



## 4 an Ellaborating overview of Configurable software: a Tertiary Study

FOSD 2023 Meeting | Rahel Arens, Thomas Thüm | March 31, 2023



## 4 an Ellaborating overview of Configurable software: a Tertiary Study - 4 ECTS

FOSD 2023 Meeting | Rahel Arens, Thomas Thüm | March 31, 2023

# Motivation

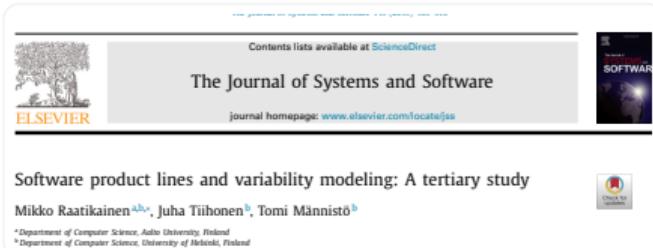
The figure displays three research papers from the journal "The Journal of Systems and Software".

**Top Paper:** "Software product lines and variability modeling: A tertiary study" by Mikko Raatikainen<sup>a,b,\*</sup>, Juha Tiihonen<sup>b</sup>, Tomi Männistö<sup>b</sup>. The paper is available at [www.elsevier.com/locate/jss](#).

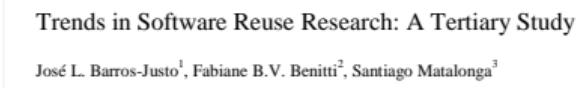
**Middle Paper:** "Systematic Studies in Software Product Lines: A Tertiary Study" by Marimuthu C and K. Chandrasekaran.

**Bottom Paper:** "Trends in Software Reuse Research: A Tertiary Study" by José L. Barros-Justo<sup>1</sup>, Fabiane B.V. Benitti<sup>2</sup>, Santiago Matalonga<sup>3</sup>.

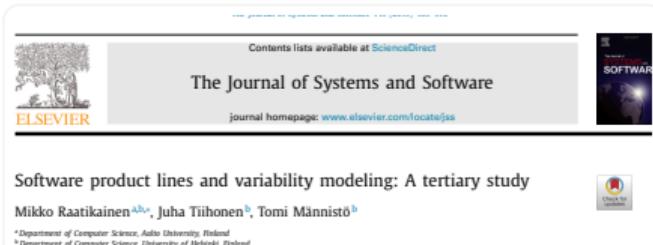
# Motivation



Raatikainen et al. 2019  
Resulted in 86 found studies



# Motivation



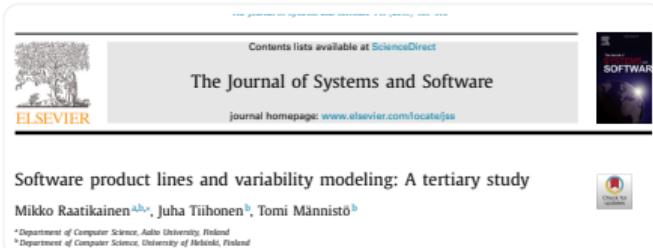
Raatikainen et al. 2019  
Resulted in 86 found studies



Marimuthu and Chandrasekaran  
2017  
Resulted in 60 found studies



# Motivation



Raatikainen et al. 2019  
Resulted in 86 found studies



Marimuthu and Chandrasekaran  
2017  
Resulted in 60 found studies



Barros-Justo et al. 2019  
Resulted in 56 found studies

# Motivation

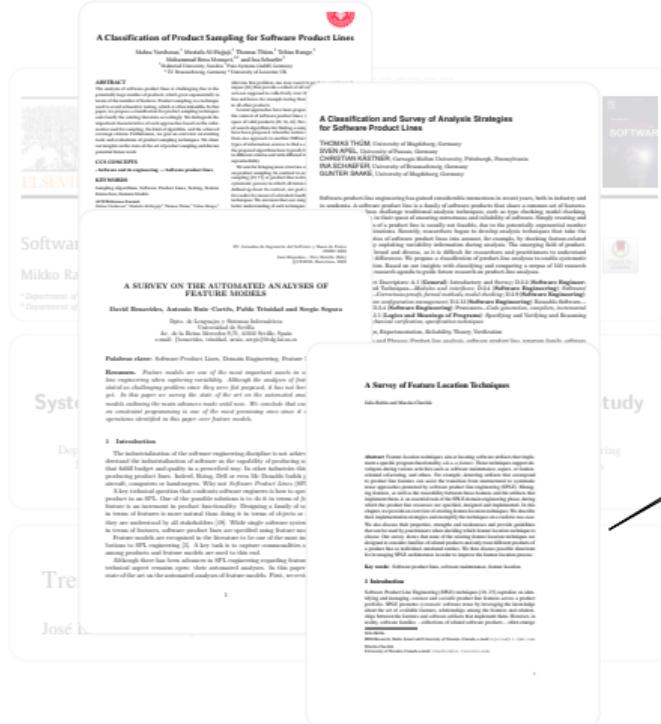


Raatikainen et al. 2019  
Resulted in 86 found studies

Marimuthu and Chandrasekaran  
2017  
Resulted in 60 found studies

Barros-Justo et al. 2019  
Resulted in 56 found studies

# Motivation



And what about us?

# Methodology

1. Created Golden Sample (38 publications)

# Methodology

1. Created Golden Sample (38 publications)
2. Forward and Backward snowballing for all publications after 2013 (19/38 publications)

# Methodology

1. Created Golden Sample (38 publications)
2. Forward and Backward snowballing for all publications after 2013 (19/38 publications)
3. Criteria:
  - Publication has to be in english
  - Publications including product-line development, configurable software or feature-oriented development
  - Access to the paper is available
  - Peer-reviewed (with a benefit of the doubt)

# Methodology

1. Created Golden Sample (38 publications)
2. Forward and Backward snowballing for all publications after 2013 (19/38 publications)
3. Criteria:
  - Publication has to be in english
  - Publications including product-line development, configurable software or feature-oriented development
  - Access to the paper is available
  - Peer-reviewed (with a benefit of the doubt)
4. Results:
  - 73 publications found with backward snowballing
  - 33 publications found with forward snowballing

# Methodology

1. Created Golden Sample (38 publications)
  2. Forward and Backward snowballing for all publications after 2013 (19/38 publications)
  3. Criteria:
    - Publication has to be in english
    - Publications including product-line development, configurable software or feature-oriented development
    - Access to the paper is available
    - Peer-reviewed (with a benefit of the doubt)
  4. Results:
    - 73 publications found with backward snowballing
    - 33 publications found with forward snowballing
- ⇒ Overall a collection of literature reviews containing 144 publications

# **1. (Software) Product Lines**

# (Software) Product Lines

Software and Systems Modeling (2021) 20:1063–1077  
https://doi.org/10.1007/s11219-020-00619-w

REGULAR PAPER

## Analysis of variability models: a systematic literature review

Maria Polta<sup>1</sup> · Agustina Bucella<sup>1</sup> · Alejandra Cachic<sup>1</sup>

Received: 16 June 2019 / Revised: 11 September 2020 / Accepted: 27 October 2020 / Published online: 22 November 2020  
© Springer Nature Switzerland, part of Springer Nature 2020

### Abstract

Dealing with variability, during Software Product Line Engineering (SPLs), means trying to allow software engineers to develop a set of similar applications based on a manageable range of variable functionalities according to expert users' needs. Particularly, variability management is an activity that allows flexibility at a high level of reuse. In this paper, we have conducted a systematic literature review of variability models and their application in SPLs. We have identified 20 variability models, and we have analyzed them for SPLs in general, and for its analysis in particular. More precisely, a specific field has emerged, named (automated) variability analysis, focusing on verifying variability models across the SPLs' phases. In this paper, we introduce a systematic literature review of variability models and their application in SPLs. The main contribution of this paper is the definition of a framework for variability models, which is composed of 20 sub-characteristics addressing general aspects, such as scope and validation, as well as model-specific aspects, such as variability primitive, manner type. The framework allows to look at the analysis of variability models during its whole life cycle, from design to deployment, according to the activities involved during an SPL development.

Pol'ta et al.  
2021

### Introduction

Software Product Line Engineering (SPLs) [14, 57, 77] is focused on defining commonalities and variabilities of several program families, called products. The set of common features, or *functionalities*, of a Software Product Line (SPL) is known as a *product line*, which serves as a basis for building each resulting product.

A product line is a collection of related products, sharing a common core, a common architecture, quality, or characteristics of a certain system or system [30].

The ability to produce a task in function.

Communicated by Ansgar Wenzel.

This work is partially supported by the UNICO Project (UNICO: “Nuevo de software modular a demanda – Parte II” part of the project UNICO: “Nuevo de software modular a demanda – Parte I”

Agustina Bucella  
agustina.bucella@unc.edu.ar

<sup>1</sup> GECIB Research Group, Departamento de Ingeniería de Sistemas, Facultad de Informática, Universidad Nacional de Córdoba, Avda. Alvaro Alfonzo 950, 5000 Córdoba, Argentina



Received: 6 January 2017 / Revised: 23 May 2017 / Accepted: 26 August 2017  
DOI: 10.1007/s11219-017-0222

WILEY Software and Systems Modeling

SPECIAL ISSUE PAPER

## A systematic mapping study of information visualization for software product line engineering

Roberto Erick Lopez-Herrejon<sup>1</sup> · Sheny Illescas<sup>2</sup> · Alexander Egyed<sup>2</sup>

<sup>1</sup>École de technologie supérieure, Université du Québec, Montréal, Canada  
Université du Québec à Montréal, Line Justice  
Correspondence  
Roberto Erick Lopez-Herrejon, Department of Software Engineering, École de technologie supérieure, Université du Québec à Montréal, Canada  
Email: erick.lopez@etri.umontreal.ca  
Funding information  
American Society for Quality (ASQ), Canadian Foundation for Innovation (CFI), Natural Sciences and Engineering Research Council of Canada (NSERC), NSERC Graduate Scholarships (GSC), NSERC Discovery Grant (DG)

**Abstract**  
Software products from (SPLs) are families of related systems whose members are distinguished by the set of features they provide. One of the main research and practice areas in SPLs is the application of visualization techniques for SPLs, such as better understanding of individual software reuse, and faster time to market. Typically, SPLs offer multiple feature configurations that are combined to form the final product. Because of the sheer amount of information and its complexity, visualizations become increasingly useful for better understanding of SPLs. This paper presents a systematic mapping study of information visualization for SPLs. The results show that information visualization is used for better understanding of individual software reuse, and faster time to market.

Lopez-Herrejon et al.  
2018

### 1 INTRODUCTION

Software product line (SPLs) are families of related systems whose members are distinguished by the set of features they provide.<sup>1</sup> Variability is the capacity of software artifacts to change, and re-use management and reutilization is at the core of successful SPL development.<sup>2</sup> Features are core-line structures that establish the relationship between features, and have become the de-facto standard for modeling variability.<sup>3</sup> Over the last decades, extensive research and practical work in academic and industry related to the value-added benefits of applying SPLs practices.<sup>4–7</sup> Among the benefits are better customization, improved software reuse, and faster time to market.

Information visualization is a discipline that studies how data can be represented and communicated to humans.<sup>8</sup> It involves the process of extracting a large number of features that are combined to form the final product. Such flexibility is an advantage for reuse, as it allows reuse of individual software systems without first identifying, defining, and configuring variability types, confronting the process known as variability management. This is particularly important for reuse, as reuse is one of the main drivers of SPL development. In general, variability is finely modeled by selecting domain requirements as a set of restrictions and then associated to avoid inconsistencies and ambiguities.

In this context, an important research field has emerged in the last years, named (automated) analysis of variability models by [10], as a set of techniques, activities and methods.

We believe that by adding 2 new requirements (R9Q and R10Q) that concern the type of variability and the presence or the absence of variability models, we can improve the quality of the results of the systematic mapping study. We also added 2 new requirements (R11Q and R12Q) that concern the type of variability and the presence or the absence of variability models.

1 https://doi.org/10.1007/s11219-017-0222

2 https://doi.org/10.1007/s11219-017-0222

3 https://doi.org/10.1007/s11219-017-0222

4 https://doi.org/10.1007/s11219-017-0222

5 https://doi.org/10.1007/s11219-017-0222

6 https://doi.org/10.1007/s11219-017-0222

7 https://doi.org/10.1007/s11219-017-0222

8 https://doi.org/10.1007/s11219-017-0222

9 https://doi.org/10.1007/s11219-017-0222

10 https://doi.org/10.1007/s11219-017-0222

11 https://doi.org/10.1007/s11219-017-0222

12 https://doi.org/10.1007/s11219-017-0222

13 https://doi.org/10.1007/s11219-017-0222

14 https://doi.org/10.1007/s11219-017-0222

15 https://doi.org/10.1007/s11219-017-0222

16 https://doi.org/10.1007/s11219-017-0222

17 https://doi.org/10.1007/s11219-017-0222

18 https://doi.org/10.1007/s11219-017-0222

19 https://doi.org/10.1007/s11219-017-0222

20 https://doi.org/10.1007/s11219-017-0222

21 https://doi.org/10.1007/s11219-017-0222

22 https://doi.org/10.1007/s11219-017-0222

23 https://doi.org/10.1007/s11219-017-0222

24 https://doi.org/10.1007/s11219-017-0222

25 https://doi.org/10.1007/s11219-017-0222

26 https://doi.org/10.1007/s11219-017-0222

27 https://doi.org/10.1007/s11219-017-0222

28 https://doi.org/10.1007/s11219-017-0222

29 https://doi.org/10.1007/s11219-017-0222

30 https://doi.org/10.1007/s11219-017-0222

31 https://doi.org/10.1007/s11219-017-0222

32 https://doi.org/10.1007/s11219-017-0222

33 https://doi.org/10.1007/s11219-017-0222

34 https://doi.org/10.1007/s11219-017-0222

35 https://doi.org/10.1007/s11219-017-0222

36 https://doi.org/10.1007/s11219-017-0222

37 https://doi.org/10.1007/s11219-017-0222

38 https://doi.org/10.1007/s11219-017-0222

39 https://doi.org/10.1007/s11219-017-0222

40 https://doi.org/10.1007/s11219-017-0222

41 https://doi.org/10.1007/s11219-017-0222

42 https://doi.org/10.1007/s11219-017-0222

43 https://doi.org/10.1007/s11219-017-0222

44 https://doi.org/10.1007/s11219-017-0222

45 https://doi.org/10.1007/s11219-017-0222

46 https://doi.org/10.1007/s11219-017-0222

47 https://doi.org/10.1007/s11219-017-0222

48 https://doi.org/10.1007/s11219-017-0222

49 https://doi.org/10.1007/s11219-017-0222

50 https://doi.org/10.1007/s11219-017-0222

51 https://doi.org/10.1007/s11219-017-0222

52 https://doi.org/10.1007/s11219-017-0222

53 https://doi.org/10.1007/s11219-017-0222

54 https://doi.org/10.1007/s11219-017-0222

55 https://doi.org/10.1007/s11219-017-0222

56 https://doi.org/10.1007/s11219-017-0222

57 https://doi.org/10.1007/s11219-017-0222

58 https://doi.org/10.1007/s11219-017-0222

59 https://doi.org/10.1007/s11219-017-0222

60 https://doi.org/10.1007/s11219-017-0222

61 https://doi.org/10.1007/s11219-017-0222

62 https://doi.org/10.1007/s11219-017-0222

63 https://doi.org/10.1007/s11219-017-0222

64 https://doi.org/10.1007/s11219-017-0222

65 https://doi.org/10.1007/s11219-017-0222

66 https://doi.org/10.1007/s11219-017-0222

67 https://doi.org/10.1007/s11219-017-0222

68 https://doi.org/10.1007/s11219-017-0222

69 https://doi.org/10.1007/s11219-017-0222

70 https://doi.org/10.1007/s11219-017-0222

71 https://doi.org/10.1007/s11219-017-0222

72 https://doi.org/10.1007/s11219-017-0222

73 https://doi.org/10.1007/s11219-017-0222

74 https://doi.org/10.1007/s11219-017-0222

75 https://doi.org/10.1007/s11219-017-0222

76 https://doi.org/10.1007/s11219-017-0222

77 https://doi.org/10.1007/s11219-017-0222

78 https://doi.org/10.1007/s11219-017-0222

79 https://doi.org/10.1007/s11219-017-0222

80 https://doi.org/10.1007/s11219-017-0222

81 https://doi.org/10.1007/s11219-017-0222

82 https://doi.org/10.1007/s11219-017-0222

83 https://doi.org/10.1007/s11219-017-0222

84 https://doi.org/10.1007/s11219-017-0222

85 https://doi.org/10.1007/s11219-017-0222

86 https://doi.org/10.1007/s11219-017-0222

87 https://doi.org/10.1007/s11219-017-0222

88 https://doi.org/10.1007/s11219-017-0222

89 https://doi.org/10.1007/s11219-017-0222

90 https://doi.org/10.1007/s11219-017-0222

91 https://doi.org/10.1007/s11219-017-0222

92 https://doi.org/10.1007/s11219-017-0222

93 https://doi.org/10.1007/s11219-017-0222

94 https://doi.org/10.1007/s11219-017-0222

95 https://doi.org/10.1007/s11219-017-0222

96 https://doi.org/10.1007/s11219-017-0222

97 https://doi.org/10.1007/s11219-017-0222

98 https://doi.org/10.1007/s11219-017-0222

99 https://doi.org/10.1007/s11219-017-0222

100 https://doi.org/10.1007/s11219-017-0222

101 https://doi.org/10.1007/s11219-017-0222

102 https://doi.org/10.1007/s11219-017-0222

103 https://doi.org/10.1007/s11219-017-0222

104 https://doi.org/10.1007/s11219-017-0222

105 https://doi.org/10.1007/s11219-017-0222

106 https://doi.org/10.1007/s11219-017-0222

107 https://doi.org/10.1007/s11219-017-0222

108 https://doi.org/10.1007/s11219-017-0222

109 https://doi.org/10.1007/s11219-017-0222

110 https://doi.org/10.1007/s11219-017-0222

111 https://doi.org/10.1007/s11219-017-0222

112 https://doi.org/10.1007/s11219-017-0222

113 https://doi.org/10.1007/s11219-017-0222

114 https://doi.org/10.1007/s11219-017-0222

115 https://doi.org/10.1007/s11219-017-0222

116 https://doi.org/10.1007/s11219-017-0222

117 https://doi.org/10.1007/s11219-017-0222

118 https://doi.org/10.1007/s11219-017-0222

119 https://doi.org/10.1007/s11219-017-0222

120 https://doi.org/10.1007/s11219-017-0222

121 https://doi.org/10.1007/s11219-017-0222

122 https://doi.org/10.1007/s11219-017-0222

123 https://doi.org/10.1007/s11219-017-0222

124 https://doi.org/10.1007/s11219-017-0222

125 https://doi.org/10.1007/s11219-017-0222

126 https://doi.org/10.1007/s11219-017-0222

127 https://doi.org/10.1007/s11219-017-0222

128 https://doi.org/10.1007/s11219-017-0222

129 https://doi.org/10.1007/s11219-017-0222

130 https://doi.org/10.1007/s11219-017-0222

131 https://doi.org/10.1007/s11219-017-0222

132 https://doi.org/10.1007/s11219-017-0222

133 https://doi.org/10.1007/s11219-017-0222

134 https://doi.org/10.1007/s11219-017-0222

135 https://doi.org/10.1007/s11219-017-0222

136 https://doi.org/10.1007/s11219-017-0222

137 https://doi.org/10.1007/s11219-017-0222

138 https://doi.org/10.1007/s11219-017-0222

139 https://doi.org/10.1007/s11219-017-0222

140 https://doi.org/10.1007/s11219-017-0222

141 https://doi.org/10.1007/s11219-017-0222

142 https://doi.org/10.1007/s11219-017-0222

143 https://doi.org/10.1007/s11219-017-0222

144 https://doi.org/10.1007/s11219-017-0222

145 https://doi.org/10.1007/s11219-017-0222

146 https://doi.org/10.1007/s11219-017-0222

147 https://doi.org/10.1007/s11219-017-0222

148 https://doi.org/10.1007/s11219-017-0222

149 https://doi.org/10.1007/s11219-017-0222

150 https://doi.org/10.1007/s11219-017-0222

151 https://doi.org/10.1007/s11219-017-0222

152 https://doi.org/10.1007/s11219-017-0222

153 https://doi.org/10.1007/s11219-017-0222

154 https://doi.org/10.1007/s112

# (Software) Product Lines

Information and Software Technology 72 (2014) 1–16

Contents lists available at ScienceDirect

Information and Software Technology journal homepage: [www.elsevier.com/locate/infsof](http://www.elsevier.com/locate/infsof)



A bibliometric analysis of 20 years of research on software product lines

Rutha Heradio<sup>a,\*</sup>, Héctor Pérez-Moreno<sup>b</sup>, David Fernández-Amoros<sup>c</sup>, Francisco Javier Calzón<sup>c</sup>, Enrique Herrera-Viedma<sup>c</sup>

<sup>a</sup>Department of Computer Science, University of Almería, Almería 04071, Spain

<sup>b</sup>Department of Computer Science and Artificial Intelligence, University of Granada, Granada 18071, Spain

<sup>c</sup>Department of Computer Science and Artificial Intelligence, University of Valencia, Valencia 46100, Spain

**ARTICLE INFO**

**Article history:** Received 11 November 2012; Received in revised form 21 December 2013; Accepted 22 December 2013; Available online 2 December 2013.

**ABSTRACT**

Software product line engineering has proven to be an efficient paradigm to developing families of related artifacts. In order to support the evolution of software product lines (SPLs), it becomes necessary to manage these variations. This paper presents a bibliometric analysis of the literature on SPLs from 1993 to 2012, identifying the most influential publications, the most-researched topics, and how the interest in these topics has evolved along the period.

