Tesis Doctoral por Compendio de Publicaciones

Achieving Energy Efficiency with a Software Product Line Engineering Approach

UNIVERSIDAD DE MÁLAGA

Daniel Jesús Muñoz Guerra

Programa de Doctorado en Tecnologías Informáticas
Departamento de Lenguajes y Ciencias de la Computación

Universidad de Málaga

Supervised by

Prof. Dr. Lidia Fuentes Fernández
Dr. Mónica Pinto Alarcón

July 2023

AUTOR: Daniel Jesús Muñoz Guerra

https://orcid.org/0000-0002-1398-9423

EDITA: Publicaciones y Divulgación Científica. Universidad de Málaga

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD DE LA TESIS PRESENTADA PARA OBTENER EL TÍTULO DE DOCTOR

D./Dña DANIEL JESÚS MUÑOZ GUERRA
Estudiante del programa de doctorado EN TECNOLOGÍAS INFORMÁTICAS de la Universidad de Málaga, autor/a de la tesis, presentada para la obtención del título de doctor por la Universidad de Málaga, titulada: ACHIEVING ENERGY EFFICIENCY WITH A SOFTWARE PRODUCT LINE ENGINEERING APPROACH

Realizada bajo la tutorización de LIDIA FUENTES FERNÁNDEZ y dirección de LIDIA FUENTES FERNÁNDEZ Y MÓNICA PINTO ALARCÓN (si tuviera varios directores deberá hacer constar el nombre de todos)

DECLARO QUE:

La tesis presentada es una obra original que no infringe los derechos de propiedad intelectual ni los derechos de propiedad industrial u otros, conforme al ordenamiento jurídico vigente (Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia), modificado por la Ley 2/2019, de 1 de marzo.

Igualmente asumo, ante a la Universidad de Málaga y ante cualquier otra instancia, la responsabilidad que pudiera derivarse en caso de plagio de contenidos en la tesis presentada, conforme al ordenamiento jurídico vigente ya que nada ha sido utilizado en tesis anteriores.

En Málaga, a 14 de JULIO de 2023

Fdo.: DANIEL JESÚS MUÑOZ GUERRA Doctorando/a	Fdo.: LIDIA FUENTES FERNÁNDEZ Tutor/a
Fdo.: LIDIA FUENTES FERNÁNDEZ Director/es de tesis	MÓNICA PINTO ALARCÓN

Edificio Pabellón de Gobierno. Campus El Ejido. 29071
Tel.: 952 13 10 28 / 952 13 14 61 / 952 13 71 10
E-mail: doctorado@uma.es

UNIVERSIDAD DE MÁLAGA
DEPARTAMENTO DE LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN

La Prof. Dr. Doña Lidia Fuentes Fernández, Catedrática de Universidad, perteneciente al Área de Ingeniería Telemática, y la Dr. Doña Mónica Pinto Alarcón, Titular de Universidad, perteneciente al Área de Lenguajes y Sistemas Informáticos, de la E.T.S. de Ingeniería Informática de la Universidad de Málaga,

certifican que Don Daniel Jesús Muñoz Guerra, Ingeniero en Informática, Astrónomo y Astrofísico, ha realizado en el Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga, bajo nuestra dirección, el trabajo de investigación correspondiente a su Tesis Doctoral titulado:

Achieving Energy Efficiency with a Software Product Line Engineering Approach

En ella se han propuesto aportaciones originales a la ingeniería del software y los resultados han dado lugar a las publicaciones indexadas que avalan esta "Tesis por compendio de publicaciones" y que no han sido utilizadas en tesis anteriores:

1. Transforming Numerical Feature Models into Propositional Formulas & the Universal Variability Language
2. Uniform Random Sampling Product Configurations of Feature Models That Have Numerical Features
3. Finding Correlations of Features Affecting Energy Consumption and Performance of Web Servers Using the HADAS Eco-Assistant
4. Energy-Aware Environments for the Development of Green Applications for Cyber-Physical Systems
5. Detecting Feature Influences to Quality Attributes in Large and Partially Measured Spaces Using Smart Sampling and Dynamic Learning
6. Category Theory Framework for Variability Models with Non-functional Requirements
7. Quality-aware Analysis and Optimisation of Virtual Network Functions

Por todo ello, consideran que esta Tesis es apta para su presentación al Tribunal que ha de juzgarla. Y para que conste a efectos de lo establecido, se AUTORIZA la presentación de esta Tesis en la Universidad de Málaga.

Málaga, julio de 2023

Fdo.: Lidia Fuentes Fernández Fdo.: Mónica Pinto Alarcón

Acknowledgements

This PhD. thesis has been supported by the European Union's H2020 research and innovation programme under grant agreement *DAEMON* H2020-101017109, by the projects *IRIS* PID2021-12281 2OB-I00 (co-financed by FEDER funds), *Rhea* P18-FR-1081 (MCI/ AEI/ FEDER, UE), and *LEIA* UMA18-FEDERIA-157, and the PRE2019-087496 grant from the Ministerio de Ciencia e Innovación.

Special Acknowledgements

Destiny Is All!

† † † † † † † † †

Es un dilema complicado ponerse a escribir esto ahora que ya parece que ha acabado el largo proceso, pero toca reconocer la parte que les toca a los que estuvieron conmigo en el camino.

A mis supervisoras, o como yo las llamo fuera, las jefas. Habrá más tesis, pero sólo una es la primera.

A los compañeros del laboratorio 333 que siguen luchando por la investigación en este país que poco lo agradece. Mención especial a las risas durante los almuerzos y comidas, que hacen que el volver a sentarse a trabajar sea menos pesado.

A los amigos que he hecho y que sigo realizando en el planeta Tierra, que por suerte cada vez son más y mejores. Y en especial a los que siempre han estado ahí "no matter what".

A la familia, que siempre está ahí y ha contribuido a que esto sea posible, aunque nunca vayan a leer este documento.

Y a Josemi, por ser mi gran apoyo en el grupo, y por dejar que copie toda esta dedicatoria de su tesis.

Contents

List of Figures

List of Tables

Part I

Thesis

Chapter 1

Introduction

According to the *Organisation for Economic Co-operation and Development* (OECD), energy consumption is a fuel for economic growth, particularly in the fast-developing economies of the world [1]. Unfortunately, energy consumption is closely related to global climate change through greenhouse gas emissions. Any energy source, particularly the most common, burning fossil fuels, is the primary source of human-induced greenhouse gas emissions. The gasses in the atmosphere absorb solar energy and trap heat close to the Earth's surface, causing global warming. This is proven to cause extreme weather, wildfires, droughts, food supply disruptions, and even geopolitical wars, which will modify the existing international order [2]. The recent Russia and Ukraine war has led directly to emissions of 33 million tons of greenhouse gases by smart weapons, as well as an increase in electricity and gas prices which shows the dependency of first-world countries on petrostates [3]. **To minimise these problems, we need to reduce the energy requirements of the world**.

1.1 The Energy Consumption of the Software

The software runs on hardware, and as the former grows, so does reliance on the machines to make it run. Software systems do not consume energy themselves but affect hardware utilisation, leading to indirect energy consumption. This is especially critical in edge computing systems based on smart devices like *Cyber-Physical*

Systems (**CPSs**), which are powered by low batteries due to lithium scarcity (e.g., smartwatches) [4]. Reducing the energy consumption of CPS requires a comprehensive study of the various actors, components and surrounding environment. Energy consumption is not only limited to the energy impact of the device itself but the entire ecosystem impacted by the usage of CPS (such as networking, servers and data centres, database storage, etc.). According to a report by the *International Energy Agency* (IEA), just data centres and running networks consume 1% of the world's electricity[5]. The researchers have proved mathematically that relatively simple hardware modifications could cut the energy requirements in half. Energy-aware software design can reduce overall energy consumption by 30% to 90% [6]. In measuring the energy consumption of a software application, one must account for the processing elements and their interactions with the memory sub-system and other components during the execution of algorithms.

1.1.1 Modelling and Storing Energy Information

We must directly measure the energy consumption of the hardware components in real time to measure the running software's energy consumption accurately. There are different ways to obtain energy information depending on the type of system and level of detail needed. One way is to use simulation tools that can model the energy consumption of a system based on different parameters. Another way is to use specialised hardware to measure and store real-time energy information. **The measuring tools usually depend on the device or system to measure, so we may look for solutions that balance measuring accuracy, usability and availability**.

The most balanced energy measuring tools are power meters, which are devices that plug into the power outlet and measure the amount of electricity flowing through it, allowing users to measure total system or device-specific power consumption (e.g., Watts Up Pro?, USB meters) [1]. Another good alternative is hardware energy models with predicting software tools that display how much power any component uses for a specific process (e.g., Intel Power Gadget, Intel

[1]When calculating the energy consumption of a running process with power meters, we need first to measure and then subtract to the recording values the energy consumption of the operating system and other unrelated tasks.

PCM).

Depending on the tool, the energy readings are provided as the total energy consumption in Joules or the energy consumption rate in Watts. The Joules per task is a more interesting metric for battery-powered devices, so we can perform analyses to stretch battery life. But the most common metric is the Watts per task, which analyses aim to reduce the electricity bill, and hence it is more interesting for electricity-powered devices [6]. Nevertheless, as the measuring tools also report the run-time in Seconds, we can easily convert Joules in Watts and vice-versa [1].

