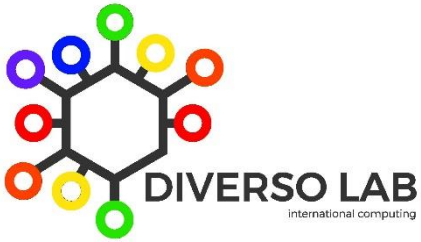
Data-intensive product lines: embracing past results and new variability challenges

Prof. David Benavides

benavides@us.es

Tokyo, SPLC 2023



https://grupo.us.es/diversolab/

@davbencue

SPLC former keynote speakers



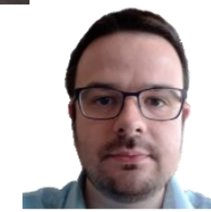












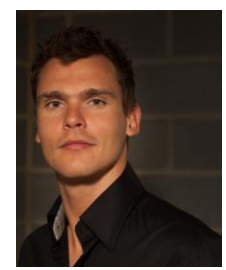














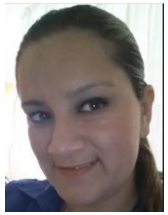



DIVERSO LAB
international computing







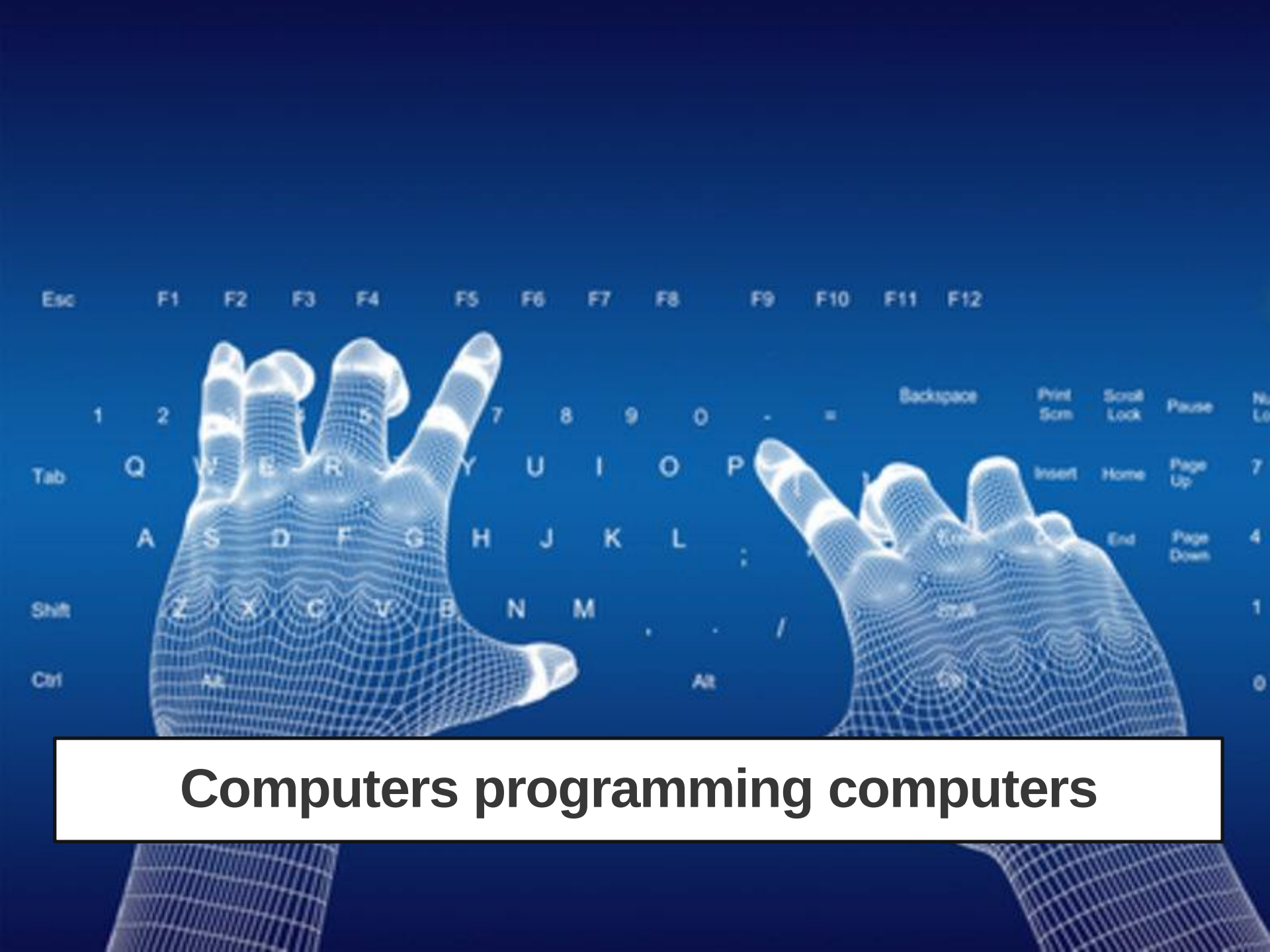




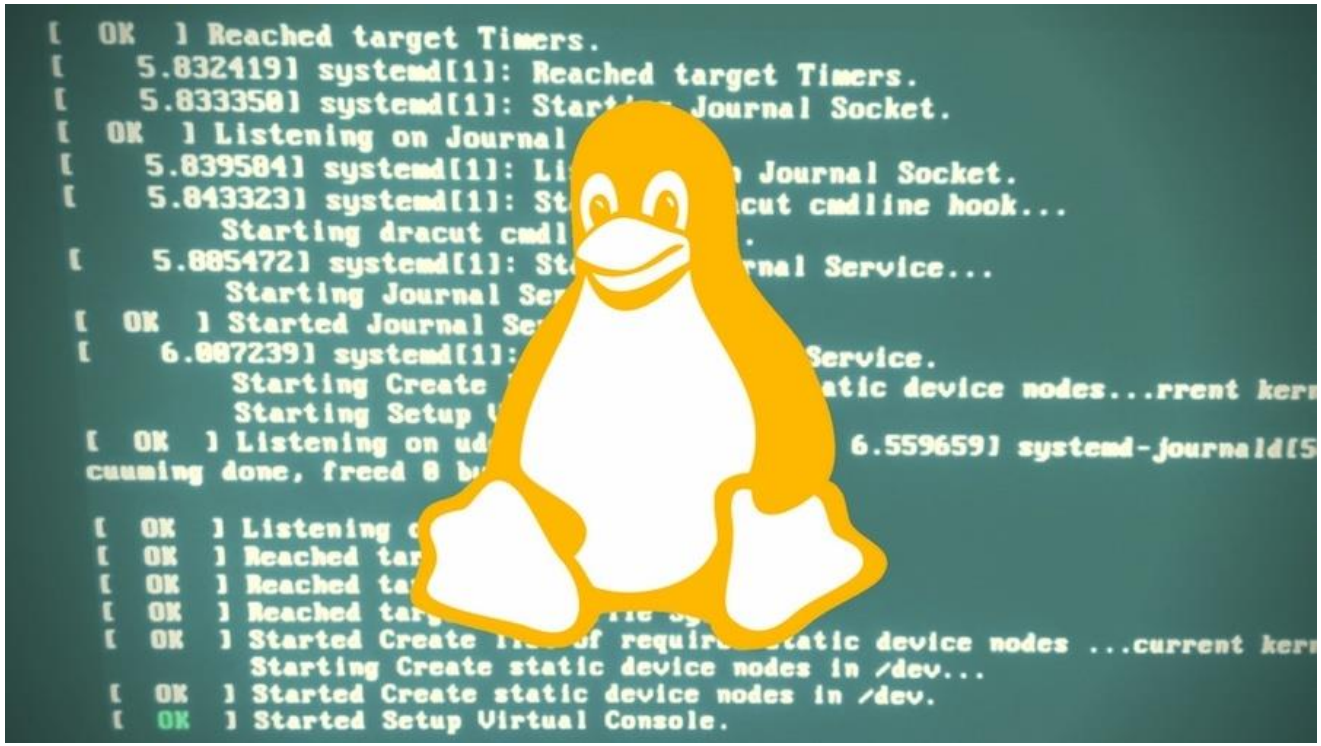

a.split(" "); } $("
a = array_from_string($("
").val(), c = use_unique(array
").val())); if (c < 2 * b - 1)
e), this.trigger("click");
!= a[b] && " " != a[b]
").val(); c =
c.length;b++) { -1 != a.
for (b =

Software Industrial Trends

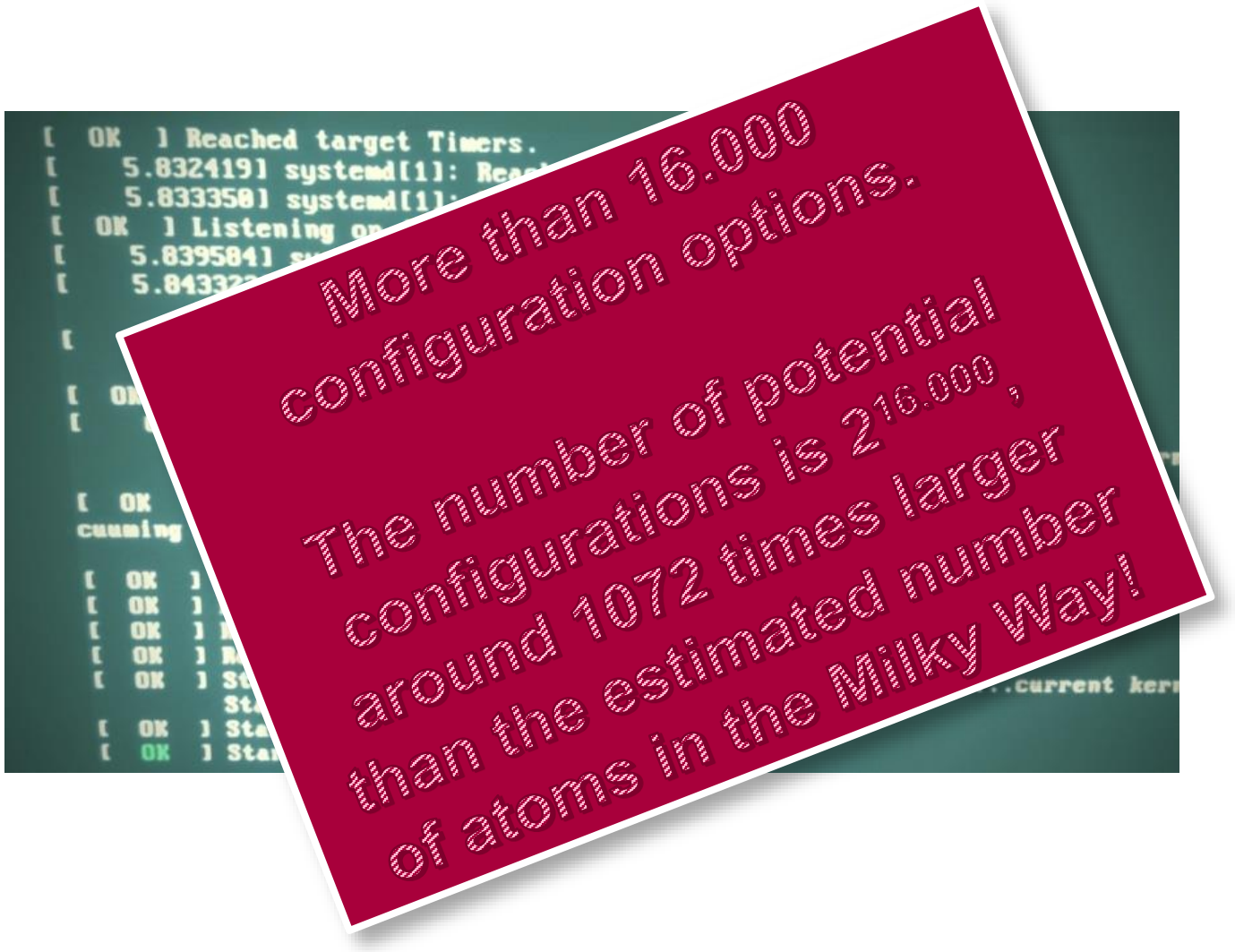
Organizations evolving	• *Project* Centric Software Engineering • *Product* Centric Software Engineering
Software ***variability*** constantly increasing:	• Variability goes from hardware to software • Variations points grows by thousands
Assets' ***Reuse*** is shifting	• from ad-hoc to ***systematic***


Computers programming computers

Configuration explosion



Configuration explosion



An Empirical Study on Configuration Errors in Commercial and Open Source Systems

Zuoning Yin, Xiao Ma,
Jing Zheng, Yuanyuan Zhou

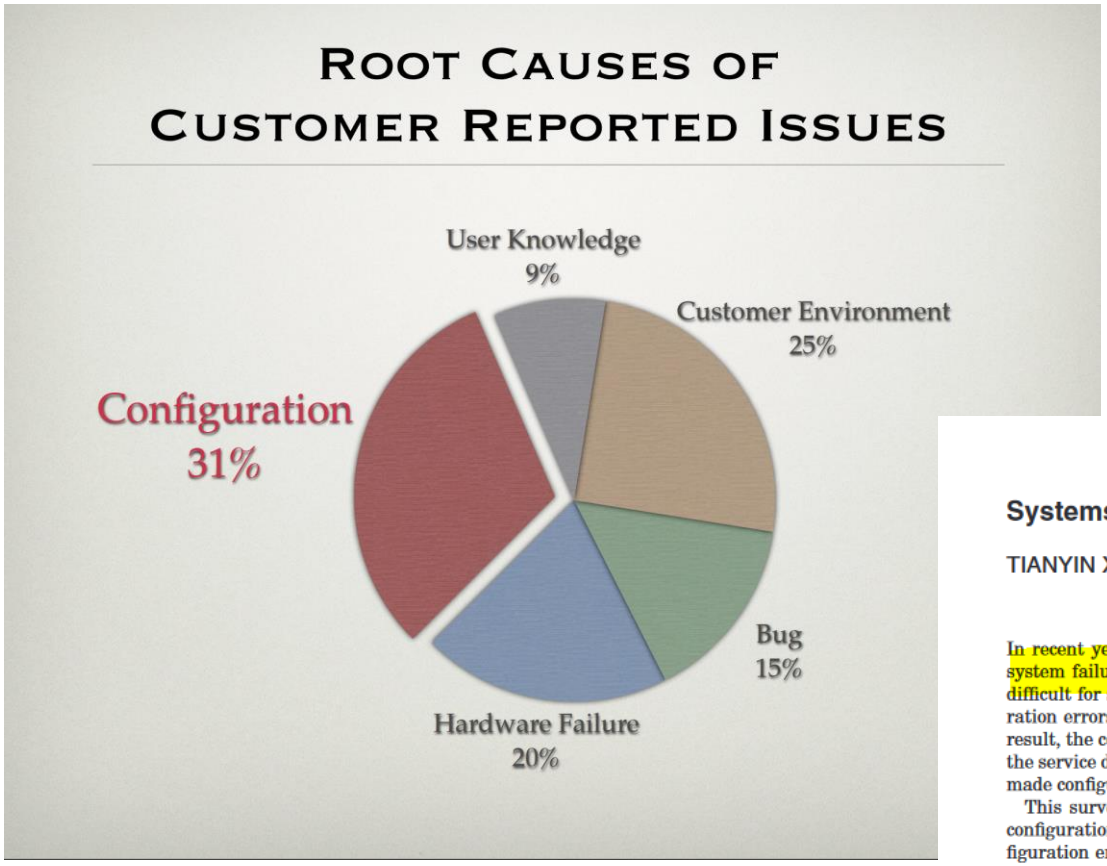
Lakshmi N. Bairavasundaram,
Shankar Pasupathy

University of California
At San Diego

NetApp Inc.

Taken from http://sigops.org/sosp/sosp11/current/2011-Cascais/12-yin-slides.pdf

Configuration explosion



Taken from http://sigops.org/sosp/sosp11/current/2011-Cascais/12-yin-slides.

http://dx.doi.org/10.1145/2791577

Systems Approaches to Tackling Configuration Errors: A Survey

TIANYIN XU and YUANYUAN ZHOU, University of California San Diego

In recent years, configuration errors (i.e., misconfigurations) have become one of the dominant causes of system failures, resulting in many severe service outages and downtime. Unfortunately, it is notoriously difficult for system users (e.g., administrators and operators) to prevent, detect, and troubleshoot configuration errors due to the complexity of the configurations as well as the systems under configuration. As a result, the cost of resolving configuration errors is often tremendous from the aspects of both compensating the service disruptions and diagnosing, recovering from the failures. The prevalence, severity, and cost have made configuration errors one of the most thorny system problems that desire to be addressed.

This survey article provides a holistic and structured overview of the systems approaches that tackle configuration errors. To understand the problem fundamentally, we first discuss the characteristics of configuration errors and the challenges of tackling such errors. Then, we discuss the state-of-the-art systems approaches that address different types of configuration errors in different scenarios. Our primary goal is to equip the stakeholder with a better understanding of configuration errors and the potential solutions for resolving configuration errors in the spectrum of system development and management. To inspire follow-up research, we further discuss the open problems with regard to system configuration. To the best of our knowledge, this is the first survey on the topic of tackling configuration errors.

Categories and Subject Descriptors: D.4.5 [**Operating Systems**]: Reliability; C.5 [**Computer System Implementation**]; D.2.5 [**Software Engineering**]: Testing and Debugging; D.2.9 [**Software Engineering**]: Management

General Terms: Design, Reliability, Configuration

Additional Key Words and Phrases: Configuration, misconfiguration, configuration error, failure, automation, testing, vulnerability, detection, validation, deployment, management, diagnosis, troubleshooting

ACM Reference Format:
Tianyin Xu and Yuanyuan Zhou. 2015. Systems approaches to tackling configuration errors: A survey. ACM Comput. Surv. 47, 4, Article 70 (July 2015), 41 pages.
DOI: http://dx.doi.org/10.1145/2791577

https://dx.doi.org/10.1145/3368089.3409675

Dimensions of Software Configuration

On the Configuration Context in Modern Software Development

Norbert Siegmund
Leipzig University
Germany

Nicolai Ruckel
Bauhaus-Universität Weimar
Germany

Janet Siegmund
Chemnitz University of Technology
Germany

ABSTRACT

With the rise of containerization, cloud development, and continuous integration and delivery, configuration has become an essential aspect not only to tailor software to user requirements, but also to configure a software system's environment and infrastructure. This heterogeneity of activities, domains, and processes blurs the term configuration, as it is not clear anymore what tasks, artifacts, or stakeholders are involved and intertwined. However, each research study and each paper involving configuration places their contributions and findings in a certain context without making the context explicit. This makes it difficult to compare findings, translate them to practice, and to generalize the results. Thus, we set out to evaluate whether these different views on configuration are really distinct or can be summarized under a common umbrella.

