarity metrics and recommender algorithms in any of the three scenarios presented in this thesis.

The *output manager* interacts directly with the stakeholder using the repository manager and the recommender manager to generate the list of suggestions for the implementation components. It is designed in HTML5 and JavaScript, supported by the Semantic UI framework^{†3} used for the design of the interfaces. The interaction between the stakeholder and RESDEC is done through a web browser.

In general, the *output manager* is responsible for receiving the requests of the stakeholders and informing the *recommender manager* of the requirements so the appropriate algorithm is run with the information that the *repository manager* responds, it finally displays the generated recommendations on the screen.

A.2.2 Web Application

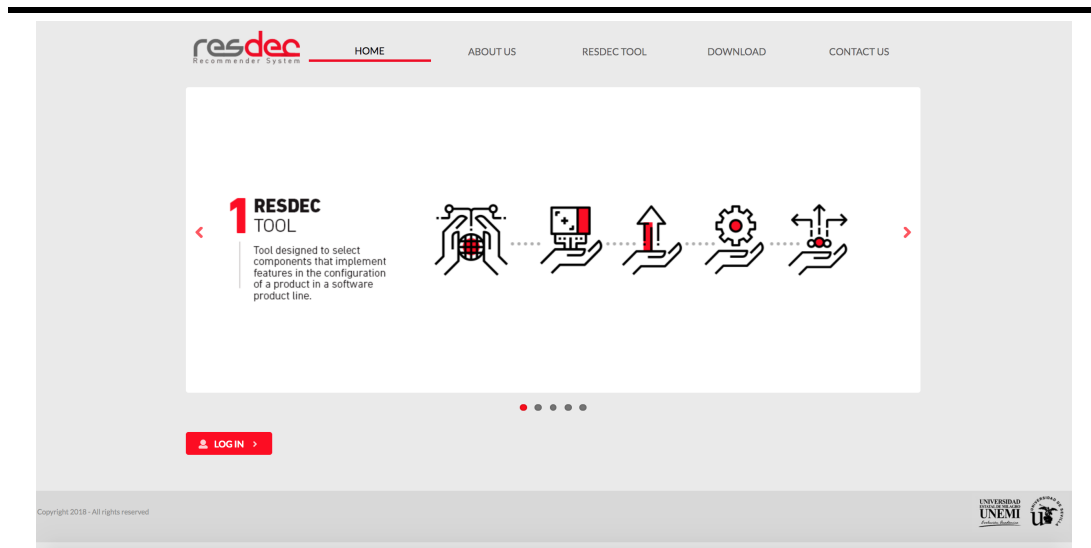


Figure A.2: RESDEC web application

^{†2}Scikit-learn website: <http://scikit-learn.org/stable/index.html>

^{†3}Semantic website: <https://semantic-ui.com/>

To make our work accessible to the community, we present a RESDEC web application that eases the generation of recommendations to stakeholders that require guided assistance in the selection of plugins to configure a products line based on WordPress websites (see Figure §A.2).

RESDEC is available at *www.resdec.com* and uses a dataset from information extracted from Wordpress. Specifically plugins, tags and ratings that users do about them. To enter RESDEC app, press the Login button and then enter the login credentials. To access as a guest, we have created a test user profile User: admin - Password: admin123 (see Figure §A.3).

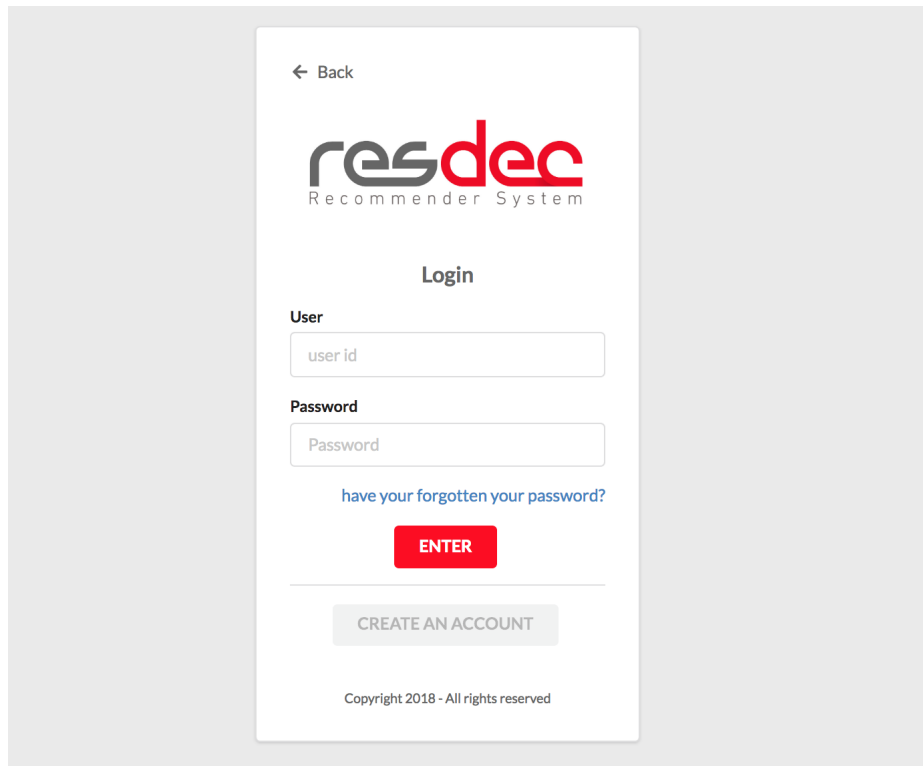


Figure A.3: RESDEC login

The main screen of RESDEC presents a menu with three recommendation scenarios where stakeholders or users can configure an SPL, additionally incorporates a case study about e-commerce websites developed in WordPress (see Figure §A.4). Next, we describe the different functionalities of our tool through an example based on the configuration of a tourism website in WordPress.