To model and store these readings, we could follow the standard approach of describing them as a characteristic of individual components. The most straightforward case is the monetary cost, where the total cost aggregates the individual costs. For example, the price of a computer is the added cost of the *Computer Processing Unit* (**CPU**), the motherboard, the different memories and other peripherals. While this could be the case for other efficiency qualities such as run-time, it would be highly inaccurate for energy consumption. For example, every CPU has an advertised peak energy-consumed value (i.e., Wattage), but the constant use of peak power would destroy (i.e., burn) that hardware. Even in the fictional scenario of running a stress-CPU benchmark, the thermal throttle of the CPU will down-clock it to prevent burning the electronics, reducing the true Wattage compared to the peak value. Consequently, the **energy consumption values present many interactions between components; hence, it is not feasible to describe them by individual and static energy values. Further, the aggregation function will be utterly complex, as it must register all levels of interaction and different behaviours.**

We can find several intents to store and collaboratively populate databases with the energy information of running systems. One of them is the already mentioned IEA which provides free datasets, including world aggregated data for different countries in different periods [7]. Another database is Datarade which provides energy consumption data commonly used for energy efficiency and sustainability analysis [8]. While the energy information is accurate, the system's description is

[1] *Joules* (**J**) is the unit of energy. It is defined as the amount of work done when a force of one newton is applied over a distance of one meter. *Seconds* (**S**) is the unit of time. *Watts* (**W**) is the unit of power. It is the rate at which work is done, or energy is transferred per unit time. One watt is equal to one joule per second.

scarce. The reason is that **databases do not scale for highly configurable systems**, as complex database queries suffer from the curse of dimensionality [1] when performing many-to-many table-joins [9].

1.1.2 Unmeasured Alternatives in Colossal Systems

To measure the energy consumption of a complete family of systems is generally impractical due to the requiring the full availability of hardware and measuring tools, as well as the expertise in running and evaluating all the different methods. This is especially proven with CPSs, which are characterised by high configurability and adaptability to changing environments. Consequently, they present many alternatives and a colossal number of different running systems. For example, just the kernel of their operating system (i.e., Linux) comprises 6467 alternatives and Boolean variables, which represents a family of $\sim 3.90 \times 10^{1672}$ different versions of CPS systems [10]. This number is called the size of the search/solution space. Its proper definition is the set of all possible points (sets of values of the choice variables) of an optimisation problem that satisfy the problem's constraints, potentially including inequalities, equalities and integer constraints. Linux presents a colossal solution space, making an exhaustive exploration intractable to optimise the energy consumption of CPSs executing Linux software.

Consequently, **we tend to work with incomplete solution spaces that are not entirely measured for energy consumption**. Hence, we deal with partially measured solution spaces, where some family members are not fully known and measured. Somewhat unknown solution spaces are common in many fields, such as computer engineering, machine learning, artificial intelligence and goal-oriented optimisation.

1.2 Constraint Satisfaction Problems

Constraint Satisfaction Problems (**CSPs**) are mathematical questions defined as a set of objects whose state must satisfy several constraints [11]. CSPs represent the

[1] The Curse of Dimensionality is a phenomenon typically increasing computational efforts required for the processing and analysis of high-dimensional spaces.

entities in a problem as a homogeneous collection of finite constraints over variables solved by constraint satisfaction methods. The solution space of software and systems can be represented as CSPs, where the different configurable alternatives are the objects and the limitations are the constraints. CSP on finite solution spaces is typically solved using a form of search. The most used techniques to generate the solution space of a CSP are variants of backtracking, constraint propagation, and local search [12].

1.2.1 Variability Models and Numerical Characteristics

Variability models (**VMs**) are a tree-like structure used to represent the commonalities and differences in a CSP. Hence, we can describe a family of software and systems as VMs. The nodes of the tree represent alternatives to build different solutions, and the edges represent propositional dependencies between the nodes [13]. An example of a technical dependency is that to execute a `Java Library`, we require a `Java Virtual Machine` running – we can represent both components as VM nodes located in different branches of the same VM tree.

Numerical characteristics are those that have a value in a numerical domain (e.g. integer). They can be used in VMs to represent the quantitative properties of the system. A different value in a numerical characteristic identifies a different software family member of the solution space. An example of a CPS numerical node is `Buffer Size` $\in$ `[1, 1024]` bytes. When considering numerical characteristics in a VM, the dependencies involving those numerical nodes are arithmetic formulas. For example, `Buffer Size` + `Cache Size` $\leq$ `1024 Bytes`.

Unfortunately, **most tools compatible with VMs do not support numerical characteristics** [14]. Additionally, **numerical characteristics increment the size of the solution space by multiplying it by its domain size, turning large solution spaces into colossal ones**. For example, the entire model of the Linux Kernel comprises an extra 55 numerical variables added to the 6467 alternatives; this increases the size of the solution space from $\sim 3.90 \times 10^{1672}$ to $\sim 5.66 \times 10^{1953}$ [15] different solutions.

1.2.2 Issues of Modelling Variability with Quality Values

Quality Models (**QMs**) are another tree-like structure used to determine which quality characteristics will be taken into account when evaluating the properties of a system [16]. There are different QM formalisations that we can consider as standards. The most popular is ISO/IEC 25010, which considers eight super-type quality characteristics: functionality, reliability, usability, efficiency, maintainability, portability, compatibility and security.

The natural approach to integrating these quality characteristics is to provide each VM node with a value registering their share of that quality in the running system (e.g., the CPU cost is 300€) alongside an aggregation formula (e.g., cost addition) [17]. But, as specified in previous paragraphs, **energy consumption is an efficiency quality linked to complete running systems and cannot be described by independent characteristics of individual nodes**. Another approach is to merge VMs and QMs through propositional dependencies where specific VM nodes forming a running system require specific QM-valued nodes [18], although at the **cost of reasoning performance due to the number and size of crossed relationships**. For example, `hardware nodes` + `software nodes` require 500 W.

1.3 Variability and Quality Analysis and Reasoning

Automated reasoning is the automation of formal logical reasoning to compute different types of information about models of systems [19]. It is commonly used in hardware and software verification, theorem-proving, and artificial intelligence. Automated reasoning tools are known in the literature as solvers. A direct example would be to provide a variability and quality model to a reasoning tool and request the generation of the entire solution space. In other words, to generate all the sound systems of the family represented in a provided model. Other examples are calculating the size of the solution space, checking the satisfiability of the model, and only generating optimal systems based on qualities-goal functions (e.g., energy-efficient ones). When we pair reasoning tools with statistical methods and

learning, we can detect energy-consuming concerns – the alternative characteristics of a family of systems that notably increase energy consumption negatively.

Unfortunately, **most solvers only support regular VMs and cannot offer quality-aware reasoning** [14]. The few supporting quality information are limited to superficial quality characteristics at the node level and simple optimisation goals (e.g., minimising cost). The main reason is that merged VMs and QMs do not scale due to the many restrictions used to link them, which decreases the tool's efficiency. Hybrid solutions, where one tool is used for native VM reasoning and another for native QM reasoning, perform better at the native level but still do not scale when merging their results, especially with colossal solution spaces. Remember that each energy characteristic registered in the model duplicates the solution space size. For example, if we measure for energy consumption and runtime a solution space size of 10^{100}, the resulting quality-measured solution space size is $2 * 10^{100}$. Consequently, we should aim to find a native modelling and reasoning approach for a unified *Variability and Quality model*(**VQM**).

In this thesis, we tackle the issues mentioned above and complexities by following a *Software Product Line Engineering* (**SPLE**) approach. Specifically, we have developed: a) a tool to support the modelling and optimisation-oriented reasoning of numerical characteristics in the existing state-of-the-art solver [20, 21], b) an algebraic framework for unified QVMs allowing extended modelling and native multi-objective reasoning [22, 23], c) an online eco-assistant that provides graphs and advises to optimise a user-restricted and quality-measured solution space [24, 25], and d) an algorithm and web-tool to learn the energy and characteristics influences of user restricted, domain-unknown, and partially measured solution spaces [26].

Chapter 2

Background

This section presents the background of this thesis. It summarises what is already known about this topic and how our research will contribute to the existing knowledge. It also provides the context and justification for the research question and aims to explain why the research was conducted. Concretely, we divide it into three sections corresponding to the three areas upon which we built this thesis: *Software Product Lines*(**SPLs**), *Category Theory*(**CT**), and statistical analyses.

2.1 Feature-Oriented Software Product Lines

SPLs are software engineering methods, tools and techniques for creating a collection of similar software systems from a shared set of software assets using a standard means of production [27]. SPLs consist of a family of software systems with some common reusable functionality and some variable and distinctive functionality. An SPL engineering approach involves dealing with one or many of the following aspects of a family of systems: the compatibility and developer requirements, the architecture, and the implementation of components. This approach reconciles mass production and mass customisation and helps to reduce development costs and time-to-market while improving product quality and customer satisfaction.

In this thesis, we focus on the feature-oriented branch of SPLs, which aims to increase the efficiency and quality of software development by focusing on reusable

software artefacts called features. These features are combined to create an SPL that guides developers to build, maintain and update products (e.g., a software program) with minimum time and effort [13].

2.1.1 Numerical Variability Models

Variability Models(**VMs**) are a common way for the representation of commonalities and specifics of the software artefacts. VMs are widely used during the SPL development process, specifically as the input of automated reasoners to generate other assets such as statistics, documents, architecture definitions, and even code.