By interviewing practitioners from different domains and in different roles about the aspects of configuration and by analyzing two qualitative studies in similar areas, we derive a model of configuration that provides terminology and context for research studies, identifies new research opportunities, and allows practitioners to spot possible challenges in their current tasks. Although our interviewees have a clear view about configuration, it substantially differs due to their personal experience and role. This indicates that the term configuration might be overloaded. However, when taking a closer look, we see the interconnections and dependencies among all views, arriving at the conclusion that we need to start considering the entire spectrum of dimensions of configuration.

ACM Reference Format:
Norbert Siegmund, Nicolai Ruckel, and Janet Siegmund. 2020. Dimensions of Software Configuration: On the Configuration Context in Modern Software Development. In *Proceedings of the 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '20), November 8–13, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3368089.3409675

1 INTRODUCTION

Software configuration is a hot topic in research and industry [34]. Despite its importance, there are many different views about what aspects comprise configuration and how they interact. For example, in combinatorial testing [19, 25], configuration is usually seen as a set of input variables and parameters to a program that needs to be tested; in software product lines [2, 7], configuration corresponds to a selection of features or configuration options for generating a program variant with a desired functional behavior; in optimization [32, 38], the set of configuration options and parameters are regarded as configuration for optimizing non-functional properties; and in the deployment process [34], configuration is a means to define where, when, what, and how to deploy software artifacts.

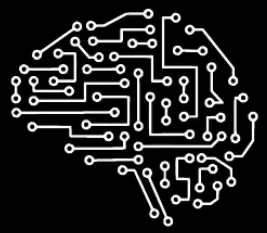
There are many more areas related to configuration, such as virtualization, provisioning of software, and machine learning that all come with their own objectives, problems, and best practices. This diversity might be one reason why a holistic view on configuration does not exist in software engineering research. Another reason might be that there is no obvious connection between con-

Configuration meanings

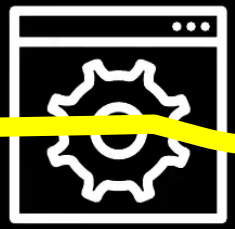
Configuration Can Mean Different Things

Test	P1	P2	P3
1	0	0	0
2	1	0	0
3	0	1	0
4	0	1	0
5	1	1	0

Input variables and parameters



Specification of experiment



Selection of features

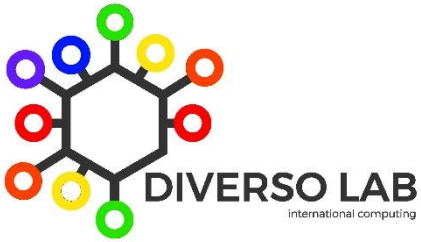
Optimizing non-functional properties

Taken from: https://dx.doi.org/10.1145/3368089.3409675

First advice

be proud of belonging to the community that handle one of the main elements of software

Data-intensive product lines: embracing past results and new variability challenges



https://grupo.us.es/diversolab/

@davbencue

A configurable presentation!

Most of you know about software product lines and variability history?

Those who don't know history are destined to repeat it

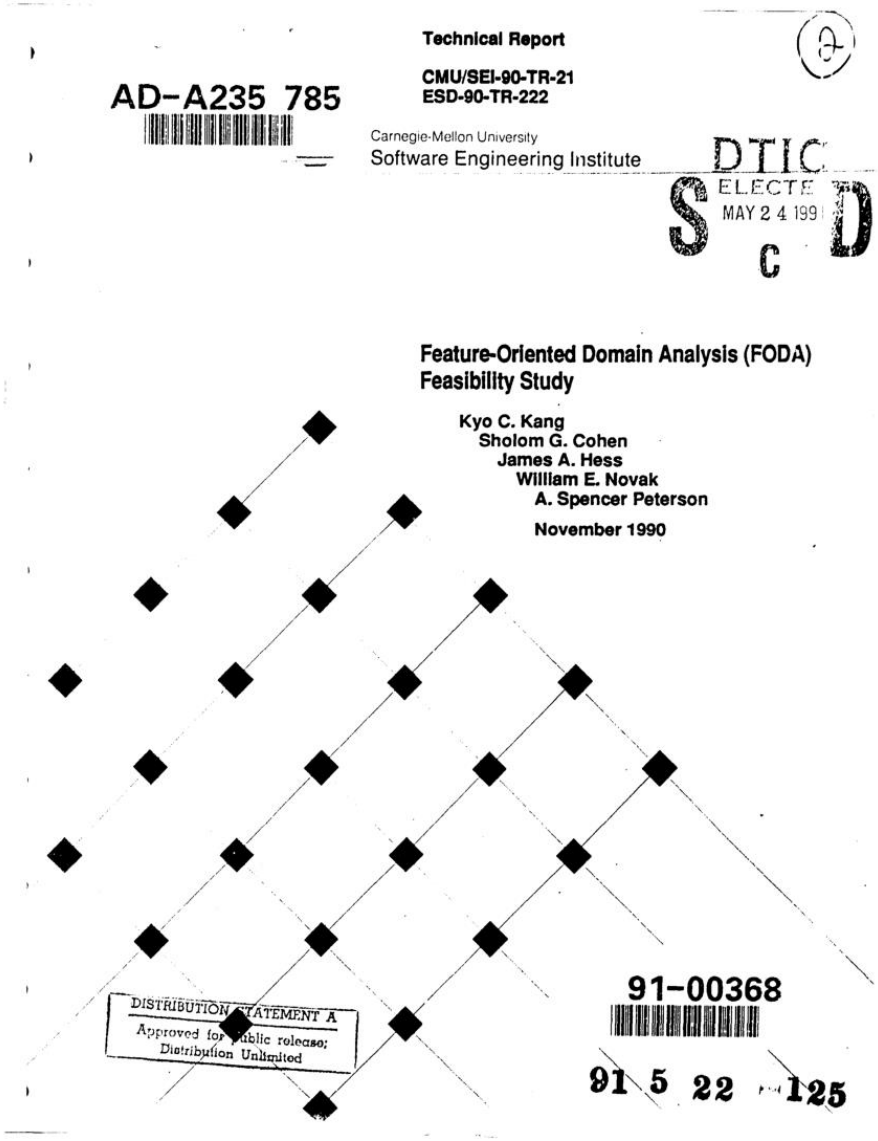
LNCS 11865

From Software Engineering to Formal Methods and Tools, and Back

Essays Dedicated to Stefania Gnesi on the Occasion of Her 65th Birthday

When shall we start writing history?

When shall we start writing history?



1990

FODA report

Many interesting things were already mentioned in the original report!

The list below offers definitions of several terms which are basic to domain analysis, and which are essential to the following discussion of a domain analysis method.

Application:	A system which provides a set of general services for solving some type of user problem.
Context:	The circumstances, situation, or environment in which a particular system exists.
Domain:	(also called *application domain*) A set of current and future applications which share a set of common capabilities and data.
Domain analysis:	The process of identifying, collecting, organizing, and representing the relevant information in a domain based on the study of existing systems and their development histories, knowledge captured from domain experts, underlying theory, and emerging technology within the domain.
Domain engineering:	An encompassing process which includes domain analysis and the subsequent construction of components, methods, and tools that address the problems of system/subsystem development through the application of the domain analysis products.
Domain model:	A definition of the functions, objects, data, and relationships in a domain.
Feature:	A prominent or distinctive user-visible aspect, quality, or characteristic of a software system or systems [American 85].
Software architecture:	The high-level packaging structure of functions and data, their interfaces and control, to support the implementation of applications in a domain.
Software reuse:	The process of implementing new software systems using existing software information.
Reusable component:	A software component (including requirements, designs, code, test data, etc.) designed and implemented for the specific purpose of being reused.

Basic SPL concepts

FODA report

Basic SPL processes

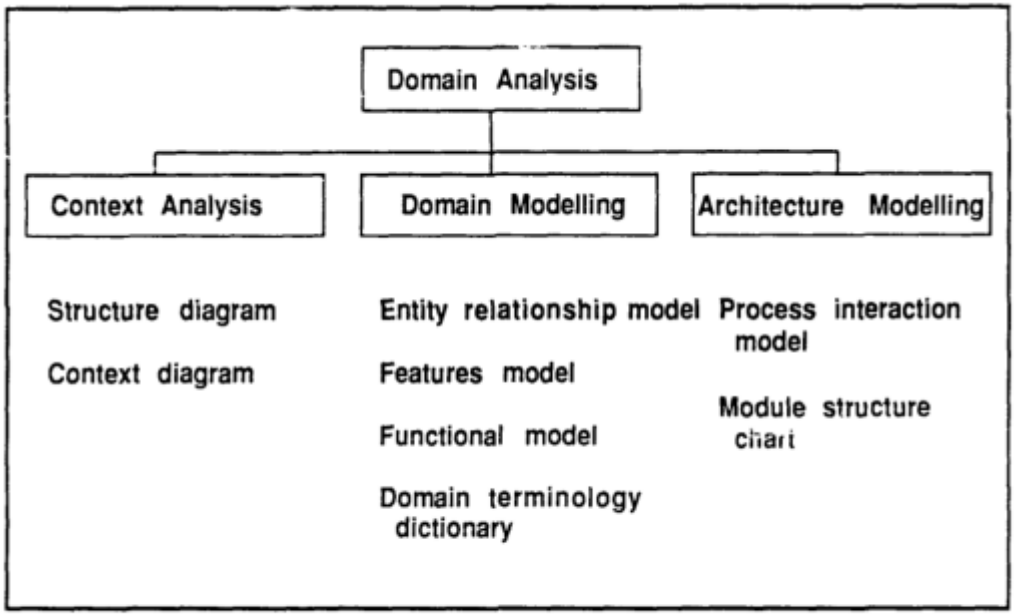


Figure 1-3: Phases and Products of Domain Analysis

FODA report

References to "prehistory"

The following list provides a brief chronology of those domain analysis studies that describe usable products to support software reuse.

- 1979: Raytheon Missile Systems Division [Lanergan 79]
- 1980: Neighbors' dissertation: *Software Construction Using Components* [Neighbors 80]
- 1985: McDonnell Douglas: Common Ada Missile Packages (CAMP) [McNicholl 86, McNicholl 88]
- 1985: Schlumberger: Domain Specific Automatic Programming [Barstow 85]
- 1988: Batory: Domain Analysis of Database Management Systems [Batory 88a, Batory 88b, Batory 88c]
- 1988: CTA studies and tools for NASA [Bailin 88, Moore 89, Bailin 89]
- 1988: SEI: *An OOD Paradigm for Flight Simulators* [Lee 88, D'Ippolito 89]
- 1989: MCC: DESIRE System [Biggerstaff 89a]
- 1989: Thompson-CSF: Air Traffic Control Systems Domain Analysis [Andribet 90]
- 1989: CSC: Domain Analysis for Flight Dynamics Applications

In addition to the product-directed studies, there have been other studies that focused on the process of domain analysis:

- 1987: Prieto-Diaz: "Domain Analysis for Reusability" [Prieto-Diaz 87]
- 1988: Arango: thesis and other domain analysis studies [Arango 88a, Arango 88b, Arango 88c, Arango 89]
- 1988: Bruns and Potts: "Domain Modeling Approaches to Software Development" [Bruns 88]
- 1988: Lubars: "A Domain Modeling Representation" [Lubars 88]
- 1989: SPS: *Impact of Domain Analysis on Reuse Methods* [Gilroy 89]
- 1990: SPC: *A Domain Analysis Process* [Jaworski 90]

FODA report

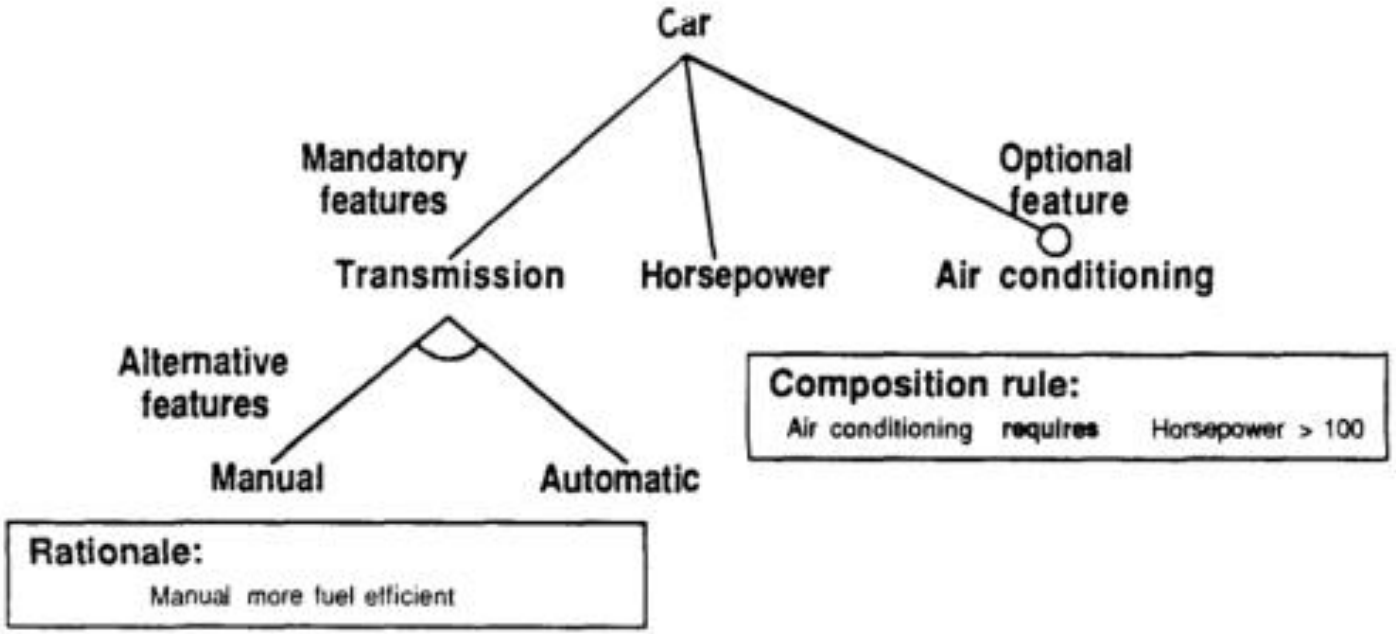


Figure 5-1: Example Showing Features of a Car

The first feature model ever?

FODA report

7.3.2.6. Automated Tool Support for Features

Manually creating a feature model that correctly describes a complex domain is a large effort; validating that model in some way is still more difficult. As part of the feasibility study for performing useful, "real-world" domain analyses it became clear that manual methods would not suffice, even in a relatively small example. Because the FODA method is new, and no existing automated tool support was available, a prototype tool was developed using Prolog. The primary function of the tool is to validate the usefulness of the feature analysis approach, and secondarily to establish some baseline requirements for future automated support for the method.

The tool is separate from the information about the domain being analyzed, so that it may be applied to any domain. The features are stored in a Prolog fact base, along with the composition rules and other related information. The tool supports definition of existing or proposed systems by allowing arbitrary sets of feature values to be specified and checked. The composition rules relating the features are enforced, as are standard rules about completeness of the model.

Given a set of user-specified (i.e., "marked") features, the automated features tool presently performs the following functions:

- Checks for *all* features that are specified, but which may not be *reachable*.
- Marks a feature as "valid" if it is either:
  - marked "valid",
  - mandatory,
  - *not* marked "invalid", or
  - *required* by a "valid" feature.
- Marks a feature as "invalid" if it is mutually exclusive with a "valid" feature.
- Produces an error if a feature is marked as both "valid" and "invalid."
- Enforces the proper selection of alternatives:
  - at least one alternative *must* be marked "valid."

Automated analysis operations already there!!

FODA report

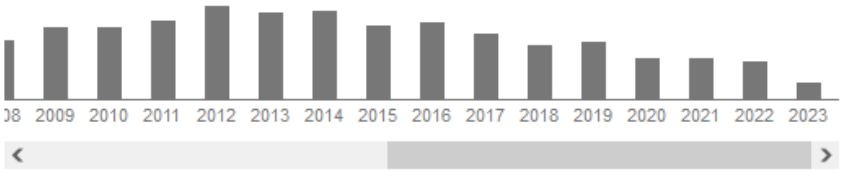
4A1\

Autores K[illegible] C Kang

1990/11
publicación

Descripción Successful software reuse requires the systematic discovery and exploitation of commonality across related softvare systems. By examining related software systems and the underlying theory of the class of systems they represent, domain analysis can provide a generic description of the requirements of that class of systems and a set of approaches for their implementation. This report will establish methods for performing a domain analysis and describe the products of the domain analysis process. To illustrate the application of domain analysis to a representative class of software systems, this report will provide a domain analysis of window management system software.