- i. The *Cold Start* option recommends components when there is no in-

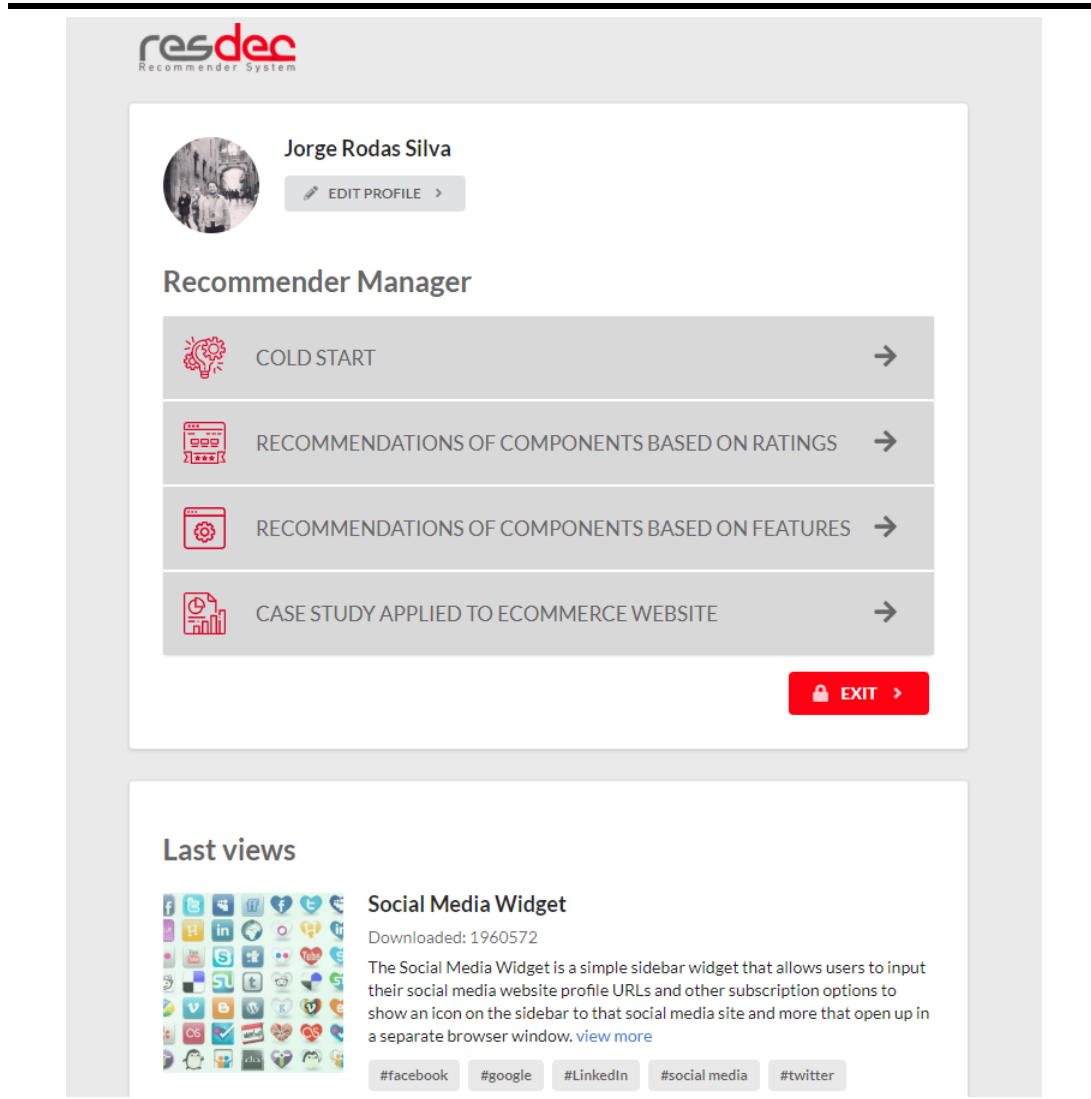


Figure A.4: RESDEC main screen

- formation associated with the user profile, i.e., when the user has no experience and is setting up a website for the first time. Suppose we are going to set up a new tourism website and we need to implement the *Social Media* function. In this case, the user selects the tag or tags associated with *Social Media* and specifies the number of desired recommendations. With the information provided by the user, RESDEC sets the recommendations based on the popularity of WordPress plugins and does not use the information associated with the user's profile (see Figure §A.5).
- ii. The second option, *Recommendations of implementation components*