**Heradio et al.  
2016**



**Strategies for Consistency Checking on Software Product Lines: A Mapping Study**

Alcimar Rodrigues Santos<sup>a</sup>  
Rauel de Oliveira<sup>a</sup>  
Raphael Pereira de Oliveira<sup>a</sup>  
Eduardo Santana de Almeida<sup>a</sup>  
Fábio Henrique Project Center for  
Software and Systems Engineering  
Federal University of Bahia  
Salvador, Brazil  
[alciar@dei.ufba.br](mailto:alciar@dei.ufba.br)

Eduardo Santana de Almeida<sup>a</sup>  
Raphael Pereira de Oliveira<sup>a</sup>  
Alcimar Rodrigues Santos<sup>a</sup>  
Fábio Henrique Project Center for  
Software and Systems Engineering  
Federal University of Bahia  
Salvador, Brazil  
[esantana@dei.ufba.br](mailto:esantana@dei.ufba.br)

**ABSTRACT**

Context. Software Product Line (SPL) has become one of the most effective ways to produce the software with the SPL paradigm, it becomes necessary to manage these variations. The literature on SPLs has been growing and diverging among the SPL articles. Thus, a number of initiatives address the management of such configurations. Objective. This paper analyzes the literature on product lines from 1993 to 2012, identifying the most influential publications, the most-researched topics, and how the interest in these topics has evolved along the period.

**Santos et al.  
2015**

**Abstract** Context. Variability is a, the ability of software systems or artifacts to be adjusted for different contexts because a key property of many systems. Objective. We analyze existing research on variability in software systems. We investigate variability handling in software systems by means of a systematic literature review. A manual search covered 12 peer-reviewed journals and 18 peer-reviewed conferences, resulting in 76,432 papers searched and 178 papers considered for analysis. To compare reliability and to increase reproducibility, we complemented the manual search with a search on Google Scholar. Results. We found that the majority of the papers focused on the reuse of variability. Variability is studied in all software engineering phases, but testing is underrepresented. Data to evaluate the applicability of current approaches are not sufficient, reuse designs are vaguely described. Conclusions. To conclude our findings we provide a discussion on variability handling, reuse, and reuse designs. Finally, we discuss the challenges of applying variability handling and propagation of variability in software systems, specifying future directions or timely targeted research themes in the software engineering field.

**Keywords** Variability, systematic review, software engineering

**Galster et al.  
2013**

# (Software) Product Lines

Author & Venue	Title
Pol'la et al. (SoSyM'21)	Analysis of variability models: a systematic literature review
Yousaf et al. (JSEA'19)	Investigation of Tools, Techniques and Languages for Model Driven Software Product Lines (SPL): A Systematic Literature Review
Lopez-Herrejon et al. (Journal of Software: Evolution and Process'18)	A systematic mapping study of information visualization for software product line engineering
Lopez-Herrejon et al. (VIS-SOFT'16)	Visualization for Software Product Lines: A Systematic Mapping Study
Heradio et al. (INST'16)	A bibliometric analysis of 20 years of research on software product lines
Santos et al. (EASE'15)	Strategies for consistency checking on software product lines: A mapping study
Galster et al. (TSE'13)	Variability in Software Systems—A Systematic Literature Review
Holl et al. (IST'12)	A systematic review and an expert survey on capabilities supporting multi product lines
Schaefer et al. (STTT'12)	Software Diversity: State of the Art and Perspectives

# (Software) Product Lines

Author & Venue	Title
Chen and Babar ('11)	A systematic review of evaluation of variability management approaches in software product lines
Istoan et al. ('11)	A Metamodel-based Classification of Variability Modeling Approaches
Rabiser et al. ('10)	Requirements for product derivation support: Results from a systematic literature review and an expert survey
Chen et al. ('09)	Variability management in software product lines: a systematic review
Chen et al. ('09)	A status report on the evaluation of variability management approaches
Chen and Ali Babar ('09)	A survey of scalability aspects of variability modeling approaches
Günter and Kühn (XPS'99)	Knowledge-Based Configuration: Survey and Future Directions
Sabin and Weigel (IS'98)	Product Configuration Frameworks: A Survey

# **(Software) Product Lines** Tertiary Studies

DOI 10.1109/ICSE-EASE55555.2021.9506182

2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)

# A Systematic Study as Foundation for a Variability Modeling Body of Knowledge

Koen Finkenzeller

LIT Software Lab

JKU Linz, Austria

koen.finkenzeller@jku.at

Günther Meierhofer

Software Lab

TU Wien, Austria

klaus.meierhofer@tuw.ac.at

Rudi Rabitsch

CDL-RICS, LIT CPS Lab

JKU Linz, Austria

rudi.rabitsch@jku.at

Stefan Häfner

Information Systems Eng.

TU Wien and CSDP, Austria

stefan.hafner@tuw.ac.at

Abstract—The software reuse field has been growing, and researchers have used variability models to explicitly represent the reuse mechanism. This study aims to provide a foundation for a variability modeling body of knowledge. For over three decades, including 20 literature reviews, 100+ research articles, and 10+ books, we have collected 100+ variability models and compared them. We found that reuse models can be categorized into reuse mechanisms, which are reuse models based on reuse mechanisms, and reuse models, which are reuse models based on reuse models and searching for relevant studies for specific research questions. In addition, we found reuse models could be categorized into reuse models based on reuse mechanisms and reuse models based on reuse models.

Despite a multitude of studies, the field of variability modeling remains insufficiently organized, and it is hard to compare the results of systematic studies. Much effort went into creating a body of knowledge, but the results of the individual types of analyzed papers but differ in the investigated research questions and the methods used to answer them. Nevertheless, the order of the publications they found is often unclear, as a list of papers. Review of paper data sets would speed up and improve the quality of literature studies.

# Raatikanen et al. 2019 Tertiary Study

## 1. Introduction

Software Product Lines (SPLs) aim to develop software products through systematic reuse to address particular market segments, instances, or customers [1]. Engineers use variability models to support reuse by defining the variability of the parts of the system [2], [3], [4]. In the last three decades, a plethora of variability modeling approaches has been proposed [5]. The reuse model approach is the most prominent one [5], followed by decision modeling [4] and other approaches [6]. Further, industry and the open-source community have developed their own variability modeling approaches [6], [7].

In recent years, several works [8]–[17] have analyzed, compared, and contrasted reuse models and their characteristics and their limitations. In addition, SLRs [8] and SMEs [9] on variability modeling have been published (e.g., [5], [10], [11], [12], [13], [14], [15], [16], [17]). These studies are secondary studies. These secondary studies started from scratch by searching and analyzing primary studies (i.e., variability modeling approaches). To find evidence for specific research questions,

selection process and improve the study result quality by providing a body of knowledge blocks of the publications.

We aim at supporting researchers conducting SLRs and SMEs in the variability modeling research area by providing a body of knowledge blocks of the publications. In this paper, we first describe the tertiary SME [8] process we conducted to build a body of knowledge blocks of the publications. We then discuss how to work with key publications from variability modeling, including meta-information and reuse for compilation. The provided knowledge blocks are intended to support reuse models and increase the quality of conducting an SLRs or an SMEs. Our goal is that the curated data set provides a “living” body of knowledge blocks that can be updated and improved, particularly through its use, evaluation, and update requests by the community. Therefore, we present a feedback loop, using GitHub to facilitate the update requests. Finally, we discuss how to publish the data set to the public. As a result, it will be easier for reuse models to reuse the knowledge blocks.

<https://github.com/SECPU/TKMRE>

2020 © The Authors. Environment and Software Engineering and Applications. All rights reserved.

978-1-60558-270-8/20/\$31.00 ©2020 IEEE

DOI 10.1109/ICSE-EASE55555.2020.9210012

Author(s) licensed under a license granted to IEEE Xplore Literature Repository. Downloaded on March 24, 2021 at 14:02:40 UTC from IEEE Xplore. Restrictions apply.

# (Software) Product Lines Tertiary Studies

ACCEPTED MANUSCRIPT

## Trends in Software Reuse Research: A Tertiary Study

José L. Barros-Justo<sup>1</sup>, Fabiane B.V. Benini<sup>2</sup>, Santiago Matalonga<sup>3</sup>

<sup>1</sup>School of Informatics (EII), Universidad de Vigo, Ourense 32004, Spain

<sup>2</sup>Universidade Federal de Santa Catarina (UFSC), Florianópolis, Santa Catarina, Brazil

<sup>3</sup>School of Computing, Engineering and Physical Sciences, University of the West of Scotland, Paisley PA1 2BE, United Kingdom

### Abstract:

Context: The reuse of software has been a research topic for more than 50 years. Throughout that time, many approaches, models, and reuse techniques have reached maturity. However, it is not yet a standard practice and remains interesting to be further investigated. The latest reuse research results date back to 2008 and we think that this should be updated.

Objective: To identify the current trends in software reuse research.

Method: A tertiary study based on systematic secondary studies published up to July 2018.

Barros-Justo et al.  
2019  
Tertiary Study  
SPLs & software reuse

### Keywords:

Software reuse; trends in software reuse; systematic literature review; tertiary study.

### 1. Introduction

The term software reuse was born 50 years ago, at the first International Conference on Software Engineering (ICSE) in 1968 [1]. Since then, researchers and practitioners have looked into software reuse for increases in productivity and cost savings. The increasing number of publications along the years show that software reuse continues to be a topic of interest in the software engineering community.

Throughout these years, several definitions have been proposed for software reuse. For the purposes of this paper, we will use the definition provided by the IEEE Standard for Information Technology-System and Software Life Cycle Processes-Reuse Processes [2].

"Software reuse entails capitalizing on existing software and systems to create new products."

## Systematic Studies in Software Product Lines: A Tertiary Study

Mari�uthu C  
Department of Computer and Engineering  
National Institute of Technology Karnataka  
Mangalore, Karnataka 575025, India  
csmari@nitk.ac.in

K. Chandrasekaran  
Department of Computer and Engineering  
National Institute of Technology Karnataka  
Mangalore, Karnataka 575025, India  
kchandra@nitk.org

### ABSTRACT

Software product lines are widely used in the software industry to increase the reusability and its decrease maintenance cost. On the other hand, systematic reviews are mainly used to identify the gaps in the research field and to provide the overview of the research field and practitioners guideline. Researchers have conducted many systematic studies on the different aspects of SPLs. In the last 10 years, however, till now there is no tertiary study conducted on systematic studies in the area of SPLs. This study aims to identify and to coordinate a systematic mapping of existing systematic studies to support the overview of the findings for

the reuse of SPLs. SPLs can be defined as "a set of software-intensive systems sharing a common, managed set of assets to support simultaneous development in a cost-effective way" [7]. The major reason for using Software product line is it increases reuse, reduces development cost, and decreases time-to-market [1]. Software product line engineering (SPLs) [4, 1, 2] has got significant attention from the researchers and practitioners as it proves to be an efficient way to develop SPLs. Paul Clements claims that large range of software-intensive systems can

Marimuthu and Chandrasekaran  
2017  
Tertiary Study

### CCS CONCEPTS

• General and reference → Surveys and overviews;

### KEYWORDS

software product line; tertiary study; systematic review

### ACM Reference format:

Marimuthu C, Chandrasekaran. 2017. Systematic Studies in Software Product Lines: A Tertiary Study. In Proceedings of the 2017 ACM SIGART International Conference on Software Engineering Advances (SIGART '17). ACM, New York, NY, USA, 1–10. DOI: 10.1145/309000.3090016.

This document is made available under the terms of the Creative Commons Attribution Non-Commercial-ShareAlike 4.0 International license (<http://creativecommons.org/licenses/by-nc-sa/4.0/>). The full-text must not be sold in whole or in part, without the express permission of the copyright holders. The full-text must not be changed in any way without the express permission of the copyright holders. The full-text must not be sold in whole or in part, without the express permission of the copyright holders. The full-text must not be changed in any way without the express permission of the copyright holders.

For reuse, reuse studies can be interpreted with practical experience and human values in the decision-making process regarding the development and maintenance of software. The reuse studies can be used as a reference material as secondary studies or systematic studies as it provides the context and the evidence from the primary studies on the particular reuse studies. The purpose of systematic studies used in the software engineering domain are Systematic studies in the reuse of existing software, reuse of existing studies [18, 22, 23], etc. As stated in [18, 23], the systematic studies can help the researchers for further research and can also help the practitioners for the application of particular method and tools in practice.

Recently, researchers have conducted several systematic studies for different aspects of SPLs including software product lines testing [9, 11, 21], quality attributes and metrics [12, 13, 14, 15, 16, 17, 18, 19], reuse of modeling languages for SPL [20], domain analysis solutions for SPLs [12], and variability management in SPL [3, 4]. One of the main goals of this study is to identify the gaps from the existing systematic studies. This study is referred to as

# (Software) Product Lines Tertiary Studies

Author & Venue	Title
Feichtinger et al. (SEAA'21)	A Systematic Study as Foundation for a Variability Modeling Body of Knowledge
Raatikainen et al. (JSS'19)	Software product lines and variability modeling: A tertiary study
Barros-Justo et al. (CSAI'19)	Trends in software reuse research: A tertiary study
Marimuthu and Chandrasekaran (SPLC'17)	Systematic Studies in Software Product Lines: A Tertiary Study

# (Software) Product Lines Industrial Applications

 20 years of Industrial Experience at SPLC: A Systematic Mapping Study

Máider Anzana  
University of the Basque Country  
Euskal Herriko Unibertsitatea  
San Sebastián, Spain  
mainer.azana@ub.edu

Leticia Montalvillo  
Barclay Research Center  
Universidad del País Vasco  
San Sebastián, Spain  
leticia.montalvillo@deusto.es

Oscar Diaz  
University of the Basque Country  
Euskal Herriko Unibertsitatea  
San Sebastián, Spain  
oscar.diaz@ub.edu

**ABSTRACT**  
Software Product Lines (SPLs) have been around since the late 1970s, and they have provided us with a way to deal with product reuse. The use of configurations around the globe can pay back investment in the development of reusable components, however, how to manage and reuse other product line components, which would help them in the SPL journey? This work intends to analyze the application of variability tools in the last 20 years. We focused from 2000 up to 2020, and we found 103 articles. We analyzed these 103 articles to find the main trends in the use of variability tools.

**Azanza et al.**  
**2021**  
**Industrial studies**  
**Only publications from SPLC**

© Software and its engineering -- Software product lines : General and reference -- Survey and overview.

ACM Reference Format:  
Anzana, M., Montalvillo, L., and Diaz, O.: 20 years of the industrial experience at SPLC: A Systematic Mapping Study. In: *2020 ACM/IEEE International Conference on Software Engineering: Software Product Lines (ICSE-SPL) (2020)*, Ljubljana, Slovenia. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3367003.3370589>

This is the peer reviewed version of the following article: Anzana, M., Montalvillo, L., and Diaz, O.: 20 years of the industrial experience at SPLC: A Systematic Mapping Study. In: *2020 ACM/IEEE International Conference on Software Engineering: Software Product Lines (ICSE-SPL) (2020)*, Ljubljana, Slovenia. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3367003.3370589>. It has been deposited in accordance with the publisher terms. For further information, please refer to the publisher's Terms and conditions.

bioRxiv preprint doi: https://doi.org/10.1101/343406; this version posted April 14, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under a [aCC-BY-ND 4.0 International license](https://creativecommons.org/licenses/by-nd/4.0/).

Journal of Universal Computer Science, vol. 19(2020), 1722-1732  
submitted 26/2/19; accepted 27/6/20; appeared 24/8/20; CTS: 83; DOI: 10.3217/jucs-019-0978-6

Have Variability Tools Fulfilled the Needs of the Software Industry?

Ana Paula Allian  
State University of Maringá and  
University of São Paulo  
Maringá, Brazil  
ana.allian@ufop.br

Edson Oliveira Jr  
State University of Maringá  
Maringá, Brazil  
edson.oliveira@ufop.br

**1 INTRODUCTION**

Allian et al.  
2020  
Talk about tool support  
Qualities that tools need for use  
in the industry

Empirical Software Engineering (2020) 25:1755–1787  
<https://doi.org/10.1007/s10664-019-03787-6>

 The state of adoption and the challenges of systematic variability management in industry

Thorsten Berger<sup>1</sup> · Jan-Philipp Steghöfer<sup>1</sup> · Tewfik Ziadi<sup>2</sup> · Jacques Robin<sup>3</sup> · Jaber Martínez<sup>4</sup>

<sup>1</sup> Published online: 4 April 2020  
© The Author(s) 2020

**Abstract**  
Handling large-scale software variability is still a challenge for many organizations. After Berger et al.  
2020  
Contains only a "lightweight literature study"  
Challenges of SPLs for industry

**Keywords** Variability management · Software product lines · Multiple, case study · Challenges

**1 Introduction**

Companies often need to engineer a portfolio of software variants instead of one-size-fits-all solutions. Creating variants allows tailoring systems towards varying stakeholder

<sup>1</sup>Julian Martínez contributed to this work during his PostDoc position at the Sorbonne University, France  
Communicated by: Tao Yu  
<sup>2</sup>Thorsten Berger  
thorsten.berger@chalmers.se

Extended author information available on the last page of the article.

# (Software) Product Lines Industrial Applications

Author & Venue	Title
Azanza et al. (SPLC'21)	20 years of Industrial Experience at SPLC: a Systematic Mapping Study
Allian et al. (J.UCS.'20)	Have Variability Tools Fulfilled the Needs of the Software Industry?
Berger et al. (ESE'20)	The state of adoption and the challenges of systematic variability management in industry





# (Software) Product Lines Tool Support

Author & Venue	Title
Horcas et al. (SoSyM'22)	Empirical analysis of the tool support for software product lines
Horcas et al. (SPLC'19)	Software Product Line Engineering: A Practical Experience
Bashroush et al. (CSUR'17)	CASE Tool Support for Variability Management in Software Product Lines
Pereira et al. (ICSR'15)	A Systematic Literature Review of Software Product Line Management Tools
Meinicke et al. (SPLat'14)	An Overview on Analysis Tools for Software Product Lines
Lisboa et al. (INST'10)	A Systematic Review of Domain Analysis Tools

# (Software) Product Lines Evolution

Information and Software Technology 134 (2022) 100046

Contents lists available at ScienceDirect

Information and Software Technology journal homepage: [www.elsevier.com/locate/infsof](http://www.elsevier.com/locate/infsof)

 ELSEVIER

Visualizations for the evolution of Variant-Rich Systems: A systematic mapping study

Rafel Medeiros<sup>a,\*</sup>, Júlio Martínez<sup>a</sup>, Oscar Díaz<sup>b</sup>, Jean-Rémy Falleri<sup>c</sup>

<sup>a</sup> Institute for Software Research and Technology (ISyR), University of Valencia, Spain

<sup>b</sup> Faculty of Computer Science, Dalhousie University, Halifax, NS, Canada

<sup>c</sup> Institute for Software and Systems Engineering, Clausthal University of Technology, Germany



**ARTICLE IN PRESS** **ABSTRACT**

**Medeiros et al.**  
2022  
Variant-Rich Systems, including  
SPLs  
41 visualisation techniques

understanding involves facing large amounts of data from heterogeneous data sources including version control systems, bug tracking systems, mailing newsletters, discussion lists, as well as the implementation of specific tools and environments. This article proposes a methodology to build diagrams and models that allow grasping the past and present, while predicting the future of variant-rich systems. The article is organized into four main sections: introduction, related work, research questions, and conclusion and evaluation [2]. In this article, our contributions can be found along two main axes: the purpose of the evolution (i.e., evolution in functionality, e.g., adding or removing features or requirements) and the nature, e.g., refactoring) and the subject of evolution (i.e., variant-rich systems).

\* Corresponding author.  
E-mail addresses: [rafel.medeiros@uv.es](mailto:rafel.medeiros@uv.es) (R. Medeiros), [julio.martinez@uv.es](mailto:julio.martinez@uv.es) (J. Martínez), [oscar.diaz@cs.dal.ca](mailto:oscar.diaz@cs.dal.ca) (O. Díaz), [jrfalleri@isyr.ub.edu](mailto:jrfalleri@isyr.ub.edu) (J.-R. Falleri).

<https://doi.org/10.1016/j.infsof.2022.100046>  
Received 14 January 2022; Received 18 September 2022; Accepted 22 September 2022  
Available online 27 October 2022  
© 2022 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (<https://creativecommons.org/licenses/by/4.0/>).

 MDPI

**Systematic Review**

**Managed Evolution of Automotive Software Product Line Architectures: A Systematic Literature Study**

Christoph Knieke<sup>a</sup>, Andreas Rauch<sup>a</sup>, Mirco Schindler<sup>a</sup>, Arthur Stürmer<sup>a</sup> and Martin Vogel<sup>a</sup>

<sup>a</sup> Institute for Software and Systems Engineering, Clausthal University of Technology, 38678 Clausthal-Zellerfeld, Germany; [christoph.knieke@tu-clausthal.de](mailto:christoph.knieke@tu-clausthal.de); [andreas.rauch@tu-clausthal.de](mailto:andreas.rauch@tu-clausthal.de); [mirco.schindler@tu-clausthal.de](mailto:mirco.schindler@tu-clausthal.de); [arthur.stuermer@tu-clausthal.de](mailto:arthur.stuermer@tu-clausthal.de); [martin.vogel@tu-clausthal.de](mailto:martin.vogel@tu-clausthal.de)

**Abstract** The rapidly growing number of software-based features in the automotive domain as well as the special requirements in this domain ask for dedicated engineering approaches, models, and processes. Nowadays, software development in the automotive sector is generally developed as product lines. The managed evolution of these product lines is challenging due to the individual modularity of the software in different vehicle variants. In addition, reuse also plays an important

**Knieke et al.**  
2022  
Evolution with focus on automotive SPLs

**1. Introduction**

The automotive has become the most technically complex consumer product [1]. The field of increasing customer requirements and strict legal requirements with regard to the reduction of fuel consumption and pollutant emissions, as well as the higher demands on safety and new driver assistance systems resulted in a steady increase in the deployment of onboard electronics systems and software. Software-intensive systems and functions account for up to 40% of the total vehicle weight [2]. Electronics and software are responsible for up to 80% of the innovation [3]. Electronics and software account for up to 40% of the production costs of a car [4].