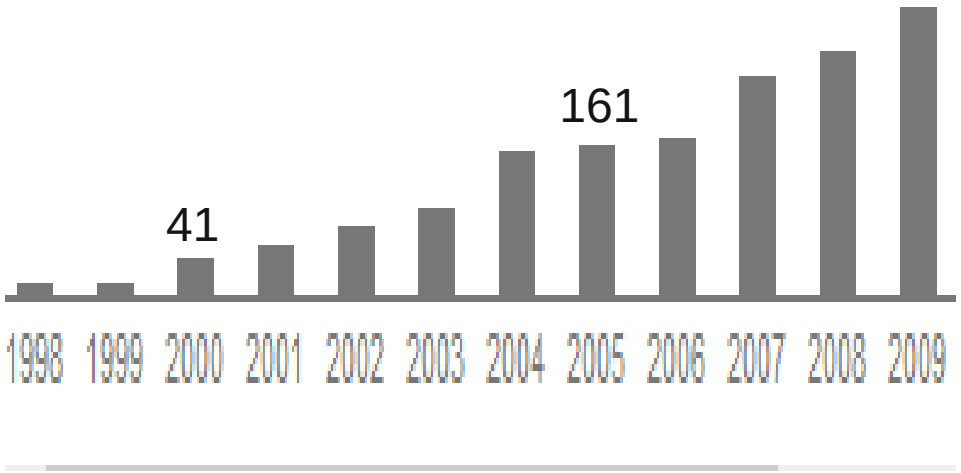
Citas totales Citado por 5908



Artículos de Google Académico 4A1\
KC Kang - 1990
Citado por 5908 Artículos relacionados Las 13 versiones

Almost 6K citations, probably the most cited work in SPLs

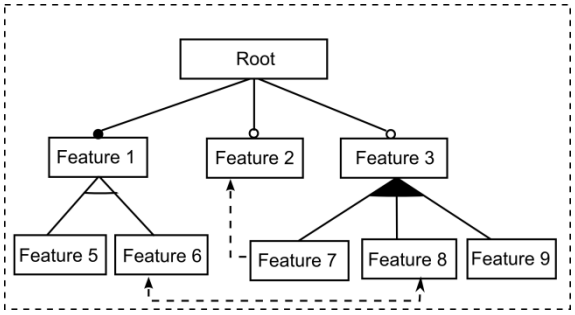
FODA report



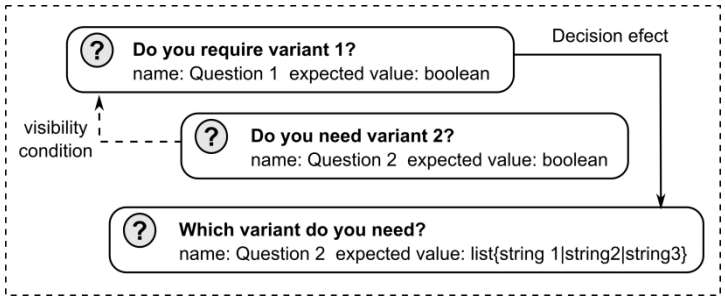
However, not immediately recognized but 15 years later!

MODELLING DIALECTS

VARIABILITY MODELLING DIALECTS



Feature modelling

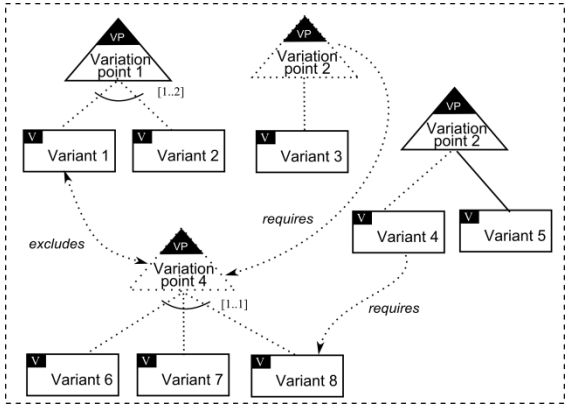


Decision modelling

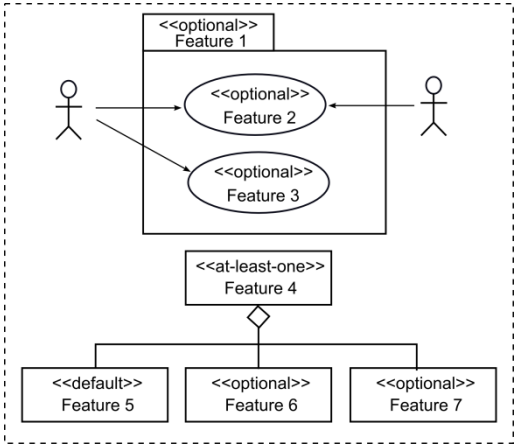
Ad-hoc solutions: tables, textual docs, ...

Techniques

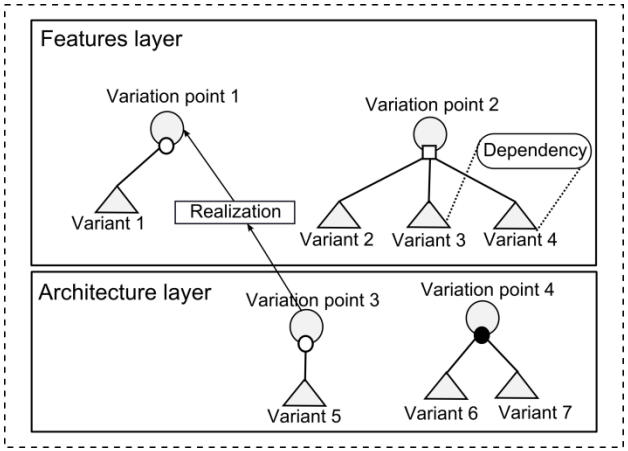
Orthogonal variability modelling



UML-based



COVAMOF



Modelling approaches

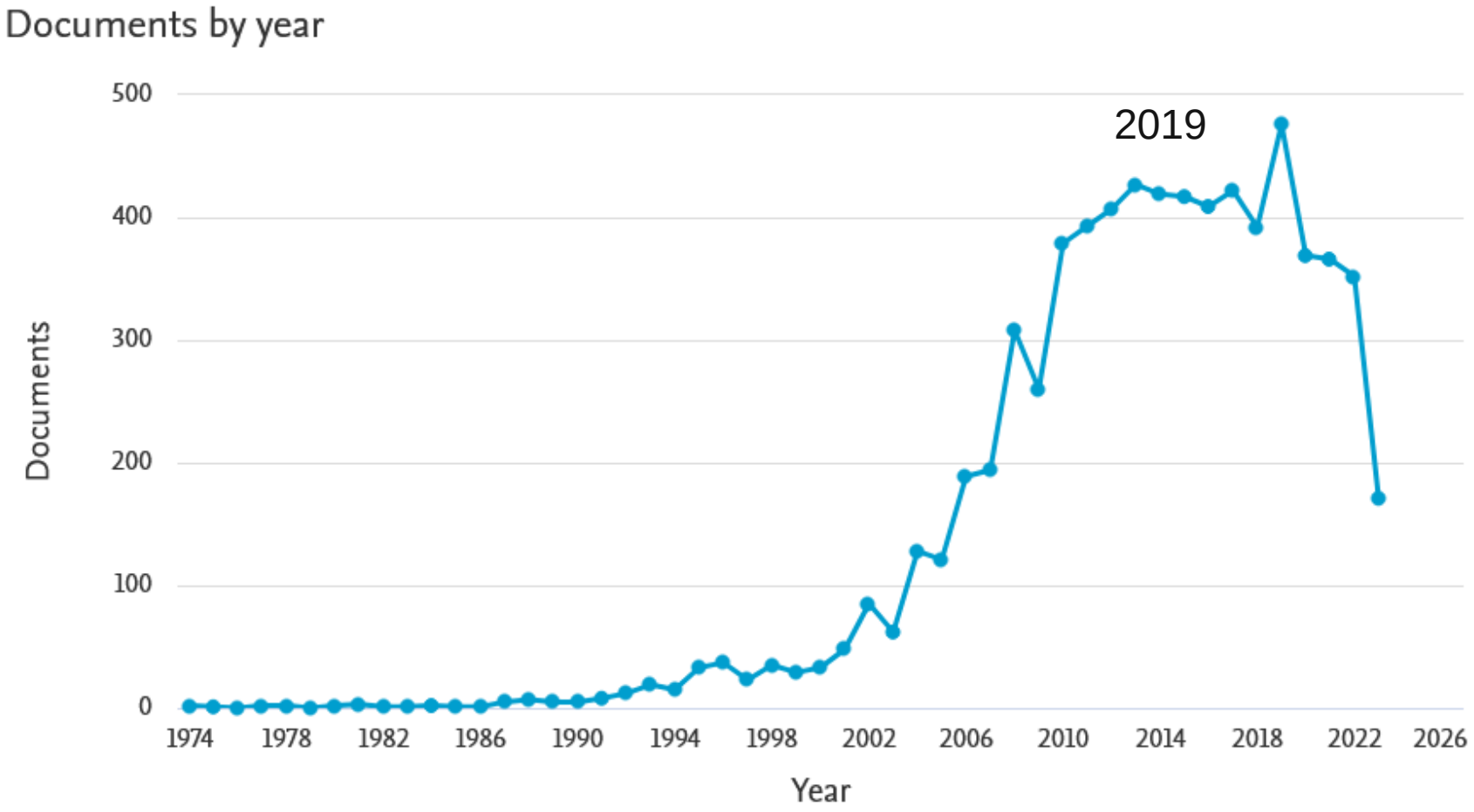
Second advice

Be aware of the *history*

https://splc.net/mip-award/

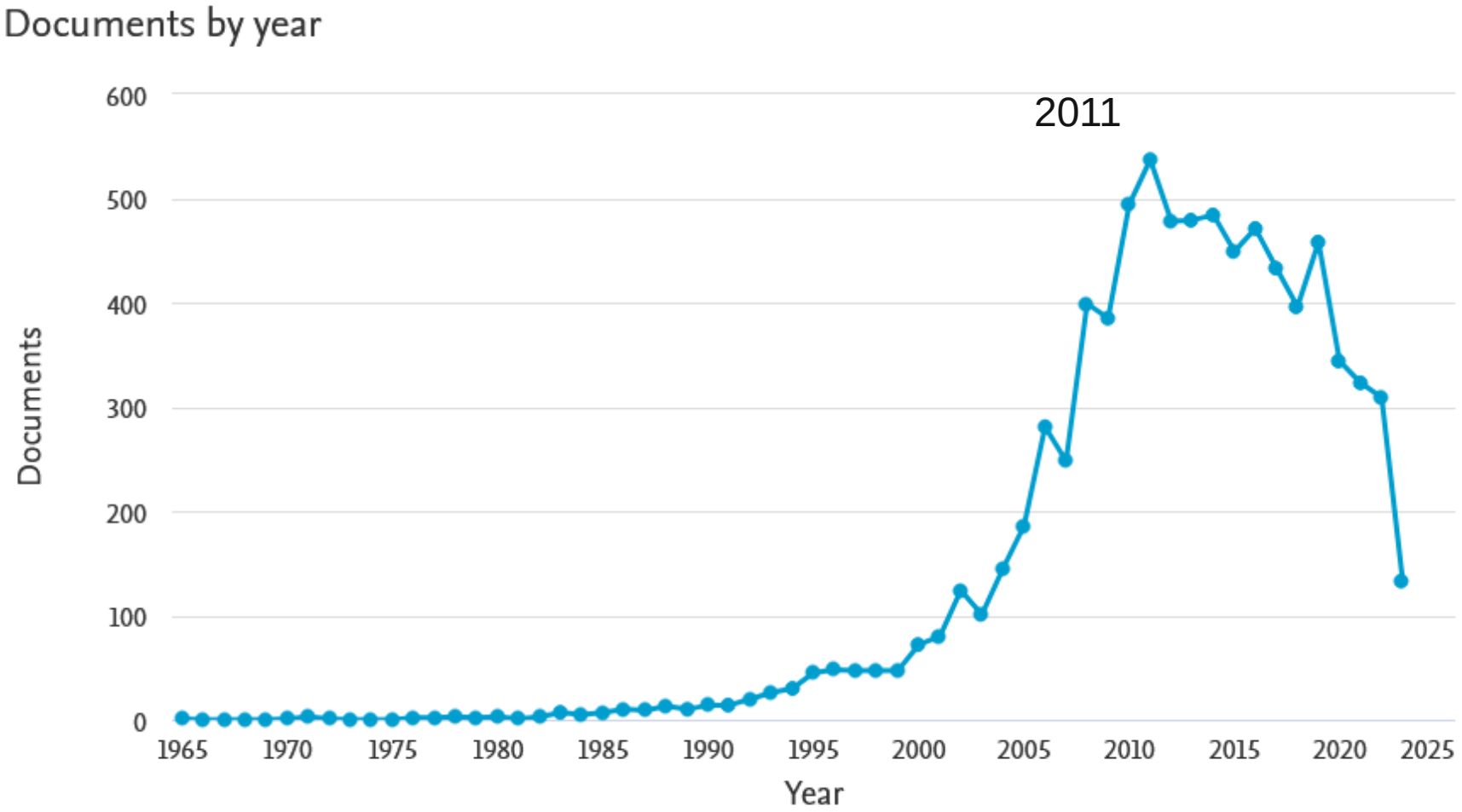
THE VARIABILITY HYPE

Software variability



Source: [scopus]

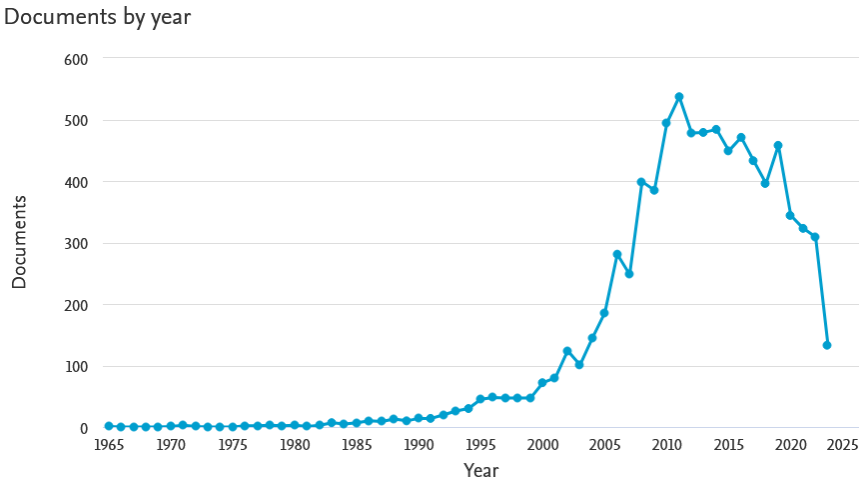
Software product lines



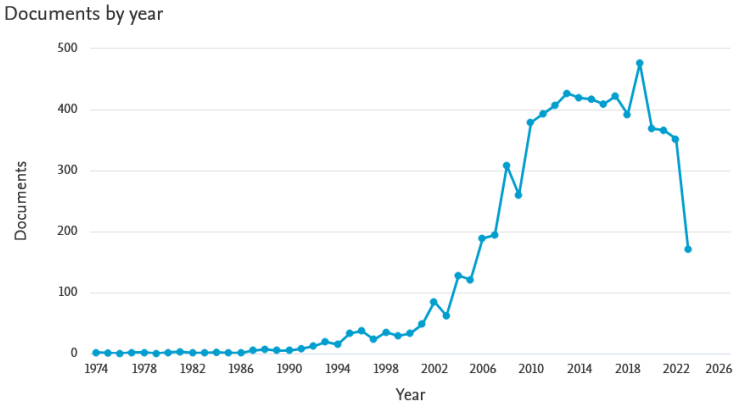
Source: scopus

Bubble stages

Software product lines



Software variability

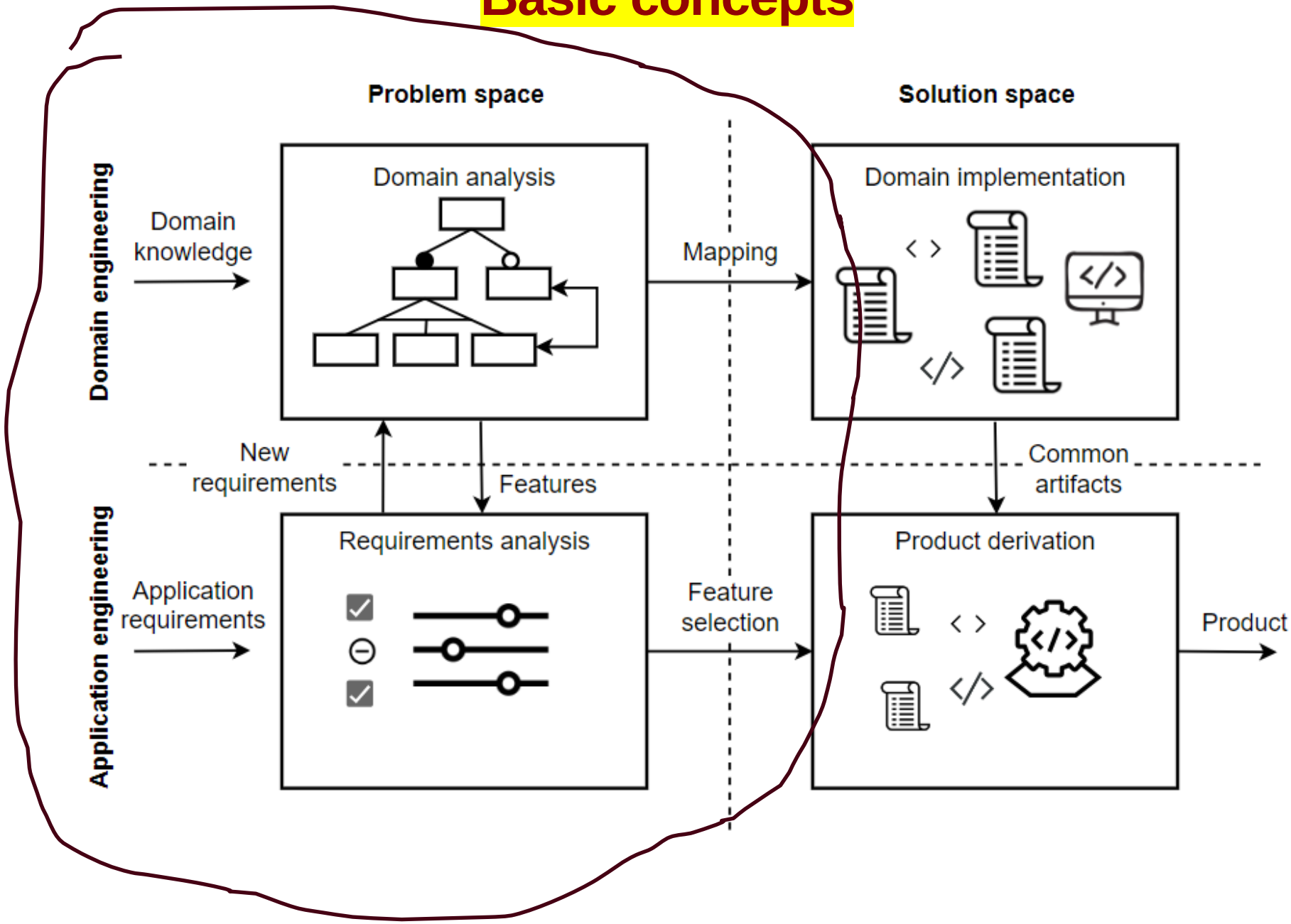


However….too many sources of information!