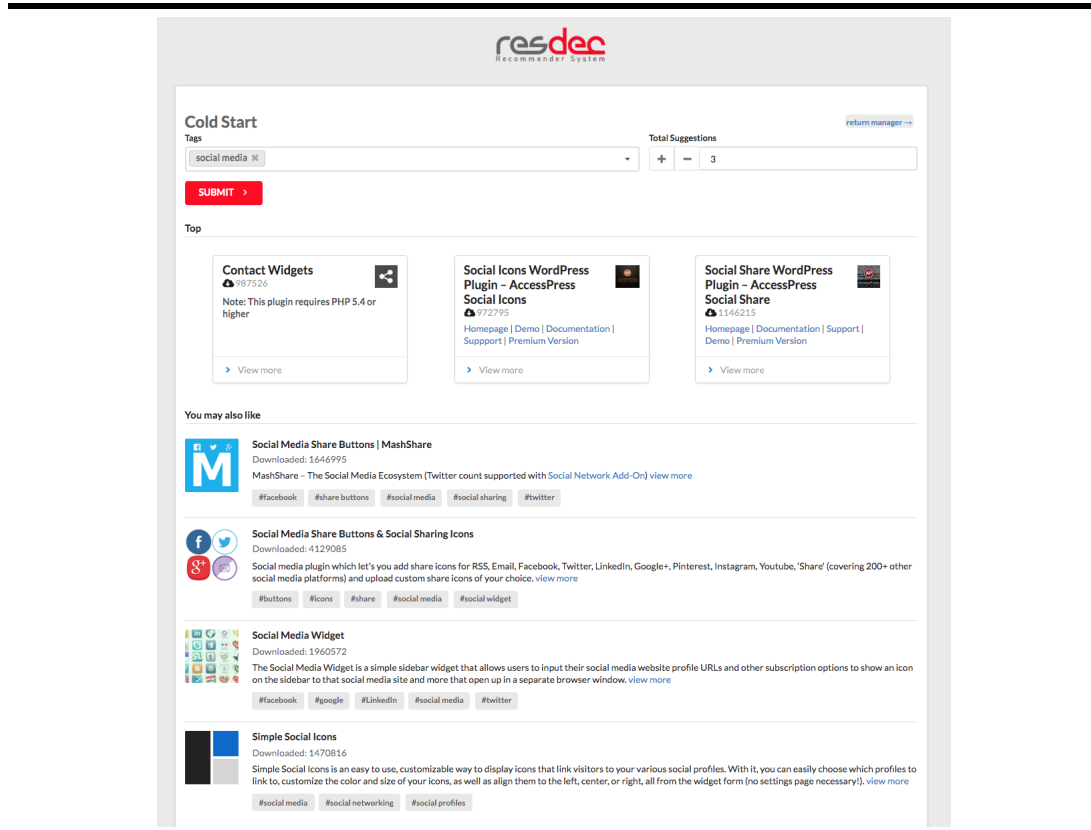


Figure A.5: Cold start scenario

based on ratings, from a component used by the user in previous configurations, recommends those components that users with similar profiles have used in the configuration of a product. Suppose that we are going to configure a *tourism website* that has already been implemented and in which we need new recommendation alternatives for *Social Media* function. In this case, the user selects the implemented plugin, *social-media-widget*, then specifies the number of desired recommendations and selects the algorithm to execute (SVD, item-item KNN or user-user KNN). With the information provided by the user, RESDEC establishes the recommendations based on the ratings that plugins similar to the one selected have been used by other users. In this scenario, RESDEC uses the information associated with the user's profile (see Figure §A.6).

- iii. The third option, *Recommendations of implementation components based on features*, recommends implementation components based on the features of a component used by the user in previous configurations. That is, the list of recommendations is established based on the descriptive information of the components associated with the user profile. Sup-