The requirements of the automotive electronics differ significantly from other areas of consumer electronics (cf., e.g., [5]). For automotive functions, hard real-time requirements exist while the storage and computational power on ECUs have to be kept as small as possible. Systems in an automobile must be reliable, safe, and secure in all situations as well as be able to withstand extreme environmental conditions. The manufacturer, i.e., the OEM has the duty to offer service and spare parts for at least 15 years after the purchase of a vehicle due to the long product life cycles. In contrast, software is changed at comparatively short intervals. During the production period and even during the development phase, individual vehicle variants are produced. The combination of high software complexity, high innovation and long life cycles, a huge number of versions and configurations exist, which in turn also makes maintenance very difficult.

**Electronics 2022, 11, 1860; <https://doi.org/10.3390/electronics11121860>** <https://www.mdpi.com/journal/electronics>

# (Software) Product Lines Evolution

# Software Product Line Evolution: a Systematic Literature Review

The Journal of Systems and Software 102 (2014) 100–103

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: [www.elsevier.com/locate/jss](http://www.elsevier.com/locate/jss)

ELSEVIER

## Requirement-driven evolution in software product lines: A systematic mapping study

Leticia Montalvillo, Oscar Diaz

*University of the Basque Country (UPV/EHU), IKERBIDE Research Group – Periodical of Application – San Sebastián, Spain*

### ARTICLE INFO

**Article history:**  
Received 1 December 2013  
Accepted 10 April 2014

### ABSTRACT

Software Product Lines (SPLs) aim to support the development of a whole family of related products through systematic reuse of shared assets. As SPLs exhibit a long life-span, it is more promising than the single-project for the purpose of this work, evolution refers to the adaptation of the SPL to its environment.

# Montalvillo and Diaz 2016 107 resulting publications

Software Product Lines (SPLs) aim to support the development of a whole family of related products through systematic reuse of shared assets (Clemons and Hwang, 2005). SPL engineering has been widely studied in the last years (Bass et al., 2003; Clemons and Hwang, 2005; Clemons et al., 2006; Gómez, 2011; van der Linde et al., 2007; Weiss, 2008). As the SPL domain matures, evolution becomes one key player (Hwang, 2005). Evolution is the process by which a long line begins recognized as being maintained with diverse changes (Hwang, 2005). The term “requirement-driven evolution in software product lines” refers to the adaptation of the SPL as a result of changing SPL requirements. From this perspective, evolution is triggered by requirement changes, and not as much by bug fixing or refactoring.

Initially, in their infancy, SPLs strive to the market. At adulthood, SPLs might face less defects, but their wider customer base have more specific needs. As a result, the SPLs need to adapt to these needs. Therefore, SPL long-life-span requires evolution a long priority, so the firm becomes competitive. The main challenges of evolution in SPLs are: (1) evolution of application-specific challenges include (1) reorganization of development and reuse (engineering) (RE) activities, (2) reuse of existing assets, (3) reuse of code, (4) reuse of requirements, (5) reuse of architecture, and (6), high number of interactions between requirements and reuse assets (Bass et al., 2003; Weiss, 2008); (2) evolution of organizational challenges include (1) the first walk through the evolution of the SPL (Bass et al., 2003; Weiss, 2008); (3) evolution of the market, which includes the market evolution and the market demand evolution (Bass et al., 2003; Weiss, 2008); (4) evolution of the firm, which includes the firm evolution and the firm demand evolution (Bass et al., 2003; Weiss, 2008); (5) evolution of the technology, which includes the technology evolution and the technology demand evolution (Bass et al., 2003; Weiss, 2008).

Two exceptions are (Barroso and Pinto, 2014; Mignogna, 2005). The former addressed the evolution of the SPL as a result of market demand, and direct the evolution of SPL assets (Hwang, 2005). Barroso and Pinto (2014) presented the most recent research

Corresponding author.

E-mail addresses: [lecia.montalvillo@ehu.es](mailto:lecia.montalvillo@ehu.es), [l.montalvillo.ehu.es](mailto:l.montalvillo.ehu.es).

<sup>1</sup> Refer to <http://tinyurl.com/lnqjw4t> for a list of selected 107 references.

<http://www.sciencedirect.com/science/journal/08905401>

© 2014 Elsevier Inc. All rights reserved.

# (Software) Product Lines Evolution

Author & Venue	Title
Medeiros et al. (INST'22)	Visualizations for the evolution of Variant-Rich Systems: A systematic mapping study
Knieke et al. (Electronics'22)	Managed Evolution of Automotive Software Product Line Architectures: A Systematic Literature Study
Marques et al. (INST'19)	Software product line evolution: A systematic literature review
Montalvillo and Diaz (JSSO'16)	Requirement-driven evolution in software product lines: A systematic mapping study

# (Software) Product Lines Processes

# Searching for Common Ground: Existing Literature on Automotive Agile Software Product Lines

# (Software) Product Lines Processes

SOFTWARE – PRACTICE AND EXPERIENCE  
Softw. Pract. Exper. 2011; 41(10):987–1002  
Published online 1 May 2011 in Wiley Online Library (wileyonlinelibrary.com) DOI: 10.1002/spe.9778

## Agile software product lines: a systematic mapping study

Ivonnei Pratića da Silva<sup>1,3,4,5,†</sup>, Paulo Avelino da Mota Silveira Neto<sup>2,3</sup>,  
Pádraig O’Leary<sup>2,3</sup>, Eduardo Santana de Almeida<sup>2,3</sup>, and  
Silvio Romano de Lemos Meira<sup>2</sup>

<sup>1</sup>Information Center, Federal University of Paraná, Brazil; <sup>2</sup>PE, Brazil;  
<sup>3</sup>Computer Engineering Department, Federal University of Paraná, Brazil;  
<sup>4</sup>IMEC (Institute of Electrical Engineering), Brazil; <sup>5</sup>FEUP, Portugal

### SUMMARY

**Background:** Software product lines and Agile methods have been an effective solution for dealing with the growing complexity of software and the needs of modern organizations. They also share common goals, such as improving productivity, reducing time-to-market, decreasing development costs, and increasing quality. **Purpose:** The purpose of this study is to evaluate the potential of Agile and SPLs could provide further benefits and solve many of the most common causes surrounding

the use of Agile methods in product line environments.

da Silva et al.  
2011

Copyright © 2011 John Wiley & Sons, Ltd.

Received 10 August 2010; Revised 29 November 2010; Accepted 22 February 2011

**KEY WORDS:** systematic mapping study; Agile methods; software product lines; Agile principles

### 1. INTRODUCTION

Software development is a complex composition of processes, technologies, tools and people, often in domains with volatile requirements. Furthermore, with the growing demand for higher quality software products, the need for reuse has increased. In this context, the reuse of existing code and reuse of existing approaches has been raised as a possible solution. When the *Software Product Lines* (SPL) engineering provides a systematic approach to reuse common artifacts to facilitate the delivery of different products [1]. Agile methods focus on working code while reducing up-front design and planning activities. Agile methods are based on iterative development of features, feedback and principles [3]. Despite the differences in how both approaches perform software development, they propose to deliver common benefits, such as improving productivity, reducing time-to-market,

<sup>†</sup>Correspondence to: Ivonnei Pratića da Silva, Information Center, Federal University of Paraná, Brazil; Tel.: +55 41 3602-7000; E-mail: ivon@uel.br

Copyright © 2011 John Wiley & Sons, Ltd.

SOFTWARE - PRACTICE AND EXPERIENCE  
Softe. Pract. Exper. 2013; 41(9):1–30  
Published online 1 May 2011 in Wiley Online Library (wileyonlinelibrary.com) DOI: 10.1002/spe.2897

# (Software) Product Lines Processes

Author & Venue	Title
Klünder et al. (ICSSP'18)	Becoming agile while preserving software product lines: An agile transformation model for large companies
Hohl et al. (ICSSP'17)	Searching for common ground: Existing literature on automotive agile software product lines
da Silva et al. (SPE'11)	Agile software product lines: A systematic mapping study
Díaz et al. (SPE'11)	Agile product line engineering - a systematic literature review

## **(Software) Product Lines Applications of Product Line Techniques**

 <p><b>Information and Software Technology</b> 00 (2000) 00–00  <b>Contents lists available at ScienceDirect</b>  <b>Information and Software Technology</b>  <i>journal homepage: <a href="http://www.elsevier.com/locate/infsof">www.elsevier.com/locate/infsof</a></i></p>																																											
<p>A systematic mapping study of search-based software engineering for software product lines</p> <p>Roberto E. Lopez-Herrejon<sup>a</sup>, Lukas Linsmaier<sup>b</sup>, Alexander Egyed<sup>a</sup></p> <p><sup>a</sup>Institute for Software Systems Engineering, Johannes Kepler University Linz, Altenbergerstr. 69, 4040 Linz, Austria</p>																																											
<hr/> <p><b>ARTICLE INFO</b></p> <p><b>Article history:</b>  Received 10 August 2014  Revised 10 January 2015  Available online 16 January 2015</p> <hr/> <p><b>ABSTRACT</b></p> <p><b>Central Search-Based Software Engineering (SBSE) is an emerging discipline that focuses on the application of search-based methods and algorithms to support the development of Software Product Lines (SPLs). One feature of related software systems whose outcomes are distinguished by the use of</b></p>																																											
<h1>Lopez-Herrejon et al. 2015 Search-based SE for SPLs</h1>																																											
<hr/> <p>ing and mapping review to research on the design and creation of mapping types research programs.  © 2015 Elsevier B.V. All rights reserved.</p> <hr/>																																											
<p><b>Keywords:</b></p>																																											
<table border="0" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10px;">1.</td> <td>Introduction</td> <td style="width: 10px;">34</td> </tr> <tr> <td>2.</td> <td>Systematic mapping study</td> <td>35</td> </tr> <tr> <td>2.1.</td> <td>Definition of research questions</td> <td>35</td> </tr> <tr> <td>2.2.</td> <td>Search strategy</td> <td>35</td> </tr> <tr> <td>2.3.</td> <td>Screening of papers for inclusion and exclusion</td> <td>35</td> </tr> <tr> <td>2.4.</td> <td>Data extraction and synthesis</td> <td>36</td> </tr> <tr> <td>2.4.1.</td> <td>SPL life-cycle stage classification</td> <td>36</td> </tr> <tr> <td>2.4.2.</td> <td>Type of search engines classification</td> <td>37</td> </tr> <tr> <td>2.4.3.</td> <td>Type of statistical analysis classification</td> <td>37</td> </tr> <tr> <td>2.4.4.</td> <td>Type of case studies classification</td> <td>37</td> </tr> <tr> <td>2.4.5.</td> <td>Type of system requirements classification</td> <td>38</td> </tr> <tr> <td>2.5.</td> <td>Data reduction and mapping study</td> <td>38</td> </tr> <tr> <td>2.5.1.</td> <td>Results R1—SPL life-cycle stages</td> <td>38</td> </tr> <tr> <td>2.5.2.</td> <td>Results R2—SPL life-cycle stages</td> <td>38</td> </tr> </table>		1.	Introduction	34	2.	Systematic mapping study	35	2.1.	Definition of research questions	35	2.2.	Search strategy	35	2.3.	Screening of papers for inclusion and exclusion	35	2.4.	Data extraction and synthesis	36	2.4.1.	SPL life-cycle stage classification	36	2.4.2.	Type of search engines classification	37	2.4.3.	Type of statistical analysis classification	37	2.4.4.	Type of case studies classification	37	2.4.5.	Type of system requirements classification	38	2.5.	Data reduction and mapping study	38	2.5.1.	Results R1—SPL life-cycle stages	38	2.5.2.	Results R2—SPL life-cycle stages	38
1.	Introduction	34																																									
2.	Systematic mapping study	35																																									
2.1.	Definition of research questions	35																																									
2.2.	Search strategy	35																																									
2.3.	Screening of papers for inclusion and exclusion	35																																									
2.4.	Data extraction and synthesis	36																																									
2.4.1.	SPL life-cycle stage classification	36																																									
2.4.2.	Type of search engines classification	37																																									
2.4.3.	Type of statistical analysis classification	37																																									
2.4.4.	Type of case studies classification	37																																									
2.4.5.	Type of system requirements classification	38																																									
2.5.	Data reduction and mapping study	38																																									
2.5.1.	Results R1—SPL life-cycle stages	38																																									
2.5.2.	Results R2—SPL life-cycle stages	38																																									
<hr/> <p><small><sup>a</sup> Corresponding author. Tel.: +43 732 2480 2400.  E-mail addresses: <a href="mailto:roberto.lopez@jku.at">roberto.lopez@jku.at</a> (R.E. Lopez-Herrejon), <a href="mailto:linsmaier@jku.at">linsmaier@jku.at</a> (L. Linsmaier), <a href="mailto:alexander.egyed@jku.at">alexander.egyed@jku.at</a> (A. Egyed).</small></p>																																											
<p><small><a href="http://dx.doi.org/10.1016/j.infsof.2015.01.008">http://dx.doi.org/10.1016/j.infsof.2015.01.008</a>  © 2015 Elsevier B.V. All rights reserved.</small></p>																																											

2014 Eighth Brazilian Symposium on Software Components, Architectures and Reuse

# (Software) Product Lines Applications of Product Line Techniques

## A Study on Service Identification Methods for Software Product Lines

Tasso Vale  
Federal University of Paraná  
tval@cpes.ufpr.br

Gustavo Brilzourt  
Figueiredo  
Federal University of Paraná  
gustavo@cc.ufrj.br

### ABSTRACT

The combination of service-oriented and software product line engineering, called Service-Oriented Product Line Engineering (SOPLE), has been an active research area in the last years. There are several proposals for service identification in the last years, and there are no addressed to mobile environments. This paper presents two approaches for service identification in mobile environments.

Eduardo Santana de Almeida  
Federal University of Paraná  
ealmeida@ufpr.br

Silvio Romero de Lemos  
Meira  
Federal University of Paraná  
srlm@cpes.ufpr.br

combination is now called Service-Oriented Product Line Engineering (SOPLE) [1].  
The main idea of SOPLE is to achieve dynamic reusability from services in most of the SPL approaches [2] all variations are instantiated before a product is delivered to the customer, making it difficult to make any changes in the system.

Vale et al.  
2012

## Service Identification methods for SPLs

Service Identification, Software Product Lines, Service-Oriented Computing, Service-Oriented Product Lines

### 1. INTRODUCTION

The combination of Software Product Lines (SPLs) and Service-Oriented Computing (SOC) has been an active research area in the last years. There are several issues in each other, and there are no addressed to mobile environments.

Positioning a mobile device or part of or all of this work for personal or commercial is problematic for providers that expect are not made or not made the full solution on the first page. Configuration is one of the most work needed. To enjoy efficient and effective positioning with configuration, it is necessary to provide specific positioning and a lot of effort. Copyright 2012 ACM 978-1-4503-1380-3/12/08... \$10.00.

feature for mobile services in a more integrated manner. A common strategy in the business process driven approach, block approach identifies services by decomposing the basic functions into smaller ones. This approach can identify a eight-ground functionality which can be reusable [1].

The layering approach is another way to identify the features of the system in place. This approach enables architects to identify the relevance of the source code and reuse it in different layers. This approach is also called a "service-in-the-middle" (also called hybrid) approach combining bottom-up and top-down strategies.

In such a scenario, mobile devices need specific and appropriate service identification methods in several contexts. However, an analysis of the literature shows that there is no work that can be found on helping SOPLE stakeholders to choose the most suitable method according to their needs. Thus, we are interested in identifying the most suitable method for the mobile context, since most of the proposals are not focused on this context. We address it proposing an in-depth comparison

2009 Sixth International Conference on Information Technology: New Generation

## A Systematic Review of Software Product Lines Applied to Mobile Middleware

Yuri Morais, Thais Butty, Gládson Elias

COMPSE – Component Oriented Service Engineering Group  
Informatics Department, Federal University of Paraná, Brazil

Abstract  
Mobile computing imposes several restrictions to software development due to diversity in network connectivity, platform capability and resource availability. In this context, Software Product Lines (SPLs) offer some advantages to reuse code and reduce costs from application development, resolving common challenges of distributed systems. However, middleware built to provide a wide set of features to meet the needs of multiple problem domains. As a result, extra features not used by installed applications contribute to unnecessary code size and configuration complexity [2]. Due to that, it is important to identify the most suitable middleware, which can be adapted to exactly suit the needs of installed applications and the capacity of the underlying hardware and software platforms.

Morais et al.

## Applying SPLs to mobile middleware

According to Kishimoto et al. [3], a systematic literature review is a means of identifying all available research that is relevant to a particular research question or topic area. A systematic review is a formalized search with an explicit protocol in order to generate evidence. Systematic reviews must be undertaken in accordance with a predefined search strategy, which must allow the comparison of results across studies. This will assist the reader to repeat the research using the same criteria. Thus, the purpose of the systematic review is to provide a summary of the state-of-the-art [3].

The remainder of the paper is structured as follows. Section 2 presents the background of the review planning and conduction. In Section 3 we present an overview of the most relevant identified proposals. Section 4 presents the results and Section 5 presents concluding remarks and future directions.

Rahel Arens

4ECTS – FOSD 2023 – 1. (Software) Product Lines

25

# (Software) Product Lines Applications of Product Line Techniques

Author & Venue	Title
Lopez-Herrejon et al. ('15)	A systematic mapping study of search-based software engineering for software product lines
Queiroz and Braga ('14)	Development of critical embedded systems using model-driven and product lines techniques: A systematic review
Santos Rocha and Fantinato ('13)	The use of software product lines for business process management: A systematic literature review
Vale et al. ('12)	A study on service identification methods for software product lines
Morais et al. ('09)	A systematic review of software product lines applied to mobile middleware

**(Software) Product Lines** Applications of Product Line Techniques - Service-Oriented Product Lines

International Journal of Basic Sciences & Applied Research, Vol. 4 (1), 60-75, 2019  
Available online at [www.ijbsar.org](http://www.ijbsar.org)  
ISSN 2147-3749 G2019

ACM SIGSOFT Software Engineering Notes Page 1

March 2014 Volume 39 Number 2

## Service-Oriented Product Lines: A Systematic Mapping Study

Daniela Castelluccia

University of Bari  
Department of Computer Science  
70131 Bari, Italy  
+39 080 5442127

daniela.castelluccia@uniba.it

Nicola Boffoli

University of Bari  
Department of Computer Science  
70131 Bari, Italy  
+39 080 5442370  
nicola.boffoli@uniba.it

### ABSTRACT

Service-oriented line engineering and service-oriented architectures both enable organizations to capture an reuse of existing software assets and capabilities and compute services on demand. In this paper we present a systematic mapping study to map both approaches to accommodate variations of software products and services. We also discuss the reuse of computing services according a new classification. Therefore, variability management in service-oriented Product Lines (SoPL) is one of the main challenges of the future. Finally, we present emerging evidence-based results from the research community.

Item-to-item modular and maintenance settings. Therefore, integrating planning and variability management are the key factors that usually allow the degree to which a software product line is reusable.

Growingly, variability management assets handling variability in software artifacts (such as requirements, code, and architecture) is becoming a key factor to support the product line, in order to manage variability among different configurations of the same product line [1, 2, 3, 4, 5, 6, 7]. Variability management involves extremely complex and challenging tasks, which need to be supported (or automated) by appropriate tools and frameworks. Every one of a product line's

# Castelluccia and Boffoli 2009

selected efficient solutions to variability management in SoPL.

### Categories and Subject Descriptors

D.2.1.1. [Software engineering]: Software architecture: (Requirements engineering and requirements analysis); D.2.1.1. [Software engineering]: Requirements engineering: Requirements engineering

usually it is referred as product derivation.

3. Implementation: variability is also discussed at this step as each product line is derived from a common base product line according to different corporate constraints; usually it is referred as product configuration.

4. Evolution and Validation: reuse rules and constraints influence the process of building products, which should be composed according to specific business rules and suppressed variability. The reuse rules and constraints are used to validate and validate of the completeness and consistency of this configuration process as well as the reuse rules and constraints.

5. Evolution and Change: variability management arises after in maintenance by managing variability and architecture evolution, which should be done in a timely manner. This step is concerned with all above steps, since it is a product line's maintenance.

The aforementioned steps usually constitute the list of topics addressed in the most conferences and journals. However, considering the lack of a general framework, we have no effort to systematically focus on the issues concerning variability management in service-oriented product lines. Therefore, to understand the evolutionary path towards: Service-oriented Product Lines (SoPL), in the first place, to create a systematic mapping study. This paper is the first step of the research in SoPL, in order to summarize and highlight the emerging evidence-based results from the research community.

### 1. INTRODUCTION

Software product lines (SPLs) [1] and service-oriented architectures (SOAs) [2] are two different approaches to reuse existing software assets and capabilities and reuse the coding of new software. Based on a set of core assets, several products or services can be derived and adapted to a specific market segment or area, leading to product families

<http://doi.acm.org/10.1145/2579281.2579294>

DOI: 10.1145/2579281.2579294

**(Software) Product Lines** Applications of Product Line Techniques - Service-Oriented Product Lines

Information and Software Technology 53 (2011) 946–959

Contents lists available at ScienceDirect

Information and Software Technology

Journal homepage: [www.elsevier.com/locate/infsof](http://www.elsevier.com/locate/infsof)

REVIEW

## Combining service-orientation and software product line engineering— A systematic mapping study

Barzia Mohabbati<sup>a</sup>, Muhamed Audi<sup>a</sup>, Dragos Golevici<sup>a,b</sup>, Marek Matula<sup>a</sup>, Hossni A. Müller<sup>c</sup>

<sup>a</sup> School of Computer Science, University of Alberta, Edmonton, AB T6G 2E8, Canada  
<sup>b</sup> Faculty of Computer Science, Dalhousie University, Halifax, NS B3H 2W5, Canada  
<sup>c</sup> University of Western Ontario, 1155 London Road, London, ON N6A 3K7, Canada

**ARTICLE INFO**

Article history:

Received: 16 December 2010

Revised: 10 March 2011

Accepted: 24 May 2011

Available online:

24 June 2011

**ABSTRACT**

**Context:** Service-Oriented Computing (SOC) is a rapidly emerging paradigm for the design and development of distributed systems. Software Product Line Engineering (SPE) is another paradigm that has been adopted as a promising and successful software reuse development paradigm over the last decade and appears to be a promising paradigm for SOC.