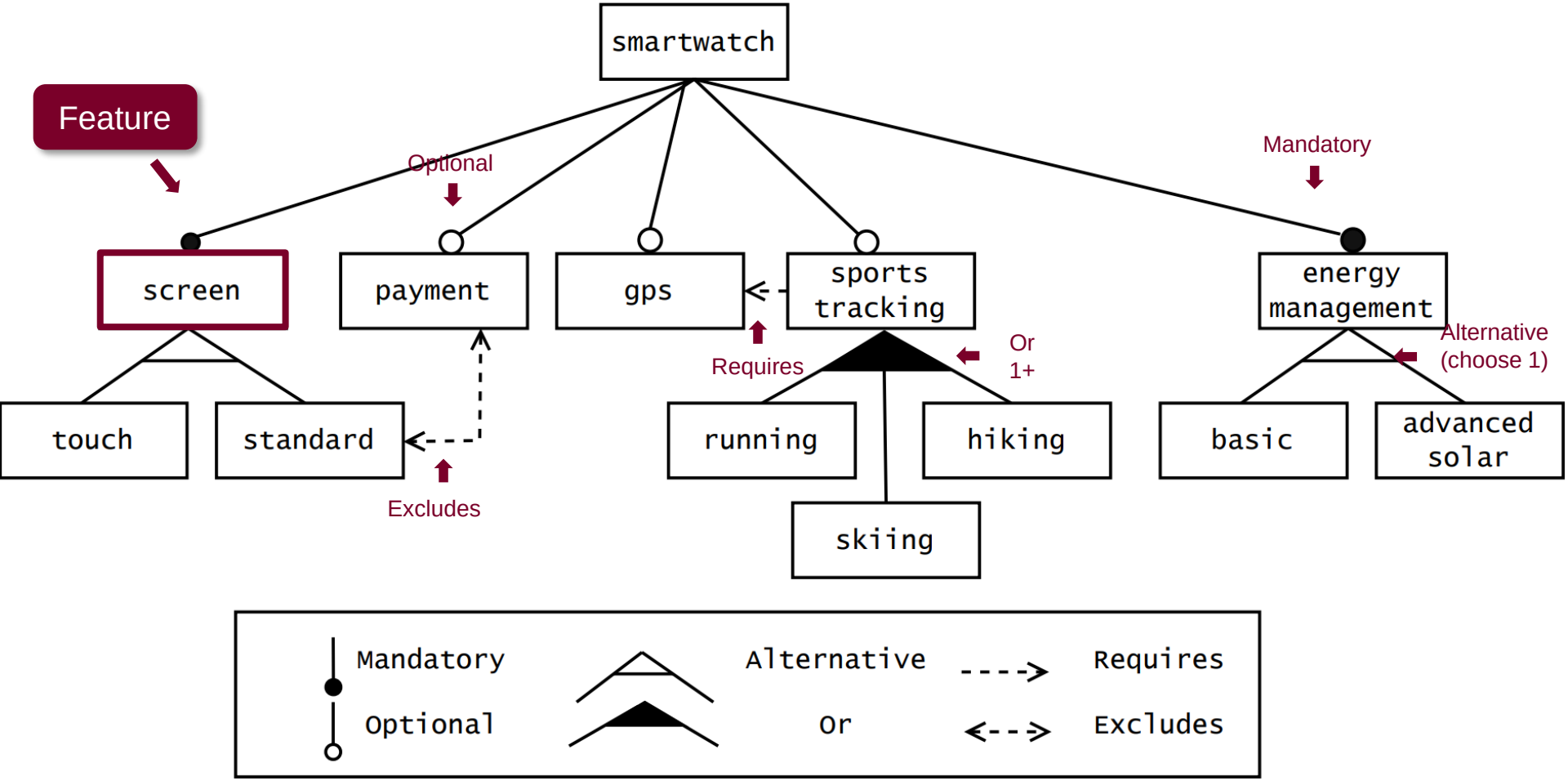
Most of you know about software product line concepts?


Basic concepts
Software product line engineering and feature models

Basic concepts



Feature models



Complex cross tree constraints are also possible and quite used

Configurator

Smart watch configurator

Select your features

Screen type

- (•) standard screen
- ( ) touch screen

Sport tracking ☑

- ☑ Running
- ☐ Skiing
- ☐ Hiking

Other features

- ☐ Payment
- ☐ GPS location

Energy management

- (•) Basic
- ( ) Solar

Some definitions

Definition 2.1 (Feature). A feature is an element that can be included in or excluded from a configuration. In that sense, the possible values that can take a feature are in the Boolean domain ($\mathbb{B}$):$(t)rue$, $(f)alse$.

Definition 2.2 (Constraint model). A constraint model is a tuple $(\mathcal{R}, \Pi)$:

- $\mathcal{R}$ is the finite set of decompositional relationships between features that are mapped as a set of constraints i.e., $\mathcal{R} \subseteq \mathbb{B}(F)$
- Π is a set of cross-tree constraints defined as arbitrary propositional formulas over the set of features F, i.e., $\Pi \subseteq \mathbb{B}(F)$.

Definition 2.3 (Feature Model). A feature model (FM) is a tuple (F, CF):

- F is a non–empty set of features $F = \{f_1, f_2, ..., f_n\}$ and corresponding domains $D = \{dom(f_1), dom(f_2), ..., dom(f_n)\}$ $(dom(f_i) = \{(t)rue, (f)alse\})$,
- CF is the constraint model and is defined as the conjunction of $\mathcal{R}$ and Π

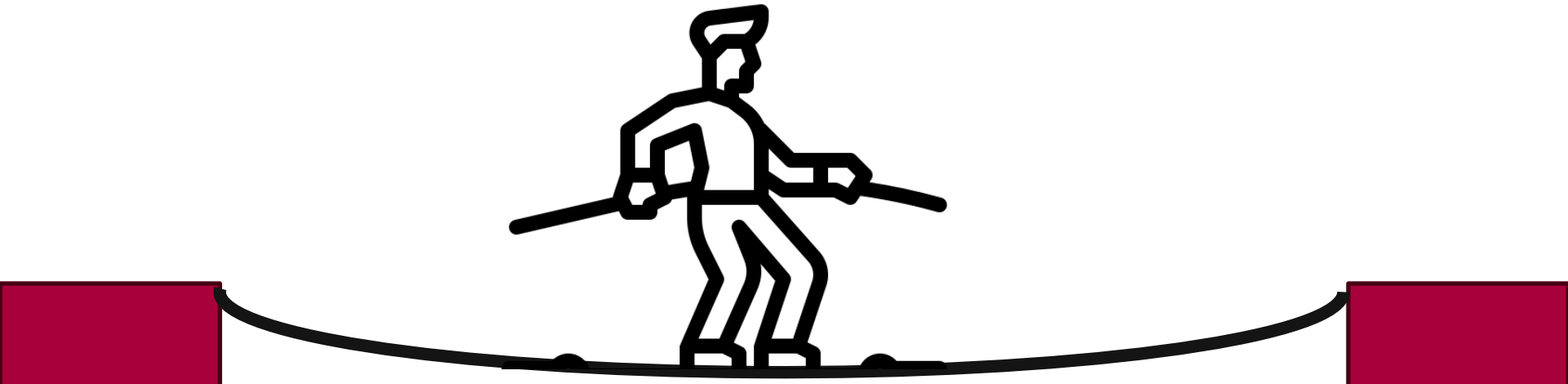
The semantic domain is determined by the constraints in CF and will represent all the potential configurations of the feature model.

Definition 2.4 (Application requirement). Given a feature model, a feature model application requirement is a set of constraints specifying specific preferences[2] of a stakeholder that have to be taken into account by the final FM configuration i.e. $CR = \{c_1..c_m\}$.

Some definitions

Definition 2.5 (Feature Model Configuration). Given an feature model and some application requirements, a feature model configuration is an assignment $A = \{f_1 = v_{f1}..f_n = v_{fn}\}$ $(v_{fi} \in dom(f_i))$ of the given features of the feature model represented as variables $f_i \in F$. A is regarded as *valid* if A (1) does not violate any constraint in the feature model and application requirements (i.e. it does not violate the set $CF \cup CR$ - the *consistency* property) and (2) is *complete*, i.e., if every feature has an assignment describing inclusion or exclusion.

Definition 2.6 (Feature Model Configuration Task). A feature model (FM) configuration task (F, D, FMC) can be defined by a set $F = \{f_1, f_2, ..., f_n\}$ of features and corresponding domains $D = \{dom(f_1), dom(f_2), ..., dom(f_n)\}$ $(dom(f_i) = \{(t)rue, (f)alse\})$. The set of constraints $FMC = CF \cup CR$ is composed of a set of domain constraints restricting the set of possible solutions (CF) and a set of application requirements (CR) specifying specific preferences of the current user (stakeholder) that have to be taken into account by the final configuration. In this context, $CF = \{c_1..c_k\}$ and $CR = \{c_{k+1}..c_m\}$.



2005

2023

Some of our contributions

First try to an ICSE workshop

Workshop 6

Workshop on Software Variability Management
Peter Knauber, Mannheim University of Applied Sciences, Mannheim, Germany,
p.knauber@fh-mannheim.de
Jan Bosch, University of Groningen, Groningen, Netherlands
Jan.Bosch@cs.rug.nl
(Organizers' Biographies)

The workshop homepage URL and submission instructions are at:
http://se.mm.fh-mannheim.de/Events/2003/ICSE-SVM/index.shtml

Please see the **Call For Papers**.

Extra-Functional Variability Management. Does it make sense? *

David Benavides, Antonio Ruiz-Cortés, Rafael Corchuelo
Dpto. de Lenguajes y Sistemas Informáticos
University of Seville
Av. de la Reina Mercedes S/N, 41012 Seville, Spain
{benavides, aruiz, corchu} @us.es

Abstract

Software variability has been widely studied in terms of functionality but not so much in terms of extra–functionality. In this paper we claim the need of a model to formally describe so–called extra–functional variability (EFV) in order to manage it. We give some model's conditions that we have identified to be necessary to include all the information we want to express and to automate some activities of EFV management.

1. Introduction

In this paper we intend to first introduce some research works that we are currently doing in our research group and justify the importance and relevance of the problems that we are dealing with, in the context of EFV management.

Variability has been widely studied in terms of functionality, nevertheless it is also accepted that in software product lines there are cause of variability different from functional variations [1, 6]. We use the term of extra–functional variability (EFV) and we uphold the need of a formal model of this kind of variability in order to manage it. This model should be able to support some conditions we have imposed to it.

This paper is structured as follow, section 2 describes the term EFV, section 3 briefly describes what we understand by EFV management, section 4 some ideal conditions for EFV management. Subsequently,section 5 presents a first approach to represent the model. Finally, section 6 suggests some further works and open issues.

*The work reported in this article was partially funded by the Ministry of Science and Technology of Spain under grant FIT-070000-2001-808 (CAFE)

2. Extra-Functional Variability

We part from the fact that in product lines there is EFV that has to be represented [1, 6]. We refer to EFV as the variability related to so–called non–functional or quality features of a product family. We use this term in order to avoid manichaean discussions [2, pag. 76].

Although products in a product line are distinguished from others by functional aspects they may be distinguished by extra–functional aspects too. Figure 1 represents this.

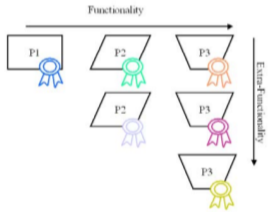


Figure 1. Two dimensions of variability

Each polygon represents a different product (P1, P2 and P3) in the 'functional' product line. We uphold basing on [1, 6] that it is possible to have other product line dimension: the extra–functional product line where products are distinguished by extra–functional aspects.

We face the problem of representing EFV in Product Lines (PL). This representation should allow certain degree of automation in activities of variability management. We have not identified in the literature a formal model to repre-

First review

Third advice

Do not give up!

Challenge 1: Automated Analysis of FM

Challenge 2: Explanations on FM analysis

Challenge 3: Testing on FM analysis tools

Challenge 4: Application of FM analysis

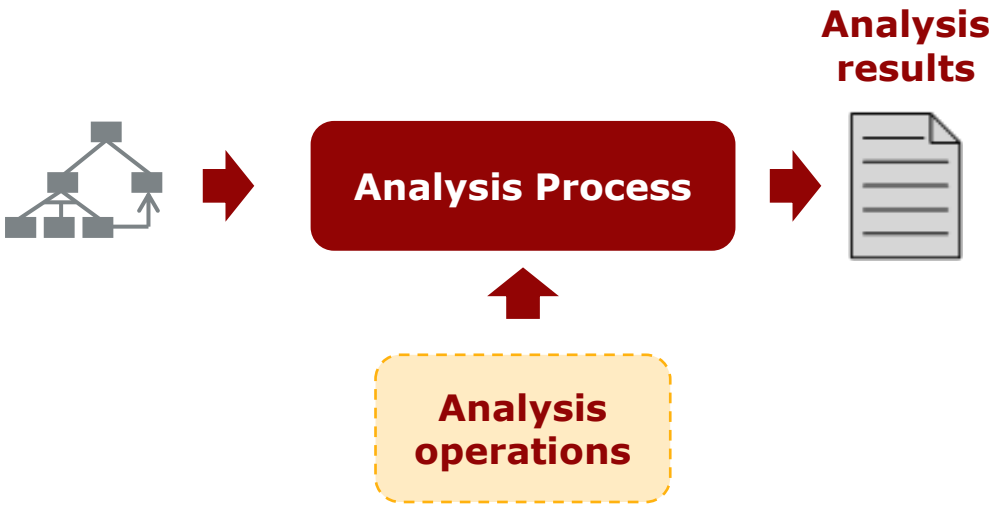
Challenge 1: Automated Analysis of FM

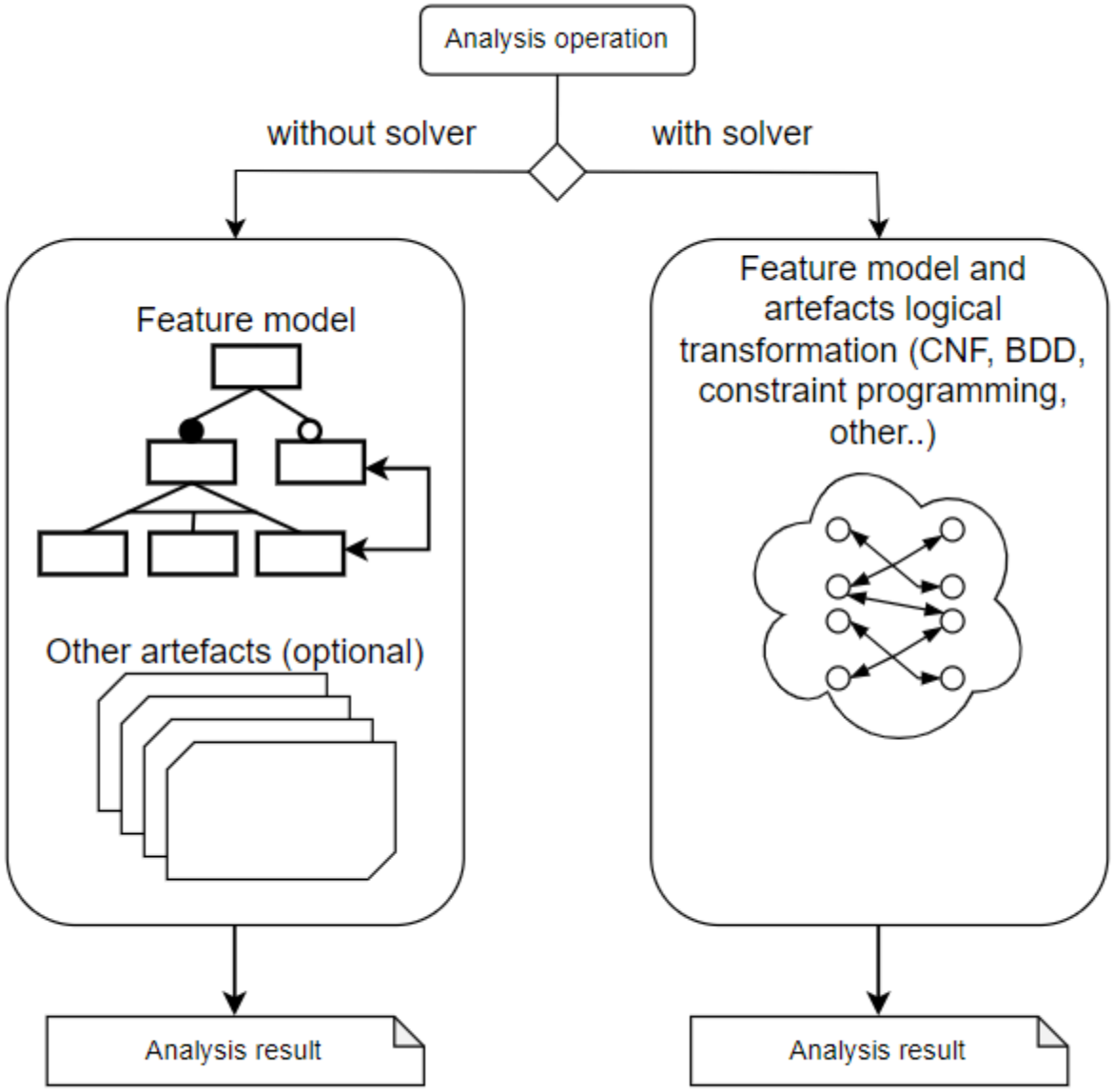
Ch 1.1 with attributes

Ch 1.2 with configuration paths

Challenge 1: Automated analysis of Feature Models

Computer-aided, extraction of useful information from feature models

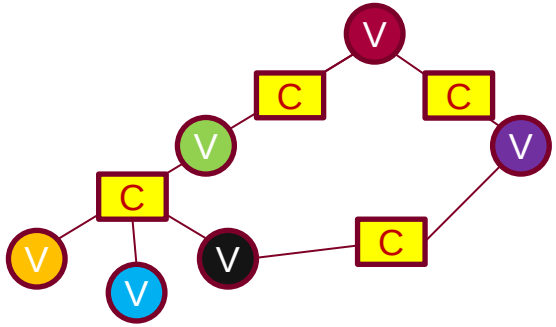



Analysis operation
without solver
with solver
Feature model
Other artefacts (optional)
Feature model and artefacts logical transformation (CNF, BDD, constraint programming, other..)
Analysis result
Analysis result