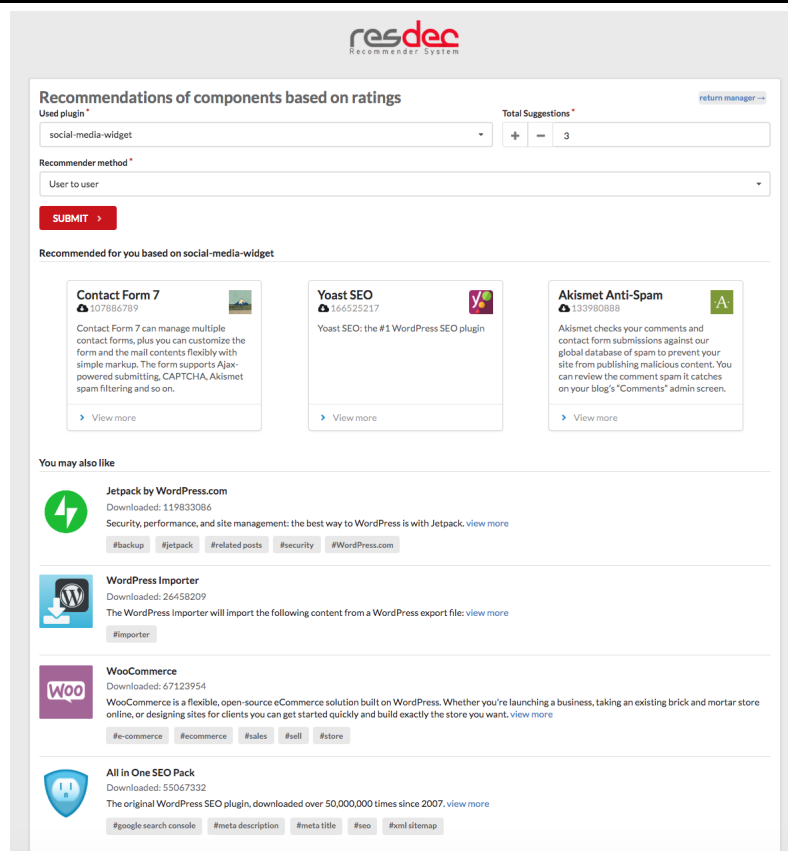


Figure A.6: Recommendations of implementation components based on ratings

pose that we are going to configure a *tourism website* that has already been implemented and we need to replace or complement the feature *Social Media*, in this case, first the user selects the plugin implemented, *social-media-widget*, then the system will display the list of tags associated with the plugin through which it will establish recommendations. Then, we specify the number of desired recommendations and select the execution algorithm (TF-IDF). In this case, RESDEC establishes the recommendations based on the features, that is, on the similarity of the tags associated to the selected plugin with other plugins that use one or more of these tags. In this scenario, RESDEC also uses the information associated to the user's profile (see Figure §A.7).

- iv. The option *Case Study applied to e-commerce website*, shows a Feature Model that was built from information about websites designed in WordPress that are available on the Internet. This Feature Model has been implemented in an interactive way and describes the relations be-

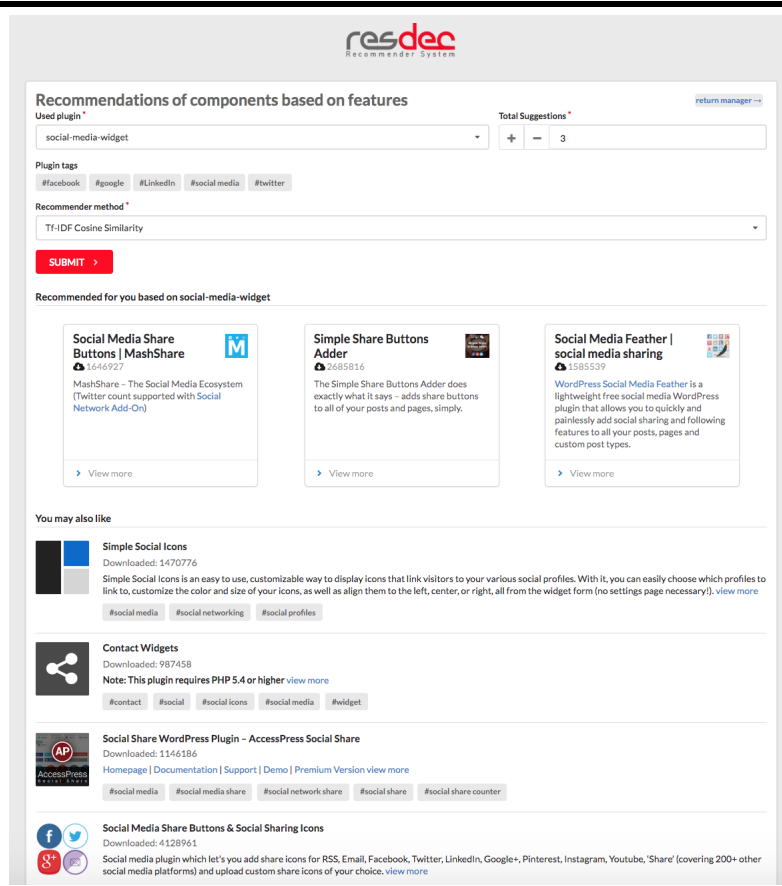


Figure A.7: Recommendations of implementation components based on features

tween features, the same ones that can appear when configuring an e-commerce website on this platform.

In the Feature Model, for example, by clicking on the *Shopping Cart* feature, the lower part of the model is configured for each scenario, showing only information associated with the selected feature. Thus, for scenario 1 it will show only the list of tags associated to that feature, in the same way in scenarios 2 and 3, it will display only the plugins that implement that feature. The recommendations in each scenario are executed in a similar way as described above (see Figure §A.8).

All recommendations presented in the different scenarios, show for each plugin a brief description and a button called “View More” that links to the WordPress website to learn more about it. Additionally, there is an additional list called “You may also like” that shows other plugin options similar to those recommended and that may be of interest to the user (see Figure §A.9).

The screenshot displays the RESDEC Recommender System interface for an eCommerce website case study. At the top, a hierarchical tree diagram shows the structure of the website's features, starting from 'eCommerce Site' and branching into categories like 'Layout Manager', 'Catalogue', 'Search', 'Online Shopping', 'Contact', 'CRM', 'Security', and 'Marketing'. A legend indicates that solid circles represent mandatory features, while open circles represent optional features. Below the tree, a red banner states: 'You have selected the feature: Shopping Cart, now you can use any of the 3 scenarios.' Underneath, there are two tabs: 'Cold Start' and 'Recommendations of components based on ratings'. The 'Cold Start' section is active, showing a search bar with the tag 'woocommerce' and a 'SUBMIT' button. To the right, it shows 'Total Suggestions' as 3. Below this, three recommended plugins are displayed in a grid:

- WooCommerce PDF Invoices & Packing Slips** (ID: 2443592): This WooCommerce extension automatically adds a PDF invoice to the order confirmation emails sent out to your customers. Includes a basic template (additional templates are available from WP Overnight) as well as the possibility to modify/create your own templates. In addition, you can choose to download or print invoices and packing slips from the WooCommerce order admin.
- YITH WooCommerce Quick View** (ID: 883183): Sometimes the halfway is better: what if you are looking to a product in a list and its image is still too small?
- WooCommerce Services** (ID: 4290266): WooCommerce Services makes basic eCommerce features like shipping more reliable by taking the burden off of your site's infrastructure.