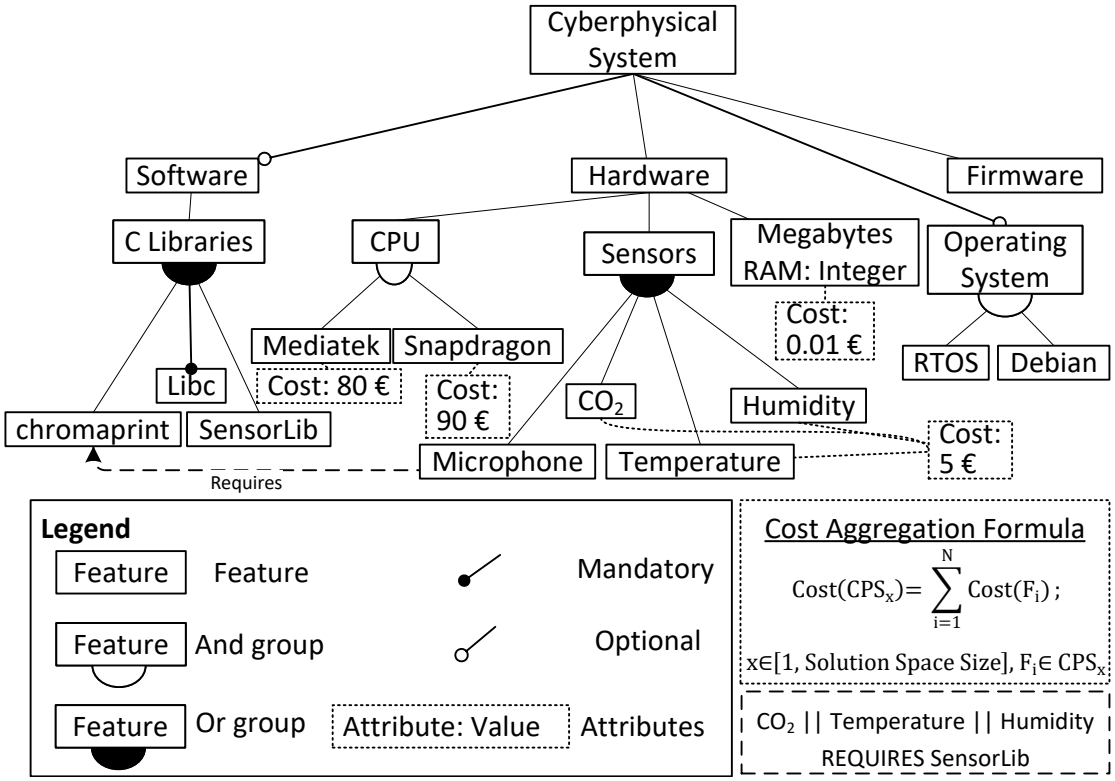

Figure 2.1: Example of an extended numerical feature model of a CPS family

Graphically, they are tree-like structures where the nodes of the tree (i.e., features) represent the alternatives to build different systems, the hierarchical edges represent cardinality-based parent-children relationships, and the cross-tree edges represent dependencies between branched nodes. In Figure 2.1, we can see an ex-

ample of a VM representing a reduced family of CPS comprising `C Libraries`, optional `Operating System`, and alternative `Hardware`. While the hierarchical relationships, we can visualise two cross-tree dependencies among features: the `Microphone` requires the `chromaprint` C library, and the rest of the `Sensors` require the `SensorLib` C library.

The SPL community considers *Feature Models* (**FM**) as the VM standard. *Feature-oriented Domain Analysis* (FODA) was the first formalisation of variability modelling and reasoning [28] as FMs. Feature models have been widely used during the whole SPL development process. They are commonly the reasoners' input to produce other assets such as documents, architecture definitions, or pieces of code.

A *Propositional Formula* (**PF**) is a type of logical formula which is well formed and has a truth value. If the values of all (Boolean) variables in a PF are given, it determines a unique truth value. As the classical FM formalisation defines SPL variability by Boolean-valued features and Boolean constraints (i.e., dependencies), we can transform classical FMs into PFs. Consequently, a PF becomes a relationship among features where the presence or absence of some features requires or excludes other features. Another alternative is to represent FMs as *Binary Decision Diagrams* (**BDD**s) [29], as they are Boolean functions in the form of directed acyclic graphs [30]. Consequently, we call configuration to: the different valid combinations of features [13], the sets of values determining a PF to `True` [31], and the complete paths of a BDD [29]. Finally, the set of all legal configurations is the SPLs' solution space.

Unfortunately, classical FMs are insufficient to represent real-world SPLs with numerical characteristics. An FM that supports *Numerical Features* (**NFs**) and mixed propositional logic and arithmetic constraints is a *Numerical Feature Model* (**NFM**) [20]. Its name, domain and metric define an NF. The domain can be a constant value (i.e., discrete NF) or a range of values. We find an example of the latter in Figure 2.1 with the NF `Megabytes of RAM` available and its `Integer` valued domain. Another example is the SPL of Linux repositories where packages have different versions and configurable parameters like `X86 MINIMUM CPU FAMILY` [32].

If we need to add specific information about individual features, we can do

it through the so-called *feature attributes* in *extended FMs*. Each leaf feature [1] can have a set of attributes which consist of valued variables of any domain (e.g., Boolean, numerical, string) [17]. An example could be the authorship of features if more than one modelling expert created the FM.

2.1.2 Modelling Quality Attributes

Quality Attributes (**QAs**) are characteristics of a system to evaluate its performance from the perspective of the end-user. Similarly to VMs, *Quality Models* (**QMs**) are tree-like structures used to determine which quality characteristics will be taken into account when evaluating a system's properties. Similarly to FMs, the community considers the ISO/IEC 25010 the most popular standard, comprising eight quality super-types: functionality, reliability, usability, efficiency, maintainability, portability, compatibility and security.

There are two types of relationships between QAs and a system's variability [33]:

- **Feature-wise**: The system's QA value is associated with the respective QA values of individual features and an aggregation formula. In other words, they belong to the feature space. This relationship matches features with non-interacting QA values, as we can define a simple and constant aggregation function. The extended NFM of Figure 2.1 acts as an example with the attribute `Cost` as a valued natural number representing the price of `Hardware` in €. The `Cost Aggregation Formula` adds the features cost per configuration.

- **Variant-wise**: The absolute QA value of a system, configuration or product. In other words, they belong to the solution space. This relationship matches interacting features, as even if we can approximate an aggregation function, it will be utterly complex and deprecates if we update the quality information. Energy consumption is one of these QAs. Each of the 565248 configurations of Figure 2.1[2] is linked to an absolute energy consumption value. For ex-

[1] A feature is a leaf if it is a terminal node in a tree data structure, meaning it has no children.
[2] 565248 configurations considering a maximum RAM of 4096 megabytes.

ample, the configuration [`Debian, Chromaprint, Libc, Mediatek, Microphone` and `64 megabytes of RAM`] consumes 5 Watts.

2.1.3 Modelling and Automated Reasoning with Solvers

Constraint Satisfaction Problems (**CSPs**) are mathematical questions defined as a set of objects whose state must satisfy several constraints. Constraint satisfaction methods calculate the solution space of CSPs represented as VMs, typically using a search form. The most used methods are variants of backtracking, constraint propagation, and local search.

The most popular types of CSPs for which we can find solver tools are *sat*isfiability (**SAT**), *Satisfiability Modulo Theories* (**SMT**) and *Constraint Programming* (**CP**):

- SAT: This CSP determines the existence of an interpretation that satisfies a given Boolean formula. In other words, we can find at least one value that makes the input **PF** resolve to true. SAT solvers support FMs if we transform them into PFs [34].

- SMT: This CSP determines whether a mathematical formula is satisfiable. It generalises SAT to more complex formulas involving numerical variables, arithmetic and various data structures such as lists, arrays and bit vectors. In practice, SMT solvers combine SAT solvers with more theory solvers [35]. SMT solvers support NFs and arithmetic.

- CP: This CSP determines whether a combinatorial problem is satisfiable. It solves Boolean, numerical and linear problems by backtracking and constraint propagation – reduce domains of variables, strengthen constraints, or create new ones [11]. CP solvers also support NFs and arithmetic.

Further, the basic and most used operations that the solvers of these problems support are:

- Model checking: It checks if the model does not present contradictions and if it represents any solution space. Practically, they try to generate one solution to check model correctness.

- Model enumeration: It records all solutions of a formula. In other words, it generates all the configurations of the solution space.

- Model count: Directly calculate the solution space size instead of a computing-based enumeration. The # versions of the solvers truly perform this operation. # Solvers are any solver modified to perform a true model counting. For example, a #SAT solver checks that the solution space size is above zero instead of solving the PF once as a classical SAT solver [36]. An alternative outperforming approach is to count the different complete paths of the respective BDD [29].

- Optimisation: It is a goal-oriented process to find optimal solutions within the solution space. This operation requires some support from QAs.

2.2 Fundamentals of Category Theory

Category Theory (CT) is an algebraic theory of mathematical structures [37]. It allows us to capture and relate similar aspects of structures while abstracting from the individual specifics of their dissimilarities. Consequently, it is a viable algebra worth testing in unifying variability, quality modelling, and reasoning.

A category $\mathcal{C}$ represents spaces as a collection of *objects* related to one another via *arrows* (i.e., *morphisms*). Two examples are the categories $\mathcal{V}ec$, where the objects are vector spaces and the arrows are linear maps, and $\mathcal{S}et$, where objects are sets and arrows are functions from one set to another. The main concepts of CT are:

- **Object**: a structured template $X \in \mathrm{Ob}(\mathcal{C})$, graphically depicted as a node $\bullet^X$.

- **Arrow**: a structure-preserving function $a \in \mathrm{Arr}(\mathcal{C})$ with source and target objects $X = src(a)$ and $Y = tgt(a)$, respectively, depicted $\overset{X}{\bullet} \overset{a}{\rightarrow} \overset{Y}{\bullet}$.

  - **Identity**: for every $X \in \mathrm{Ob}(\mathcal{C})$ exists exactly one arrow $\overset{X}{\bullet} \overset{id}{\rightarrow} \overset{X}{\bullet}$.

  - **Composition**: if $\overset{X}{\bullet} \overset{a_1}{\rightarrow} \overset{Y}{\bullet}$ and $\overset{Y}{\bullet} \overset{a_2}{\rightarrow} \overset{Z}{\bullet}$, then $\overset{X}{\bullet} \overset{a_2 \circ a_1}{\longrightarrow} \overset{Z}{\bullet}$.

Composition is associative, i.e.,
$a_1 \circ (a_2 \circ a_3) = (a_1 \circ a_2) \circ a_3$.