**Purpose:** The purpose of this study is to identify and characterize the state-of-the-art SPE techniques used in the context of SOC and to evaluate their potential for reuse by identifying promising research areas.

**Design/methodology/approach:** This study uses a systematic mapping approach to identify promising research areas by integrating methods and techniques from these two areas.

**Originality/value:** This study is the first to identify and characterize the state-of-the-art SPE techniques used in the context of SOC and to evaluate their potential for reuse by identifying promising research areas.

# Mohabbati et al. 2012

and discusses its practical dimensions. Finally, it concludes by discussing software product lines.

**Conclusion:** The study identifies and characterizes the state-of-the-art SPE techniques used in the context of SOC and evaluates their potential for reuse by identifying promising research areas.

**License:** Copyright © 2011 Published by Elsevier Ltd. All rights reserved.

**Contents**

1. Introduction	1046
2. Systematic mapping and research method	1046
2.1. Review protocol	1047
2.2. Search strategy	1047
2.3. Data sources	1047
2.4. Data extraction	1047
2.5. Included and excluded criteria	1048
2.6. Data synthesis	1048
3. Characteristics of the included studies	1049
4. Results	1049
4.1. Demographic data	1049
4.2. Research focus	1049
4.2.1. Service-oriented modeling	1050
4.2.2. Service identification	1051
4.2.3. Service reuse	1052

<sup>a</sup> Corresponding author. Tel.: +1 780 492 7655.

<sup>b</sup> E-mail addresses: mohabbati@cs.ualberta.ca (B. Mohabbati), mohabbati@cs.ualberta.ca (M. Audi), drgolevici@cs.dal.ca (D. Golevici), matula@cs.ualberta.ca (M. Matula), hmueller@cs.ualberta.ca (H.A. Müller).

0893-9618/\$ - see front matter © 2011 Published by Elsevier Ltd. All rights reserved.

<http://dx.doi.org/10.1016/j.infsof.2011.05.006>

## (Software) Product Lines Applications of Product Line Techniques - Service-Oriented Product Lines

Author & Venue	Title
Khoshnevis et al. ('15)	A Systematic Literature Review of Variability Management in Service-Oriented Products and Product Lines
Castelluccia and Boffoli ('14)	Service-oriented product lines: A systematic mapping study
Mohabbati et al. ('13)	Combining service-orientation and software product line engineering: A systematic mapping study
Murugesupillai et al. ('11)	A preliminary mapping study of approaches bridging software product lines and service-oriented architectures

# (Software) Product Lines Applications of Product Line Techniques - Dynamic SPL



Article

## Variability Management in Dynamic Software Product Lines for Self-Adaptive Systems—A Systematic Mapping

Oscar Aguayo<sup>1</sup> and Daniel Sepúlveda<sup>2,3</sup>

Departamento de Ciencias de la Computación e Información, Centro de Políticas en Ingeniería de Software, Universidad de La Frontera, Temuco 1912128, Chile

<sup>2</sup> Comisión Chilena de Investigación Científica y Tecnológica

**Abstract.** Context: Dynamic software product lines (DSPLs) have considerably increased their adoption for variability management for self-adaptive systems. The most widely used models for managing the variability of DSPLs are the MAPLE-K control loop and context-aware feature models (CfMs). Aim: In this paper, we review and synthesize evidence of using variability constraint approaches, methodologies, and challenges for DSPLs. Method: We conducted a systematic mapping, including a search strategy, study selection, and synthesis of results. Results: The main results show that open-dynamic variability shows a presence in 57.1% of the selected papers, and on the other hand, closed-dynamic variability appears in 38.7%. The most commonly used methodology for managing a DSPL environment is based on proprietary architectures (60.7%), while the use of CfMs predominates. For open-dynamic variability approaches, the MAPLE-

Aguayo and Sepúlveda  
2022

Information for Authors

Self-Adaptive Systems—A Systematic Mapping  
11, 10240 | https://doi.org/10.3390/app120510240

Academic Editor: Vito Costantini

Received: 19 August 2022  
Accepted: 10 September 2022  
Published: 13 September 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Appl. Sci. 2022, 12, 10240 | https://doi.org/10.3390/app120510240

<https://www.mdpi.com/journal/applied>

2016–2018 Euroconic Conference on Software Engineering and Advanced Applications

## On the Dependability for Dynamic Software Product Lines A Comparative Systematic Mapping Study

Jana Díaz Alvar, Sardón Elizalde<sup>1</sup>  
USAL, Campus Grande, Spain  
[jana.diaz@usal.es](mailto:jana.diaz@usal.es)Andrea Henssler, Paola Lefebvre<sup>2</sup>  
UNIR, Florida, Italy  
[andrea.henssler@unir.it](mailto:andrea.henssler@unir.it), [pao.lefebvre@unir.it](mailto:pao.lefebvre@unir.it)Felipe Nunes Góis<sup>3</sup>  
UFSC, Joinville, Brazil  
[fngois@ufsc.br](mailto:fngois@ufsc.br)Germán Núñez Rodríguez<sup>3</sup>  
UFSC, Joinville, Brazil  
[gnrodriguez.ufsc.br](mailto:gnrodriguez.ufsc.br)Carla Mary Fischer Rubira<sup>3</sup>  
UFSC/CEAD, Campus Joinville, Brazil  
[cmrufira@ufsc.br](mailto:cmrufira@ufsc.br)

*Abstract.* Software Product Lines (SPLs) are techniques where

On the other hand, the software industry seeks processes

Eleuterio et al.  
2016

Eleuterio et al. [1] studied the dependability of DSPLs. They found that there was no evidence about dependability in DSPL context. We selected nine primary studies in this regard. We performed a comparative study of the results, analyzing their strengths and weaknesses, aiming for a better understanding of this research area.

**Keywords:** dynamic software product lines; dependency; adaptive systems; dependency systematic mapping study

### 1. INTRODUCTION

There is a growing need for systems to be able to self-adapt to changes in their environment. This is particularly true for DSPLs, which are systems that can produce different products with the ability to change the configuration at runtime by themselves [2]. Dynamic variability (dynamic variability management) is a discipline that studies how systems can be designed to support self-adaptation and reconfiguration. This discipline is closely related to the discipline of Dynamic Software Product Line (DSPL), proposed by Halpin et al. [3].

There is a lack of research for solutions to Dependability in DSPLs, showed few results in the literature. According to Kachabdar et al. [4], Dependability is one of a set of quality requirements that must be met in order to satisfy the needs of a particular market segment or mission (Shabani et al., 2007). However, SPLs just support the development of new products, but they do not support the reuse of existing ones. Therefore, the currently complex systems need to deal with dynamic aspects, such as successive modifications, at runtime. Dependability is a discipline that studies how systems can be designed to support self-adaptation and reconfiguration. This discipline is closely related to the discipline of DSPLs. In this context, Eleuterio et al. [1] studied the dependability of DSPLs. They found that there was no evidence about dependability in DSPL context. We selected nine primary studies in this regard. We performed a comparative study of the results, analyzing their strengths and weaknesses, aiming for a better understanding of this research area.

Eleuterio et al. [1] studied the dependability of DSPLs. They found that there was no evidence about dependability in DSPL context. We selected nine primary studies in this regard. We performed a comparative study of the results, analyzing their strengths and weaknesses, aiming for a better understanding of this research area.

2016-00000-00-00-0000 ISSN 2073-4336  
DOI: 10.3390/app120510240

123



## Requirements Engineering and Variability Management in DSPLs Domain Engineering: A Systematic Literature Review

Luisom M. P. da Silva<sup>1</sup>, Carla I. M. Bezerra<sup>2,3,4</sup>, Rosana M. C. Andrade<sup>2,3</sup> and José Maria S. Monteiro<sup>2</sup><sup>1</sup>Federal University of Ceará (UFC), Quixadá Campus, ZIP: 63002-500, Quixadá, CE, Brazil<sup>2</sup>Federal University of Ceará (UFC), Fortaleza Campus, ZIP: 60455-760, Fortaleza, CE, Brazil<sup>3</sup>School of Computer Science and Technology, UFSC, Brazil<sup>4</sup>Belo Horizonte ZIP: 31260-902, Belo Horizonte, MG, Brazil[luisom@ufc.br](mailto:luisom@ufc.br), [carla.bezerra\\_receveu@gmail.com](mailto:carla.bezerra_receveu@gmail.com), [rosana.andrade@ufc.br](mailto:rosana.andrade@ufc.br), [jose.monteiro@ufc.br](mailto:jose.monteiro@ufc.br)

**Keywords:** Dynamic Software Product Lines; Requirements Engineering; Variability Management.

**Abstract:** Recently, Software Product Lines (SPLs) have been used successfully for building products families. However, the currently and complex software products demand more adaptive features. Today, many application domains demand capabilities for more adaptation and post-deployment reconfiguration. In this context, Dynamic Software Product Lines (DSPLs) have been proposed as a solution for dealing with these requirements. DSPLs have been used to support the reuse of existing products and to adapt them to the changes in the product use environment. DSPLs present some interesting characteristics, such as the reuse of existing products and the possibility of adapting them to the changes in the product use environment. This paper is available at <https://www.mdpi.com/2073-4336/12/5/10240>.

da Silva et al.  
2016

### 1. INTRODUCTION

Software Product Lines (SPLs) can be defined as a set of software-intensive systems sharing a common infrastructure to support similar products or product families needs of a particular market segment or mission (Shabani et al., 2007). However, SPLs just support the development of new products, but they do not support the reuse of existing ones. Therefore, the currently complex systems need to deal with dynamic aspects, such as successive modifications, at runtime. Dependability is a discipline that studies how systems can be designed to support self-adaptation and reconfiguration. This discipline is closely related to the discipline of DSPLs, proposed by Halpin et al. [3].

There is a lack of research for solutions to Dependability in DSPLs, showed few results in the literature. According to Kachabdar et al. [4], Dependability is one of a set of quality requirements that must be met in order to satisfy the needs of a particular market segment or mission (Shabani et al., 2007). However, SPLs just support the development of new products, but they do not support the reuse of existing ones. Therefore, the currently complex systems need to deal with dynamic aspects, such as successive modifications, at runtime. Dependability is a discipline that studies how systems can be designed to support self-adaptation and reconfiguration. This discipline is closely related to the discipline of DSPLs. In this context, Eleuterio et al. [1] studied the dependability of DSPLs. They found that there was no evidence about dependability in DSPL context. We selected nine primary studies in this regard. We performed a comparative study of the results, analyzing their strengths and weaknesses, aiming for a better understanding of this research area.

Eleuterio et al. [1] studied the dependability of DSPLs. They found that there was no evidence about dependability in DSPL context. We selected nine primary studies in this regard. We performed a comparative study of the results, analyzing their strengths and weaknesses, aiming for a better understanding of this research area.

2016-00000-00-00-0000 ISSN 2073-4336  
DOI: 10.3390/app120510240

544

Ana L. M. Bezerra, Carla I. M. Bezerra, Rosana M. C. Andrade, Luisom M. P. da Silva

Institute of Computing, Federal University of Ceará (UFC), Fortaleza, Brazil

E-mail: [rosana.andrade@ufc.br](mailto:rosana.andrade@ufc.br); [carla.bezerra\\_receveu@gmail.com](mailto:carla.bezerra_receveu@gmail.com); [luisom@ufc.br](mailto:luisom@ufc.br); [ana.bezerra@ufc.br](mailto:ana.bezerra@ufc.br)Copyright © 2016 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

# (Software) Product Lines Applications of Product Line Techniques - Dynamic SPL

2015 IX Brazilian Symposium on Components, Architectures and Reuse Software

## Variability Management in Dynamic Software Product Lines: A systematic mapping

Gabriela Guedes<sup>1</sup>, Carla Silveira<sup>2</sup>, Montse Roemer<sup>1</sup>, Justino Castro<sup>3</sup>  
<sup>1</sup>Information Center (CIn), UFPE - Recife, PE, Brazil  
<sup>2</sup>UFPE Campus Cognitiva - Capivara, PE, Brazil  
ggs, cslv, mro, jcastro@de.ufpe.br

**Abstract:** Dynamic Software Product Lines (DSPLs) are SPLs in which the variability configuration may occur at runtime. Despite being proposed in 1997, the DSPL approach has been an area of research and development for more than two decades. This paper presents the results of one systematic mapping, which identifies the main approaches used to manage variability in DSPLs. The paper also highlights the main challenges and opportunities for future research. The results show that the DSPL approach is still immature and needs further research.

Keywords: Dynamic Software Product Lines (DSPLs), variability management, systematic mapping, software reuse.

**Guedes et al.  
2012**

on SPLs in which the variability configuration occurs at runtime. The DSPL manages the environment and when necessary changes, it can have a new configuration [1] (see below [6]).

Dynamic variability management, as proposed by DSPL, has been approached in various ways such as: dynamic variability management (DVM), self-\* systems, pervasive systems and context-aware methods. In this paper, we propose a classification of the DSPLs based on the development of dynamically adaptive systems (DAS) in general, i.e., systems capable of adapting themselves to certain changes in their environment.

DSPLs provide a way of modelling DAS by using SPL variability management techniques to deal with the variable features of the final system [1]. Thus, DSPLs are a DSPLs. That is, dynamically adapt to system itself is a DSPL, and every variation of the DSPL should be seen as a product of the DSPL [6].

In general, approaches for DSPL should be capable of supporting the generation of configurations for different variants, and use them variability models at runtime. For the case of the DSPLs, the variability models are generated at runtime in order to detect changes that require the system to adapt. When adaptation is required, a new product configuration must be generated. This can be done through the use of variability models, e.g., contexts, meta-functional requirements (MFRs), etc.

DOI: 10.1109/ICARIS.2012.20

Authorized access granted to IEEE Xplore. Downloaded on March 27, 2023 at 10:00 07 UTC from IEEE Xplore. Restrictions apply.

978-1-4673-4421-2/12/\$31.00 ©2012 IEEE

DOI: 10.1109/ICARIS.2012.20

90



## Characterizing Dynamic Software Product Lines – A Preliminary Mapping Study

Vanilson Andrade Bezerra and Sílvia Lemos Meira  
Universidade Federal de Pernambuco,  
CIn – Center of Informatics,  
Ribeirão Preto Institute of Engineering  
(rads, rlema)@cina.ufpe.br

Eduardo Santana da Almeida  
Universidade Federal de Pernambuco,  
Ribeirão Preto Institute of Engineering  
(edsa)@cina.ufpe.br

**Abstract:** Dynamic Software Product Lines (DSPLs) has been implemented in a variety of domains, such as consumer products, medical devices, and industrial equipment. However, it still represents a current topic with many open questions, such as: how to manage variability in DSPLs? How to support variability in DSPLs? How to evaluate variability in DSPLs? This paper aims to investigate in more detail relevant issues related to DSPLs. It also proposes a classification to structure the Dynamic Software Product Line field.

During the last few years, Software Product Lines has been implemented in a variety of domains, such as consumer products, medical devices, and industrial equipment. However, it still represents a current topic with many open questions, such as: how to manage variability in DSPLs? How to support variability in DSPLs? How to evaluate variability in DSPLs? This paper aims to investigate in more detail relevant issues related to DSPLs. It also proposes a classification to structure the Dynamic Software Product Line field.

**Burégio et al.  
2010**

**I. INTRODUCTION**  
Currently, the complexity of the majority of applications on current emerging domains, such as context-aware systems, pervasive and ubiquitous computing, sensor networks, mobile devices, and distributed systems, requires the use of variability modeling to correctly model the variability configuration.

Section II describes the main concepts of variability and solution strategies. Section III presents the results of the systematic mapping study. Section IV discusses the limitations of the study. Finally, Section V concludes the paper and discusses future work such as Section VI.

### II. RELATED WORK

One main goal was understanding the evolution and challenges of variability management, which include: the development of generic principles, identification and selection of variability models, and the application of variability models to real systems. The following section presents a brief assessment of included studies, and concluding the results.

To deal with these kinds of systems, several studies have been developed, such as: the Systematic Mapping Studies on Dynamic Software Product Lines (DSPLs) [31] has been developed to identify the main characteristics of DSPLs. The DSPL approach enables the generation of software variants at runtime and can be seen as an extension of the concept of Software Product Lines (SPLs).

From the point of view of variability models, the main goal was to identify the main types of variability models studied related to DSPL and characterize such types, mapping the main characteristics of each type.

As a result, we expect to provide an overview of the current state of DSPLs, identifying the main characteristics as well as the contributions already available within the field. Furthermore, this study should provide a basis for future research, helping to identify gaps and areas for improvement in the DSPL area.

The rest of this paper is organized as follows. Section 2 discusses some related work. Section 3 presents the methodology used to perform the systematic mapping study. Section 4 describes the outcomes obtained from

## (Software) Product Lines Applications of Product Line Techniques - Dynamic SPL

Author & Venue	Title
Aguayo and Sepúlveda ('22)	Variability Management in Dynamic Software Product Lines for Self-Adaptive Systems—A Systematic Mapping
Eleuterio et al. ('16)	On the dependability for dynamic software product lines: A comparative systematic mapping study
da Silva et al. ('16)	Requirements Engineering and Variability Management in DSPLs Domain Engineering: A Systematic Literature Review
Guedes et al. ('15)	Variability management in dynamic software product lines: A systematic mapping
Burégio et al. ('10)	Characterizing dynamic software product lines-a preliminary mapping study

(Software) Product Lines Management

The image shows the front cover of the journal 'Information and Software Technology'. At the top, it says 'Information and Software Technology 55 (2013) 136–148'. Below that is a small Elsevier logo. The main title 'Information and Software Technology' is in large, bold, black font. Underneath it, it says 'Contents lists available at ScienceDirect' and 'journal homepage: www.elsevier.com/locate/infsof'. There's also a small logo for 'DowMath'. The central part of the cover features a large, stylized graphic of a globe or network of lines against a blue background. The bottom half of the cover has a white background with some faint, illegible text.

**Systematic Review of Solutions Proposed for Product Line Economics**

Mahvish Khurram, Tony Gorschek, Kent Petersson  
*Blacksburg Institute of Technology, Department of Systems and Software Engineering,  
 5-372 25, Rosedale, Sweden  
[mkhurram@bilkent.edu.tr](mailto:mkhurram@bilkent.edu.tr) <http://www.powersoft.bilkent.edu.tr>*

**Abstract**

This paper presents a systematic review of all the solutions proposed for product line economics in the product line (PL) e.g. SPL adoption mechanisms, PL architecture decisions, SPL cost savings project, cost estimation, and revenue modeling. The primary goal of the review is to analyze the level of empirical support for each solution. The second goal is to analyze the purpose of mapping maturity as well as to what extent proposed solutions might proceed in terms of feasibility and applicability.

**Khurram et al.  
 2008**

Software product families have received significant attention from researchers and practitioners since the 1990s [1, 2]. The concept of product lines aims towards having a set of systems that share a common infrastructure and reuse components for a particular needs of a market segment, developed from a customer set of core assets in a certain given way [1]. The first approach to reuse was the reuse-oriented approach for reuse in software development [3] with the major benefits of product lines adoption reported as reduced costs, improved quality, and improved improved quality [4, 5, 6]. For these reasons many companies developing software intensive products have adopted the reuse-oriented approach instead of a SPL approach [5, 6].

To make sound decisions about product line processes and products, it is important to answer the question "Does it pay?" [8]. In order to make sound decisions about product line processes, one must understand the business impact of those decisions and vice versa [9]. Without economic evaluation of

applied nature validated in industry. In addition to industry validation all other types of empirical results are collected and taken into consideration to offer a more comprehensive view of the state-of-the-art. To achieve this, the selected papers are categorized and analyzed from several perspectives, such as research methods, scope, and application. The empirical results in relation to feasibility and usefulness of the proposed solutions are also evaluated. As the main purpose performs evaluating using a certain solution, the results of the study can be used as an indicator of the potential success or failure of the risk of a certain solution. From an academic point of view, research planning studies and evaluation of a solution are the two main parts of a systematic study design as the evaluation criteria of the review presented in this paper could be seen as a checklist to rate a solution's feasibility and usefulness.

**2. Related Work**

# (Software) Product Lines Management

Author & Venue	Title
Marchezan et al. ('22)	Software product line scoping: A systematic literature review
Tüzün and Tekinerdogan (INST'15)	Analyzing impact of experience curve on ROI in the software product line adoption process
Khurum et al. ('08)	Systematic review of solutions proposed for product line economics

## 2. Problem Space

# Problem Space Representation

**Bischoff et al.**  
2019  
**Integrating two (or more) feature models with one another to get an output-desired feature model**

Integration of feature models: A systematic mapping study

Vinicius Bischoff, Kleineren Fuster, Lucian José Gonçalves, Jorge Luis Vicentini Barbosa

Feature Models in Applied Computing (FIMAC), University of São Paulo, São Paulo, Brazil

Information and Software Technology

journal homepage: [www.elsevier.com/locate/infsof](http://www.elsevier.com/locate/infsof)

ARTICLE INFO

ABSTRACT

The integration of feature models has been widely investigated, but the literature seems to disagree on how to do it. This paper presents a systematic mapping study to identify the main approaches for integrating feature models. The results show that there is no single best way to integrate feature models, as each approach has its own strengths and weaknesses. The study also identifies some common challenges and opportunities for future research.