Below the recommendations, there is a 'You may also like' section with two more plugins:

- YITH WooCommerce Zoom Magnifier** (Downloaded: 1674841): Improve the user experience, improve your sales [view more](#). Tags: #carousel, #magnifier, #slider, #woocommerce, #zoom.
- YITH WooCommerce Wishlist** (Downloaded: 6649450): What can really make the difference in conversions and amount of sales is without a doubt the freedom to share your own wishlist, even on social networks, increasing indirect sales: can you imagine the sales volume you can generate during holidays or birthdays, when relatives and friends will be looking for the wishlist of your clients to buy a gift? [view more](#). Tags: #e-commerce, #products, #Wishlist, #woocommerce, #yit.

Figure A.8: Case Study applied to e-commerce Website

Finally, there is an option called “About this case study”, when you click on this option, it shows in detail the process that was carried out to build the feature model. In addition, We present a video demonstration of the RESDEC web tool that is available on YouTube at https://youtu.be/iz_bpj0rJLE (see Fig-

Top

Auto Terms of Service and Privacy Policy (WP AutoTerms)
 1145674

WP AutoTerms plugin helps you with a wide range of legal requirements your WordPress website might be required to keep up with, such as the GDPR law or the requirement to have a disclosure for affiliate links.

> View more

Nextend Social Login and Register (Facebook, Google, Twitter)
 1541338

Nextend Facebook Connect, Nextend Google Connect and Nextend Twitter Connect are discontinued and Nextend Social Login takes their place. Feel free to update the old plugins and enjoy Nextend Social Login.

[Demo](#) | [Tutorial videos](#) | [Docs](#) | [Support](#) | [Pro Addon](#)

> View more

Open Graph for Facebook, Google+ and Twitter Card Tags
 1017616

This plugin inserts Facebook Open Graph Tags into your WordPress Blog/Website for more effective and efficient Facebook sharing results.

> View more

You may also like

Facebook Widget
 Downloaded: 848537

One of the most popular & lightweight plugin for Facebook page feeds widget with over 8,50,000 downloads and 1,00,000+ active installs. [view more](#)

#facebook #Facebook fan pages #facebook like box #facebook like button #Facebook simple like

Facebook Like Box Widget
 Downloaded: 792570

Facebook Like Box Widget is a social plugin that enables Facebook Page owners to attract and gain Likes & Recommendation Comments from their own website. The Like Box enables users to: see how many users already like this page, and which of their friends like it too, read recent posts from the page and Like the page with one click, without needing to visit the page. [view more](#)

#facebook #facebook badge #facebook button #facebook like #facebook like button

PixelYourSite - Facebook Pixel, Google Analytics, Head & Footer (WooCommerce, EDD, Events)
 Downloaded: 950697

Manage your Facebook Pixel or Google Analytics code with a single plugin and add ANY other script (Head & Footer feature). The Pinterest Tag can be implemented via [free add-on](#). [view more](#)

Facebook Pixel, Google Analytics, ANY other script, WooCommerce & EDD

Figure A.9: Screen “You may also like” that show recommendation alternatives

ure §A.10).

resdec
Recommender System

UNIVERSIDAD
ESTATAL DE MILAGRO
UNEMI
Enseñanza Avanzada



Figure A.10: Screen of RESDEC tool demo

Bibliography

- [1] A. Abele, Y. Papadopoulos, D. Servat, M. Törngren, and M. Weber. The cvm framework-a prototype tool for compositional variability management. *VaMoS*, 10:101–105, 2010
- [2] M. Acher, P. Collet, A. Gaignard, P. Lahire, J. Montagnat, and R. B. France. Composing multiple variability artifacts to assemble coherent workflows. *Software Quality Journal*, 20(3-4):689–734, 2012
- [3] M. Acher, P. Collet, P. Lahire, and R. B. France. Familiar: A domain-specific language for large scale management of feature models. *Science of Computer Programming*, 78(6):657–681, 2013
- [4] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, June 2005
- [5] M. Al-Hajjaji, T. Thüm, J. Meinicke, M. Lochau, and G. Saake. Similarity-based prioritization in software product-line testing. In *Proceedings of the 18th International Software Product Line Conference-Volume 1*, pages 197–206. ACM, 2014
- [6] A. Anandhan, L. Shuib, M. A. Ismail, and G. Mujtaba. Social media recommender systems: Review and open research issues. *IEEE Access*, 6:15608–15628, 2018
- [7] S. Apel and C. Kästner. An overview of feature-oriented software development. *Journal of Object Technology*, 8(5):49–84, 2009
- [8] J. Arriel. *Personalized recommender systems for software product line configurations*. PhD thesis, Otto von Guericke Universität Magdeburg, 2018
- [9] M. A. Babar, L. Chen, and F. Shull. Managing variability in software product lines. *Software, IEEE*, 27(3):89–91, 2010