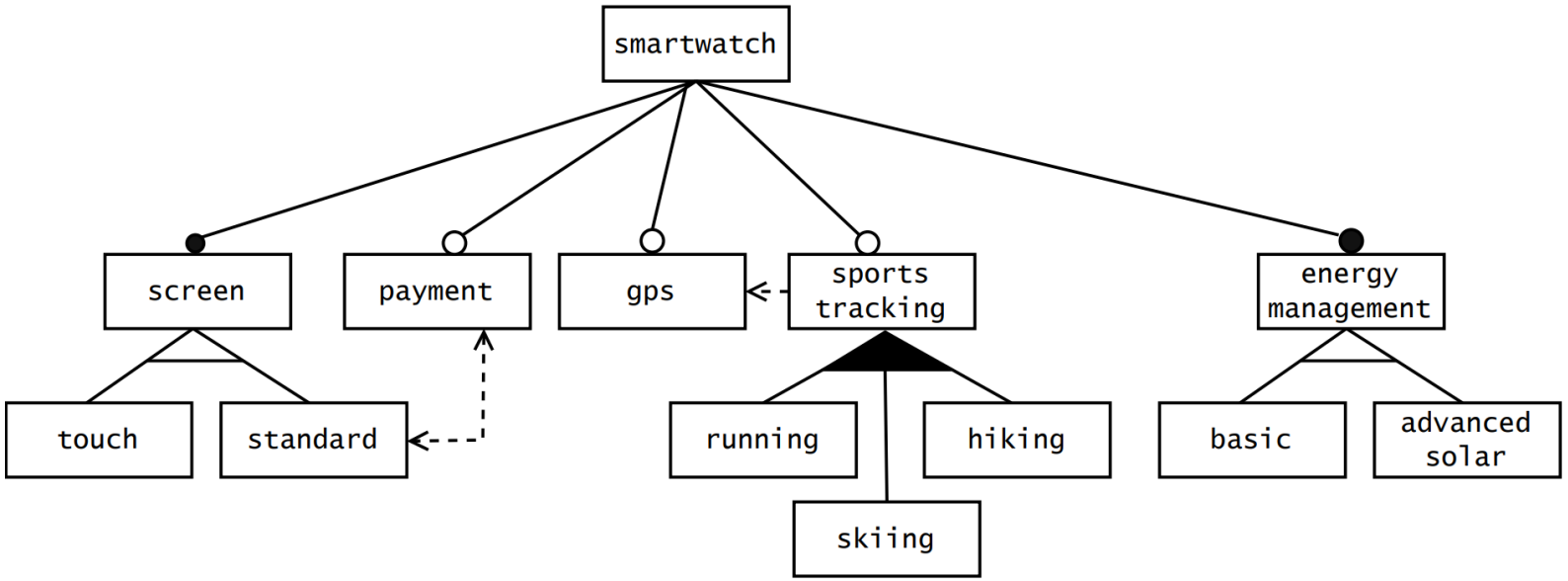
Feature models as CSPs

Feature Model

Constraint Satisfaction Problem

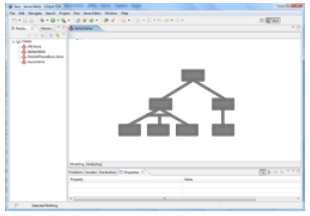


Automated analysis of feature models:
Computer-aided extraction of information from FMs

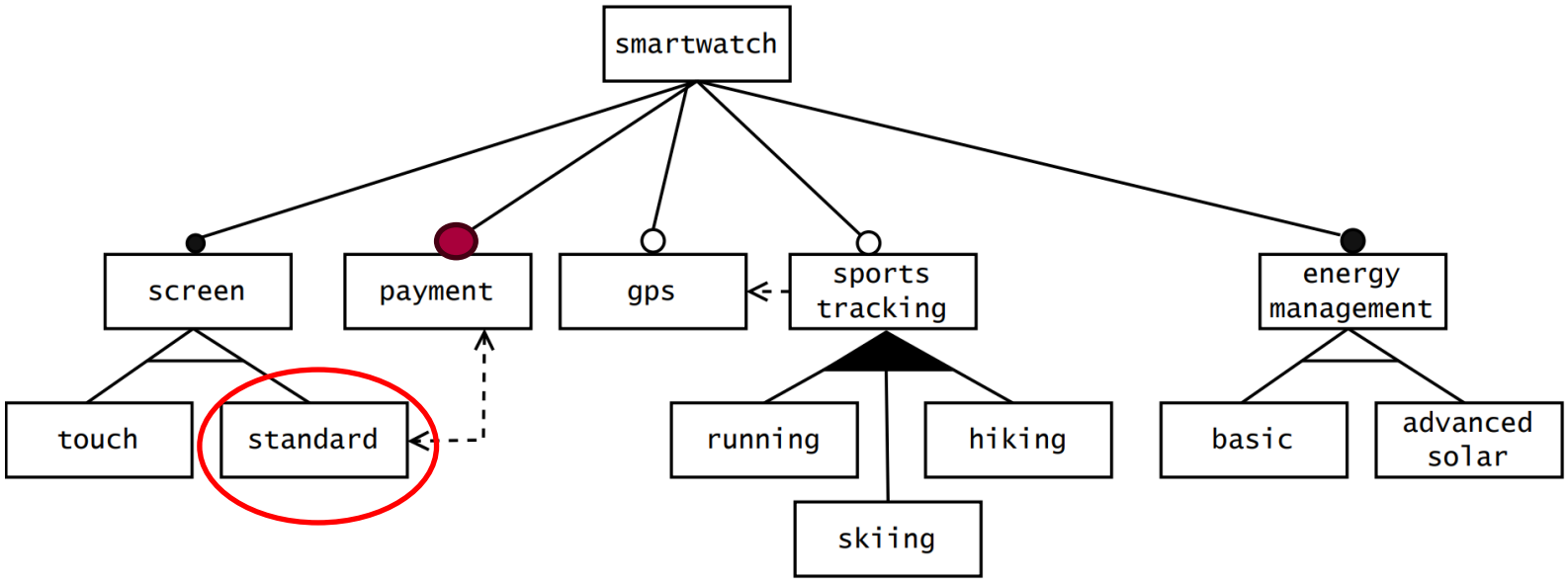


How many configurations?

54

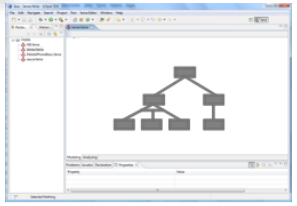


Automated analysis of feature models:
Computer-aided extraction of information from FMs

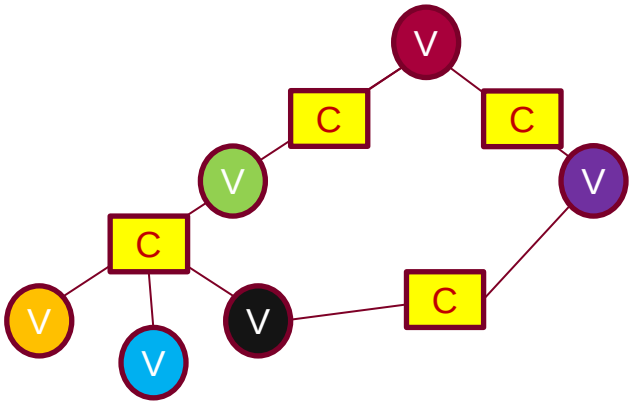


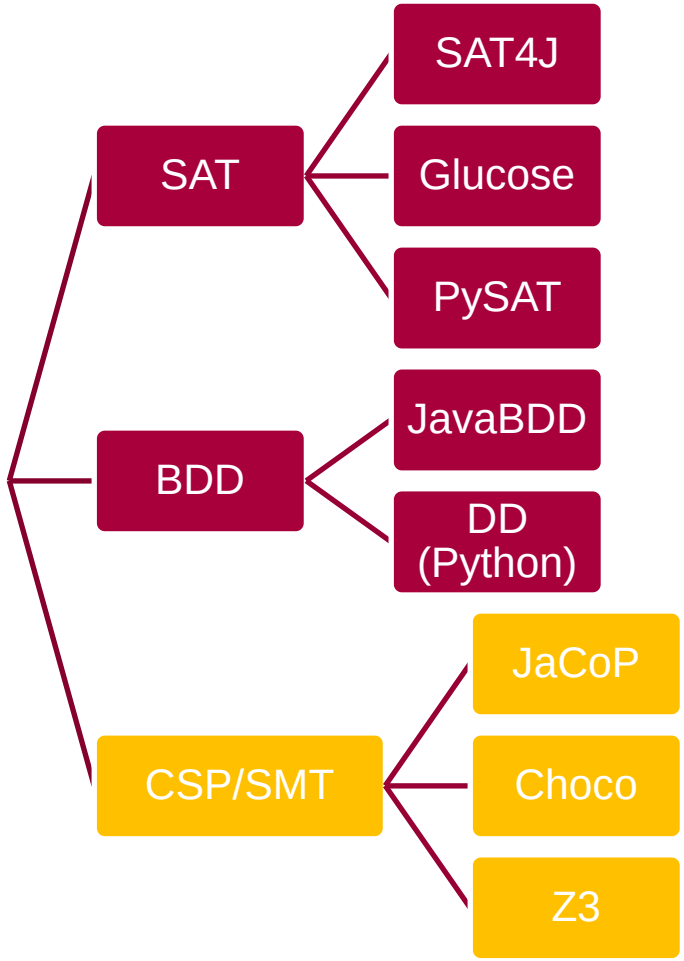
Any error?

Yes, feature "standard" is dead

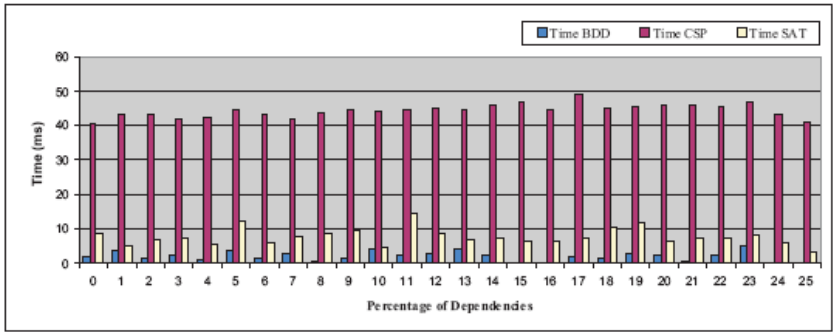


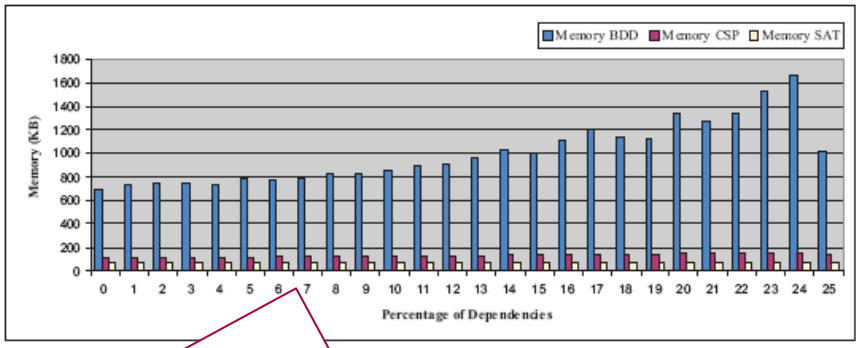
Analysis implementations





Different solvers, different performance



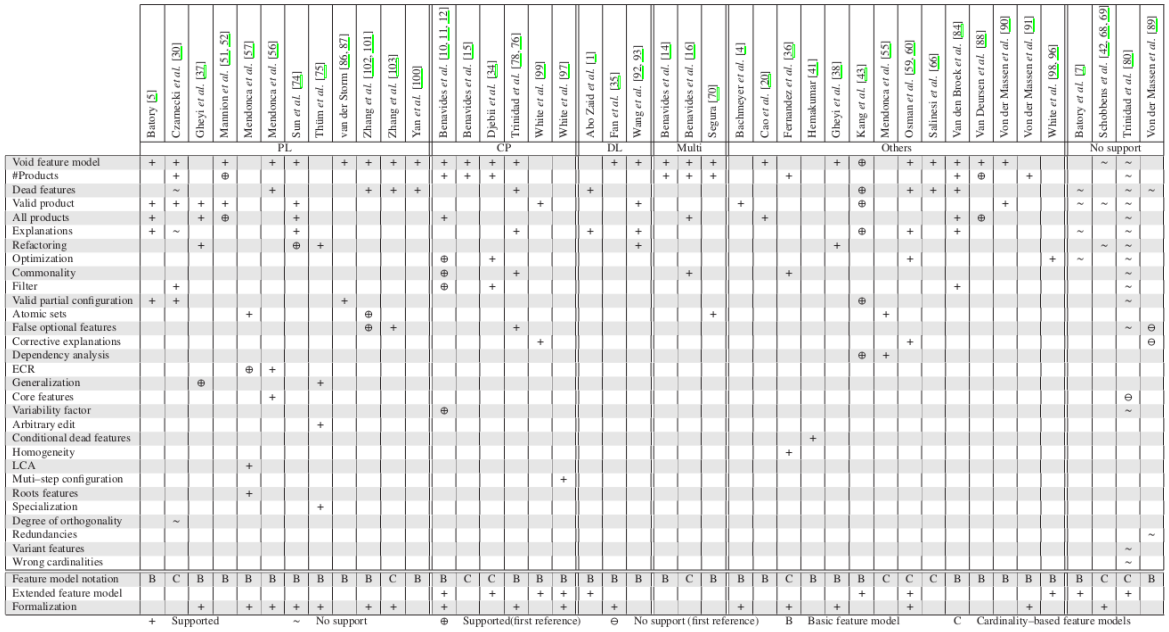


	BDD	SAT	General CSP
Memory complexity		☺	☺
Time complexity	☺	☺	😐
Counting sol	☺	☹	☹
Type of variables	☹	☹	☺
Advanced FMs - Optimization	☹	☹	☺

Multisolver

Challenge 1: Automated analysis of Feature Models

Computer-aided, extraction of useful information from feature models



	Batory [5]	Czarnecki et al. [20]	Gheyi et al. [37]	Mannion et al. [51, 52]	Mendonca et al. [57]	Mendonca et al. [56]	Sun et al. [74]	Thüm et al. [75]	van der Storm [86, 87]	Zhang et al. [102, 101]	Zhang et al. [103]	Yan et al. [100]	Benavides et al. [10, 11, 12]	Benavides et al. [15]	Djebbi et al. [24]	Trinidad et al. [28, 26]	White et al. [99]	White et al. [97]	Abo Zaid et al. [1]	Fan et al. [35]	Wang et al. [92, 93]	Benavides et al. [14]	Benavides et al. [16]	Segura [70]	Bachmeyer et al. [4]	Cao et al. [20]	Fernandez et al. [36]	Hemakumar [41]	Gheyi et al. [38]	Kang et al. [43]	Mendonca et al. [55]	Osman et al. [59, 60]	Salinesi et al. [66]	Van den Broek et al. [84]	Van Deursen et al. [88]	Von der Massen et al. [90]	Von der Massen et al. [91]	White et al. [95, 96]	Batory et al. [7]	Schobbens et al. [42, 68, 69]	Trinidad et al. [80]	Von der Massen et al. [89]
	PL												CP						DL			Multi			Others														No support			
Void feature model	+	+		+		+	+		+	+	+	+	+	+	+	+				+	+	+	+	+		+			+	⊕		+	+	+	+	+			~	~	
#Products		+		⊕									+	+	+							+	+	+				+							⊕		+				~	
Dead features		~				+				+	+	+				+			+											⊕		+	+	+					~		~	~
Valid product	+	+	+	+			+										+								+					⊕					+				~	~	~	
All products	+		+	⊕			+						+								+		+			+				⊕				+	⊕				~		~	
Explanations	+	~					+									+			+											⊕		+		+					~		~	
Refactoring			+				⊕	+													+							+												~	~	
Optimization													⊕		+																	+								~		~	
Commonality													⊕			+						+					+														~	
Filter		+											⊕		+																			+							~	
Valid partial configuration	+	+							+																					⊕										~	
Atomic sets					+					⊕														+							+											
False optional features										⊕	+					+																										~	⊖
Corrective explanations																	+															+										⊖
Dependency analysis																														⊕	+										
ECR					⊕	+																																				
Generalization			⊕					+																																		
Core features						+																																			⊖	
Variability factor													⊕																													~	
Arbitrary edit								+																																		
Conditional dead features																											+															
Homogeneity																										+																
LCA					+																																					
Muti–step configuration																		+																									
Roots features					+																																					
Specialization								+																																		
Degree of orthogonality		~																																								
Redundancies																																										~
Variant features																																									~	
Wrong cardinalities																																									~	
Feature model notation	B	C	B	B	B	B	B	B	B	B	C	B	B	C	C	B	B	B	B	B	B	B	C	B	B	B	C	B	B	B	C	C	C	B	B	B	B	B	C	C	B
Extended feature model													+		+		+	+	+		+									+		+					+		+		+	
Formalization			+		+	+	+	+		+	+		+			+		+		+					+		+		+			+							+		

+ Supported ~ No support ⊕ Supported(first reference) ⊖ No support (first reference) B Basic feature model C Cardinality–based feature models

Table 3: Summary of operations and support

- Third most visited paper in the history of the journal (Scopus)
- The most cited in the period of 2009-2014 (Scopus)
- More than 1.500 citations (GS)