© 2019 Elsevier GmbH. All rights reserved.

Keywords: Feature model, Integration, Systematic mapping, Feature modeling, Variability modeling

Received 1 October 2018; revised 10 January 2019; accepted 21 January 2019; available online 1 September 2019  
0020-0255/\$ - see front matter © 2019 Elsevier GmbH. All rights reserved.  
<https://doi.org/10.1016/j.infsof.2019.01.012>

**Eichelberger and Schmid**  
2015  
**Investigate 11 different textual variability modeling languages**

Mapping the design-space of textual variability modeling languages: a refined analysis

Holger Eichelberger · Klaus Schmid

© Springer-Verlag Berlin Heidelberg 2014

Abstract Variability modeling is a space set of needs, variability modeling languages are a means to represent these needs. In this paper we investigate 11 different textual variability modeling languages. We compare them with respect to their expressiveness and their usefulness for reuse. The results show that there is no single best language, as each language has its own strengths and weaknesses. The study also identifies some common challenges and opportunities for future research.

This article is based on our initial analysis of textual variability modeling languages in [1].

H. Eichelberger (✉) · K. Schmid

Institute für Systemtechnik und Softwareengineering, University of Hildesheim, Heinrich-Plett-Strasse 43, 31135 Hildesheim, Germany

e-mail: [ehelberger@informatik.uni-hildesheim.de](mailto:ehelberger@informatik.uni-hildesheim.de)

K. Schmid

e-mail: [schmid@informatik.uni-hildesheim.de](mailto:schmid@informatik.uni-hildesheim.de)

Published online: 11 December 2014

Springer

**Hubaux et al.**  
2013  
**Concerns regarding feature diagrams**

Separation of Concerns in Feature Diagram Languages: A Systematic Survey

ARNOLD HUBAUX, University of Namur, Belgium  
THOMAS DE LAET, The Open University, United Kingdom  
PATRICK HEYMANS, University of Namur, Belgium

The need for flexible combination of large featureable software systems, according to requirements of many stakeholders, has become an important problem in software engineering. Among many approaches, feature diagrams have emerged as a promising solution. The notion of Feature Model (FM) has emerged as a major modeling concept. Drawing from established disciplines of manufacturing, FM is a graphical representation of a system's configuration space. It is used to represent what specific stakeholder requirements can be fulfilled. A major difficulty of FM approaches stems from addressed concerns such as the separation of concerns, the reuse of features, the reuse of configurations, the reuse of configurations in different contexts, and the reuse of configurations in different product configurations of a feature set. These have been several proposals for feature diagram decomposition, but they have not been widely adopted. This paper reviews

Keywords Feature diagram, Separation of concerns, Feature model, Configuration, Feature reuse, Feature decomposition

Received: 10 March 2012; revised: 10 July 2013; accepted: 11 August 2013  
Available online: 1 September 2013  
© 2013 Springer-Verlag Berlin Heidelberg

Author's address: A. Hubaux ([ahubaux@cs.unamur.be](mailto:ahubaux@cs.unamur.be)), PRISM Research Center, Faculty of Computer Science, University of Namur, Belgium; T. De Laet ([tdelet@open.ac.uk](mailto:tdelet@open.ac.uk)), The Open University, United Kingdom; P. Heymans ([pheyman@cs.unamur.be](mailto:pheyman@cs.unamur.be)), PRISM Research Center, Faculty of Computer Science, University of Namur, Belgium

For digital or hard copies of part or all of this work for personal or classroom use in private without the prior written consent of or from or arrangements with the publisher is illegal. For other permissions, or for permission to photocopy material for internal or classroom use, for digital or hard copies, or for republication on other media, or for other kinds of copying, such as copying for general distribution for advertising or promotional purposes, for creating new collective works, or for resale, or for use in other works, please apply for permission and/or a fee. Permissions may be requested through Rights and Permissions, Springer, Haberstraße 7, 69126 Heidelberg, Germany, or via e-mail: [permissions@springer.com](mailto:permissions@springer.com), or through Rights and Permissions, Springer, 233 Spring Street, New York, NY 10013-1578, USA, or via e-mail: [permissions@springer.com](mailto:permissions@springer.com).  
DOI 10.1007/978-3-642-31134-0\_11  
2013 10.1007/978-3-642-31134-0\_11  
[http://dx.doi.org/10.1007/978-3-642-31134-0\\_11](http://dx.doi.org/10.1007/978-3-642-31134-0_11)

# Problem Space Representation

Author & Venue	Title
Bischoff et al. ('19)	Integration of feature models: A systematic mapping study
Eichelberger and Schmid ('15)	Mapping the design-space of textual variability modeling languages: a refined analysis
Hubaux et al. ('13)	Separation of Concerns in Feature Diagram Languages: A Systematic Survey
Czarnecki et al. ('12)	Cool Features and Tough Decisions: A Comparison of Variability Modeling Approaches
Schmid et al. ('11)	A comparison of decision modeling approaches in product lines
Hubaux et al. ('10)	A Preliminary Review on the Application of Feature Diagrams in Practice
Sinnema and Deelstra ('07)	Classifying variability modeling techniques
Schobbens et al. ('06)	Feature Diagrams: A Survey and a Formal Semantics

# Problem Space Analysis



Article  
Reasoning Algorithms on Feature Modeling—A Systematic Mapping Study

Saúl Sepúlveda and Asia Cravero

Departamento de Ciencias de la Computación e Información, Centro de Estudios en Ingeniería de Software, Universidad de La Rioja, Logroño, Spain. <http://ceis.ulr.es>

\* Correspondence: saul.sepulveda@ulr.es

Abstract Context: Software product lines (SPLs) have reached a considerable level of adoption in the software industry. The most commonly used models for managing the variability of SPLs are feature

**Sepúlveda and Cravero  
2022**

**Contains 66 papers published  
between 2010 and 2020  
Overview of reasoning algorithms  
for feature models**

Received: 03/03/2022

Published: 03/04/2022

Publication's Note: MDSI encourages  
the reuse of its content for non-profit  
purposes and educational activities.



Copyright: © 2022 by the authors.

Licensee: MDPI Ltd.

This article is an open access article  
distributed under the terms and  
conditions of the Creative Commons  
Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Appl. Sci. 2022, 12, 1563; doi:10.3390/app1201563

<https://www.mdpi.com/journal/appliedsciences>



Computing manuscript No.  
(will be inserted by the editor)

Automated analysis of feature models: Quo vadis?

José A. Galindo · David Sepúlveda · Pablo Tránsito ·  
Antonio Manuel Gratacés-Fernández · Alvaro  
Roldán-Cortés

Received: date / Accepted: date

**Galindo et al.  
2019  
Database search + snowballing  
+ experience  
Contains 423 publications pub-  
lished after 2010**

In this paper we point different approaches and current literature on the literature of feature modeling (FMs). The information that can be obtained from FMs is extensive, and the mechanism for obtaining it are likewise varied. In fact, this area known as Automated Analysis of Feature Models (AAFM) [3] has recently been identified as one of the most important areas in the SPL [2].

The AAFM is the most widely cited research area in feature models [4] and can be summarized in three steps. First, FMs are translated to a logical representation. Second, an off-the-shelf solver or specific algorithm is used to perform a given analysis operation (such as counting the number of possible configurations) [5]. Finally, the results are interpreted and used in a determined context to perform other tasks such as product configuration or derivation.

A comprehensive list of proposals and operations for the AAFM was presented by Sepúlveda et al. [4] in 2019 that sorted the emerging subbranches of the discipline. In particular, three main operational

approaches that consider FMs as sets of constraints [6], general models [7] and feature models [8] were

**Draft version of the paper published in 115086, Oct. 2013, doi:10.3390/got21813401150086**

**A Literature Review on Feature Diagram Product Counting and  
Its Usage in Software Product Line Economic Models**

Ruben Heradio<sup>1,\*</sup>, David Fernández-Antoruz<sup>1</sup>, José Antonio Cerrada-Semolinos<sup>1,2</sup>, and  
Ismael Abad<sup>3,4</sup>

<sup>1</sup> Dpto. de Ingeniería Informática, Universidad Nacional de Educación a Distancia, Madrid, Spain

**Heradio et al.  
2013  
Product Counting  
Draws a connection to the eco-  
nomic aspects**

building and reusing a PL as compared to traditional development approaches.

There has been a great deal of research on the economics of software reuse, and a large number of economic models have been proposed. However, these models are mainly oriented to reuse and analyses 17 of them. However, general models in the context of software reuse can only be applied in a restricted way, since PL development involves some fundamental assumptions that are not reflected in these models [4]. To overcome this situation, several economic models oriented specifically to PLs have been proposed.

# Problem Space Analysis

## Automated Analysis of Feature Models 20 Years Later: A Literature Review<sup>1)</sup>

David Benavides, Sergio Segura and Antonio Ruiz-Cortés  
Dpto. de Lenguajes y Sistemas Informáticos, University of Seville  
Av. de la Reina Mercedes s/n, 41012 Seville, Spain

### Abstract

Software product line engineering is about producing a set of related products that share more commonalities than variabilities. Feature models are widely used for variability and commonality management in software product

**Benavides et al.**  
**2010**  
**1543 Citations**  
**Automated Analyses of feature models**

organizing their production in a mass production environment.

However, in this highly competitive and segmented market, companies need to offer more individualized customer's personalization. Software product line engineering promotes the production of a family of software systems that can be configured to meet different needs, producing them one by one from scratch. This is the key change: software product line engineering is about producing families of similar systems rather than the production of individual systems.

Software product lines have found a broad adoption in several branches of software production such as automotive, telecom, consumer electronics, embedded software and avionics [95]. However, adopting other types of software and systems applications such as desktop, web or data-intensive applications is a current challenge.

An organization decides to set up a software product line and faces the following issue: how is a particular

engineering discipline, the management of software products in known as software product line [24] or simply as SPL [10, 11, 12, 13, 14, 15, 16, 17, 18]. In order to support the customer's personalization, software product line engineering promotes the production of a family of software systems that can be configured to meet different needs, producing them one by one from scratch. This is the key change: software product line engineering is about producing families of similar systems rather than the production of individual systems.

Software product lines have found a broad adoption in several branches of software production such as automotive, telecom, consumer electronics, embedded software and avionics [95]. However, adopting other types of software and systems applications such as desktop, web or data-intensive applications is a current challenge.

An organization decides to set up a software product line and faces the following issue: how is a particu-

<sup>1)</sup>A very preliminary version of this paper was published in the proceedings of the XV Jornadas de Ingeniería del Software y Sistemas de Datos (JISD'07) held at the University of Seville, Spain, on October 2007. The authors would like to thank the anonymous reviewers for their useful comments. (David Benavides, Sergio Segura and Antonio Ruiz-Cortés)  
Preprint submitted to Information Systems

January 1, 2010

XV Jornadas de Ingeniería del Software y Sistemas de Datos  
JISD'07 2007  
José M. Gómez, Francisco J. Martínez  
CUCINE, Seville, Spain

## A SURVEY ON THE AUTOMATED ANALYSES OF FEATURE MODELS

David Benavides, Antonio Ruiz-Cortés, Pablo Trinidad and Sergio Segura  
Dpto. de Lenguajes y Sistemas Informáticos  
Universidad de Sevilla  
Av. de la Reina Mercedes s/n, 41012 Seville, Spain  
e-mail: {benavides, trinidad, arcls, sergio}@usdg.us.es

**Benavides et al.**  
**2006**  
**108 Citations**  
**Automated Analyses of feature models**

A key technical question that confronts software engineers is how to specify a particular product in an SPL. One of the possible solutions is to do it in terms of *feature*, where a feature is an increment in product functionality. Designing a family of software systems in terms of feature is more natural than doing it in terms of objects or classes because the former is closer to the actual needs [10]. In fact, the requirements of the system are specified in terms of features, whereas product lines are specified using feature models.

Feature models are recognized in the literature to be one of the most important contributions to SPL engineering [3]. A key task is to capture commonalities and variabilities among products and feature models are used to this end.

Although there has been advances in SPL engineering regarding feature models, a key technical aspect remains open: their automated analyses. In this paper we survey the state-of-the-art on the automated analyses of feature models. First, we review the different

# Problem Space Analysis

Author & Venue	Title
Sepúlveda and Cravero (Applied Sciences'22)	Reasoning Algorithms on Feature Modeling—A Systematic Mapping Study
Galindo et al. (Computing'19)	Automated analysis of feature models: Quo vadis?
Heradio et al. (IJSEKE'13)	A Literature Review on Feature Diagram Product Counting and Its Usage in Software Product Line Economic Models
Benavides et al. (Information Systems'10)	Automated Analysis of Feature Models 20 Years Later: A Literature Review
Benavides et al. (Jornadas de Ingeniería del Software y Bases de Datos'06)	A Survey on the Automated Analyses of Feature Models

# Problem Space Configuration

## Learning Software Configuration Spaces: A Systematic Literature Review

JULIANA ALVES PEREIRA, Univ Rennes, Inria, CNRS, IRISA, France

HUGO MARTIN, Univ Rennes, Inria, CNRS, IRISA, France

MATHIEU ACHER, Univ Rennes, Inria, CNRS, IRISA, France

JEAN-MARC JÉZÉQUEL, Univ Rennes, Inria, CNRS, IRISA, France

GOETZ BOTTERWECK, University of Limerick, Lero-The Irish Software Research Centre, Ireland

ANTHONY VENTRESQUE, University College Dublin, Ireland

Most modern software systems featuring systems like Linux or Android. Web browsers like Firefox or Chrome, video encoders like ffmpeg, xvid or VLC, mobile and cloud applications, etc.) are highly configurable. Hundreds of configuration options, features, or plugins can be combined, each potentially with distinct functionality and effects on execution time, memory, energy consumption, etc. Due to the combinatorial explosion and the cost

Pereira et al.  
2021

Sampling  
Also classify about machine  
learning algorithms

Abstract: Configuration management is the process of selecting the right combination of thousands of options (a.k.a. features or parameters) to configure various kinds of software systems and, in the case of highly configurable systems (e.g., mobile devices, software quality management, energy consumption, etc.), it is now acknowledged that software comes in many variants and is highly configurable through conditional compilations, command-line options, runtime parameters, configuration files, or plugins. Software product lines (SPLs), software generators, dynamic, self-adaptive systems, variability-intensive systems are well studied in the literature and enter in this class of configurable software systems [8, 12, 13, 32, 38, 71, 81, 95, 98].

From a different point of view, a software configuration is simply a combination of options' values. Through enumeration, a highly desired feature introduces exponential complexity due to the combinatorial explosion of possible variants. For example, the Linux kernel has 15,000+ options

Author's address: Juliana Alves Pereira, Univ Rennes, Inria, CNRS, IRISA, Rennes, France, juliana.alves.pereira@inria.fr; Hugo Martin, Univ Rennes, Inria, CNRS, IRISA, Rennes, France, hugo.martin@inria.fr; Mathieu Acher, Univ Rennes, Inria, CNRS, IRISA, Rennes, France, mathieu.acher@inria.fr; Jean-Marc Jézéquel, Univ Rennes, Inria, CNRS, IRISA, Rennes, France, jean-marc.jezequel@inria.fr; Goetz Botterweck, University of Limerick, Lero-The Irish Software Research Centre, Limerick, Ireland, goetz.botterweck@phc.ie; Anthony Ventresque, University College Dublin, Dublin, Ireland, anthony.ventresque@ucd.ie.

| 7 Jun 2019

## A Systematic Literature Review on the Semi-Automatic Configuration of Extended Product Lines

Lina Ochoa<sup>a,\*</sup>, Oscar González-Rojas<sup>a</sup>, Juliana Alves Pereira<sup>b</sup>,  
Harold Castro<sup>a</sup>, Gustav Sander<sup>c</sup>

<sup>a</sup>Universidad de los Andes, Bogotá, Colombia  
<sup>b</sup>Systèmes et Informatique, Université Paris-Saclay, School of Engineering  
<sup>c</sup>University of Magdeburg, Magdeburg, Germany

Ochoa et al.  
2018  
Extended product lines  
Papers published between 2000  
and 2016

consider non-functional properties in the product line modelling. We compare and classify a total of 66 primary studies from 2000 to 2016. Mainly, we give an in-depth view of techniques used by each work, how these techniques are applied and their main contribution. As main contributions, we have identified i) a need to improve the quality and evaluate of existing approaches, ii) a lack of hybrid solutions to support multiple configuration constraints, and iii) a need to improve scalability and performance conditions.

Keywords:  
Extended product line, product configuration, systematic literature review

Preprint submitted to Journal of Systems and Software

October 15, 2019

Computer Standards & Interfaces 60 (2014) 30–48

## Computer Standards & Interfaces

journal homepage: [www.elsevier.com/locate/csf](http://www.elsevier.com/locate/csf)



## Intelligent software product line configurations: A literature review

Umma Afzal<sup>a,b,\*</sup>, Tariq Mahmood<sup>a,b</sup>, Zahoor Shahid<sup>a,c</sup>

<sup>a</sup>National University of Emerging Sciences, Experiments (COMSATS Science Campus), Lahore

<sup>b</sup>Software Engineering and Technology, College of Computing & Information Sciences, Khalid, Pakistan

### ARTICLE INFO

Received date:

Accepted date:

Available online date:

© Software product line (SPL) is a set of individual software instances produced for configuring similar soft-

ware over their lifetime, goal, use or as decomposing its life span

© Author(s). Published by Elsevier Ltd. This is an open access article under the terms of the Creative Commons Attribution Non-Commercial License (<http://creativecommons.org/licenses/by-nc/3.0/>), which permits unrestricted non-commercial use, distribution, and reproduction in other forms, provided the original author(s) and source are credited.

Abstract: The adoption of an inconsistency resolution process is critical for the success of intelligent software product line configurations. As work in conflict with each other, leading to reduced productivity and morale, reduce each other's, and non-compliance with business product requirements. Hence, the need for an intelligent software product line configuration system that can handle conflicts to resolve them in selecting features [10][11][12]. These problems have already been highlighted in different case studies and research papers. However, there is no clear solution to these problems. The challenge is to ensure an automatic, inconsistency-free configuration in the face of complex feature models, customer requirements and business product requirements [13][14][15][16][17][18][19][20]. Artificial intelligence (AI) techniques [16][17][18][19] can provide the best solution to these problems. AI techniques can help to generate configurations of intelligent programs which attempt to reproduce the way business act naturally. AI techniques have some widespread applications in various industries, e.g., healthcare, engineering and other

\* Corresponding author at: National University of Emerging Sciences, Experiments (COMSATS Science Campus), Lahore, Pakistan.

<sup>a</sup> E-mail addresses: [ummamafzal@comsats.edu.pk](mailto:ummamafzal@comsats.edu.pk), [zahoor.s@comsats.edu.pk](mailto:zahoor.s@comsats.edu.pk).

<sup>b</sup> Tel.: +92 33 25000474/75/76/77/78; fax: +92 33 25000110.

<sup>c</sup> Tel.: +92 33 25000474/75/76/77/78; fax: +92 33 25000110.

<https://doi.org/10.1016/j.csf.2014.06.002>

# Problem Space Configuration

Author & Venue	Title
Pereira et al. (JSSO'21)	Learning Software Configuration Spaces: A Systematic Literature Review
Ochoa et al. (JSSO'18)	A systematic literature review on the semi-automatic configuration of extended product lines
Afzal et al. (Computer Standards & Interfaces'16)	Intelligent software product line configurations: A literature review

# 3. Solution Space

# Solution Space Requirements

Requirements Modelling Languages for Software Product Lines: a Systematic Literature Review

Samuel Sepúlveda<sup>a\*</sup>, Ásia Caceres<sup>b</sup>, Cristina Caceres<sup>b</sup>  
<sup>a</sup>Departamento de Ciencias de la Computación e Información, Centro de Estudios en Ingeniería de Sistemas, Universidad de la República, Montevideo, Uruguay  
<sup>b</sup>Departamento de Lenguajes y Sistemas Informáticos, Universidad de Alcalá, España

## Abstract

**Context:** Software product lines (SPLs) have reached a considerable level of adoption in the software industry, having demonstrated their cost-effectiveness for developing higher quality products with lower costs. For this reason, in the

**Sepúlveda et al.**  
2016

**Method:** We have conducted a Systematic Literature Review with six research questions that cover the main objective. It includes 51 studies, published from 2000 to 2013.

**Results:** The mean level of maturity of the modelling languages is 2.50 over 5, with 40% of them falling within level 2 or below - no implemented abstract syntax reported. They show a level of expressiveness of 0.7 over 1.0. Some constructs (feature, mandatory, optional, alternative, exclude and refine) are present in all the languages, while others (cardinality, attribute, constraint and label) are less common. Only 9% of the languages have been empirically val-

\*This document is a collaboration effort.  
Corresponding author. Tel.: +598 21 274108.  
Email addresses: samuel.sepulveda@udec.edu.uy (Samuel Sepúlveda),  
asia.caceres@deci.udec.edu.uy (Ásia Caceres), caceres@deci.udec.edu.uy (Cristina Caceres)

Preprint submitted to Information and Software Technology

January 06, 2015

Torres et al. / Journal of Software Engineering Research and Development (2015) 4:4  
DOI 10.1007/s40101-015-0020-0

Journal of Software Engineering  
Research and Development  
a SpringerOpen® Journal

## REVIEW

## Open Access

### Templates for textual use cases of software product lines: results from a systematic mapping study and a controlled experiment

Iomarla S Santos<sup>1</sup>\*, Rosana MC Andrade<sup>2</sup> and Pedro A Santos Neto<sup>2</sup>

<sup>1</sup>Graduate Program in Computer Science, Federal University of Paraná, Brazil  
2Department of Computer Science, Federal University of Paraná, Brazil

**Abstract**  
Use case templates can be used to describe functional requirements of a Software Product Line (SPL). To our knowledge, to the best of our knowledge, no efforts have been made to collect and summarize these existing templates and no empirical evaluation of the use cases' comprehensibility provided by these templates has been addressed yet. The context of this work is the use of SPLs in the domain of mobile devices. This paper aims to study about the SPL variability description using textual use cases. From this mapping, we found 11 use case templates available and evaluated the use cases according to the use

**Santos et al.**  
2015

Received: 10 January 2014; Accepted: 10 January 2015  
Published online: 13 February 2015  
Keywords: Use case, Systematic mapping study, Software product line, Controlled experiments

## 1 Introduction

The paradigm of Software Product Line (SPL) has emerged together with large-scale reuse of software components [1]. A Software Product Line (SPL) is defined as “a set of related families of systems that share a common managed feature set, satisfying a particular market segment's specific needs or mission and that is developed from a common set of core assets in a prescribed way”.