- **Category**: consists of $\mathrm{Ob}(\mathcal{C}) \cup \mathrm{Arr}(\mathcal{C})$ in a labelled directed graph.
- **Functor**: a process F between categories $C = src(F)$ and $D = tgt(F)$, depicted $\overset{C}{\bullet} \overset{F}{\rightarrow} \overset{D}{\bullet}$, which preserves identity and function composition.

Also, we shall introduce algebraic data integration CT concepts [38]:

- **Path**: a finite sequence of composed arrows: $\overset{X_0}{\bullet} \overset{a_1}{\longrightarrow} \overset{X_1}{\bullet} \cdots \overset{X_{n-1}}{\bullet} \overset{a_n}{\longrightarrow} \overset{X_n}{\bullet}$.
- **Element**: for $X \in \mathrm{Ob}(\mathcal{C})$, a *generalised element* of X is a morphism $\overset{U}{\bullet} \overset{elem}{\longrightarrow} \overset{X}{\bullet}$, where U is a select "unit" object. For example $\overset{U_x}{\bullet} \overset{Name}{\longrightarrow} \overset{String}{\bullet}$.
- **Instance**: a set-valued functor **Inst** that assigns values to elements. For example, $\overset{U_x}{\bullet} \overset{Name}{\longrightarrow} Java$ and $\overset{U_x}{\bullet} \overset{Value}{\longrightarrow} True$.

2.3 Statistical Analyses and Machine Learning

In statistical analyses, *machine learning* can be used to predict the outcome of a model or the behaviour of some of its features. This can be useful when the model is too large or complex to solve analytically or when there are too many variables to consider. This is exacerbated when measuring variant-wise and feature-wise QAs for the respective reasons. Machine learning can also optimise predictive models (e.g., performance models) by finding the best set of parameters that fit the data.

Instead of analysing an ample solution space, we can work with a reduced version, meaning a subset of solutions called samples. *Sampling* is the selection and measurement of a subset of the search space formed by pairs [configuration, QA] and [feature, QA] [39]. Random and guided sampling are popular solutions in the literature, although we can find many types depending on probabilities, knowledge and heuristics [40, 41, 42]. Unfortunately, choosing a proper representative set of samples is complex, as most real-world QAs and feature relationships do not follow a normal distribution. Additionally, we want to generate different sample sets to ensure that the features and interactions that we are identifying as the ones most

affecting a QA are meaningful. We can check that the analysed samples are as generally distributed as the system – the null hypothesis. In the context of our problem, a statistical normality test tells if two sets of samples have equally distributed interactions. To tackle abnormal distributions, the variability community integrates non-parametric tests. The most common ones are the Student's t-test, the Unequal Variance t-test, and the *Mann-Whitney U test* (MWU). The most used is the Student's t-test, which is accurate for assumed normally distributed data.

On the other hand, it is stated that the Unequal Variance t-test is superior to comparing the central tendency of 2 populations based on sets of unranked data. To sum up, the most accurately flexible is the MWU, independent of the sets' distribution, size and ranking. Furthermore, for ranked data, the MWU is superior to the alternatives [43]. Therefore, the MWU is the best test to assess whether a particular feature or interaction influences a QA, such as an energy footprint.

On the other hand, machine learning is a vast conglomerate of predictive and self-learning techniques [44] that allow machines to learn from data and improve their performance on a specific task without being explicitly programmed. It is used to solve problems that are too complex or expensive for human programmers to solve by developing algorithms. Instead, a computer learns how to solve the problem by discovering its own algorithms from the given data, which will then help make predictions or decisions on new data. As processing capacity has increased and deep learning become more powerful, computers have been able to design their own models. These generated models are empirically demonstrated to be effective with small learning sets [45], something that is crucial to improve scalability by reducing the number of requested samples. The four main types of machine learning paradigms are:

- **Supervised Learning**: The data consists of labelled [1] samples.
- **Unsupervised Learning**: Learning patterns from unlabelled data.
- **Reinforcement Learning**: It balances data exploration and exploitation.

[1] Labelling: Takes a set of unlabelled data and augments them with informative tags

- **Transfer Learning** (TL): Knowledge learned from task X is re-used to improve a related task Y.

This thesis focuses on supervised learning and TL. Applied in our context, we can make use of supervised learning methods to classify features and interactions influences to a QA, and approximate QA values of unmeasured configuration by generating regression formulas. For example, completing a partially measured solution space for the energy consumption QA. On the other hand, TL registers knowledge obtained while solving specific problems and applies it to another similar situation. A simple example is directly extrapolating the energy consumption behaviour of the GNU/Linux operating system Debian from version 10 to 11.

Chapter 3

Motivation and Approach

This section defines and elaborates the three *Research Questions* (RQs) that motivates this thesis. In short, we will follow an SPL approach to automatically perform different energy efficiency analyses of emergent domain systems like CPS and B5G networks. These systems present complexities like complex numerical components and arithmetic relationships, as well as colossal solution spaces [46]. On the other hand, energy efficiency also presents complexities as variant-wise relationships and partially measured solution spaces [47].

3.1 Motivated Research Questions

Emergent domains present many features of different domains and are constantly evolving. Additionally, each developer imposes a different set of feature constraints (e.g., requiring the Windows operating system). If both the NFM and the imposed constraints vary, also does the solution space to analyse. Hence, we cannot rely on guided sampling and domain-knowledge approaches.

Model counting configurations enable a fast, unbiased random sampling of large product spaces [48, 49]. This allows locating near-optimal configurations in a solution space with statistical guarantees (e.g., x% from optimal with y% confidence), given a defined workload [49, 50, 51].

SAT, CP, and SMT solvers perform poorly on counting as they enumerate configurations, which is infeasible for solutions spaces of size $\geq 10^6$ [39]. Consequently,

we need solvers supporting true model counting, meaning # versions and BDDs. SMT and CP solvers support NFM, but while we can find some algorithms in the literature, #SMT or #CP solvers have not been successfully developed yet [52]. On the other hand, #SAT and BDD solvers exist, but they do not support NFMs. Consequently, we define the first RQ:

> **RQ1:** How can we extend the state-of-the-art (Boolean) solvers to automatically support NFMs without performance degradation?

It is not realistic to think that every developer knows how to program or update any system with the specific suggested near-optimal one, as emergent domains can involve hundreds of changes (i.e., alternative features). Following a different approach, we can focus on the *noteworthy features* that strongly affect the energy consumption —the *energy consuming concerns*. Certain features influence a particular QA more than others. Moreover, one feature can affect another feature's impact on the quality value of a specific configuration - *interacting features* (e.g., selecting features A and B degrades the behaviour of C). Identifying noteworthy and interacting features will reduce the required domain-specific knowledge, as the developer can be informed about how certain features affect the energy efficiency of its systems. Then it is in his hands to decide the realistic and feasible ones.

Unfortunately, we cannot fully automatise the building and measuring of a **variant-wise** QA-like energy consumption, as we cannot aim to have all the 'features', measuring tools and expertise available for a large solution space [53]. Likewise, it is not feasible to do it manually in a possible time frame (e.g., $\geq 10^6$ [39] measurements). Current approaches deal with QAs mostly making use of sampling [39] with machine learning [44], basically as follows: sampling selects and measures a subset of the valid configurations generating a partial solution space, and machine learning predicts the rest of the space based on the (training) samples. While some predictive approaches based on learning performance models could be re-used for energy models, their accuracy requires specific initially measured samples, and the solution space is mostly static [49].

But even if that was not an issue, to our knowledge, none of the SPL tools supports the modelling and reasoning of variant-wise QAs like energy consumption [54]. Further, the lack of community consensus has led to entirely different

ideas which are finally not materialised in a sound analysis tool [55, 56], ending in the literature eluding the discussion of this issue [57]. The three assets which we can work with are:

- Generate certain configurations of a specific solution space with a solver that supports NFMs.
- Model variant-wise QAs in a QM.
- Having non-guided energy measurements of feasible configurations stored in data centres.

This means we lack the interconnection between a) configurations from a specific solution space given by an NFM and b) an incomplete measured solution space given by a QM and a data repository. But even if we can perform automated analyses of an interconnected NFM and QM, providing configurations and interacting features insights of large and partially measured solutions spaces is not straightforward. Quality reasoning methods for extended NFMs are linear techniques, but variant-wise information requires exponential techniques to discern statistically noteworthy interactions [39]. In other words, a brute-force approach is impossible in a reasonable time [58]. In summary, we need domain-agnostic methods that identify and analyse meaningful features and interacting features for variant-wise QAs when many configuration measurements are unavailable and cannot be requested on the fly. Consequently, we define the second RQ:

RQ2: How can we automatically provide energy efficiency insights when dealing with colossal, partially unknown, and partially-measured solution spaces based on NFMs and variant-wise QAs?

While answering RQ1 and RQ2 would allow obtaining energy efficiency insights for systems developers, it will not be a rounded solution. In other words, constantly executing a process that interconnects NFM and QM solutions spaces would be less scalable than a completely integrated solution with native reasoning. It can also present maintainability issues, like the respective reasoners being outdated or deprecated, forcing us to update the interconnecting process if we want to support

the new versions or solvers. Further, based on the limited reasoning operations that the NFM and QM reasoners support, we are bounded by what the interconnection process implements. Additionally, those base operations are not QA-aware; for example, while we can test the satisfiability of the NFM and the correctness of the QM, we cannot formally define and natively check the satisfiability of their union. An example would be a configuration with two energy consumption measurements with different values; At the same time, this is feasible in an integrated energy and variability model; the formal definition of satisfiability does not consider any QA. This ambiguity can create formal questions: Is having two different energy consumption values for the same configuration unsatisfiable? Why?

Furthermore, a rounded solution will support the modelling and reasoning of any extended NFM alongside varia-wise QAs. An important one is the interactions and constraints between feature-wise and variant-wise QAs. A modelling example could be a constraint where the monetary cost of a feature is proportional to the energy consumption of the configurations in which that feature is involved (e.g., CPU cost). A reasoning example could be a multi-objective goal function considering a trade-off between aggregated cost and energy consumption. This is feasible to implement in an *Integrated Development Environment* (**IDE**) supporting a unified extended NFM and QM. Finally, it will be more scalable and intuitive than interconnecting two distant reasoners. Consequently, we define the third RQ:

RQ3: How can we properly unify extended NFMs and QMs while having the native support of automated reasoning tools?