- [10] K. Bak, Z. Diskin, M. Antkiewicz, K. Czarnecki, and A. Wkasowski. Clafer: unifying class and feature modeling. *Software & Systems Modeling*, 15(3):811–845, 2016
- [11] M. Bashari, E. Bagheri, and W. Du. Dynamic software product line engineering: a reference framework. *International Journal of Software Engineering and Knowledge Engineering*, 27(02):191–234, 2017
- [12] R. Bashroush, M. Garba, R. Rabiser, I. Groher, and G. Botterweck. Case tool support for variability management in software product lines. *ACM Comput. Surv.*, 50(1):14:1–14:45, March 2017
- [13] D. Batory. Feature models, grammars, and propositional formulas. In *International Conference on Software Product Lines*, pages 7–20. Springer, 2005
- [14] D. Batory, D. Benavides, and A. Ruiz-Cortés. Automated analysis of feature models: challenges ahead. *Communications of the ACM*, 49(12): 45–47, 2006
- [15] D. Benavides, S. Segura, and A. Ruiz-Cortés. Automated analysis of feature models 20 years later: A literature review. *Information Systems*, 35(6):615–636, 2010
- [16] D. Benavides, P. Trinidad, and A. Ruiz-Cortés. Automated reasoning on feature models. In *Advanced Information Systems Engineering*, pages 381–390. Springer, 2005
- [17] D. Benavides and J. A. Galindo. Automated analysis of feature models: Current state and practices. In *Proceedings of the 22Nd International Systems and Software Product Line Conference - Volume 1, SPLC '18*, pages 298–298, New York, NY, USA, 2018. ACM
- [18] D. Benavides, S. Segura, P. Trinidad, and A. Ruiz-Cortés. A first step towards a framework for the automated analysis of feature models. *Proc. Managing Variability for Software Product Lines: Working With Variability Mechanisms*, pages 39–47, 2006
- [19] D. Benavides, P. Trinidad, A. Ruiz-Cortés, and S. Segura. Fama. In *Systems and Software Variability Management*, pages 163–171. Springer, 2013
- [20] J. Bentahar, M. Taghavi, K. Bakhtiyari, and C. Hanachi. New Insights Towards Developing Recommender Systems. *The Computer Journal*, 61(3):319–348, 06 2017

- [21] M. W. Berry, S. T. Dumais, and G. W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM review*, 37(4):573–595, 1995
- [22] V. Bischoff, K. Farias, and L. Gonçalves. Evaluating the effort of integrating feature models: A controlled experiment. 07 2018
- [23] L. Blair, G. S. Blair, J. Pang, and C. Efstratiou. Feature interactions outside a telecom domain. In *FICS*, pages 15–20, 2001
- [24] J. Bobadilla, A. Hernando, F. Ortega, and J. Bernal. A framework for collaborative filtering recommender systems. *Expert Systems with Applications*, 38(12):14609 – 14623, 2011
- [25] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109–132, 2013
- [26] R. Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002
- [27] M. Chen and P. Liu. Performance evaluation of recommender systems. *International Journal of Performability Engineering*, 13(8), 2017
- [28] P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. The SEI series in software engineering. Addison-Wesley, Boston and Mass. and London, 2001
- [29] K. Czarnecki and U. Eisenecker. *Generative programming: Methods, tools, and applications*. Addison Wesley, Boston, 2000
- [30] K. Czarnecki and S. Helsen. Feature modeling. *Generative Programming*, pages 82–130, 1998
- [31] P. Donohoe. Introduction to software product lines. In *2011 15th International Software Product Line Conference*, pages 350–350, Aug 2011
- [32] A. Durán, D. Benavides, S. Segura, P. Trinidad, and A. R. Cortés. FLAME: a formal framework for the automated analysis of software product lines validated by automated specification testing. *Software and System Modeling*, 16(4):1049–1082, 2017
- [33] J. A. Galindo, D. Benavides, and S. Segura. Debian packages repositories as software product line models. towards automated analysis. In *International Workshop on automated configuration and tailoring of applications (ACoTA)*, pages 29–34, 2010

- [34] J. A. Galindo, D. Benavides, P. Trinidad, A.-M. Gutiérrez-Fernández, and A. Ruiz-Cortés. Automated analysis of feature models: Quo vadis? *Computing*, pages 1–47, 2018
- [35] J. A. Galindo, D. Dhungana, R. Rabiser, D. Benavides, G. Botterweck, and P. Grünbacher. Supporting distributed product configuration by integrating heterogeneous variability modeling approaches. *Information and Software Technology*, 62:78–100, 2015
- [36] J. A. Galindo, H. Turner, D. Benavides, and J. White. Testing variability-intensive systems using automated analysis: an application to android. *Software Quality Journal*, 24(2):365–405, 2016
- [37] Y. Gonzalez-Fernandez, S. Hamidi, S. Chen, and S. Liaskos. Efficient elicitation of software configurations using crowd preferences and domain knowledge. *Automated Software Engineering*, pages 1–37, 2018
- [38] F. Heidenreich, J. Kopcsek, and C. Wende. Featuremapper: Mapping features to models. In *Companion of the 30th International Conference on Software Engineering, ICSE Companion '08*, pages 943–944. ACM, New York, NY, USA, 2008
- [39] S. Ida and S. Ketil. Technology research explained. Technical report, 2007
- [40] M. Javed, M. Naeem, and H. A. Wahab. Towards the maturity model for feature oriented domain analysis. *Computational Ecology and Software*, 4(3):170–183, 2014
- [41] W. Jirapanthong. Experience on re-engineering applying with software product line. *arXiv preprint arXiv:1206.4120*, 2012
- [42] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson. Feature-oriented domain analysis (FODA) feasibility study. Technical report, DTIC Document, 1990
- [43] K. C. Kang, S. Kim, J. Lee, K. Kim, G. J. Kim, and E. Shin. Form: A feature-oriented reuse method with domain-specific reference architectures. *Annals of Software Engineering*, 5:143–168, 1998
- [44] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86. ACM, 2010