Because the products of an SPL reuse common artifacts among themselves, the SPL approach maximizes reuse and minimizes development costs. Some advantages of using SPLs are: (i) reuse of code and Guagliardi 2007, (ii) better reuse and Chaves 2006, (iii) increased productivity, (iv) better time to market, and (v) higher product quality.

In the SPL paradigm, the requirements engineering activity needs to cope with common and variable requirements for the whole set of products in the family [6–10].

© 2015 Springer. This is an Open Access article distributed under the terms of the Creative Commons Attribution License

(http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium,

provided original author is properly cited.



Springer

2014 Eighth Brazilian Symposium on Software Components, Architectures and Tools

### How to Describe SPL Variabilities in Textual Use Cases: A Systematic Mapping Study

Iomarla S. Santos, Rosana M. C. Andrade

Department of Computer Science  
Federal University of Paraná,  
Foz do Iguaçu, Paraná, Brazil  
{iomarla.santos, rosana}@cpes.ufpr.br

Pedro A. Santos Neto  
Department of Computer Science  
Federal University of Paraná,  
Curitiba, Paraná, Brazil  
{pedro.santos.neto}@ufpr.br

**Abstract** In the Software Product Line (SPL) paradigm, the variability description is an important issue for the requirements engineering process. In this scenario, there are several approaches to handle variability in use cases. However, to the best of our knowledge, there is no work that studies how to handle variability in use cases. This paper aims to summarize the existing templates for textual use case description in the SPL paradigm. This paper addresses the

review indicated that textual, feature and use cases are the leading formats to express requirements in an SPL.

In both the feature and use cases formats, the requirements are described at high level, which could make difficult for communicating to all stakeholders the final behavior of the system. Therefore, this paper aims to propose a template to summarize the existing templates for textual use case description in the SPL paradigm. This paper addresses the

**Santos et al.**  
2014

## 1. INTRODUCTION

The paradigm of Software Product Line (SPL) has emerged together with large-scale reuse of software components [1]. According to Noschese [2], an SPL is “a set of software-intensive systems that share a common managed feature set, satisfying a particular market segment's specific needs or mission and that is developed from a common set of core assets in a prescribed way”. The use of SPLs has [3] productivity improvement, (i) best time to market and (ii) higher product quality.

In the SPL paradigm, the requirements engineering activity needs to cope with common and variable requirements for the whole set of products in the family [6–10].

The SPL requirements should then be modeled from the reuse perspective by exploring the commonality and variability [11].

In this way, one challenge for SPL development relates to how to handle variability in the requirements [12]. According to Alves et al. [4], which presented a Systematic Literature Review [5] about Requirements Engineering (RE) in SPLs, there are two approaches for the documentation of SPL requirements, such as the Feature Model (FM) and the Object Model (OM) [13].

For the FM, the variability model is a good approach to

handle the variability of the SPL functional requirements whereas the OM is a good approach to handle the variability of the domain modeling in a more generic way [14].

It is important to note that in the SPL paradigm, the use cases are a good approach to handle variability [15, 16]. Thus, there are proposals for the description of SPL requirements using the use cases and variability models. Furthermore, the results of this systematic

SPM 1.4.7795\_7405\_143\_EU\_001-001-2014-000

DOI 10.1007/s40101-015-0020-0

Springer

Authorized licensed use limited to: Aalto University Library. Downloaded on March 27, 2023 at 12:48:41 UTC from IEEE Xplore. Restrictions apply.



# Solution Space Requirements

Author & Venue	Title
Sepúlveda et al. ('16)	Requirements modeling languages for software product lines: A systematic literature review
Santos et al. ('15)	Templates for textual use cases of software product lines: results from a systematic mapping study and a controlled experiment
Santos et al. ('14)	How to describe spl variabilities in textual use cases: A systematic mapping study
Alférez et al. ('13)	Evaluating Scenario-Based SPL Requirements Approaches — The Case for Modularity, Stability and Expressiveness
Alves et al. ('10)	Requirements engineering for software product lines: A systematic literature review
Khurum and Gorschek ('09)	A systematic review of domain analysis solutions for product lines

# Solution Space Architecture & Design

**Evaluating variability at the software architecture level: An Overview**

Ana Paula Allian<sup>1</sup>, Bruno Senna<sup>2</sup>, Elisa Yumi Nakagawa<sup>3</sup>

<sup>1</sup>University of São Paulo  
<sup>2</sup>University of São Paulo  
<sup>3</sup>University of São Paulo

**ABSTRACT**

Software architectures are designed for developing software systems under different requirements. In order to support reuse, it is necessary to deal with a significant amount of variability in functional and quality attributes to create different products. Due to this variety, the reuse of existing architectures is often difficult and complex, as different alternatives of systems might be developed leading to an expensive and time consuming task. Several methods have been proposed to evaluate variability in software evolution items (SEI) aiming to assess whether or not the architecture can be reused. However, these methods have shown little success in the existing evaluations methods in most cases for evaluating variability in software architectures, instead of only considering PLAs. Understanding and explicitly consider

**Allian et al.**  
2019

real effectiveness as most studies present gaps and lack of evidence about the usage of each technique in an industrial environment.

**CCS CONCEPTS**  
• Software engineering → Software architectures.

**KEYWORDS**  
software architecture, software variability, evaluation, systematic mapping study

**ACM Reference Format:**  
Ana Paula Allian<sup>1</sup>, Bruno Senna<sup>2</sup>, and Elisa Yumi Nakagawa<sup>3</sup>. 2019. Evaluating variability at the software architecture level: An Overview. In *Proceedings of the 12th ACM/IEEE/IFIP Symposium on Applied Computing (SAC '19)*, April 8–12, 2019, New York, NY, USA, 8 pages. <https://doi.org/10.1145/3297487.3297507>

**1 INTRODUCTION**

Variability is understood as the ability of a software system to be modified to satisfy different needs. It is also called as variability (i.e., change), variability, and variability [21]. Although variability has been further explored in software product lines, there is still a lack of studies that evaluate the real effectiveness of methods for evaluating variability in software architectures, instead of only considering PLAs. Understanding and explicitly consider-

**Science of Computer Programming 123 (2018) 40–60**

Contents lists available at ScienceDirect  
**Science of Computer Programming**  
[www.elsevier.com/locate/scpro](http://www.elsevier.com/locate/scpro)

**ELSEVIER**

**basic behavioral models for software product lines: Expressiveness and testing pre-orders**

Harish Beohar, Malha Varsiosha, Mohammad Reza Mousavi<sup>1</sup>

<sup>1</sup>Graduate School of Technology, University of Twente, Enschede, The Netherlands

**ARTICLE INFO**

**Article history:**  
Received 10 October 2014  
Received in revised form 11 June 2015  
Accepted 12 June 2015  
Available online 9 July 2015

**ABSTRACT**

In order to provide a rigorous foundation for Software Product Lines (SPLs), several fundamental approaches have been proposed to their formal modeling. In this paper, we provide a structured overview of these formalisms based on labeled transition systems (LTSs). We also propose a new behavioral model for SPLs, namely, basic behavioral models (BBMs). BBMs are LTSs that represent the behavior of a set of objects that interact with each other. Moreover, we define the notion of tests for each of these LTSs and show that our

**Beohar et al.**  
2016

**I. Introduction**

**II. Motivation**

Software product lines (SPLs) are becoming increasingly popular as efficient means for mass production and mass customization of software. Hence, establishing formal foundations for specification and verification of SPLs can benefit a large community and can have substantial impact. In the last few years, many researchers have spent substantial effort in establishing formal foundations for mass production and customization in the area of software engineering [1], providing comprehensive overviews.

In this paper, we put some structure in the body of knowledge regarding one of the main fundamental questions of behavioral models for SPLs, namely those based on labeled transition systems, such as state transition systems as studied in [2]. These basic models can serve as semantic models for extraction of higher level models such as domain specific languages (DSLs), or those based on the Unified Modeling Language (UML) state or sequence diagrams. Hence, bringing more structure into the field of behavioral models than the current computational models can help the language designers of higher level language to make the right choices when defining the semantics of their language.

**CCS Concepts**  
• Software engineering → Software architecture; Software product lines.

**Keywords**  
Software Product Lines; Product Line Architectures; Variability; Metamodels; Systematic Literature Review

**1. INTRODUCTION**

In recent years, Software Product Lines (SPLs) have attracted increasing interest from both industry and academia. In this context, a consequence, more organizations are adopting SPLs [3], which consists of a portfolio of related products sharing commonalities and

**A Systematic Review on Metamodels to Support Product Line Architecture Design**

Crescencio Lima<sup>1</sup>, Fernanda Oliveira<sup>2</sup>, Fabrício de Oliveira<sup>3</sup>, Vitoria da Cunha<sup>4</sup>, Brazil  
<sup>1</sup>crescencio@polim.uba.br

**ABSTRACT**

Product Line Architectures (PLA) design is a key activity for developing software systems. The main challenge in performing PLA design is a difficult task, mostly due to the complexity of the software models that a PLA deals with, and their variability. Metamodels models have been proposed to support the representation of models that comprise a PLA. SPL variability and its impact on the design of a PLA have been studied. In this paper, we propose a structured review of these formalisms based on labeled transition systems (LTSs). We also propose a new behavioral model for SPLs, namely, basic behavioral models (BBMs). BBMs are LTSs that represent the behavior of a set of objects that interact with each other. Moreover, we define the notion of tests for each of these LTSs and show that our

parts systematically [4]. According to Diao et al. [20], the main challenge in performing PLA design is a difficult task, mostly due to the complexity of the software models that a PLA deals with, and their variability. Metamodels models have been proposed to support the representation of models that comprise a PLA. SPL variability and its impact on the design of a PLA have been studied. In this paper, we propose a structured review of these formalisms based on labeled transition systems (LTSs). We also propose a new behavioral model for SPLs, namely, basic behavioral models (BBMs). BBMs are LTSs that represent the behavior of a set of objects that interact with each other. Moreover, we define the notion of tests for each of these LTSs and show that our

**Heradio et al.**  
2016

**and programmatic sections in the language, when and how they interact?**

Mahesh and Aditya [21] stated that the engineering process of a system is divided into three phases: (i) gathering requirements needed for product line engineering methods, (ii) help in the design of a product line architecture, and (iii) help in the implementation of the architecture. The first phase is to identify the concepts and how they are evaluated in different methods, and (ii) provide the conceptual schema required for reuse.

In this paper, we present a Systematic Literature Review (SLR) to support the design of a PLA. The main goal of this SLR is to answer research questions, and establish the state of knowledge regarding metamodels in the context of PLA design. The main contributions of this SLR are to (i) characterize the metamodels used for PLA designs, and characterize their main characteristics, (ii) analyze the main challenges and (iii) propose a methodology for reuse and breeding.

Therefore, we define the goal of our study using the Goal Question Metric (GQM) approach [22] as follows:

- Analyze the main digital artifacts or at least part of the work for reuse and breeding of metamodels used for PLA design.
- Analyze the literature on product line architectures design and reuse, and identify the main challenges and opportunities to gather information from the point of view of reuse and breeding.

For this purpose, we conducted a search in the following databases: IEEE Xplore (20.12.2016), Google Scholar (20.12.2016), and SCOPUS (20.12.2016).

**Keywords**  
Software Product Lines; Product Line Architectures; Variability; Metamodels; Systematic Literature Review

**1. INTRODUCTION**

In recent years, Software Product Lines (SPLs) have attracted increasing interest from both industry and academia. In this context, a consequence, more organizations are adopting SPLs [3], which consists of a portfolio of related products sharing commonalities and

# Solution Space Architecture & Design

2015 IX Brazilian Symposium on Components, Architectures and Reuse Software

## Initial Evidence for Understanding the relationship between Product Line Architecture and Software Architecture Recovery

Conrado Rodrigues Lima Neto<sup>a</sup>, Matheus Paroco Santa Catarina<sup>b</sup>,  
Christina de Paula Garcia Chaves<sup>c</sup>, Edvaldo Soares de Almeida<sup>c</sup>,  
Computer Science Department - Federal University of Bahia (UFBA/UFBA)  
Federal Institute for Education, Science, and Technology of Bahia (IFBA)  
[\[email protected\]](http://www.ifba.edu.br/~neto/), [\[email protected\]](http://www.ufba.br/~matheus/), [\[email protected\]](http://www.ufba.br/~christina/), [\[email protected\]](http://www.ufba.br/~edvaldo/)

**Abstract:** Over the years, the interest in software architecture recovery has increased. Due to software product line architecture complexity, the recovery of specific line architectures is very challenging. In this paper, we present initial evidence for understanding the relationship between product line architecture and software architecture recovery. (Method) we performed a survey on the state-of-the-art.

**Keywords:** Software Architecture Recovery (SAR), Software Product Lines (SPL), Software Architecture Recovery (SAR).

**1. INTRODUCTION**

Software Architecture Recovery (SAR) is the process that retrieves the architecture of a system from its execution [1]. As defined by Koenig et al. [2] and Deneve et al. [3], SAR is the process to obtain the architecture of an implemented system, or a system in operation, to support the understanding of the running system.

The Software Product Lines (SPL) approach provides the discipline to reuse common parts of systems [4] and SPL products share a set of common features (components) and have specificiations that distinguish the application [5]. Reusing common parts of an SPL is a way to reuse common parts of different situations, processes, interfaces and activities, it is also a way to reuse best practices to pay sufficient attention to its architecture.

In this work, according to Gleison [6], on Product Line Architecture Recovery (PLA) we have the following artifacts, which describes the mandatory optional, and variable components in the SPL, and their interconnections. Moreover, the PLA must be able to support the reuse of the SPL at a high level design for the SPL, including variation points and variants (described as the variability [5]).

Journal on exception handling to understand the information about the system [7].

A first step towards addressing product line architecture recovery is to identify current methods, techniques and practices used in the field. In this paper, we present initial evidence for understanding the relationship between product line architecture and software architecture recovery from the point of view of the state-of-the-art. We present the survey results on product line architecture and software architecture recovery for the purpose of characterization with respect to gather information from the point of view of the state-of-the-art in the context of software product line research area.

Thus, the contributions of this paper are (i) review the literature on the state-of-the-art in the field of product line architecture recovery and its evolution over time, (ii) introduce the main concepts of product line architecture recovery, and (iii) present the main challenges in product line architecture research area.

This paper is organized as following: Section II presents related work, Section III details the research method. Section IV presents the outcome. In Section V, we present the main challenges in product line architecture. Finally, in Section VI, we present the concluding remarks and future work.

**Valid:** 15 January 2023 23:00, Münster, Germany  
**Copyright:** © 2015 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for sale or redistribution requires the prior written permission of the author(s) and the publisher.  
**DOI:** 10.1109/ICMCA.2015.15  
**Authorized licensed use limited to: IEEE Xplore. Downloaded on March 27, 2023 at 12:08:00 UTC from IEEE Xplore. Restrictions apply.**



**A Survey on Modeling Techniques for Formal Behavioral Verification of Software Product Lines**

Fabian Benduhn  
University of Magdeburg  
Magdeburg, Germany  
[fabian.benduhn@ovgu.de](mailto:fabian.benduhn@ovgu.de)

Thomas Thüm  
University of Magdeburg  
Magdeburg, Germany  
[tthuem@ovgu.de](mailto:tthuem@ovgu.de)

Malte Leich  
University of Magdeburg  
Magdeburg, Germany  
[malte.leich@ovgu.de](mailto:malte.leich@ovgu.de)

Gunter Saake  
University of Magdeburg  
Magdeburg, Germany  
[sakke@ovgu.de](mailto:sakke@ovgu.de)

**ABSTRACT**

As software product lines are increasingly used for safety-critical systems, researchers have adopted formal verification techniques to verify the correctness of the product lines. The main challenge is the lack of guidelines for verification recovery in SPL domain. SPL can fully benefit from community and variability identification to support the reuse of verification models. This survey presents the state-of-the-art in modeling techniques for formal behavioral verification of software product lines. We discuss the challenges in the reuse of existing modeling techniques. We survey existing approaches as a first step towards the reuse of existing modeling techniques for formal behavioral verification of software product lines. We illustrate the approaches

Software product lines are increasingly used for the development of safety-critical and mission-critical systems in which software engineers may have consequences that cannot be tolerated [34, 8]. For this reason, formal verification techniques have been applied to ensure the correctness of the product lines. The reuse of existing formal verification techniques has emerged as an approach to establish correctness of the product lines. This survey presents the state-of-the-art in modeling techniques to re-use the accompanying modeling techniques for product lines. We illustrate the approaches

In general, a verification technique consists of a formalism to model

**Keywords**  
Software Product Lines, Variability, Survey, Modeling, Verification

**1. INTRODUCTION**

Software product lines often developed in a large variety of variants to meet the requirements of different customers. Software product line engineering is a discipline of software development that aims to reuse common parts of a family of development artifacts are developed simultaneously [16, 18]. Such products are called software product lines (SPLs). A family is a collection of variants [1]. A feature is a variable characteristic of a software product [11].

Hence, under digital or hard copy of all or part of this work is granted on the understanding it is "permitted" without prior permission or license to copy, store in electronic form and to transmit to others, either free of charge or on payment of a fee. Reproductive permission is granted on payment of a fee. Reproductive permission from Ponsen & Welmers.

**2. BACKGROUND**

Software product line engineering, a significant part of the modeling techniques for product lines are based on transition systems [13]. Thus, we give a brief introduction to transition systems.

# Solution Space Architecture & Design

Author & Venue	Title
Allian et al. ('19)	Evaluating variability at the software architecture level: An Overview
Beohar et al. ('16)	Basic behavioral models for software product lines: Expressiveness and testing pre-orders
Lima and Chavez ('16)	A systematic review on metamodels to support product line architecture design
Neto et al. ('15)	Initial evidence for understanding the relationship between product line architecture and software architecture recovery
Benduhn et al. ('15)	A Survey on Modeling Techniques for Formal Behavioral Verification of Software Product Lines
Groher and Weinreich ('15)	Variability support in architecture knowledge management approaches: A systematic literature review
Stephan and Cordy ('13)	A Survey of Model Comparison Approaches and Applications
Souza et al. ('08)	Evaluating domain design approaches using systematic review
Cabrero et al. ('07)	Understanding product lines through design patterns

# Solution Space Implementation

**Conference Article**

Circulation in Computer Science  
International Conference on Engineering, Computing & Information Technology (CECIT 2017), pp-7-13

## Model-Driven Approaches for Software Product Lines: A Systematic Literature Review

**Kinza Zahra**  
National University of  
Science and Technology  
(NUST), H-12,  
Islamabad, Pakistan

**Faroque Azam**  
National University of  
Science and Technology  
(NUST), H-12,  
Islamabad, Pakistan

**Fauzia Ilyas**  
National University of  
Science and Technology  
(NUST), H-12,  
Islamabad, Pakistan

**Abdul Wahab Muzafer**  
National University of  
Science & Technology  
(NUST), H-12,  
Islamabad, Pakistan

**ABSTRACT**

The quality in software product lines (SPLs) is believed to be more important than a single software product for higher reusability. Therefore, the researchers frequently try to produce better quality SPLs. Model-driven approaches for Design Architectures (MDA) is well-known approach which is considered as a standard for producing SPLs [1]. This paper discusses the systematic literature review of 31 studies regarding model-driven architecture applied to SPLs.

There benefits, if model structure providers like model repository, reuse manager, and reuse manager for reuse implementation of software systems. Model driven architecture, Software indicators, or reusable Design Architectures are the main components of MDA. The main purpose of MDA is to reuse existing code such as MME, XMI, MDSL, or MELA. In spite of the positive applications, all of these above

**Zahra et al.  
2017  
31 studies regarding model-  
driven architecture applied to  
SPLs**

engineering fields have been used to develop a defense related product [15].

Silence system which has categorized into software products, reuse process, and various types of characteristics providing explicit requirements [2]. Software products are developed by reuse of existing code and reuse process, product formation with pair of reuse and reuse components. Software product lines present a method through which reuse of existing code can be used to facilitate fundamental to create software product family [1].

Varieties modeling tools and methods have been proposed to support reuse of existing code for developing different software products. Several dialects were defined in the perspective of reuse of existing code, reuse of specific perspective with the aim of facilitating and constituting a rational view of the system and, consequently, to support reuse of existing code [16]. In the last two decades, this field has been increasing of interest because it has emerged as an vital element in indicate engineering practice. Reuse of all

engineering fields have been used to develop a defense related product [15].

Silence system which has categorized into software products, reuse process, and various types of characteristics providing explicit requirements [2]. Software products are developed by reuse of existing code and reuse process, product formation with pair of reuse and reuse components. Software product lines present a method through which reuse of existing code can be used to facilitate fundamental to create software product family [1].