3.2 Approach Overview

This chapter also presents the general overview of the incremental approaches followed to answer RQs[1-3]. In Figure 3.1, we summarised in columns the already discussed motivated *Problem*, our scientific *Contributions*, the *Tools* that we have developed and published, and the *Output* results. We sequentially elaborate our approach divided into the different paths shown in Figure 3.1.

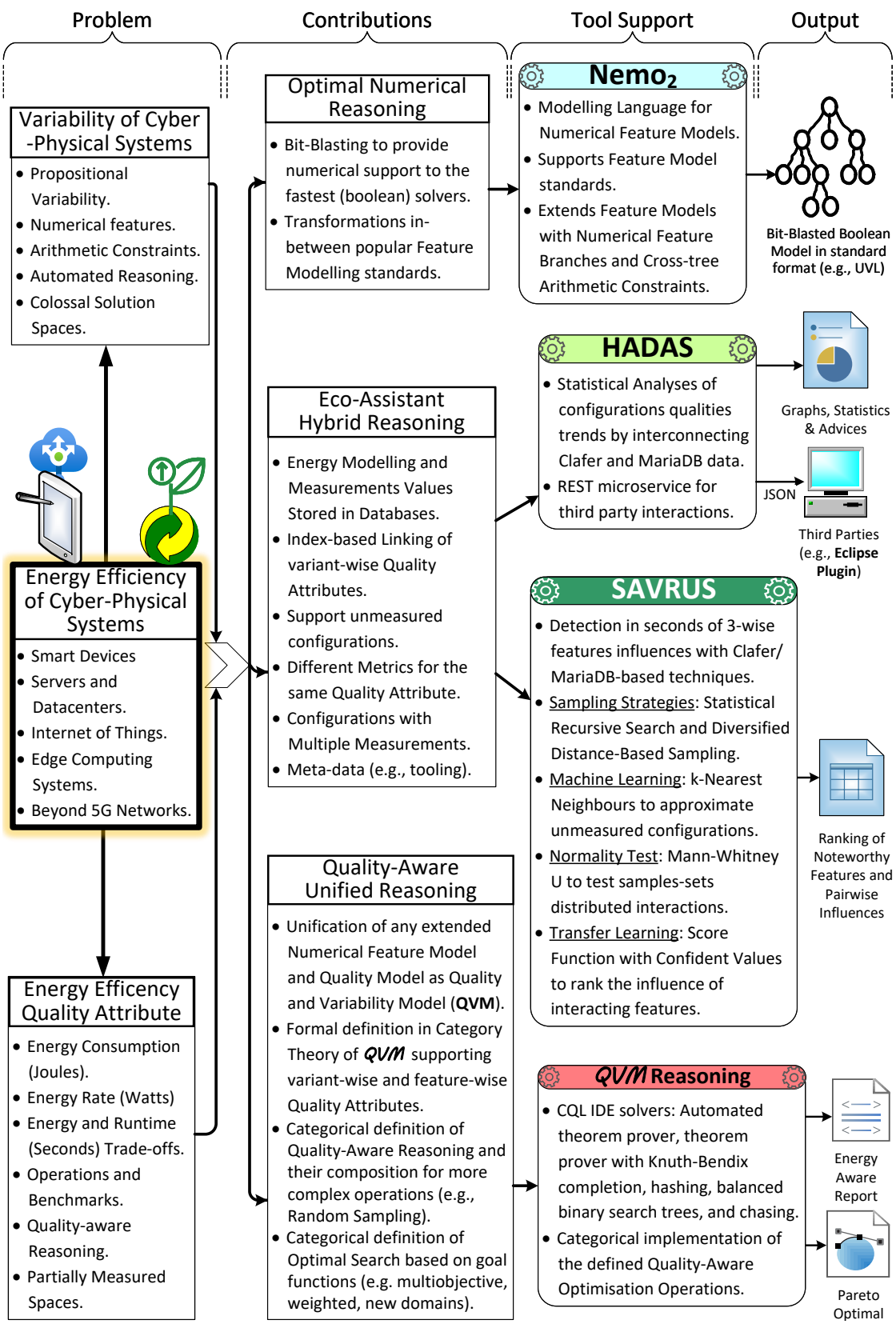

Figure 3.1: Graphical Overview of the Motivation and the Followed Approach

3.2.1 Numerical Features and Near-Optimal Search

Based on RQ1, a straightforward approach transforms NFMs into FMs and PFs, creating a direct compatibility of the fastest model counting reasoners. Thus, we can perform the state-of-the-art near-optimal search based on unbiased random sampling. This corresponds to the first path in Figure 3.1.

To transform numerical domains into Boolean domains, we can use bit-vectors - a fixed-size sequence of bits that can be used to represent numbers. In the logic community, transforming numerical variables and arithmetic formulas into bit-vectors is called flattening and **Bit-Blasting**. Numerical variables are bit-vectors, and arithmetic operations are propositional clauses that reference bits. The resulting PF is satisfiable whenever the original arithmetic formula is.

In our approach, we propose to Bit-Blast the NFs and arithmetic constraints of an NFM to transform it into any FM or PF standard. Our work focuses on the most popular arithmetic relationships and operations. We present them ordered by their usage frequency in real-world NFMs [59]: equality ($=$), inequalities ($\neq$, $>$, $\geq$), addition ($+$), subtraction ($-$), multiplication ($\star$), division ($/$), and modulo ($\%$). Furthermore, and by composition, we could transform nonlinear equations [1] (e.g., exponential encoded as a sequence of self-multiplications).

The main property of bit-vectors is the width which defines: a) the minimum and maximum value limits of numerical variables, and b) whether the vector is unsigned (i.e., binary **sign-magnitude** encoding) or signed (i.e., binary **two's complement** encoding) [2]. We also use the Big-Endian representation [3] where the first bit of the bit-vector encodes the sign as positive (0) or negative (1). For the concrete details of the transformations, we kindly refer the readers to chapters 7 and 8.

Manually applying Bit-Blasting to arithmetic requiring large bit-widths will take too much time. Therefore, we automated the process by developing the tool $Nemo_2$. As summarised in Figure 3.1, it defines a complete NFM modelling lan-

[1] Nonlinear equation: It is an equation that does not have a linear relationship between its variables. In other words, it looks like a curve when graphed.

[2] Two's complement negative integer encoding is the binary complement of the positive encoding plus one.

[3] Big-Endian: An order of bits in which the 'Big end' (most significant value in the sequence) is first in the sequence.

guage supporting all Boolean and arithmetic variables and operations. Additionally, it supports the inputs and outputs standards compatible with the different state-of-the-art solvers:

- **DIMACS** [1]: A set of clauses in *Conjunctive Normal Form* (**CNF**), while the classical PF and the *Universal Variability Language* (**UVL**) [60] model can contain implications, equivalences, nested clauses, and parenthesis.

- **UVL**: This model is a textual variability tree with a Root feature and the respective hierarchical constraints among features (i.e., father and children) followed by independent cross-tree constraint clauses.

- Classical **PF**: Defined in Chapter 2.

Further, $Nemo_2$ allows to extend/compose already modelled FMs and PFs in those standards with extra NFs, arithmetic constraints and NFM branches. Finally, $Nemo_2$ presents specific model optimisations, trying to generate the smallest and fastest to reason models in any of the supported formats.

Now that we can automatically obtain a Boolean and propositional representation of an NFM, we can perform model counting followed by a fast and accurate near-optimal search of configurations in a QA-measured solution space. Given a random integer j in $[1..|\mathbb{C}|]$, where $\mathbb{C}$ is the solution space, the trick is to convert j into the j^{th} configuration in $\mathbb{C}$. This is done by a binary search by choosing a feature f and counting the size of the space of configurations with f. If $j \leq |\phi \wedge f_i|$, the j^{th} configuration has feature f, recurse on the space $(\phi \wedge f)$, otherwise the j^{th} configuration has feature $\neg f$ and recurse on $(\phi \wedge \neg f)$.

At each iteration, a new feature is chosen, counting is performed, and the features belonging to the j^{th} configuration are eventually found. Finally, the confidence of that configuration is checked by performing a Student t-test analysis against another set of samples. The algorithm returns the best-performing configuration ¢ in a sample of size n, requiring in $n \cdot f$ calls to a counting tool (i.e., #SAT or BDD). Our sampling strategies rely on unbiased and random techniques:

[1] DIMACS is the de-facto standard for SAT solvers: `http://archive.dimacs.rutgers.edu/pub/challenge/satisfiability`

- **Uniform Random Sampling** (URS) [49]: We select samples randomly. Its accuracy depends on the uniform distribution of the search space and the randomiser.

- **Statistical Recursive Search** (SRS) [49]: It is a refined URS with recursion by fixing statistically meaningful features. We force the noteworthy features that passed the Student t-Test on each recursion in the next set of samples.

In case of URS, configuration ¢ is *on average* $\frac{100}{n+1}$ percentiles away from *the* best-performing configuration in $\mathbb{C}$ with $\frac{100}{n+1}$ percentiles standard deviation. In case of SRS with m recursions, the average decreases to $\frac{\frac{100}{n+1}}{10*m}$. So, if 99 random samples are selected by URS, the best-performing configuration out of the 99 is an average 1%±1% away from the best configuration in $\mathbb{C}$. Details are in [49, 51].