- [45] A. Knüppel, T. Thüm, S. Mennicke, J. Meinicke, and I. Schaefer. Is there a mismatch between real-world feature models and product-line research? In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, pages 291–302. ACM, 2017
- [46] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. GroupLens: applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997
- [47] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008
- [48] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009
- [49] S. K. Lam and J. Riedl. Shilling recommender systems for fun and profit. In *Proceedings of the 13th international conference on World Wide Web*, pages 393–402. ACM, 2004
- [50] M. Lettner, J. Rodas-Silva, J. A. Galindo, and D. Benavides. Automated analysis of two-layered feature models with feature attributes. *Journal of Computer Languages*, 2019
- [51] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003
- [52] L. Lorentsen, A.-P. Tuovinen, and J. Xu. Modelling feature interactions in mobile phones. In *FICS*, pages 7–13, 2001
- [53] L. Lü, M. Medo, C. H. Yeung, Y.-C. Zhang, Z.-K. Zhang, and T. Zhou. Recommender systems. *Physics reports*, 519(1):1–49, 2012
- [54] M. Marques, J. Simmonds, P. O. Rossel, and M. C. Bastarrica. Software product line evolution: A systematic literature review. *Information and Software Technology*, 105:190 – 208, 2019
- [55] J. Martinez, G. Rossi, T. Ziadi, T. F. D. A. Bissyandé, J. Klein, and Y. Le Traon. Estimating and predicting average likability on computer-generated artwork variants. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 1431–1432. ACM, 2015

- [56] R. Mazo, C. Dumitrescu, C. Salinesi, and D. Diaz. Recommendation heuristics for improving product line configuration processes. In *Recommendation Systems in Software Engineering*, pages 511–537. Springer, 2014
- [57] I. P. McCarthy. Special issue editorial: the what, why and how of mass customization. *Production Planning & Control*, 15(4):347–351, 2004
- [58] P. Melville and V. Sindhvani. Recommender systems. In *Encyclopedia of machine learning*, pages 829–838. Springer, 2011
- [59] A. Metzger and K. Pohl. Variability management in software product line engineering. In *Software Engineering - Companion, 2007. ICSE 2007 Companion. 29th International Conference on*, pages 186–187, 2007
- [60] A. Metzger, K. Pohl, P. Heymans, P.-Y. Schobbens, and G. Saval. Disambiguating the documentation of variability in software product lines: A separation of concerns, formalization and automated analysis. In *Requirements Engineering Conference, 2007. RE'07. 15th IEEE International*, pages 243–253. IEEE, 2007
- [61] G. K. Narwane, J. A. Galindo, S. N. Krishna, D. Benavides, J.-V. Millo, and S. Ramesh. Traceability analyses between features and assets in software product lines. *Entropy*, 18(8):269, 2016
- [62] H. Nguyen-Tan, H. Huynh-Huu, and H. Huynh-Xuan. Collaborative filtering recommendation in the implication field. *International Journal of Machine Learning and Computing*, 8(3), 2018
- [63] L. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley, 2002
- [64] OMG. Mda guide version 1.0.1, 2003
- [65] OMG. Adm whitepaper: Transforming the enterprise, 2007
- [66] M. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine learning*, 27(3):313–331, 1997
- [67] M. J. Pazzani and D. Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007
- [68] J. A. Pereira. A collaborative-based recommender system for configuration of extended product lines. In *Proceedings of the 39th International Conference on Software Engineering Companion*, pages 445–448. IEEE Press, 2017

- [69] J. A. Pereira, J. Martinez, H. K. Gurudu, S. Krieter, and G. Saake. Visual guidance for product line configuration using recommendations and non-functional properties. 2018
- [70] J. A. Pereira, P. Matuszyk, S. Krieter, M. Spiliopoulou, and G. Saake. A feature-based personalized recommender system for product-line configuration. In *Proceedings of the 2016 ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences*, pages 120–131. ACM, 2016
- [71] J. A. Pereira, P. Matuszyk, S. Krieter, M. Spiliopoulou, and G. Saake. Personalized recommender systems for product-line configuration processes. *Computer Languages, Systems & Structures*, 2018
- [72] J. A. Pereira, S. Schulze, E. Figueiredo, and G. Saake. N-dimensional tensor factorization for self-configuration of software product lines at runtime. In *Proceedings of the 22Nd International Systems and Software Product Line Conference - Volume 1, SPLC '18*, pages 87–97, New York, NY, USA, 2018. ACM
- [73] J. A. Pereira, S. Schulze, S. Krieter, M. Ribeiro, and G. Saake. A context-aware recommender system for extended software product line configurations. In *Proceedings of the 12th International Workshop on Variability Modelling of Software-Intensive Systems*, pages 97–104. ACM, 2018
- [74] E. Pimenidis, N. Polatidis, and H. Mouratidis. Mobile recommender systems: Identifying the major concepts. *Journal of Information Science*, 45(3):387–397, 2019
- [75] K. Pohl, G. Bockle, and F. Van Der Linden. *Software product line engineering: Foundations, principles, and techniques*. Springer-Verlag New York Inc, 2005
- [76] M. Pol'la, A. Buccella, A. Cechich, and M. Arias. Un modelo de metadatos para la gestión de la variabilidad en líneas de productos de software. In *XLIII Jornadas Argentinas de Informática e Investigación Operativa (43JAIO)-XV Simposio Argentino de Ingeniería de Software (Buenos Aires, 2014)*, 2014
- [77] M.-O. Reiser and M. Weber. Multi-level feature trees. *Requirements Engineering*, 12(2):57–75, 2007
- [78] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM, 1994