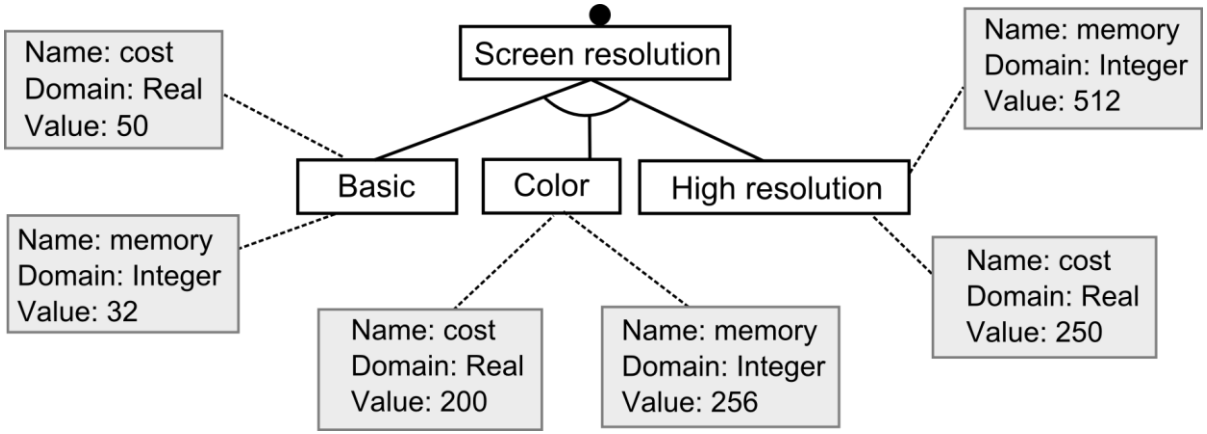
David Benavides, Sergio Segura, Antonio Ruiz Cortés: Automated analysis of feature models 20 years later: A literature review. Inf. Syst. 35(6): 615-636 (2010)

Challenge 1: Automated Analysis of FM

Ch 1.1 with attributes

Ch 1.2 with configuration paths

Challenge 1.1: Automated Analysis of Feature Models with attributes

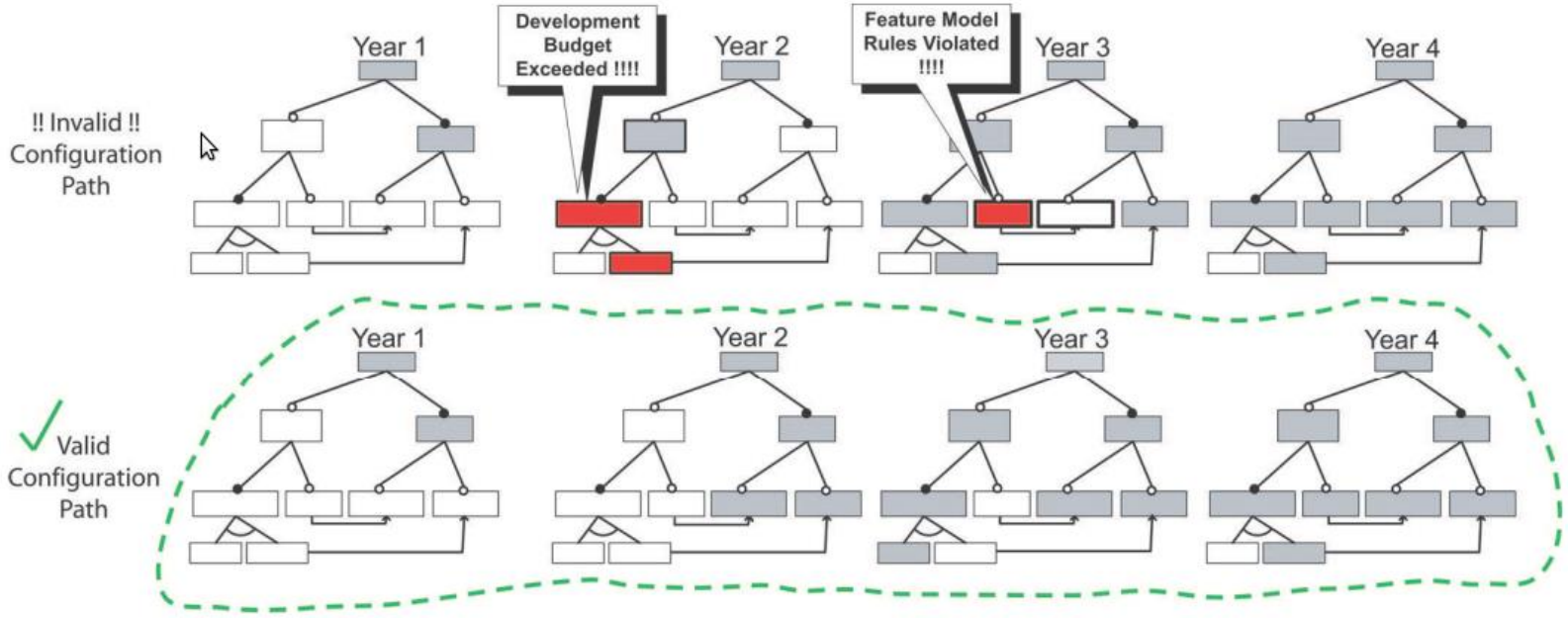


- Most Influential Paper Award 2017
- Rejected at a local conference
- Among the most influential paper in the 25th anniversary of the conference
- More tan 930 citations (Gscholar)

• David Benavides, Pablo Trinidad Martín-Arroyo, Antonio Ruiz Cortés: Automated Reasoning on Feature Models. CAiSE 2005: 491-503

• F Roos-Frantz, D Benavides, A Ruiz-Cortés, A Heuer, K Lauenroth Quality-aware analysis in product line engineering with the orthogonal variability model. Software Quality Journal

Challenge 1.2: Automated Analysis of Feature Models Configuration Paths



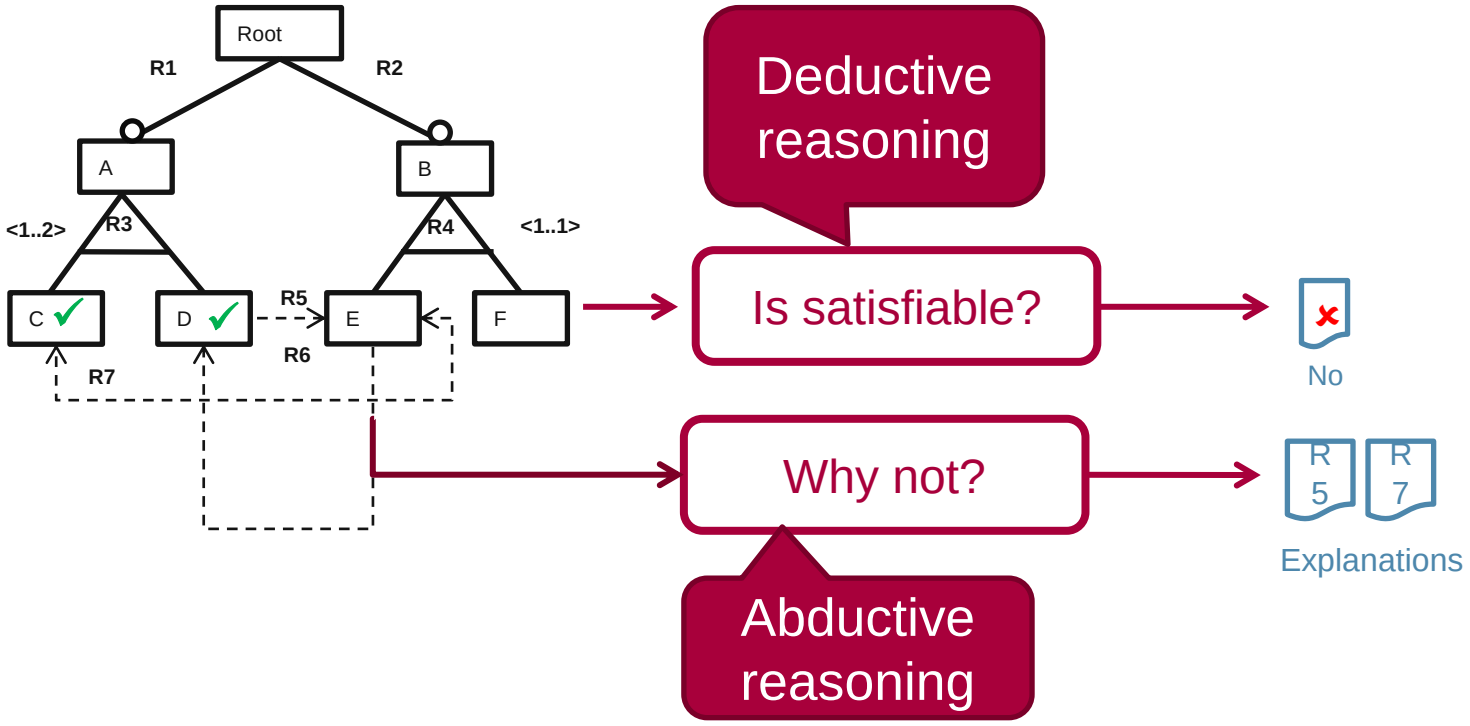
Jules White, José A. Galindo, Tripti Saxena, Brian Dougherty, David Benavides, Douglas C. Schmidt: Evolving feature model configurations in software product lines. Journal of Systems and Software 87: 119-136 (2014)

Challenge 2: Explanations on FM analysis

Ch 2.1 with feature models

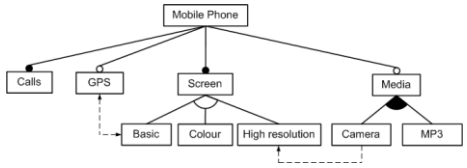
Ch 2.2 with configurations

Challenge 2: Explanations on the Automated analysis of SPL



Challenge 2: Explanations on the Automated analysis of SPL

Ch 2.1 with feature models



Pablo Trinidad, David Benavides, Amador Durán, Antonio Ruiz Cortés, Miguel Toro: Automated error analysis for the agilization of feature modeling. Journal of Systems and Software 81(6): 883-896 (2008)

Ch 2.2 with configurations

Jules White, David Benavides, Douglas C. Schmidt, Pablo Trinidad, Brian Dougherty, Antonio Ruiz Cortés: Automated diagnosis of feature model configurations. Journal of Systems and Software 83(7): 1094-1107 (2010)

Alexander Felfernig, Rouven Walter, José A. Galindo, David Benavides, Seda Polat Erdeniz, Müslüm Atas and Stefan Reiterer. Anytime Diagnosis for Reconfiguration. Journal of Intelligent Information Systems (2019).

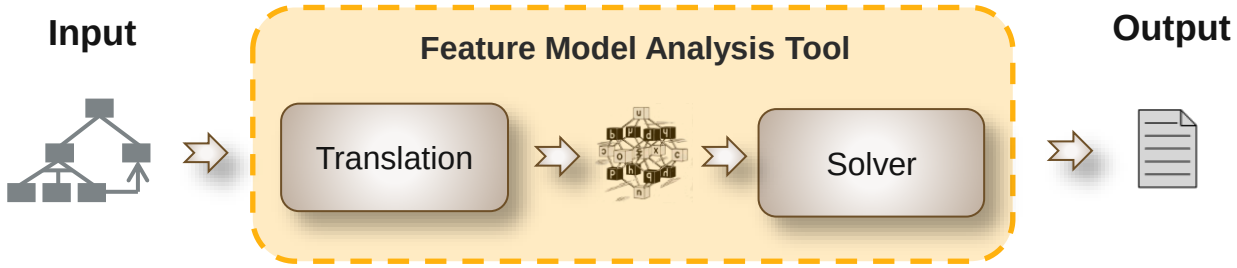
Ch 3.1
Functional
Testing

Ch 3.2
Performance
Testing

Challenge 3: Testing on FM analysis tools

Challenge 3.1: Functional Testing

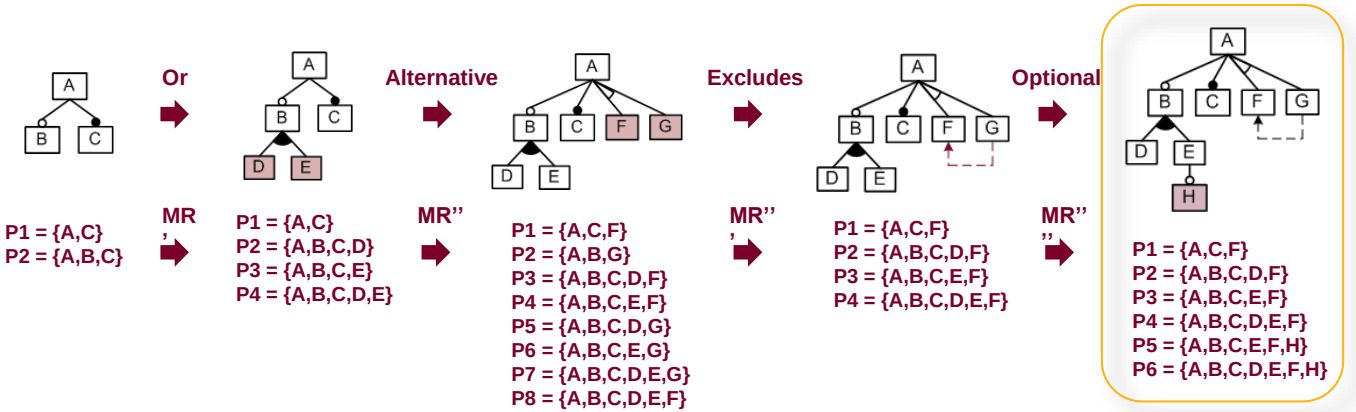
How to detect faults in feature model analysis tools?



- SAT, CSP, DL, OWL...
- Large programs.

Time-consuming
Error-prone

Challenge 3.1: Functional Testing



Operation

Does the product contain any dead feature?

Is the model consistent?

Expected output

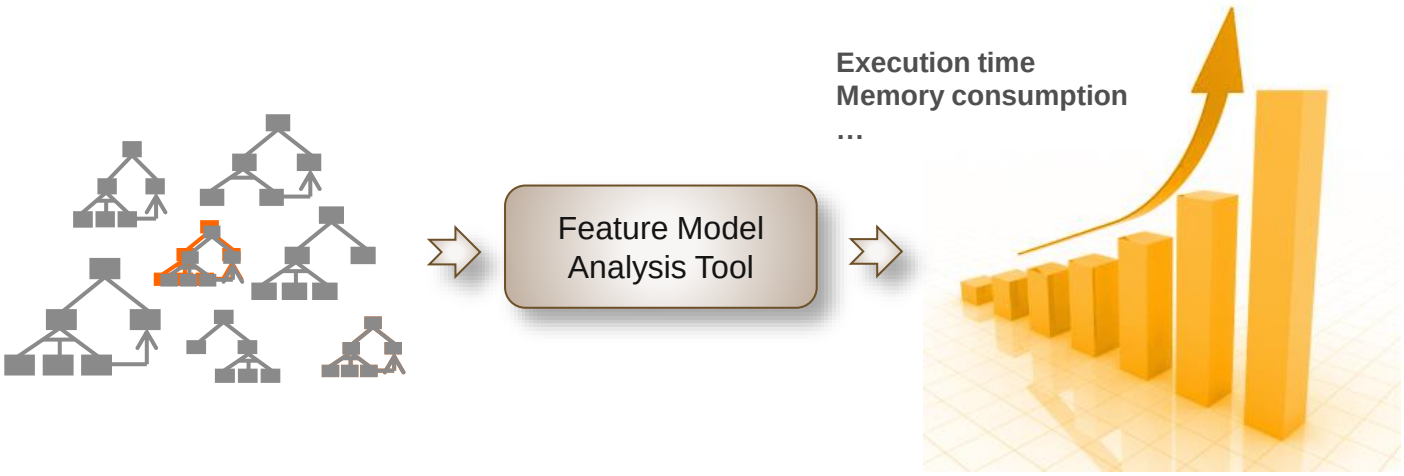
Yes, the product contains 6 products

Yes, the model is consistent

Sergio Segura, Robert M. Hierons, David Benavides, Antonio Ruiz Cortés: Automated metamorphic testing on the analyses of feature models. Information & Software Technology 53(3): 245-258 (2011)

Challenge 3.2: Performance Testing

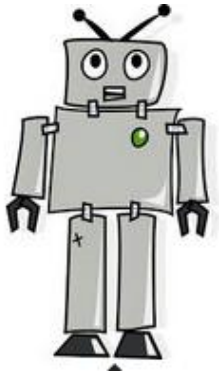
How to know the performance of FM analysis tools in pessimistic cases?