3.2.2 Hybrid Solver and Database Reasoning

Developers of CPSs have limited, and even erroneous, knowledge of how to reduce their energy consumption [61]. Consequently, they require an automated tool that acts as an *eco-assistant* by informing the CPS developers about how to improve their current systems. To propose an eco-assistant while answering RQ2, we follow two complementary approaches corresponding to the doubled second path in Figure 3.1. For the first part, we focus on including the most popular storage of energy consumption data in the reasoning process – databases [62]. Naturally, this implies interconnecting an NFM solver with a database manager. Further, mixing the variant-wise QA information of different real-world database impose three new complexities:

- Different metrics: We can use several metrics in the values of the same QA depending on its nature and final purpose. Energy is an excellent example of this; Joule is the standard unit, but kiloWatt-hour is used for large devices, and power in Watts is the most common metric to describe electronic components except for battery capacities in milliAmperes-hour. In a broader scope, the list is larger (e.g., calories, electron volts).

- Multiple measurements: A single configuration can have multiple values for the same QA. For example, it is proven that the energy measurement of the same CPS but using different measuring tools and environmental conditions will produce drastically different values [63]– although all of them are correct and valid from the perspective of variability.

- Meta-data: Specific information of a valued QA. This can involve the energy measuring tools used, the person or firm that performed the measurements, the environmental conditions like temperature, etc.

In any case, a solver should consider all the metrics and values in its reasoning operations and report interesting meta-data. For example, meta-data could help to avoid outliers [1] if we detect that a measuring tool is inaccurate. This will reduce the size of the measured solution space, which, if wholly measured, is **at least** as big as the solution space.

In our approach, we propose to define a process that, by externally linking configurations and variant-wise QA values, can provide valuable insights about the quality of each valid configuration but yet makes use of the existing tools. This means that the process will give quality-aware reasoning operations by composing the solutions of NFM solvers and database managers. Our solution is to create index-based symbolic links between valid configurations and their valued QAs. In practice, we assign a static identifier to leaf features, and a registered QA value must comprise the set of those identifiers that conforms to its associated complete configuration. As a result, we can directly perform statistical analyses of the solution space. For instance, providing quality-aware information on alternative configurations, analysing the effect of replacing a feature including a different value of an NF, or simply including a QA value requirement (e.g., generating all the configurations that consume less than 10 Watts).

We implemented this approach in our second tool coined *HADAS* – a cross-platform web-app [2] comprising:

[1] Outlier: A extremely low or extremely high straggler in a given data-set that induces inaccuracies in statistic analyses.

[2] *HADAS* web-app: `https://hadas.caosd.lcc.uma.es/`

- Solver-based NFM reasoning based on features selection and quality requirements;
- a collaborative database that stores all types of information of variant-wise QAs;
- a back-end process that interconnects solver-based (i.e., variability) and database manager (i.e., variant-wise QAs) reasonings;
- a graphical and interactive user interface for the complete reasoning process;
- a statistic influence analysis to complement the graphical results;
- a micro-service for third-party applications interaction; and
- a plugin for IntelliJ IDEA that integrates *HADAS* analyses as code improvement suggestions to the developers.

In step-by-step summary of its architecture and functionality is the following:

1. A graphical representation of an NFM internally defined in Clafer Modelling Language. This interface also supports click-based feature requirements as well as several analysis parameters.
2. Clafer-solver generates all the possible configurations of the user-restricted solution space.
3. Our proposed process generates a SQL query to retrieve the QA values of those configurations based on the identifiers of the leaf features. Here, user-defined QA constraints are also added to the query.
4. MariaDB process that query where the identifiers act as primary keys, and the specific values are *indexed* by them (i.e., foreign keys).
5. Having calculated the measured solution space, it performs a Pearson's chi-squared differentials and Bootstrapping analysis.
6. Finally, it plots graphs and statistic results showing the effect a feature change by an alternative will have in a configuration. This also includes the exclusive domain of each NF.

When using a plugin, the first and the last steps are replaced by the interface of the third application. At the same time, the data is interchanged in JSON format [1]. For more details, we kindly refer the readers to chapters 9 and 10.

Having successfully reached a viable Whendatabase reasoning approach, we encountered an extra set of complexities when taking *HADAS* into practice. They match the second part of the second path in Figure 3.1:

- Reasoning time of colossal spaces: When the number of configurations and measurements defined in *HADAS* essentially increase, and the *HADAS*' users unconstrained the NFM, hence performing reasoning of the entire space, *HADAS* runtime proportionally increased. Considering that it is an expected behaviour, users start to drop software if it takes more than 3 seconds to load [64]. While we could think that developers are professionals with trained patience, its maximum limit is around 60 minutes [65].

- T-wise Interacting features: Interactions are identified when the presence of a set of others influences one feature's behaviour. Furthermore, interactions sometimes cannot be deduced from individual behaviours, as they genuinely exist when following a complex equation. The higher the order of interactions, the harder to detect them (e.g., pairs are a quadratic increase, and triplets are cubic). This analysis increases reasoning time even further, as the number of potential interactions in a system is exponential in the number of features. Depending on T, the accuracy and computationally demands vary in an increasing pattern; up to 2 is considered balanced, while 6 reaches exponential cost [66].

- Partially Measured Solution Space: For hard-to-measure variant-wise QAs of large spaces, we probably find unmeasured configurations, whether by the impossibility of manually measuring that space size or by the infeasibility to automatise the building and measuring process. Consequently, techniques that involve specific prior knowledge (i.e., measurements) are not accurate in these scenarios, as we cannot assure that particular configurations are

[1] *JavaScript Object Notation* (**JSON**) is a data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays

measured for a QA. This makes us discard guided sampling and learning-based predictive techniques (e.g., providing an approximate QA value for a specific configuration).

Summarising, we now need to: a) in a maximum of 60 minutes, b) derive noteworthy and interacting features without predicting absolute values, c) while supporting a variable search space, and d) without relying on initial domain knowledge. Consequently, we have defined the *SAVRUS* approach that: navigates through large and hard-to-measure spaces with partially-unknown QAs values, works with just available data, and provides fast identification of features and up to pairwise interactions ranked by their importance. *SAVRUS* is a modular strategy comprising five sequential steps. We developed a prototype [1] with the following choices of techniques for each step:

1. Solver-based sampling: Based on $Nemo_2$ results, we continued using SRS, but additionally implemented *Diversified Distance-based Sampling* (**DDbS**) [67] – a faster T-wise sampling where we select fewer samples and up to T interactions, based on a configurable distance metric and probability distribution, and where diversity is increased by prioritising configurations containing the least frequently sampled features and interactions.

2. Lazy learning: We will use *k-Nearest Neighbours* (kNN) as a regression technique by assigning the mean of the k-closest observations. Our implementation uses the *Manhattan distance* [2] – also called the *City-Block distance*, as it is the recommended one in the literature for the variety of dimensions that the configurations of a CPSs SPL comprises [68]. We left the k hyper-parameter user-defined, being the default value, the minimum required for a mean (i.e., $k = 2$).

3. Statistical test: In this step, *SAVRUS* performs an MWU-based statistical test to identify with a 95% confidence the noteworthy and noticeably interacting features affecting a QA based on the sample set.

[1] *SAVRUS* web-app: `https://hadas.caosd.lcc.uma.es/savrus.php`
[2] Manhattan distance of N dimension $= |x_1 - y_1| + |x_2 - y_2| + ... + |x_N - y_N|$

4. TL: *SAVRUS* We implemented an anomaly-shifted detection with unsupervised instance selection to keep a dynamic track of their features' noteworthiness scoring and pairwise combinations based on previous executions. It uses the *Local Outlier Factor* (LOF) method, which measures the local deviation of the density of an unknown sample. The ranking scoring goes from 0 to 1, meaning not affecting and always affecting, respectively. On each *SAVRUS* execution, if any of them are detected as statistically noteworthy, the LOF is executed. And the weights are updated depending on their new LOF value.

5. Weighted sorting: *SAVRUS* uses the updated scoring weights to rank feature combinations – the highest the weights and lowest the rank, the strongest is the features-QA interaction.

Hence, *SAVRUS* is an interactive approach that developers will normally execute several times while taking decisions based on previous executions. In each execution, they can decide if using DDbS or SRS, the number of samples and the k hyperparameter. For more details, we kindly refer the readers to Chapter 11. With SAVRUS, developers can now get fast insights into features and variant-wise quality interactions of large yet partially measured models without requiring specific domain knowledge, supporting features and quality constraints, and sampling method selection and adjustment. With those insights, developers optimise the systems by directly replacing the noteworthy *redder* features and shielding the noteworthy *greener* ones.

3.2.3 Quality Variability Models and Reasoning

To answer RQ3 and become independent of maintaining an exogenous process interconnecting two reasoning tools of different application areas, we followed the unification approach of the third path in Figure 3.1. With this objective, we leveraged the modelling and reasoning flexibility of CT, formalising an SPL framework that unifies extended NFMs with any QAs related to an SPL category; this means supporting both Boolean and numerical features alongside feature-wise QAs and QMs with variant-wise values. Structuring a category is similar to defining a meta-

model, that in our case, we coined the *Quality Variability Model* ($\mathcal{QVM}$) category. $\mathcal{QVM}$ comprises three data-type categorical objects (i.e., Boolean, Integer, and String for characters sets) and five structured categorical objects:

- $\mathcal{F}$eatures: It defines the extended NFM as name and domain arrows to String, NF value to Integer, and self-object arrow for hierarchical (i.e., *Parent*) and cross-tree constraints, and additional arrows depending on the number of feature-wise.

- $\mathcal{Q}$ualities: It hosts the QM as name and domain arrows to String, variant-wise value to Integer, and again self-object arrow for hierarchical and cross-QA constraints.

- $\mathcal{C}$onfigurations: It links the leaf features of specific complete configurations by categorical instances. Please note that non-leaf features can be traced back to the root by *Parent* arrows.

- $\mathcal{QA}$s: It links valued variant-wise QAs in sets (e.g., energy consumption = 1 Joule and latency = 1 second).