Varieties modeling tools and methods have been proposed to support reuse of existing code for developing different software products. Several dialects were defined in the perspective of reuse of existing code, reuse of specific perspective with the aim of facilitating and constituting a rational view of the system and, consequently, to support reuse of existing code [16]. In the last two decades, this field has been increasing of interest because it has emerged as an vital element in indicate engineering practice. Reuse of all

```

graph TD
    SS[Silence System] --> RP[Reuse Process]
    RP --> MDA[MDA Tools]
    subgraph SP [Software Product Selection]
        SS
        RP
        MDA
    end
    subgraph MDA [MDA Tools]
        MDA
        MME[MME]
        XMI[XMI]
        MDSL[MDSL]
        MELA[MELA]
        MREU[MDA Reuse Environment]
        MREU --- MREU_label[MDA Reuse Environment]
    end

```

**Fig: Flow Diagram**

Copyright © 2017, Zahra et al. This is an open access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction is permitted, and given to the original author(s) and to others in accordance with the specified terms.

## Leveraging Software Product Lines Engineering in the Development of External DSLs: A Systematic Literature Review

David Méndez-Aranda (✉), José A. Galindo, Thomas Degruy, Renato Condeiros, Renato Renday

ESADE/SEDIKA and University of Remote I. Pumaré

---

### Abstract

The use of domain-specific languages (DSLs) has become a successful technique in the development of complex systems. Consequently, nowadays we can find a large variety of DSLs for diverse purposes. However, and although DSLs have been developed for the reuse of their code, they are usually developed as isolated, particular modeling patterns -such as state machines or petri nets- used for several purposes. In this scenario, the studies for the reuse of a language are not yet widespread, nor are the reuse techniques adopted among existing DSLs [1].

Mendez et al.  
2016  
Collect and sort the literature on  
language product line engineering

This research is focused to this need as an alternative to improve software evolution by reusing domain concepts, that hiding fine-grained implementation details and favoring the participation of domain experts in the software development process [2].

For this, an approach based on what representation is localized in a well-defined domain, and which provide the abstraction (e.g., business contracts) intended to describe certain aspect of a system under construction [3]. Naturally, the adoption of such a language-oriented vision relies on the availability of the DSLs needed to describe the diverse aspects of the system [4]; consequently, nowadays there is a large variety of DSLs available for reuse [5]. We can find, for example, DSLs to build graphical user interfaces [6], or to specify security policies [7].

Although each of the existing DSLs is unique and has been developed for a precise purpose, not all the modeling patterns are suitable for reuse. This is due to the fact that some of them are specific to a field, providing similar language constructs [8]. A possible explanation to such phenomena is the recurrent use of certain modeling patterns that, with proper adaptations, are suitable for several purposes. Consider, for

✉ Email address: daniel.mendez@esaude.es (David Méndez-Aranda); jgalindo@esaude.es (José A. Galindo); thomas.degruy@esaude.es (Thomas Degruy); renato.condeiros@esaude.es (Renato Condeiros); renato.renday@esaude.es (Renato Renday).

Preprint submitted to Journal of Computer Languages, Systems and Structures

September 27, 2016

# Solution Space Implementation

Author & Venue	Title
Zahra et al. ('18)	Model-Driven Approaches for Software Product Lines: A Systematic Literature Review
Ternava and Collet ('17)	On the Diversity of Capturing Variability at the Implementation Level
Mendez et al. ('16)	Leveraging Software Product Lines Engineering in the development of external DSLs: A systematic literature review

## Solution Space Quality Assurance

Category	Definition	Search terms	Number of publications	Studies included
Concepts	Configurable systems, safety, security, engineering, product line engineering, configuration, survey, overview.	configurable system AND safety OR security OR engineering OR product line engineering OR configuration OR survey OR overview	65	Andy Krumer, Richard May, Jacob Krueger, Kenner et al.
Participants	None		0	
Interventions	None		0	
Outcomes	None		0	
Setting	None		0	
Study design	Surveys and overviews	Survey AND overview	1	Andy Krumer, Richard May, Jacob Krueger, Kenner et al.
Total	65		65	Andy Krumer, Richard May, Jacob Krueger, Kenner et al.

Check for updates

IET Software

Review Article


**IET** **Journal of**  
 The Institution of  
 Engineering and Technology

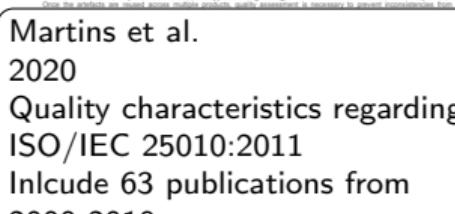
## Evolution of quality assessment in SPL: a systematic mapping

Luana Almeida Martins<sup>1</sup>, Paulo Afonso Júnior<sup>1</sup>, André Pimenta Freire<sup>1</sup>, Helor Costa<sup>1,2</sup>

<sup>1</sup>Universidade Federal de São Carlos, Federal University of Paraná - UFLA, Lages - MS, Brazil

<sup>2</sup>E-mail: helor@ufla.br

**Abstract:** Software product line (SPL) is one of the most recent and effective reuse approaches. SPL derives several products from the core assets. SPL engineering includes two processes: domain engineering, which identifies the common and variable requirements; and reuse engineering, which creates specific products. In the reuse engineering process, once the products are reused across multiple products, quality assessment is necessary to prevent inconsistencies from



Quality assessment and metrics are used for the SPL quality evaluation (e.g., ISO/IEC 25010:2011 [4]). Software metrics can be used for assessing the quality of software characteristics and for their reuse in the reuse engineering process and improving the software quality [4]. Over the last five years, many studies have been conducted to evaluate the quality of SPLs. This study presents the result of a set of well-defined quality studies.

In this paper, we present a systematic mapping of the quality characteristics used in the SPL evaluation. The objective is to provide a better understanding of the quality characteristics and metrics used by researchers and practitioners in their reuse engineering process. We performed a systematic mapping from 2000 to 2019 and analyzed these data by following a three-step process: (i) identification of relevant studies; (ii) definition of the (i) research question (RQ); (iii) search protocol and selection of papers; (iv) data extraction and classification; (v) data synthesis and (vi) reporting.

Analyzing the results of this systematic mapping, we identify the main quality characteristics used in the reuse engineering process regarding its software properties and three quality characteristics

### 2 Background and related work

This section presents the background information on SPL and quality assessment. The first part of this section presents what SPL is and to understand the differences between SPL and traditional software engineering. The second part presents what quality assessment is and the differences in perform an SPL quality assessment.

### 2.1 Software product line

SPL approach generates software products belonging to a specific domain based on the instantiation of common (modularized) and variable assets [1]. The reuse engineering process of SPLs is to reuse SPL products and establish standard behavior among these products. The reuse engineering process is divided into the behavioral of the products and create specific products.

Since SPLs are based on systematic reuse of software artifacts, quality assessment is a key factor in the reuse engineering process, first spreading in all of its products [4]. Therefore, in order to reuse the products, quality assessment must be done. SPL development SPL engineering is divided into two processes for

© IET 2020, Vol. 14 No. 4, pp. 572-587

© The Institution of Engineering and Technology 2020

572

2016-42th Euromech Colloquium on Software Engineering and Advanced Applications

# Solution Space Quality Assurance

Author & Venue	Title
Kenner et al. ('21)	Safety, Security, and Configurable Software Systems: A Systematic Mapping Study
Martins et al. ('20)	Evolution of quality assessment in SPL: a systematic mapping
Baumgart and Fröberg ('16)	Functional Safety in Product Lines - A Systematic Mapping Study
Bezerra et al. ('14)	Measures for Quality Evaluation of Feature Models
Soares et al. ('14)	Analysis of non-functional properties in software product lines: A systematic review
Montagud et al. ('12)	A systematic review of quality attributes and measures for software product lines
Mylärniemi et al. ('12)	A systematically conducted literature review: Quality attribute variability in software product lines
Montagud and Abrahao ('09)	Gathering Current Knowledge about Quality Evaluation in Software Product Lines
Etxeberria et al. ('08)	Quality aware Software Product Line Engineering
Calder et al. ('03)	Feature interaction: a critical review and considered forecast

# Solution Space Quality Assurance - Static Analysis

**Applications of #SAT Solvers on Feature Models**

Chico Sundermann  
University of Ulm  
  
Michael Nielske  
TU Braunschweig  
  
Tobias Heß  
University of Ulm  
  
Paul Maximilian Bittner  
University of Ulm  
  
Thomas Thüm  
University of Ulm  
  
Ina Schaefer  
TU Braunschweig

**ABSTRACT**  
Product lines are ubiquitous for managing variability options. The variability of product lines can typically be modeled by a feature model. Analyzing a feature model gives insight into some aspects, such as the validity of a configuration of features. Beyond analyzing individual configurations, it is also interesting to analyze all valid configurations which proves highly inefficient when using a traditional approach. In this paper, we propose a method of obtaining variable assignments of a propositional formula and counting satisfying assignments of a propositional formula in linear time. We evaluate our approach on 11 feature models from available benchmarks.

**Sundermann et al.**  
**2021**  
**Present and introduce applications for #Sat Solvers**

**A Formal Framework of Software Product Line Analyses**

THIAGO CASTRO, Systems Development Center—Brazilian Army  
LEOPOLDO TEIXEIRA, Federal University of Paraná  
VANDER ALVES, Universidade de São Paulo  
SVEN APEL, Saarland University, Saarland Informatics Campus  
MAXIME CORDY, University of Luxembourg  
ROHIT GHAYI, Federal University of Campinas Grande

**ABSTRACT**  
A number of product line analysis approaches lift analyses such as type checking, model checking, and theorem proving from the level of single programs to the level of product lines. These approaches share concepts and mechanisms that suggest an unexplored potential for reuse of key analysis steps and properties, implementation, and verification efforts. Despite the availability of taxonomies synthesizing such approaches,

**Castro et al.**  
**2021**  
**Based on Thüm et al.(CSUR'14)**  
**Formal framework for SPL Analyses**



# Solution Space Quality Assurance - Static Analysis

Author & Venue	Title
Sundermann et al. (VaMoS'21)	Applications of #SAT Solvers on Feature Models
Castro et al. (TOSEM'21)	A Formal Framework of Software Product Line Analyses
Thüm et al. ('14)	A Classification and Survey of Analysis Strategies for Software Product Lines
von Rhein et al. ('13)	The PLA Model: On the Combination of Product-Line Analyses

# Solution Space Quality Assurance - Dynamic Analysis - Sampling

## A Classification of Product Sampling for Software Product Lines

Mahsa Varchosaz<sup>1</sup>, Mstafa Al-Hajaji<sup>2</sup>, Thomas Thün<sup>3</sup>, Tobias Ranga<sup>3</sup>, Mohammad Reza Movahedi<sup>4,5</sup>, and Ira Schaefer<sup>2</sup>

<sup>1</sup>Hannover University, Berliner Platz 3, Hannover, Germany  
<sup>2</sup>TU Braunschweig, Germany  
<sup>3</sup>University of Lorraine, France  
<sup>4</sup>Software Engineering Institute, USA  
<sup>5</sup>TU Braunschweig, Germany

### ABSTRACT

The analysis of software product lines is challenging due to the potentially large number of products, which grow exponentially in complexity over time. In order to support the quality assurance of product lines, one may resort to product sampling techniques [32] that provide a subset of all valid products. These products are supposed to collectively cover the behavior of the product line. In this paper, we present a classification of sampling techniques used to avoid exhaustive testing, which is often infeasible. In this paper, we propose a classification for product sampling techniques.

### Varshosaz et al. 2018

### Expert survey

### Classification for product sampling techniques

### Overview on existing tools

Development by considering the commonalities and variations among the products is an NP-hard problem [10].

Testing the quality of software product lines is however far more challenging due to the sheer number of possible products, which makes exhaustive testing and analysis practically impossible. To

Dynamically test all configurations of an entire product line is a well-known problem in software engineering. In this paper, we propose a classification of sampling techniques for product lines. We also present a survey of existing tools for product line testing. The classification is based on the type of information that is used to generate the sample. The survey is based on the classification and provides an overview of the state-of-the-art. We also present a comparison of the proposed classification with other classifications in the literature. Finally, we discuss the challenges and opportunities for future research.

1 Department of Computer Science, University of Twente, Enschede, The Netherlands; 2 Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, Enschede, The Netherlands; 3 Chair of Software Testing Intelligent Lab (STEL), Department of Computer Science, University of Twente, Enschede, The Netherlands; 4 Chair of Software Testing Intelligent Lab (STEL), Department of Computer Science, University of Twente, Enschede, The Netherlands; 5 Chair of Software Testing Intelligent Lab (STEL), Department of Computer Science, University of Twente, Enschede, The Netherlands

✉ m.varchosaz@utwente.nl (Mahsa Varchosaz); mstafa.al-hajaji@utwente.nl (Mstafa Al-Hajaji); thomas.thun@utwente.nl (Thomas Thün); tobias.ranga@utwente.nl (Tobias Ranga); mrv@cs.tu-bs.de (Mohammad Reza Movahedi); schaefer@informatik.tu-bs.de (Ira Schaefer)

Received 15 January 2018; revised 15 April 2018; accepted 15 May 2018. This work was partially funded by the European Union's Horizon 2020 research and innovation program under grant agreement No 723520 (Project PLATINUM).

© 2018 IEEE/ACM 38th IEEE International Conference on Software Engineering. This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ICSE.2018.00020, ISSN 1084-5623.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ICSE.2018.00020, ISSN 1084-5623.

## Constrained Interaction Testing: A Systematic Literature Study

Bostan S. Ahmed<sup>1</sup>, Karan Z. Zantil<sup>2</sup>, Wasif Afzal<sup>3</sup>, and Miroslav Burša<sup>4</sup>

<sup>1</sup>Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, Enschede, The Netherlands  
<sup>2</sup>Department of Computer Science, University of Twente, Enschede, The Netherlands  
<sup>3</sup>Department of Computer Science, University of Twente, Enschede, The Netherlands  
<sup>4</sup>Department of Computer Science, University of Twente, Enschede, The Netherlands

### Abstract

Interaction testing can be used to effectively detect faults that are otherwise undetectable effectively. Interaction testing has been given other alternative names such as configuration testing, constrained interaction testing (CIT), and  $t \times n$ -wise, or  $n \times n$ -wise testing (where  $t$  is to indicate the interaction strength). However, throughout this paper, the term "interaction testing" is used as a representative

### Ahmed et al. 2017

### 103 primary studies

### Classification of the constrained interaction testing research work in four different categories

1 Department of Computer Science, University of Twente, Enschede, The Netherlands; 2 Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, Enschede, The Netherlands; 3 Chair of Software Testing Intelligent Lab (STEL), Department of Computer Science, University of Twente, Enschede, The Netherlands; 4 Chair of Software Testing Intelligent Lab (STEL), Department of Computer Science, University of Twente, Enschede, The Netherlands

✉ b.s.ahmed@utwente.nl (Bostan S. Ahmed); k.zantil@utwente.nl (Karan Z. Zantil); wafiz.afzal@utwente.nl (Wasif Afzal); m.bursha@utwente.nl (Miroslav Burša)

Received 15 June 2016; revised 15 December 2016; accepted 15 February 2017. This work was partially funded by the European Union's Horizon 2020 research and innovation program under grant agreement No 723520 (Project PLATINUM).

© 2017 IEEE/ACM 37th IEEE International Conference on Software Engineering. This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ICSE.2017.00020, ISSN 1084-5623.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ICSE.2016.00020, ISSN 1084-5623.

## A Comparison of 10 Sampling Algorithms for Configurable Systems

Fábio Medeiros<sup>1</sup>, Christian Kistner<sup>2</sup>, and Mário Rioseco<sup>3</sup>

<sup>1</sup>Federal University of Campina Grande, Paraíba, Brazil  
<sup>2</sup>Carnegie Mellon University, Pittsburgh, Pennsylvania, USA  
<sup>3</sup>Federal University of Alagoas, Maceió, Alagoas, Brazil

### Abstract

Almost every software system provides configuration options to tailor the system to the target platform and application needs found in highly configurable systems, such as the Linux Kernel, Gnu, Bind9, and Apache [2, 3, 14, 20, 22, 23, 34]. Although researchers have proposed approaches to automatically generate test cases for these systems [1, 2, 3, 4, 5, 6, 7],

### Medeiros et al. 2016

### Compare sampling algorithms by, e.g., number of faults they find, or size of sample set

1 Federal University of Campina Grande, Paraíba, Brazil; 2 Carnegie Mellon University, Pittsburgh, PA, USA; 3 Federal University of Alagoas, Maceió, Alagoas, Brazil

✉ fmedeiros@fatecpg.br (Fábio Medeiros); ckistner@cs.cmu.edu (Christian Kistner); mario.rios@ual.br (Mário Rioseco)

Received 15 March 2015; revised 15 October 2015; accepted 15 November 2015. This work was partially funded by the Brazilian National Research Council (CNPq) under grants 304321/2013-0 and 304321/2013-0.

© 2016 IEEE/ACM 36th IEEE International Conference on Software Engineering. This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ICSE.2016.00020, ISSN 1084-5623.

Rahel Arens

4ECTS – FOSD 2023 – 3. Solution Space

57

**Solution Space** Quality Assurance - Dynamic Analysis - Sampling

Journal of Computer Science

Literature Review

## Combinatorial Interaction Testing of Software Product Lines: A Mapping Study

**Mohd Zamri Sulaiman,<sup>1</sup> Abu Bakar Md Sulaiman,<sup>1</sup> Abdul Aziz Abdul Ghani<sup>2</sup> and Yolani Bahrami<sup>3</sup>**

<sup>1</sup>*Faculty of Computer Science and Information Technology, Universiti Tunku Abdul Rahman (UTAR), Malaysia*

<sup>2</sup>*Faculty of Computer Science and Technology, Universiti Malaysia Pahang (UMP), Malaysia*

Article history:

Received: 22-06-2016

Revised: 17-08-2016

Accepted: 24-08-2016

Corresponding Author:

Mohd Zamri Sulaiman

**Abstract:** Software Product Line (SPL) is a software engineering paradigm that is inspired by the concept of modularity of system design. It is formulated for different software product. Complete testing on entire SPL is a challenging task as it requires testing all possible configurations that can be produced, configured using a subset or all possible factors in the SPL. This paper reports a Systematic Mapping Study (SMS) of relevant primary

**Sahid et al.**  
2016  
44 primary studies  
**Combinatorial interaction testing techniques for SPLs**

have been developed based on the same kind of input and output types. The similarity in the program structure due to identical user requirements also contribute to the reuse of code. In order to reduce the cost of this scenario and based on the benefit of reuse principle, Software Product Line (SPL) has been developed as a software engineering paradigm. This paradigm is inspired by product line approach Clement and Northrop (2002) and Cusumano et al. (2000).

"A Software Product Line is a set of software intensive systems sharing a common, managed set of basic features and only the specificities of a particular market segment and that are developed around a common set of core assets in a prescribed manner" (Clement and Northrop, 2002).

reduced over time via various researches (Witte and Lee, 1999; Cusumano and Hwang, 2000; Thiel and Peruzzi, 2000). Two main processes in SPL are Domain Engineering phase and Application Engineering phase. In the Domain Engineering phase, few crucial activities are performed which includes, but not limited to, identifying system requirements, identifying reuse opportunities and variabilities. In the Application Engineering phase, the products are configured and generated by utilizing configuration management and reuse technologies. The most important principle that spans throughout the software engineering process is modularity based on the concept of genericity and variability.

Apart from similarities of functional properties among software, the differences of functionalities are

Science  
Publications

© 2016 Mohd Zamri Sulaiman, Abu Bakar Md Sulaiman, Abdul Aziz Abdul Ghani and Yolani Bahrami. This open access article is distributed under a Creative Commons license (CC-BY-NC-ND).

# A First Systematic Mapping Study on Combinatorial Interaction Testing for Software Product Lines

# Solution Space Quality Assurance - Dynamic Analysis - Sampling

Author & Venue	Title
Varshosaz et al. (SPLC'18)	A Classification of Product Sampling for Software Product Lines
Ahmed et al. (AZAB'17)	Constrained Interaction Testing: A Systematic Literature Study
Medeiros et al. (ICSE'16)	A Comparison of 10 Sampling Algorithms for Configurable Systems
Sahid et al. (JcS'16)	Combinatorial interaction testing of software product lines: A mapping study
Lopez-Herrejon et al. (IWCT'15)	A First Systematic Mapping Study on Combinatorial Interaction Testing for Software Product Lines

# Solution Space Quality Assurance - Dynamic Analysis

Author & Venue	Title
Sulaiman et al. ('22)	Classification Trends Taxonomy of Modelbased Testing for Software Product Line: A Systematic Literature Review
Ferreira et al. ('21)	Evaluating T-wise Testing Strategies in a Community-wide Dataset of Configurable Software Systems
Petry et al. ('20)	Model-based testing of software product lines: Mapping study and research roadmap
Ferreira et al. ('20)	On the Proposal and Evaluation of a Test-enriched Dataset for Configurable Systems
Ferreira et al. ('19)	Testing Tools for Configurable Software Systems: A Review-based Empirical Study
Ruiz et al. ('19)	A general approach to Software Product Line testing
Sahak et al. ('17)	Evaluation of Software Product Line Test Case Prioritization Technique
Machado et al. ('14)	On Strategies for Testing Software Product Lines: A Systematic Literature Review

# Solution Space Quality Assurance - Dynamic Analysis

Author & Venue	Title
Neto et al. ('12)	Mapping study on software product lines testing tools
Lee et al. ('12)	A Survey on Software Product Line Testing
Neto et al. ('11)	A systematic mapping study of software product lines testing
Engström and Runeson ('11)	Software product line testing – A systematic mapping study
Johansen et al. ('11)	A survey of empirics of strategies for software product line testing
Lamancha et al. ('09)	Software product line testing - a systematic review
Tevanlinna et al. ('04)	Product Family Testing – a Survey



# Solution Space Reengineering

Author & Venue	Title
Strüber et al. ('19)	Facing the Truth: Benchmarking the Techniques for the Evolution of Variant-Rich Systems
Sinkala et al. ('18)	A mapping study of software architecture recovery for software product lines
Li et al. ('17)	Reverse engineering variability from natural language documents: A systematic literature review
Lopez-Herrejon et al. ('15)	Genetic Improvement for Software Product Lines: An Overview and a Roadmap
Vale et al. ('14)	Bad smells in software product lines: A systematic review
Fenske et al. ('14)	A Taxonomy of Software Product Line Reengineering
Laguna and Crespo ('13)	A systematic mapping study on software product line evolution: From legacy system reengineering to product line refactoring



# Solution Space Reengineering - Migration

Author & Venue	Title
Chacón-Luna et al. ('20)	Empirical software product line engineering: A systematic literature review
Assuncao et al. ('17)	Reengineering legacy applications into software product lines: A systematic mapping
Bakar et al. ('15)	Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review
Tüzün et al. ('15)	Empirical evaluation of a decision support model for adopting software product line engineering
Assunção and Vergilio ('14)	Feature Location for Software Product Line Migration: A Mapping Study
Ferreira Bastos et al. ('11)	Adopting software product lines: A systematic mapping study

# Solution Space Reengineering - Feature Location & Traceability

2021 International Conference on Innovation and Intelligence for Information, Computing, and Technologies (IICIT)

## Reviewing Dynamic Feature Location Techniques: Basic Elements and Challenges

Faisal Buzaid  
Department of Computer Science  
University of Bahrain  
Sakhib, Bahrain  
202011222@uobmail.bh

Mustafa Hammad  
Department of Computer Science  
University of Bahrain  
Sakhib, Bahrain  
mhammad@uob.edu.bh

Abstract—Dynamic Feature Location Techniques (DFLTs) work to provide the software maintainability artifacts which are known as software features, in the relevant source code based on

to cause errors. Therefore, creating a self-adapting solution to overcome DFLTs artifacts can reduce the required effort to analyze the execution trace.