Sergio Segura, José Antonio Parejo, Robert M. Hierons, David Benavides, Antonio Ruiz Cortés: Automated generation of computationally hard feature models using evolutionary algorithms. Expert Syst. Appl. 41(8): 3975-3992 (2014)

Challenge 4: Application of FM analysis

Beyond (software) product lines



Some results from the literature

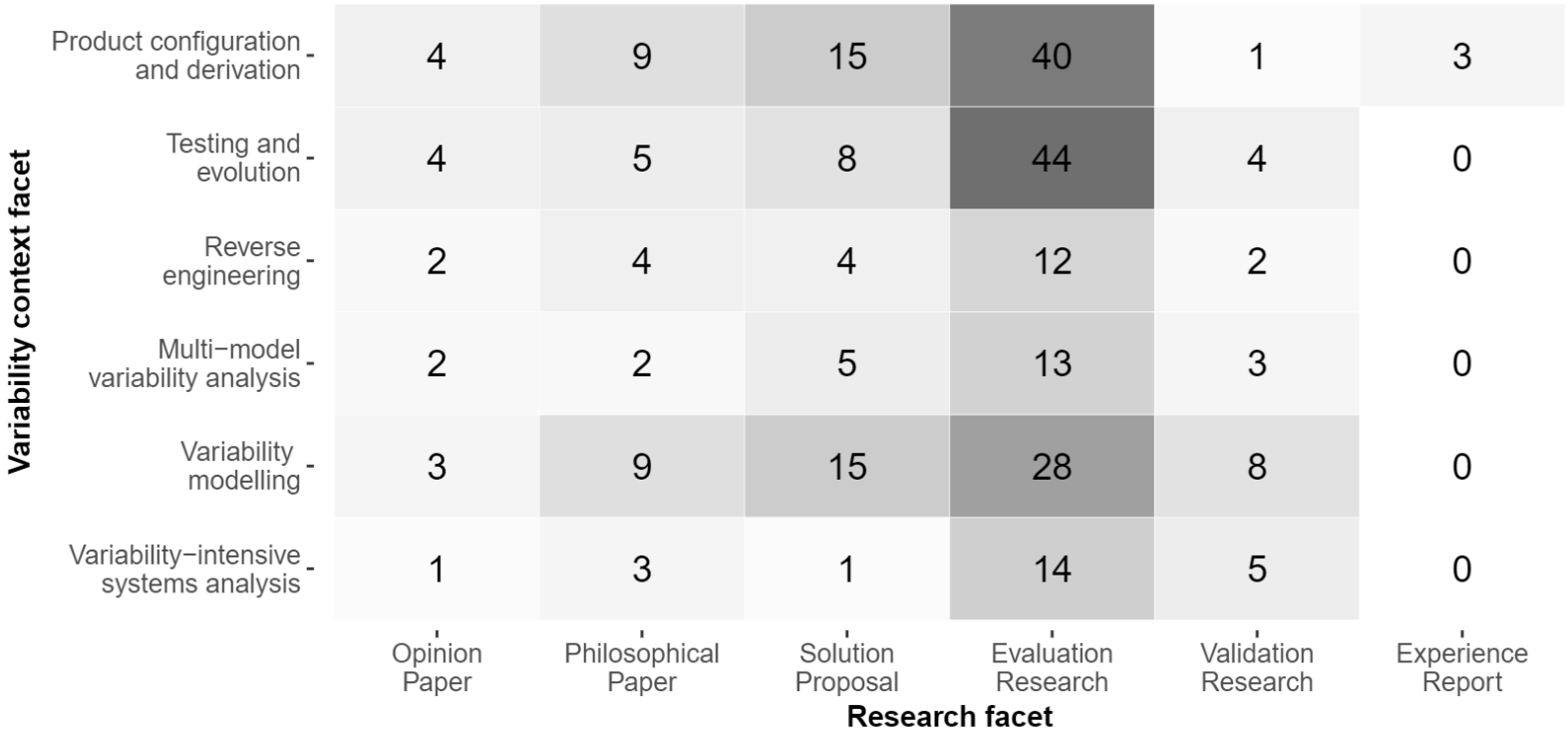
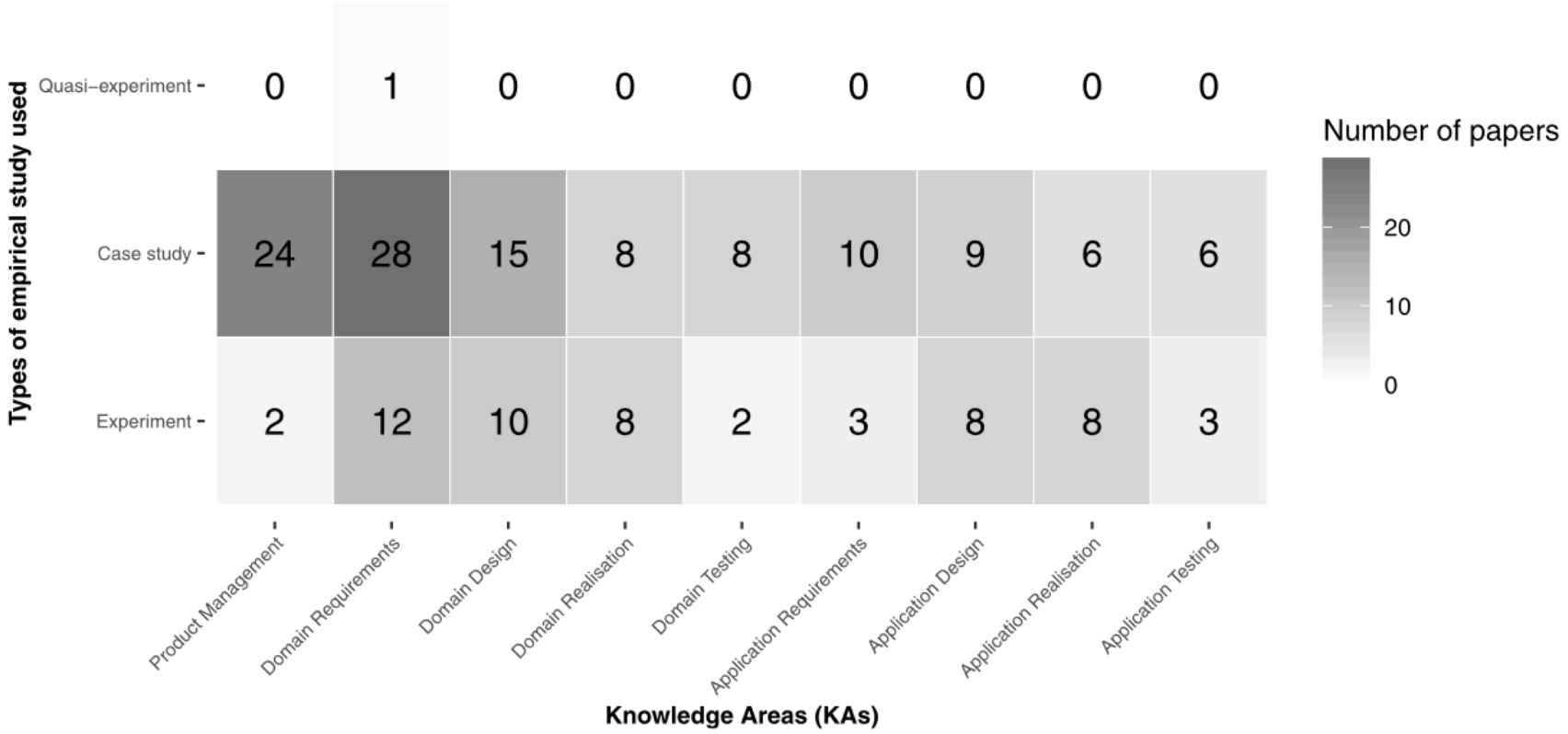


Fig. 11: Visualization of the systematic map

José A. Galindo, David Benavides, Pablo Trinidad, Antonio Manuel Gutiérrez-Fernández, Antonio Ruiz-Cortés: Automated analysis of feature models: Quo vadis? Computing 101(5): 387-433 (2019)

Some results from the literature



Ana Eva Chacón-Luna, Antonio Manuel Gutiérrez , José A. Galindo, David Benavides: Empirical software product line engineering: A systematic literature review. Inf. Softw. Technol. 128: 106389 (2020)

Examples of applications

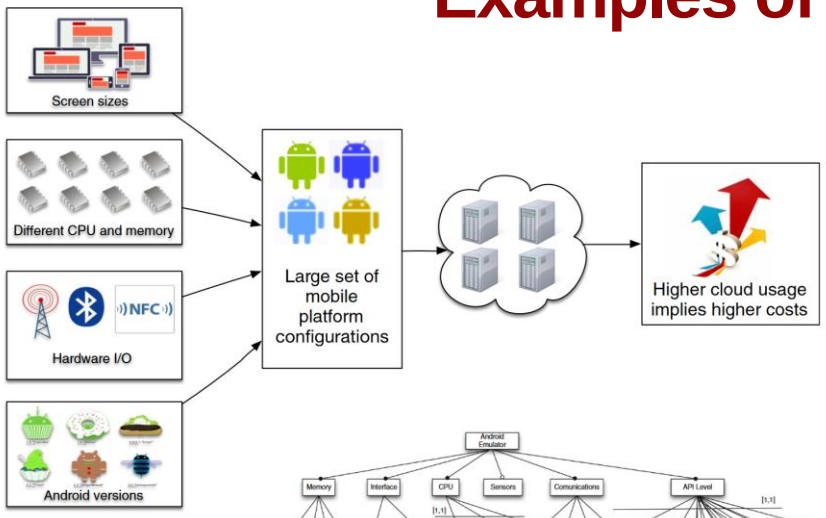


Fig. 3 Android variability

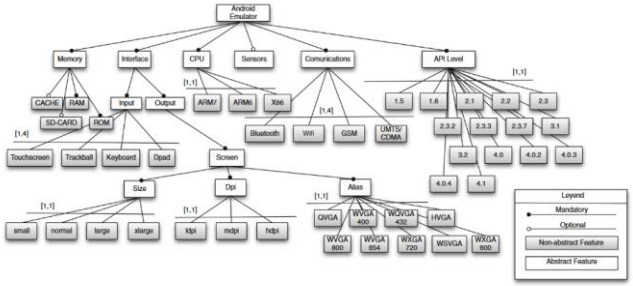


Fig. 8 An Android feature model

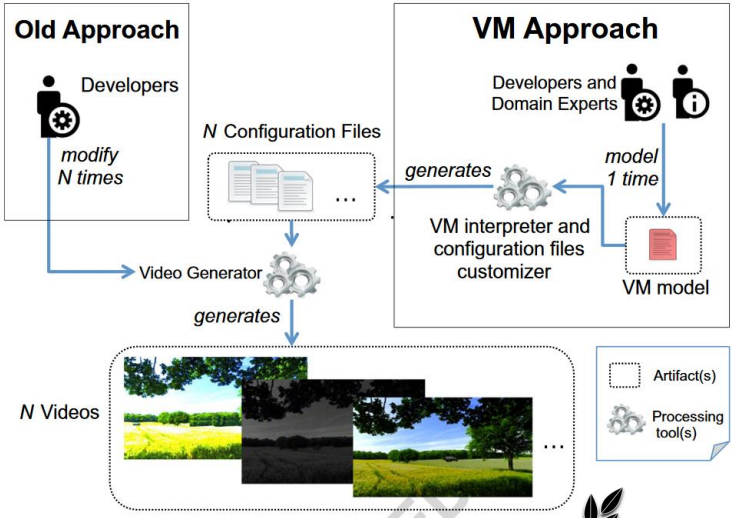


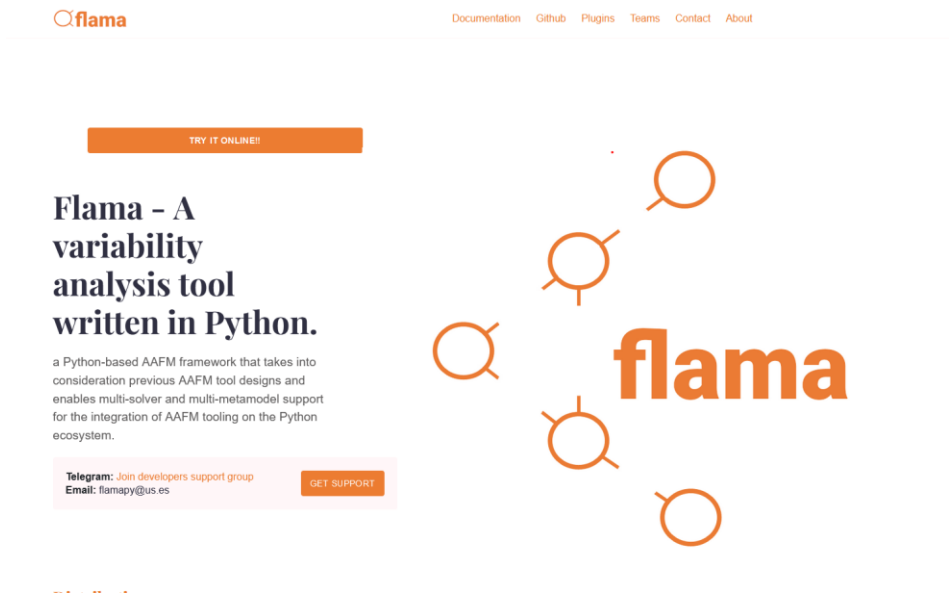
Fig. 1 Old process compared to the VM-based process for video generation

José A. Galindo, Hamilton A. Turner, David Benavides, Jules White: Testing variability-intensive systems using automated analysis: an application to Android. Software Quality Journal 24(2): 365-405 (2016)

Mauricio Alférez, Mathieu Acher, José A. Galindo, Benoit Baudry, David Benavides: Modeling variability in the video domain: language and experience report. Software Quality Journal

Tooling the Automated analysis of SPL

Tooling the Automated analysis of SPL



http://www.fama-ts.us.es/
https://github.com/FaMaFW/FaMA

https://flamapy.github.io/

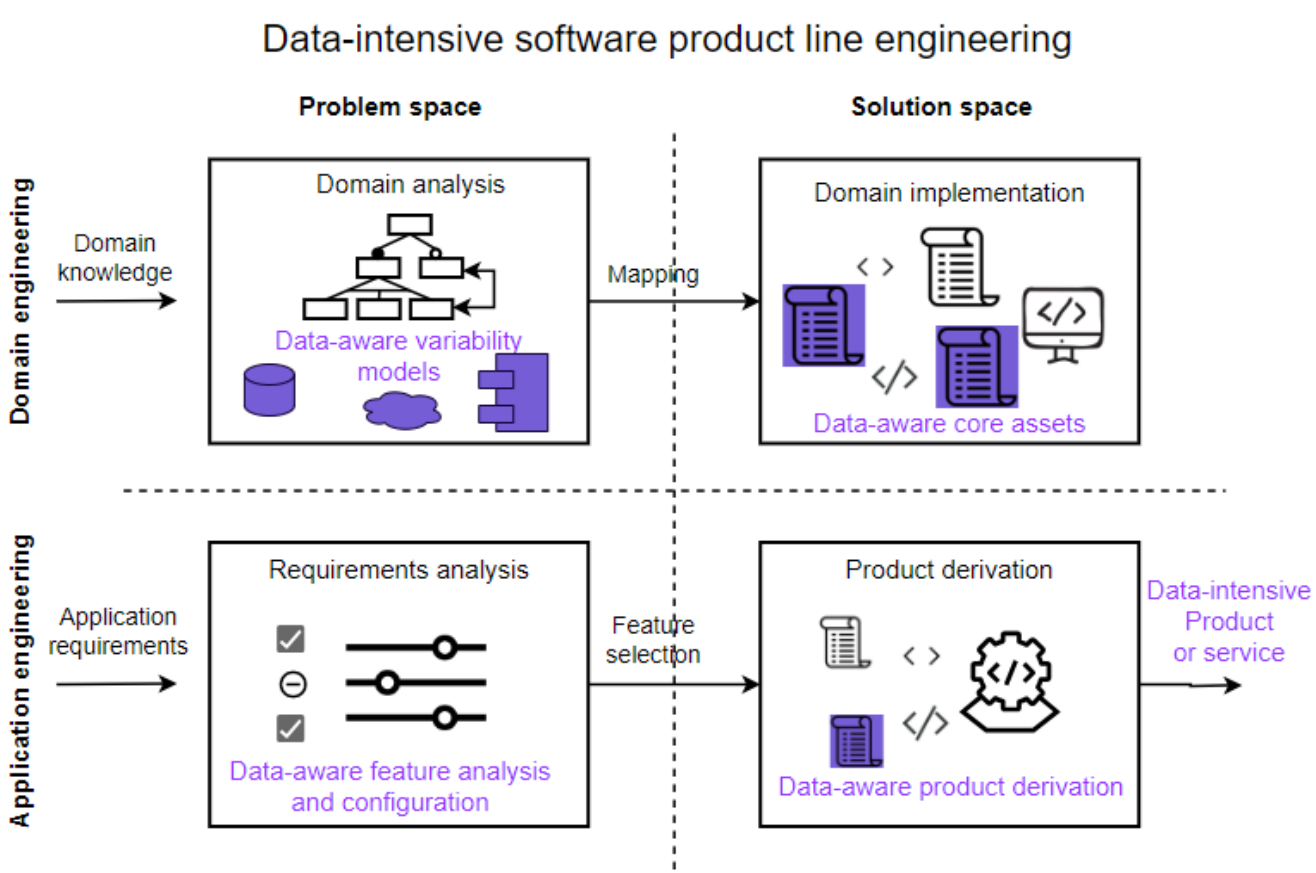
David Benavides, Sergio Segura, Pablo Trinidad, Antonio Ruiz Cortés: FAMA: Tooling a Framework for the Automated Analysis of Feature Models. VaMoS 2007: 129-134

The future

DATA-INTENSIVE
SOFTWARE
PRODUCT LINES

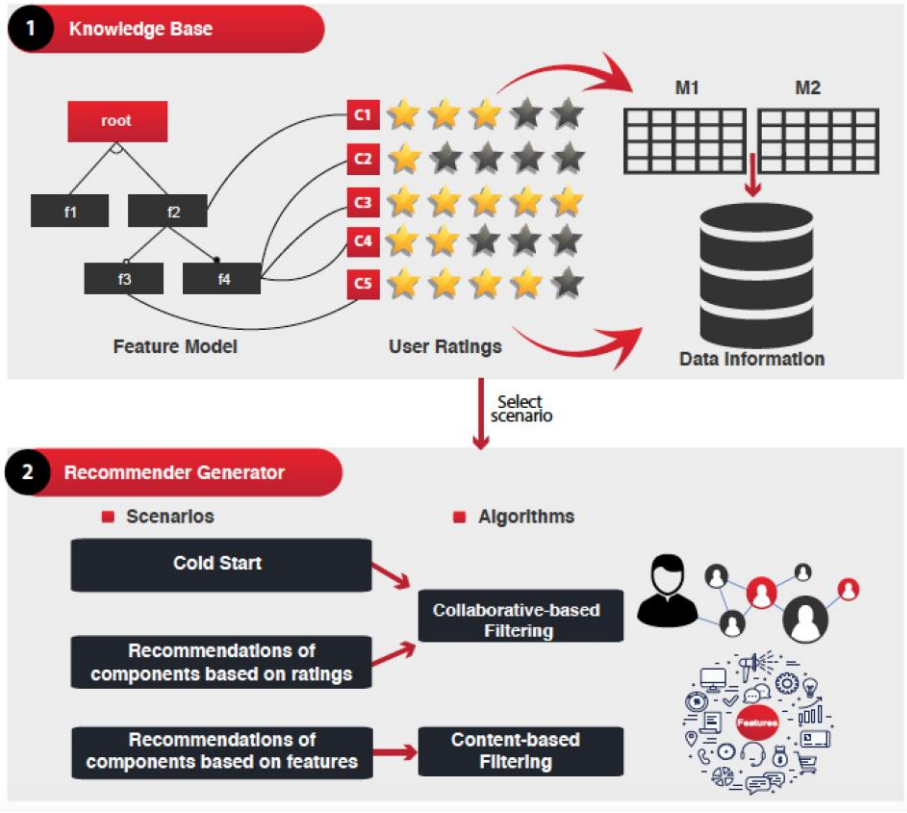
DATA-INTENSIVE SOFTWARE PRODUCT LINES

Challenge: Modeling and analysing new dimensions in data-intensive software product lines