- $\mathcal{S}$pan: This categorical property is translated into a mapping between single $\mathcal{C}$onfigurations and single sets of variant-wise values $\mathcal{QA}$s. They are pairs of arrows to identifiers acting similarly to database foreign keys.

Further, a $\mathcal{QVM}$ instance means filling the category with information matching the defined structure - i.e., creating a model based on the meta-model.

Having defined the unifying category $\mathcal{QVM}$, the next step is to define quality-reasoning operations on SPL configurations, which allow developers to directly perform *quality-aware* analyses of the now united as one space the features, solutions and measured spaces. While some of those operations are quality-aware re-definitions of the classical ones, the complex ones were impossible with the state-of-the-art previous to this thesis. We can summarise the operations as follows:

- Report: Number of features, NFs, first-order constraints, arithmetic constraints, feature-wise QAs and variant-wise QAs, and QAs meta-data.

- Satisfiability [69]: Check that there exists at least one measured configuration of a $\mathcal{QVM}$ for a set of QAs.
- Count [20]: Count the instances of a $\mathcal{QVM}$ measured solution space without generating them.
- Filter [14]: State advanced variability and QAs requirements over a $\mathcal{QVM}$ and generate its corresponding reduced and measured solution space.
- Bound [14]: Directly restrict the size of a $\mathcal{QVM}$ measured solution space.
- Random [70, 20]: Randomise the generation of a $\mathcal{QVM}$ measured solution space for any operation (e.g., random seed).
- Aggregate: Transform feature-wise QAs into variant-wise based on aggregation functions.
- Optimisation: Quality-goal function search. The function supports weighted arithmetic relationships between multiple feature-wise and quality-wise QAs and commonly is a maximisation or minimisation of a set of valued QAs.

To test them, we defined these operations as lambda functions in functional programming, as they are functors between our unified category (i.e., $\mathcal{QVM}$) and a resulting category. For example, for *Satisfiability*, the resulting category is just one categorical object with a single Boolean element, and its formal definition is $\boldsymbol{SAT} : \mathcal{QVM} \xrightarrow{Satisfiability} \mathbb{B}$.

The definition of categories like $\mathcal{QVM}$ and functor-based reasoning is supported in categorical tools where CQL IDE [1] is the state-of-the-art. CQL IDE processes our defined operations with its set of reasoners: an automated theorem prover with Knuth-Bendix completion for PFs and arithmetic equations, hashing, balanced trees and chasing for data-type and cross-object arrows. Further, the composition of operations is a natural operation in algebras and formal theories like CT and is likewise supported by CQL IDE. For example, URS reasoning is the functorial composition of *Random* and *Bounding*. The main advantage of this composition is that we can now reuse reasoning operations to create more complex ones. Finally,

[1] CQL IDE: `https://www.categoricaldata.net/`

we ultimately developed and published our CQL IDE code as a framework that allows direct SPL modelling and reasoning without notions of CT. Technically, the complete $\mathcal{QVM}$ and all the reasoning operations are categorically implemented in CQL IDE. Hence, a user can directly fill the category with the extended NFM and QM data and relationship and execute reasoning operations among the ones presented in this thesis. It is also developed to support extending the NFM and the operations for new domains, properties or reasoning goals. In other words, to support future SPL extensions, we do not need to extend CQL IDE as a tool, but its configuration – our $\mathcal{QVM}$ framework. This is the main advantage of using generalistic tools like categorical ones. The result is a unified and scalable solution to model extended NFMs with QA's relationships and reasoning of any complexity. We kindly refer the readers to chapters 12 and 13 for more details.

Chapter 4

Discussion of Results

This chapter answers the research question by discussing this thesis's main results and contributions.

Our **first contribution** starts with the approach implemented in $Nemo_2$, empirically achieving to model, extend, automatically optimise, and **transform NFMs into the most common formats of classical FMs** using bit-blasting. We can represent complex formulas up to 12 bit-width and transform real-world NFMs to colossal sizes without overhead for almost every combination of Boolean and arithmetic operations in under 15 minutes. *This contribution is elaborated in our work [15].*

Our **second contribution** is related to the $Nemo_2$ generated models, with which we can **search for (near) optimal configurations in colossal solution spaces** based on URS or SRS reaching a size of 10^{45} configurations in under 10 seconds with a BDD solver. Additionally, we can count the number of configurations of a solution space size of 10^{248} in under 5 hours with a #SAT solver. These solution spaces correspond to the NFMs of the embedded Linux Busybox version 1.18.5 and 1.28, respectively. ***This contribution and these results are elaborated in our highlighted works [20, 21] (chapters 7 and 8).***

A1: We can encode NFs and arithmetic constraints into bit-vectors and PFs. Consequently, with $Nemo_2$, we automatically extend FMs with NFM branches, run NFMs in classical (Boolean) solvers, and perform (near) optimal configuration search with a minimum performance degradation (e.g., seconds).

Our **third contribution** is to design a hybrid reasoning approach where an exogenous process **interconnects an NFM solver and a database manager that hosts real information of the respective variant-wise QAs**. This is the core of our tool *HADAS*, where a PHP backend interconnects the Clafer solver with a MariaDB database indexed-based. This database is a repository collecting from third-parties energy consumption measurements of real-world CPSs. Naturally, this information includes outliers and multiple values of a single QA and configuration. *This contribution is elaborated in our works [71, 72, 73]*

Our **fourth contribution** is to **provide rich reasoning analyses in the form of graphical energy efficiency insights, statistic correlations, and contextual code advice of CPSs** with the *HADAS* web tool and its JetBrains IDE plugins. Our experiments reached up to a 90% decrease in the required energy consumption if the CPS (i.e., Waspmote) sends large amounts of data in big batches through Wi-Fi. However, *HADAS* also detects crossed conclusions, like, and following the previous results, Bluetooth is more energy efficient for tiny bits of data. Other sets of results are to detect which level of code granularity is worth analysing and uncovering linear relationships of edge computing micro-servers with the number of requests and data size. Finally, we ran a survey with 17 CPS developers using the *HADAS* plugin while coding their applications in their work, resulting in many positive experiences with an average of 4 out of 5 Likert scale points. ***This contribution and these results are elaborated in our highlighted works [25, 24] (chapters 9 and 10)*** *and also in the publications [74, 75, 76, 77]*

For our **fifth contribution**, we defined the *SAVRUS* approach, which allows to **create a ranking of influencing features and interactions (i.e., concerns) by fast navigating through large measured spaces containing many randomly-unknown configurations**. As a result, we can uncover the energy-consuming concerns of an NFM with $\sim \mathbf{5.3 * 10^8}$ configurations with only

132,500 random measurements in 1-minute average – *SAVRUS* runtime is proportional to the selected number of samples. Additionally, we tested the ranking results with 4 real-world and completely measured NFMs. We generated several rankings after artificially degrading their measured spaces to resemble real-world databases' random and biased energy measurements; *SAVRUS* presents coverage and accuracy of $\sim 80\%$. ***This contribution and these results are elaborated in our highlighted work [26] (Chapter 11)***

A2: We can provide rich insights into the variant-wise QAs of SPL products by developing our *HADAS* index-based process that interconnects an NFM solver reasoning with database managers that host measurements such as energy efficiency values. For cases in which the measured space is large, biased and incomplete, we can provide a ranking of the energy-consuming concerns by following the *SAVRUS* sequence of techniques: solver-based sampling, lazy learning, statistical tests, transfer learning and weighted sorting.

Our **sixth contribution** is to **formally define the category $\mathcal{QVM}$ that integrates into a unified model the extended NFM and QM, natively supporting any feature, QA, and constraint among them**. We empirically evaluated this approach by transforming into a single $\mathcal{QVM}$ the NFM and database model of our *HADAS* tool and having implemented the resulting categorical instance (i.e., $\mathcal{HADAS}$) in CQL IDE, we generated the respective energy-measured solution space 10 times faster than the with the regular *HADAS* tool. In short, our categorical approach allows us to easily extend a model with new properties (e.g., ranged NFs, features with string domain) and perform native and faster reasoning. ***This contribution and result is elaborated in our highlighted work [22] (Chapter 12)*** *and also in the publication [78]*

Our **seventh contribution** is to **formally define the categorical $\mathcal{QVM}$ framework revealing 8 groups of lambda operations for quality-aware reasoning supported by the categorical solver CQL IDE**. Additionally, we show how to combine operations to form complex ones. We empirically tested this framework with 5 real-world NFMs. Two of them are part of the European

H2020 project DAEMON [1] of which we are partners, and involve incredibly complex CPSs, Beyond5G networks and virtual network functions. We also compared our framework against the state-of-the-art for up to 20 quality-aware reasoning operations. Our implementation in CQL IDE is the only solver compatible with the complete models and the 20 operations, and on average, it is also the fastest solution. However, in rare cases, one of the compared solvers was slightly shorter for a particular combination of model and operation (e.g., SATIBEA [79] for basic near-optimal search). ***This contribution and its results are elaborated in our highlighted work of an industrial track [23] (Chapter 13)*** *and also in the publications [80, 81]*

A3: With a CT approach, we can unify the extended NFM and QM as the category $\mathcal{QVM}$, as well as define any quality-aware operation for a categorical solver like CQL IDE, allowing native reason over quality-measured solution spaces.

[1] European H2020 project DAEMON: `https://h2020daemon.eu/`

Chapter 5

Related Work

This section starts by reviewing the current studies for efficient CPSs. It follows by grouping the current state-of-the-art and most influential related works matching the paths division of Figure 3.1, meaning the support of NFs and arithmetic constraints, measured variant-wise QAs in colossal spaces, and algebraic approaches to unify NFMs and QMs. Additionally, Table 5.1 summarises the current modelling and reasoning tools and alternatives.