- [79] F. Ricci, L. Rokach, and B. Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011
- [80] J. Rodas-Silva, J. A. Galindo, J. García-Gutiérrez, and D. Benavides. Selection of software product line implementation components using recommender systems: An application to wordpress. *IEEE Access*, pages 1–1, 2019
- [81] J. Rodas-Silva, D. Méndez-Acuña, J. A. Galindo, D. Benavides, and J. Cárdenas. Towards testing variability intensive systems using user reviews. In *2015 10th Computing Colombian Conference (10CCC)*, pages 39–46. IEEE, 2015
- [82] J. Rodas-Silva, J. Olivares, J. A. G. Duarte, and D. Benavides. Hacia el uso de sistemas de recomendación en sistemas de alta variabilidad. In *CEDI 2016*, 2016
- [83] M. Rosenmüller, N. Siegmund, T. Thüm, and G. Saake. Multi-dimensional variability modeling. In *Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems*, pages 11–20. ACM, 2011
- [84] A. Rossini, J. de Lara, E. Guerra, A. Rutle, and U. Wolter. A formalisation of deep metamodelling. *Formal Aspects of Computing*, 26(6):1115–1152, Nov 2014
- [85] A. Saini and S. Rajkumar. Software product line configurations generation using different types of tools—a comparison. 2017
- [86] A. B. Sánchez, S. Segura, and A. Ruiz-Cortés. The drupal framework: A case study to evaluate variability testing techniques. In *Proceedings of the Eighth International Workshop on Variability Modelling of Software-Intensive Systems (VAMOS 2014)*, number 11, Nice, France, 01/2014 2014. ACM, ACM
- [87] L. E. Sánchez, J. A. Diaz-Pace, and A. Zunino. A family of heuristic search algorithms for feature model optimization. *Science of Computer Programming*, 172:264–293, 2019
- [88] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, Minnesota Univ Minneapolis Dept of Computer Science, 2000

- [89] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001
- [90] J. B. Schafer, J. Konstan, and J. Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 158–166. ACM, 1999
- [91] P.-Y. Schobbens, P. Heymans, J.-C. Trigaux, and Y. Bontemps. Generic semantics of feature diagrams. *Computer Networks*, 51(2):456–479, 2007
- [92] R. Schröter, T. Thüm, N. Siegmund, and G. Saake. Automated analysis of dependent feature models. In *Proceedings of the Seventh International Workshop on Variability Modelling of Software-intensive Systems*, page 9. ACM, 2013
- [93] S. Segura, J. A. Galindo, D. Benavides, J. A. Parejo, and A. Ruiz-Cortés. Betty: Benchmarking and testing on the automated analysis of feature models. In *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems, VaMoS '12*, pages 63–71, New York, NY, USA, 2012. ACM
- [94] S. Segura Rueda. Functional and performance testing of feature model analysis tools extending the fama ecosystem. 2010
- [95] G. Shani and A. Gunawardana. Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer, 2011
- [96] L. Terán, A. O. Mensah, and A. Estorelli. A literature review for recommender systems techniques used in microblogs. *Expert Systems with Applications*, 103:63–73, 2018
- [97] T. Thum, C. Kastner, S. Erdweg, and N. Siegmund. Abstract features in feature modeling. In *Software Product Line Conference (SPLC), 2011 15th International*, pages 191–200. IEEE, 2011
- [98] T. Thüm, C. Kästner, F. Benduhn, J. Meinicke, G. Saake, and T. Leich. Featureide: An extensible framework for feature-oriented software development. *Science of Computer Programming*, 79:70 – 85, 2014
- [99] D. Véras, T. Prota, A. Bispo, R. Prudêncio, and C. Ferraz. A literature review of recommender systems in the television domain. *Expert Systems with Applications*, 42(22):9046 – 9076, 2015

- [100] B. Wang, W. Zhang, H. Zhao, Z. Jin, and H. Mei. A use case based approach to feature models' construction. In *2009 17th IEEE International Requirements Engineering Conference*, pages 121–130, Aug 2009
- [101] H. Zare, M. A. N. Pour, and P. Moradi. Enhanced recommender system using predictive network approach. *Physica A: Statistical Mechanics and its Applications*, 520:322 – 337, 2019
- [102] P. Zipkin. Mass customization. *MIT Sloan management review*, 2001

This document was typeset on 2019/7/26 using $\mathcal{R}\mathcal{C}\text{-}\mathcal{B}\mathcal{O}\mathcal{K}$ $\alpha 2.11$ for $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}_{2\epsilon}$.
Should you want to use this document class, please send mail to
contact@tdg-seville.info.