## Buzaid and Hammad

2021

## Dynamic feature location techniques

Feature Location (FL):  
Fault Localization, Dynamic Analysis, Evolution Trace, Program Comprehension, Software Engineering

### 1. INTRODUCTION

Dynamic Feature Location Techniques (DFLTs) are usually used through software maintenance processes to comprehend the system's behavior. The main purpose of the analysis of DFLTs may include static, visual or, as a combination between more than one type of analysis to the Feature Location (FL). Monitoring the system at runtime is the main process that is used to analyze the type of DFLTs. The main purpose of DFLTs is to map a software feature to its relevant source code, relying on the execution traces that are generated while the feature is being tested by the system.

Exploring the generated execution trace manually by developers to find and locate source code that belongs to the established software functionality is time-consuming and tiring

In other words, to clarify the DFLTs field, a precise DFLTs review paper [1] tries to use the term "feature location" for the DFLTs. The nature of such broad domain of FLs makes a clear need for the researchers to focus on the investigation of the feature dynamic criteria. It is important for the researchers to understand that DFLTs are a specific type of FLs, which are used to extract the required artifacts from the relevant source code to carry out any research on DFLTs.

This paper is categorized into two parts. First, this work reviews dynamic feature location strategies and their main applications in the field of DFLTs, including [1]-[6]. The main distinction between this survey paper and other studies are based on the type of analysis. Only dynamic analysis is used in this paper to analyze the DFLTs. Second, this paper is one aspect of dynamic analysis to profile the actual behavior of a running system and then taking into account for every study.

© 2021 IEEE. Personal use of this material is permitted. By agreeing to access this work, you accept the terms of the IEEE Copyright License (http://ieeexplore.ieee.org/about/about\_copyright.html).

## A Literature Review and Comparison of Three Feature Location Techniques using ArgoUML-SPL

Daniel Cruz  
Universidade Federal de São Carlos

Eduardo Pugnatiello  
Universidade Federal de Minas Gerais  
danielcruz@uol.com.br

Javier Martínez  
TecMilenio  
jpmartinez@tec.mx

ABSTRACT  
Over the last decades, the adoption of Software Product Line (SPL) approaches for software development has increased. As a result, it is possible to reuse some single product from a family of systems to enhance the reusability of systems (e.g., refactoring by plugins, aspects, among other forms of modularization), as long as

the reuse analysis does respect its domain to identify and

Cruz et al.  
2019  
Three feature location techniques: static, dynamic, and textual  
From 26 publications

the goal to produce variations of an SPL by maintaining selected features. In this paper, we propose a classification of the state-of-the-art of the feature location activity in its facilitation the identification of code artifacts that implemented each feature [1], and also its high-level classification into three categories: static, dynamic, and textual. In order to better reflect some of the characteristics of this work, we argue that the first part of this work is related to the literature review, and the second part, we focus on comparing technical information retrieval techniques. This kind of methods has been largely applied to the SPL reuse analysis [2]. In this paper, we present a classification of a previous work [3] investigating the benefits of information retrieval techniques for reuse analysis. We also present a comparison of the reuse analysis methods for reuse analysis [4] and reuse analysis for feature location [5].

Keywords—feature location, dynamic analysis, evolution trace, program comprehension, software engineering

## The State of Empirical Evaluation in Static Feature Location

ABDUL RAZZAQ, ASANKA WASALA, CHRIS EXTON, and JIM BUCKLEY,  
The Irish Software Research Centre, L230 and CSIS, University of Limerick

Feature location (FL) is the task of finding the source code that implements a specific, user-observable functionality in a software system. It plays a key role in many software maintenance tasks and a wide variety of Feature Location Techniques (FLTs), which rely on source code structure or textual analysis, have been proposed by researchers. As FLTs evolve and new novel FLTs are introduced, it is important to perform empirical evaluations to investigate "What is the best". The lack of transparency of the empirical evaluations suggests that performing each comparison would be an arduous process, based on the large number of techniques to be compared, the heterogeneous nature of the empirical designs, and the lack of transparency in the literature. This article presents a systematic review of 170 FLT articles, published between the years 2000 and 2018, to evaluate the state-of-the-art in FLTs. The results show that there is no consensus on what is the best, in that they are evaluating different types of novel FLTs, evaluated through standard empirical methods but, of those, only 9% use baseline technique(s) in their evaluations to allow cross comparison.

## Razzaq et al.

2018

170 publications from between  
2000-2015

<https://doi.org/10.1145/3208998>

Supported in part by Science Foundation Ireland grant 13/RC/2094

Author addresses: A. Razzaq, University of Limerick, L230-022, Limerick, Ireland; email: abdulrazaq@ieee.org;

A. Wasala, University of Limerick, L230-029, Limerick, Ireland; email: asankaw@uol.com.br; C. Exton, University of Limerick, C10-008, Limerick, Ireland; email: chrisexton@uol.com.br; J. Buckley, University of Limerick, L230-001, Limerick, Ireland; email: jimbuckley@uol.com.br

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

© 2018 Association for Computing Machinery.

978-1-4503-5612-4/18/06 ©ACM 018. \$15.00

<https://doi.org/10.1145/3208998>

# Solution Space Reengineering - Feature Location & Traceability

Author & Venue	Title
Buzaid and Hammad (3ICT'21)	Reviewing Dynamic Feature Location Techniques: Basic Elements and Challenges
Cruz et al. (VaMoS'19)	A Literature Review and Comparison of Three Feature Location Techniques using ArgoUML-SPL
Razzaq et al. (TOSEM'18)	The State of Empirical Evaluation in Static Feature Location
Vale et al. (INST'17)	Software product lines traceability: A systematic mapping study
Dit et al. (JSEP'13)	Feature Location in Source Code: A Taxonomy and Survey
Rubin and Chechik ('13)	A Survey of Feature Location Techniques

# What next?

## Applications of Product Line Techniques

The screenshot shows a research paper titled "Intelligent software product line configurations: A literature review" by Umra Alzal et al. The paper is published in the journal "Computer Standards & Interfaces" (Volume 48, 2016, pages 30–44). The journal is published by Elsevier. The abstract discusses the evolution of software product line (SPL) techniques and their application in business environments. It highlights the potential of artificial intelligence (AI) to address inconsistencies in SPL configurations. The paper concludes that AI can help in identifying and resolving such inconsistencies.

**Where should I go?**

## Configuration

# What next?

Journal of Computer Science

Literature Reviews

## Combinatorial Interaction Testing of Software Product Lines: A Mapping Study

<sup>1</sup>Mohd Zaini Salai, <sup>2</sup>Aba Bakar Md Sulsa, <sup>3</sup>Abdul Aziz Abdul Ghani and <sup>4</sup>Naimi Baharom

<sup>1</sup>Faculty of Computer Science and Information Technology, Universiti Tawiah Islam Malaysia (UTIM), Malaysia  
<sup>2</sup>Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM), Malaysia

Article history:

Received: 22-01-2016

Revised: 10-03-2016

Accepted: 19-03-2016

Corresponding Author:

Mohd Zaini Salai,

Faculty of Computer Science

and Information Technology,

Universiti Tawiah Islam

Malaysia.

E-mail: zainisalai@utim.edu.my

**Abstract:** Software Product Line (SPL) is a software engineering paradigm that is inspired by the concept of reusability of common features, formulated for different software products. Complete testing on entire SPL is a challenging task due to the large number of configurations that need to be produced, configured using a subset of all possible features in the SPL. This paper reports a Systematic Mapping Study (SMS) of relevant primary literature on the application of Combinatorial Interaction Testing (CIT) for SPL. In CIT, one has to construct a covering array, which is a set of configurations having valid feature interactions and every feature interaction occurs at least once in the array. This is also known as *cov-test*. By following the systematic mapping study guidelines, we have selected and filtered relevant studies for review. The main goal of CIT is to reduce the time required for review. The main idea of CIT is to generate a covering array using SPL test cases and an greedy algorithm followed by meta-heuristic algorithm. The motivation of SPL testing is to maximize the feature interaction problem, in which the product variants are considered to be independent from each other. An analytical approach, while some employed test configuration prioritization approach. Numerous works have been reported, but only few works managed to demonstrate the effectiveness of primary studies only due to low strength ( $r^2$  is less than 4) of review testing.

**Keywords:** Systematic Mapping Study, Secondary Study, Combinatorial Interaction Testing, Software Product Line

### Introduction

In real world, many software products developed for various domains carry some similar functionalities. These softwares share similar functionalities due to the fact that they have been developed based on the same requirements and output types. The similarity in the internal program structure due to identical user requirements allows us to reuse the code in the development process [1].

In the early 1990s, the first work on Software Product Lines (SPL) was conducted by Clements and Northrop [2]. The first work on SPL was a set of software intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment. The products are developed from a common set of core assets in a *parameterized manner* [3] (Clements and Northrop, 2002) defined SPL as follows:

"A Software Product Line (SPL) is a set of software intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment. The products are developed from a common set of core assets in a *parameterized manner*" among software, the differences of functionalities are



© 2016 Mohd Zaini Salai, Aba Bakar Md Sulsa, Abdul Aziz Abdul Ghani and Naimi Baharom. This open access article is distributed under a Creative Commons Attribution (CC-BY) 4.0 license.

## A First Systematic Mapping Study on Combinatorial Interaction Testing for Software Product Lines

Roberto E. Lopez-Herranz<sup>1\*</sup>, Stefan Fischer<sup>2</sup>, Radolf Rauter<sup>1</sup> and Alexander Egyed<sup>1</sup>

<sup>1</sup> Institute for Software Systems Engineering  
Institutes Kepler University Linz, Austria  
Email: {roberto.lopez-herranz, radolf.rauter, alexander.egyed}@jku.at  
<sup>2</sup> Software Competence Center Hagenberg, Austria  
Email: stefan.fischer@ksg.ac.at

**Abstract:** Software Product Lines (SPLs) are families of related software products that are distinguished by the set of features they provide [1]. In recent years, SPLs have become the subject of extensive research and application both in academia and industry. SPLs have been used to reduce costs, increase quality and shorten product customization and reduced time to market. Testing SPLs pose additional challenges due to the large number of configurations of product variants which make it infeasible to test every single configuration. As a result, researchers have been carrying out research on applying Combinatorial Interaction Testing (CIT) for SPL testing. In this paper we present the first systematic mapping study on the application of the techniques that have been applied to the testing of SPLs. We also report the main findings and the main challenges that have been addressed. Our goal is to identify common trends, gaps, and opportunities for further research and application.

### 1. INTRODUCTION

Software Product Lines (SPLs) are families of related systems, where the products are distinguished by the set of features they provide [1]. In recent years, SPLs have become the subject of extensive research and industry clearly aware of the significant benefits in applying SPLs [2].

From the early 1990s until 2000, the interest in SPLs

research and practice both in academia and industry clearly

increased [3, 4]. The first work on SPLs was conducted by Clements and Northrop [2].

The first work on SPL was a set of software intensive

systems sharing a common, managed set of features

that satisfy the specific needs of a particular

market segment. The products are developed from a

common set of core assets in a *parameterized manner* [3]

(Clements and Northrop, 2002) defined SPL as follows:

"A Software Product Line (SPL) is a set of software

intensive systems sharing a common, managed set of

features that satisfy the specific needs of a particular

market segment. The products are developed from a

common set of core assets in a *parameterized manner*"

among software, the differences of functionalities are

the variations of the features and their interac-

tions [1].

Combinatorial Interaction Testing (CIT) is a testing ap-

proach that models a *Feature Under Test* (FUT) as a set of factors

(choice points or parameters) each taking its values from a particular domain [5]. CIT is a well-known technique that has been successfully applied to different types of software systems [7], [8], [9]. It is particularly useful to identify interactions between features and to identify the most important parameters that can affect the behavior of the system [10].

However, variability is a key factor in SPL development [5].

Therefore, variability needs special attention [11].

For SPLs, the challenge is to generate a covering array

in a large number of different software variabilities

of the lines cannot be all tested individually.

Combinatorial Interaction Testing (CIT) is a testing ap-

proach that models a *Feature Under Test* (FUT) as a set of factors

(choice points or parameters) each taking its values from a

a particular domain [5].

CIT is a well-known technique that has been suc-

cessfully applied to different types of software

systems [7], [8], [9]. It is particularly useful to iden-

tify interactions between features and to identify the

most important parameters that can affect the be-

havior of the system [10].

However, variability is a key factor in SPL develop-

ment [5]. Therefore, variability needs special atten-

tion [11].

For SPLs, the challenge is to generate a covering ar-

ray in a large number of different software variabil-

ties of the lines cannot be all tested individually.

Combinatorial Interaction Testing (CIT) is a testing ap-

proach that models a *Feature Under Test* (FUT) as a set of factors

(choice points or parameters) each taking its values from a

a particular domain [5].

CIT is a well-known technique that has been suc-

cessfully applied to different types of software

systems [7], [8], [9]. It is particularly useful to iden-

tify interactions between features and to identify the

most important parameters that can affect the be-

havior of the system [10].

However, variability is a key factor in SPL develop-

ment [5]. Therefore, variability needs special atten-

tion [11].

For SPLs, the challenge is to generate a covering ar-

ray in a large number of different software variabil-

ties of the lines cannot be all tested individually.

Combinatorial Interaction Testing (CIT) is a testing ap-

proach that models a *Feature Under Test* (FUT) as a set of factors

(choice points or parameters) each taking its values from a

a particular domain [5].

CIT is a well-known technique that has been suc-

cessfully applied to different types of software

systems [7], [8], [9]. It is particularly useful to iden-

tify interactions between features and to identify the

most important parameters that can affect the be-

havior of the system [10].

However, variability is a key factor in SPL develop-

ment [5]. Therefore, variability needs special atten-

tion [11].

For SPLs, the challenge is to generate a covering ar-

ray in a large number of different software variabil-

ties of the lines cannot be all tested individually.

Combinatorial Interaction Testing (CIT) is a testing ap-

proach that models a *Feature Under Test* (FUT) as a set of factors

(choice points or parameters) each taking its values from a

a particular domain [5].

CIT is a well-known technique that has been suc-

cessfully applied to different types of software

systems [7], [8], [9]. It is particularly useful to iden-

tify interactions between features and to identify the

most important parameters that can affect the be-

havior of the system [10].

However, variability is a key factor in SPL develop-

ment [5]. Therefore, variability needs special atten-

tion [11].

For SPLs, the challenge is to generate a covering ar-

ray in a large number of different software variabil-

ties of the lines cannot be all tested individually.

Combinatorial Interaction Testing (CIT) is a testing ap-

proach that models a *Feature Under Test* (FUT) as a set of factors

(choice points or parameters) each taking its values from a

a particular domain [5].

CIT is a well-known technique that has been suc-

cessfully applied to different types of software

systems [7], [8], [9]. It is particularly useful to iden-

tify interactions between features and to identify the

most important parameters that can affect the be-

havior of the system [10].

However, variability is a key factor in SPL develop-

ment [5]. Therefore, variability needs special atten-

tion [11].

For SPLs, the challenge is to generate a covering ar-

ray in a large number of different software variabil-

ties of the lines cannot be all tested individually.

Combinatorial Interaction Testing (CIT) is a testing ap-

proach that models a *Feature Under Test* (FUT) as a set of factors

(choice points or parameters) each taking its values from a

a particular domain [5].

CIT is a well-known technique that has been suc-

cessfully applied to different types of software

systems [7], [8], [9]. It is particularly useful to iden-

tify interactions between features and to identify the

most important parameters that can affect the be-

havior of the system [10].

However, variability is a key factor in SPL develop-

ment [5]. Therefore, variability needs special atten-

tion [11].

For SPLs, the challenge is to generate a covering ar-

ray in a large number of different software variabil-

ties of the lines cannot be all tested individually.

Combinatorial Interaction Testing (CIT) is a testing ap-

proach that models a *Feature Under Test* (FUT) as a set of factors

(choice points or parameters) each taking its values from a

a particular domain [5].

CIT is a well-known technique that has been suc-

cessfully applied to different types of software

systems [7], [8], [9]. It is particularly useful to iden-

tify interactions between features and to identify the

most important parameters that can affect the be-

havior of the system [10].

However, variability is a key factor in SPL develop-

ment [5]. Therefore, variability needs special atten-

tion [11].

For SPLs, the challenge is to generate a covering ar-

ray in a large number of different software variabil-

ties of the lines cannot be all tested individually.

Combinatorial Interaction Testing (CIT) is a testing ap-

proach that models a *Feature Under Test* (FUT) as a set of factors

(choice points or parameters) each taking its values from a

a particular domain [5].

CIT is a well-known technique that has been suc-

cessfully applied to different types of software

systems [7], [8], [9]. It is particularly useful to iden-

tify interactions between features and to identify the

most important parameters that can affect the be-

havior of the system [10].

However, variability is a key factor in SPL develop-

ment [5]. Therefore, variability needs special atten-

tion [11].

For SPLs, the challenge is to generate a covering ar-

ray in a large number of different software variabil-

ties of the lines cannot be all tested individually.

Combinatorial Interaction Testing (CIT) is a testing ap-

proach that models a *Feature Under Test* (FUT) as a set of factors

(choice points or parameters) each taking its values from a

a particular domain [5].

CIT is a well-known technique that has been suc-

cessfully applied to different types of software

systems [7], [8], [9]. It is particularly useful to iden-

tify interactions between features and to identify the

most important parameters that can affect the be-

havior of the system [10].

However, variability is a key factor in SPL develop-

ment [5]. Therefore, variability needs special atten-

tion [11].

For SPLs, the challenge is to generate a covering ar-

ray in a large number of different software variabil-

ties of the lines cannot be all tested individually.

Combinatorial Interaction Testing (CIT) is a testing ap-

proach that models a *Feature Under Test* (FUT) as a set of factors

(choice points or parameters) each taking its values from a

a particular domain [5].

CIT is a well-known technique that has been suc-

cessfully applied to different types of software

systems [7], [8], [9]. It is particularly useful to iden-

tify interactions between features and to identify the

most important parameters that can affect the be-

havior of the system [10].

However, variability is a key factor in SPL develop-

ment [5]. Therefore, variability needs special atten-

tion [11].

For SPLs, the challenge is to generate a covering ar-

ray in a large number of different software variabil-

ties of the lines cannot be all tested individually.

Combinatorial Interaction Testing (CIT) is a testing ap-

proach that models a *Feature Under Test* (FUT) as a set of factors

(choice points or parameters) each taking its values from a

a particular domain [5].

CIT is a well-known technique that has been suc-

cessfully applied to different types of software

systems [7], [8], [9]. It is particularly useful to iden-

tify interactions between features and to identify the

most important parameters that can affect the be-

havior of the system [10].

However, variability is a key factor in SPL develop-

ment [5]. Therefore, variability needs special atten-

tion [11].

For SPLs, the challenge is to generate a covering ar-

ray in a large number of different software variabil-

ties of the lines cannot be all tested individually.

Combinatorial Interaction Testing (CIT) is a testing ap-

proach that models a *Feature Under Test* (FUT) as a set of factors

(choice points or parameters) each taking its values from a

a particular domain [5].

CIT is a well-known technique that has been suc-

cessfully applied to different types of software

systems [7], [8], [9]. It is particularly useful to iden-

tify interactions between features and to identify the

most important parameters that can affect the be-

havior of the system [10].

However, variability is a key factor in SPL develop-

ment [5]. Therefore, variability needs special atten-

tion [11].

For SPLs, the challenge is to generate a covering ar-

ray in a large number of different software variabil-

ties of the lines cannot be all tested individually.

Combinatorial Interaction Testing (CIT) is a testing ap-

proach that models a *Feature Under Test* (FUT) as a set of factors

(choice points or parameters) each taking its values from a

a particular domain [5].

CIT is a well

# What next?



And what about us?

# 4 an Ellaborating overview of Configurable software: a Tertiary Study

## 1. (Software) Product Lines

Industrial Applications

Tool Support

Evolution

Processes

### Applications of Product Line Techniques

Service-Oriented PL

Dynamic SPL

Management

## 2. Problem Space

Representation

Analysis

Configuration

## 3. Solution Space

Requirements

Architecture & Design

Implementation

Quality Assurance

Static Analysis

Dynamic Analysis

Reengineering

Migration

Feature Location & Traceability