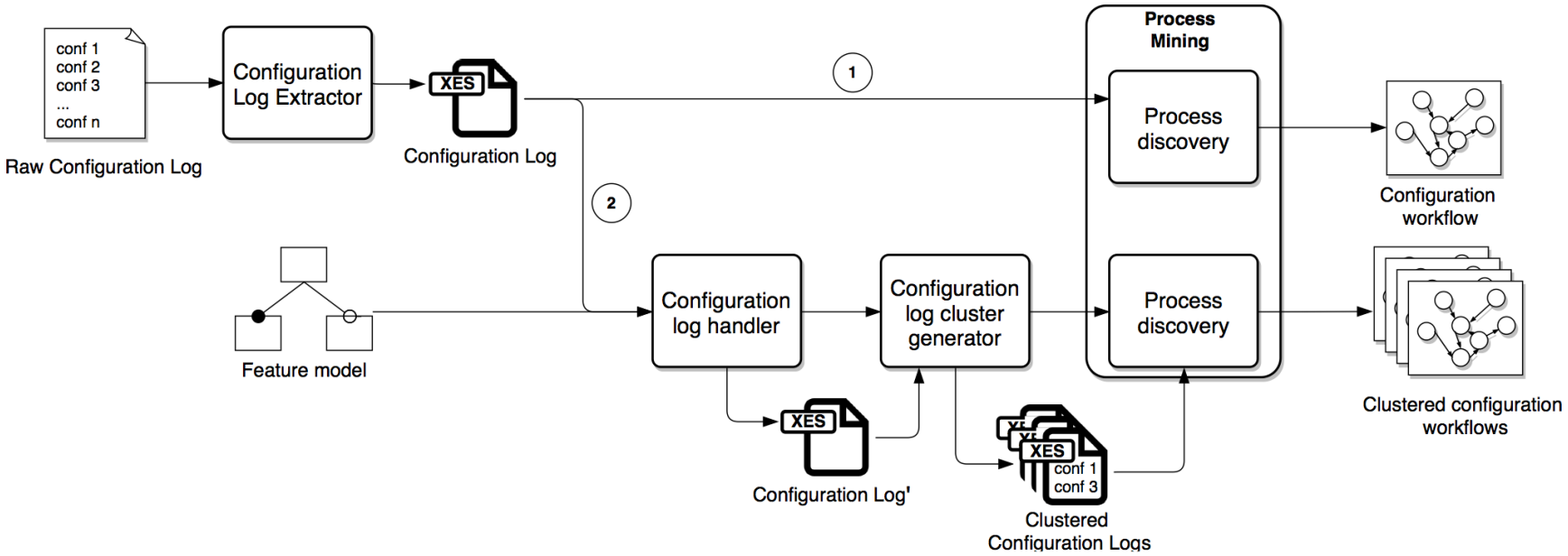
Challenge: Engineering platforms for data-intensive software product lines

Data aware configurations



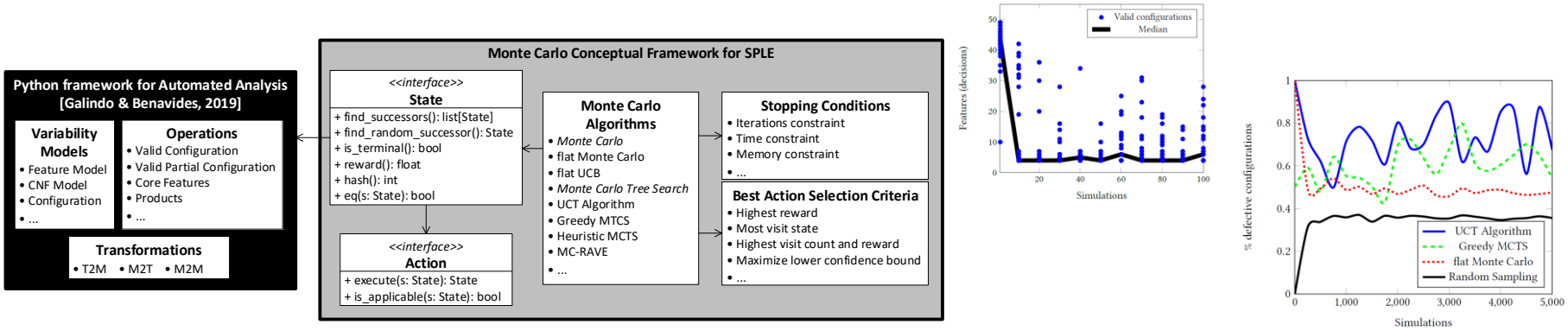
Jorge Rodas-Silva, José Angel Galindo, Jorge García-Gutiérrez, David Benavides: Selection of Software Product Line Implementation Components Using Recommender Systems: An Application to Wordpress. IEEE Access 7: 69226-69245 (2019)

Configuration workflows mining



B. Ramos-Gutiérrez, Á.J. Varela-Vaca , J. A. Galindo, M.Teresa Gómez-López, D. Benavides:Discovering configuration workflows from existing logs using process mining. Empir. Softw. Eng. 26(1): 11 (2021)

Sampling based analyses



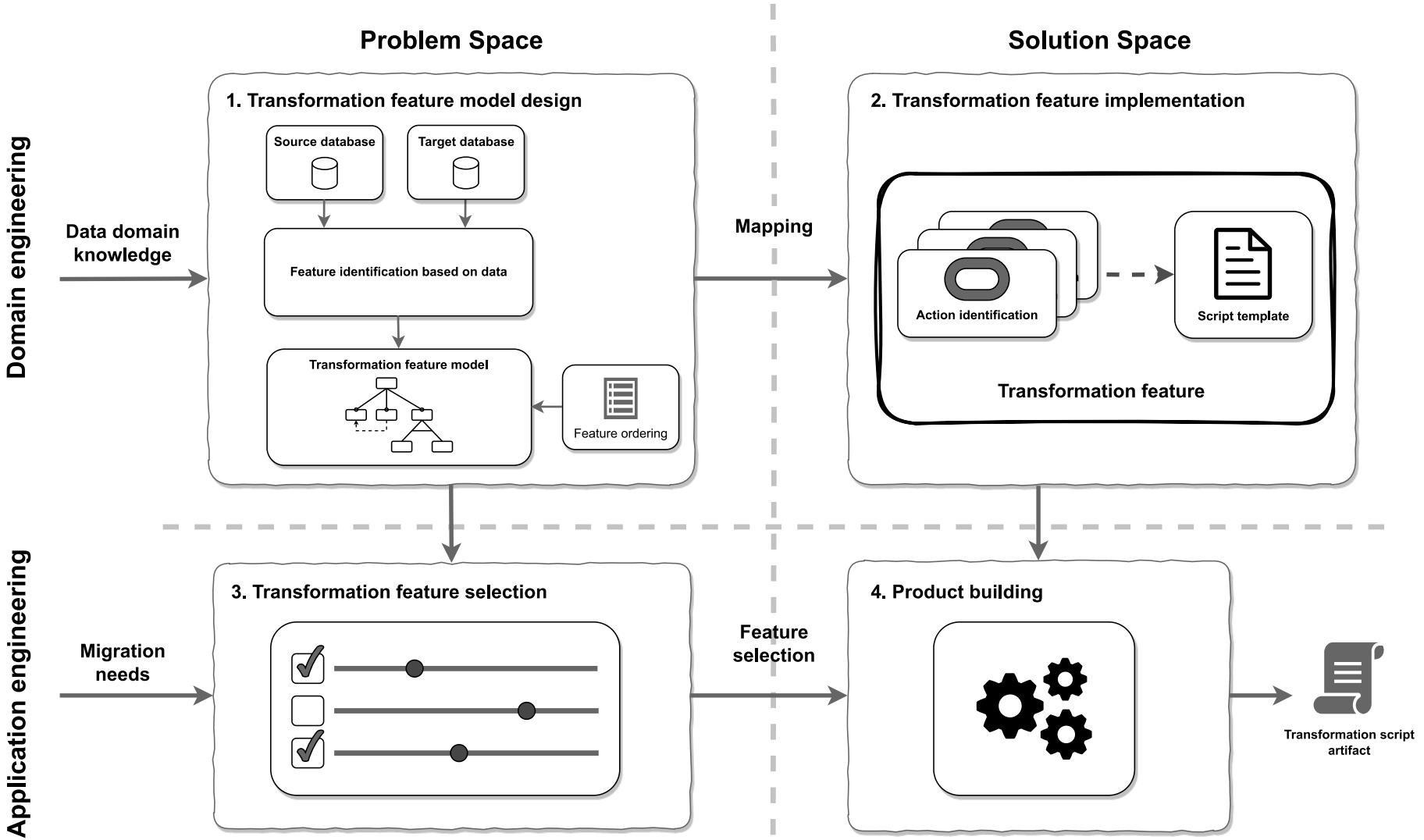
https://github.com/diverso-lab/fm_montecarlo

Ruben Heradio, David Fernández-Amorós, José A. Galindo, David Benavides, Don S. Batory:Uniform and scalable sampling of highly configurable systems. Empir. Softw. Eng. 27(2): 44 (2022)

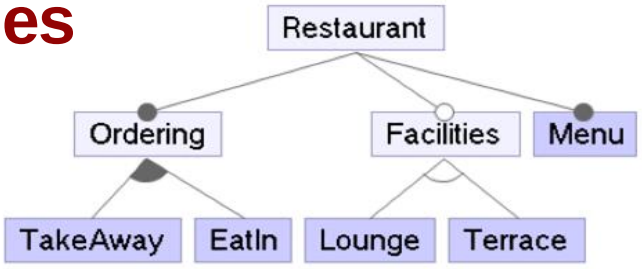
José Miguel Horcas, José A. Galindo, Ruben Heradio, David Fernández-Amorós, David Benavides: A Monte Carlo tree search conceptual framework for feature model analyses. J. Syst. Softw. 195: 111551 (2023)

José Miguel Horcas Aguilera, A. Germán Márquez, José A. Galindo, David Benavides: Monte Carlo Simulations for Variability Analyses in Highly Configurable Systems. ConfWS 2021: 37-44

Data migration product lines

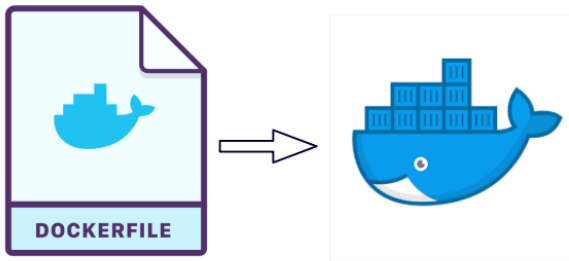


Chat bot product lines



$\neg EatIn \Rightarrow \neg(Lounge \vee Terrace)$

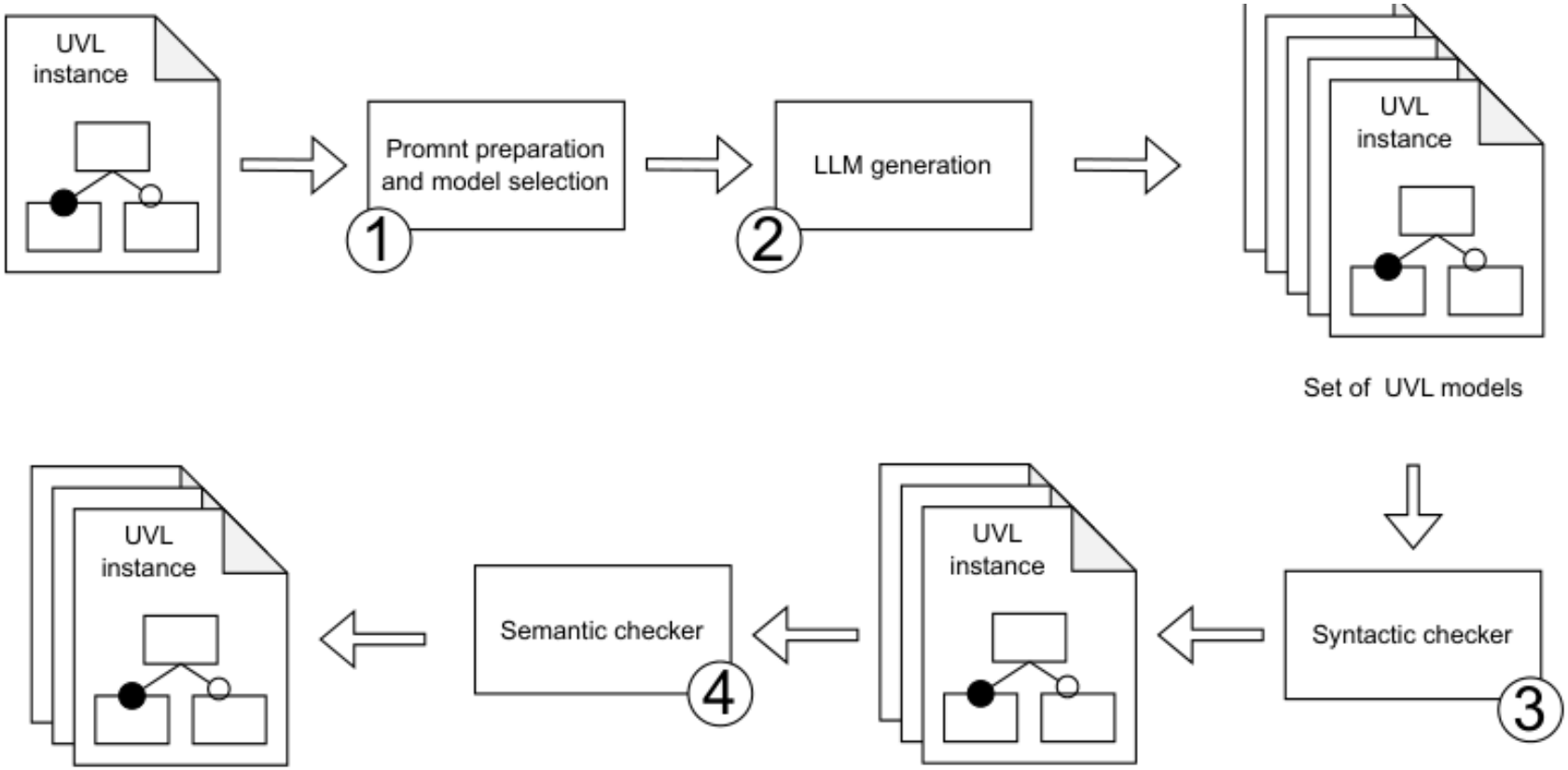
```
_id: ObjectId('64496b42be2a6a8b8ece5f7e')
owner: ObjectId('64496a4bbe2a6a8b8ece5f63')
name: "botR1"
description: "Chatbot for R1"
intents: Array
  0: ObjectId('64496aa7be2a6a8b8ece5f6a')
  1: ObjectId('64496aa7be2a6a8b8ece5f6c')
  2: ObjectId('64496aa7be2a6a8b8ece5f70')
  3: ObjectId('64496aa7be2a6a8b8ece5f6e')
fallback: "No dispongo de información relacionada
pl: ObjectId('64496aa7be2a6a8b8ece5f74')
compiled: true
createdAt: 2023-04-26T18:19:46.599+00:00
updatedAt: 2023-04-26T18:20:12.941+00:00
__v: 0
```



Docker file

Docker Image

LLMs and variability models



José A. Galindo, Antonio J. Dominguez, Jules White, David Benavides: Large Language Models to generate meaningful feature model instances. SPLC (A) 2023: 15-26

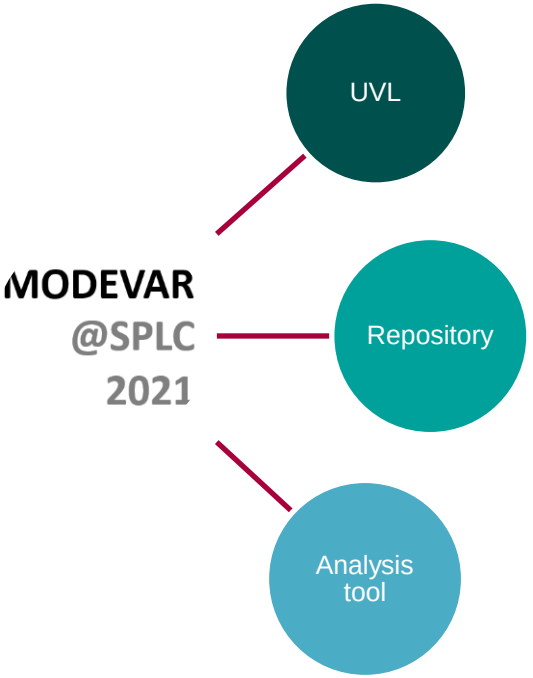
CONSOLIDATION

Batory's test of time award talk (2017)

WHAT DOES THE NIGHT SKY OF SOFTWARE SCIENCE LOOK LIKE ?

KNOWLEDGE CONSOLIDATION

2018 @ SPLC → MODEVAR@SPLC 2019. Paris, France. → MODEVAR@VaMoS 2020. Magdeburg, Germany. MODEVAR@SPLC 2020. Montréal, Canada → MODEVAR@SPLC 2021. Leicester, United Kingdom. → MODEVAR@SPLC 2022. Graz. Austria



https://modevar.github.io/

OA Publishing Agreement SPRINGER NATURE

This Open Access Publishing Agreement (this "**Agreement**") has been approved by and entered into between

Prof. David Benavides, University of Seville, ETS de Ingeniería Informática, Avenida Reina Mercedes s/n, 41012 Sevilla, Spain
Prof. Alexander Felfernig, Graz University of Technology, Inst of Softwaretechnology, Inffeldgasse 16b, 8010 Graz, Austria
Dr. Andreas Falkner, Siemens (Austria), Corporate Technology, Siemensstraße 90, 1210 Wien, Austria
Prof. Don Batory, The University of Texas at Austin, Dept of Computer Science, 2317 Speedway, Austin, TX 78712, USA
(the "**Author**")

Prof. David Benavides
(the "**Corresponding Author**")

on the one part and

Springer Nature Switzerland AG, Gewerbestrasse 11, 6330 Cham, Switzerland
(the "**Publisher**")

on the other part

together hereinafter referred to as the "**Parties**".

Some conclusion

- Variability is a fundamental part of software science. Our problems are difficult but are important to solve. If we don’t solve them, others will.
- Be aware of our own history. If you don’t know it, you can repeat it.
- We passed the variability hype. Be prepared to come back to normal.
- Do not give up when rejections come to you. Something amazing can happen afterwards
- Data intensive software product lines will be important in the following years

Data-intensive product lines: embracing past results and new variability challenges

ありがとうございました

Prof. David Benavides

benavides@us.es

Tokyo, SPLC 2023

https://grupo.us.es/diversolab/

@davbencue