Energy efficiency is critical in CPSs since it determines the device's autonomy. However, developers tend to underestimate the energy requirements of their devices at a software level [82]. For example, [83] proposes a green energy-powered architecture based on composing services in fog computing. However, the usage context and the insights for the deployed functionality are avoided. In [84], the context is considered to be lifetime increased by minimising the amount of cloud-transmitted data. Nonetheless, the focus is only on communication while tackling any operation of existing configurations. The goal of a usable high-performance environment is presented in [85] by investigating complex CPSs with application scenarios serving as case studies; instead of real devices and measurements, software simulations were used. The energy consumption of multi-core algorithms for CPSs is studied in [86]. However, the insights are assembly level – hard to implement in real-world SPLs. In [87], a framework for CPS development is formalised. They evaluate the effectiveness of their proposal in terms of resiliency and reliability but not energy consumption. In Embedded Systems, the total energy

Table 5.1: Support for reasoning about quality in feature modelling

	T:ClaferMoo	T:FAMA	T:FeatureIDE	T:pure::variants	T:SPL Conqueror	T:QAMTool	T:HADAS	T:STEAM	A:SATIBEA	A:MILPIBEA	A:GIA	A:MO-DAGAME	CT framework
Quality Modelling													
Feature-wise QAs	■	■	■	■	■	■	□	■	■	■	■	■	■
Variant-wise QAs	□	□	□	□	▣	■	■	□	□	□	□	□	■
Variability and Quality Solvers													
Declarative Paradigm (CSP, BDD, SAT, ...)	■	■	■	■	□	□	□	□	■	□	■	□	□
Hybrids (SAT solver + database, ...)	□	□	□	□	■	■	■	□	□	□	□	□	□
Search-Based Software Engineering	■	□	□	□	■	□	□	□	■	■	■	■	□
Alternative Formalisation (Category Theory)	□	□	□	□	□	□	□	■	□	□	□	□	■
Automatic Quality Reasoning													
Model Analyses Operations (sat, counting)	■	■	■	■	■	■	■	■	▣	▣	▣	▣	Q
Aggregation Function Operations													
* Addition	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q
* Product	Q	□	□	Q	Q	□	Q	□	□	□	Q	□	Q
* Mean	□	□	□	Q	□	□	□	□	□	□	□	□	Q
* Approximation arithmetic equation	□	□	□	□	Q	□	□	□	□	□	□	□	Q
Optimal Search Operations													
* Maximum	Q	Q	□	□	Q	□	□	Q	Q	Q	Q	Q	Q
* Minimum	Q	Q	□	□	□	□	□	Q	Q	Q	Q	Q	Q
* Multiobjective	Q	□	□	□	□	□	□	□	Q	Q	Q	Q	Q
* Range optimisation	Q	□	□	□	Q	□	□	□	□	□	□	□	Q

■ It supports the characteristic. ▣ Out of the scope of the approach. T: The approach is an SPL tool.
▣ It partially supports it. Q Quality-aware native operation. A: Algorithm approach for FMs.
□ It doesn't support the characteristic.

consumption is minimised while the timing constraint is satisfied in [88]. They also continue by proposing a heuristic approach to obtain a near-optimal solution efficiently. However, it is only helpful for hybrid memory systems.

5.1 Numerical Features and Arithmetic

Works dealing with NFMs are rare, and those who did it, for various reasons, did not describe how numerical variables were represented [89]. Some considered NFs as classical features with just present/absent states [90, 32, 91]. Others encoded NFs as alternative features, where each valued NF is a feature [92]. Shi [93] used 'pseudo-boolean' features only supporting `Successor` (`+1`) and `Predecessor` (`−1`) operations. Another popular approach is that each Boolean feature has attributes – a set of variables in the form (name, value, domain) [94]. However, attributes and NFs are essentially different: attributes are not nodes of the variability tree, and as opposed to an actual NF, a change in the value of an attribute does not result in a different configuration [24]. Hence, counting the size of a product space will return a lower-than-expected value.

SMT and CP solvers natively support the representation and reasoning of NFMs. However, the # versions are nonexistent. This is to be expected, as CP and SMT are unbounded by default [95], being unaware of allocated memory or domain definitions (e.g., undefined maximum of `x` in $x \geq 1$). In SAT ones, all variables are bounded (i.e., Boolean). Consequently, SMT approximation counting has been proposed but not yet empirically tested [96]. Solvers Z3 [97] and Yices [98] apply bit-blasting to every operation besides equality, which a specialised solver then handles. They also add axioms dynamically from array theory. STP solver [99] implements a bit-vector approach for counting. It performs array optimisations, arithmetic, and Boolean simplifications before bit-blasting to MiniSat [100]. While it works to test satisfiability by counting at least one, it does not preserve counting or model equivalence. This aligns with the most recent model counting competition (2020), where they tested 34 versions of the 8 fastest counting solvers. Model counting is commonly found in BDDs [30] and SAT-based [36] solvers. The results indicate that while fast, even so-called 'exact solvers' count a close but inexact number of configurations.

5.2 Colossal Variant-Wise Quality Reasoning

Typically, a CPS developer would like to obtain the configurations with a QA below a threshold (e.g., < 3 Joules) or generate the best-qualified configuration (e.g. trade-off between energy consumption and performance). We have already discussed the differences between feature-wise and variant-wise QAs. Feature-wise QAs are the most common in the literature and are supported by ClaferMoo [101], FAMA [17], FeatureIDE [56], pure::variants [1], SPL Conqueror [102] and STEAM [103]. In QAMTool [104], authors use an alternative representation and extend the FM by incorporating QA-specific features in a sub-tree. Another alternative is to have some external storage to relate features and quality measurements as usually done in genetic algorithms (SATIBEA [105], MILPIBEA [106], MO-DAGAME [107]). An exception is the GIA algorithm [108], defined to be applied to an attributed FM that also uses the Z3 solver. Only a few approaches, such as QAMTool [104] and HADAS [24], support variant-wise QAs. SPL Conqueror supports them only partially by calculating an approximated value for the feature attributes based on the set of measured configurations during the generation of the solution space. Our CT framework [22] supports both types of QAs.

SPL tools (labelled with T: in Table 5.1) that only support feature-wise QAs (ClaferMoo, FAMA, pure::variant) commonly use a declarative paradigm (e.g. CSP, BDD, SAT) to represent the FM and reason about its quality. In other cases, an external quality model is defined (e.g., a goal model), and the QAs measurements are usually linked to the configurations through a database. The FM is still represented using a declarative paradigm, but an additional structure is used to store and reason about variant-wise QAs. This is the case with the SPL Conqueror, HADAS and QAMTool tools. SPL Conqueror creates a performance model by using sampling and aggregation techniques and uses this model to approximate near-optimal configurations. The HADAS tool uses Clafer plus a relational database, and the QAMTool uses the NFR framework [55] to represent QAs in a goal model externally. For algorithms generating optimum configurations (labelled with A: in Table 5.1), a genetic algorithm is usually complemented with a representation of the FM as genes and a measurements database with the

[1] https://www.pure-systems.com/pv-update/additions/doc/latest/pv-user-manual.pdf

feature-wise QA measurements. In some cases, a declarative solver is also used, as in the SATIBEA algorithm, defined as a combination of an SAT solver, the IBEA genetic algorithm, and the GIA algorithm that uses a Z3 solver.

Prior work on SPLs performed statistical analyses to reason on colossal ($> 10^{82}$) and complex configuration spaces. To estimate the influence of a feature on performance, samples were benchmarked and compared for performance differences [109, 110]. To find optimal configurations, samples were used to search the configurations throughout the space [111, 112, 113, 114, 115, 116]. To evaluate different sampling approaches to locate variability bugs, URS was considered the baseline to compare with other approaches [117, 42, 118].

Studies for finding interactions without prior knowledge are rare, especially in energy-aware CPSs. Most of them aim to minimise the costs while maintaining the latency. In [119], we find a meta-review of predicting energy values of reusable software by integrating big data and artificial intelligence. Unfortunately, the work concludes that the accuracy of the reviewed solutions was insufficient. MIGRATE is a three-step machine learning framework for intelligent energy profiling [120]. MoMo is a dynamic variability approach but works with well-known absolute values [121]. Federated learning creates performance models based on previously learned aggregations and provides accuracy and cost balance [122]. [123] follows the same aim by using orthogonal versions of extended FMs, and a series of mappings and transformations compatible with FAMA framework. However, that mapping and those analyses are neither reusable nor extendable for evolved or user-constrained models. Deep software variability is coined in [124] to exploit multi-layered SPLs. *Thermal-aware Scheduling and Tuning* (TaSat) proposed a Pareto algorithm for the latency, energy consumption and temperature. While it does not require specific measurements to perform accurately, its domain is specifically heterogeneous [125]. Similarly, we have TOFFEE, which uses a stochastic algorithm [126]. Unfortunately, these solutions consider the quality of space as well-known.

Another approach proposes pre-defined templates based on FPGAs architecture for energy-aware edge computing [127]. These are regularly called energy models in the literature and tend to be used to detect worst-case scenarios. The tool Serapis [128] is an example, but still, they do not match our objectives. Green-

Scaler provides automatic test generation for CPS, and the results are stored in a local repository and used to detect energy-aware configurations [129].

All SPL tools (labelled with T: in Table 5.1) offer some level of model analysis operations. ClaferMoo, FAMA, FeatureIDE, pure::variants and STEAM provide implementations of all or a subset of the operations defined in [94] (e.g. satisfiability, type and number of features, type and number of model constraints, number of configurations). Others (e.g. SPL Conqueror, QAMTool, HADAS) use a third-party variability modelling language that provides such support. Algorithms (labelled with A: in Table 5.1) focus on optimisation. Regarding quality-aware operations, current approaches do not natively support the complete set of quality-aware operations. Native support would mean that the variability model implements quality-enriched operations as primitives. Regarding the aggregation function and the optimal search operations, the support is variable, as shown in Table 5.1. The operations supported by ClaferMoo are almost as complete as in our approach. It supports both addition and product aggregation functions and all the optimisation operations under consideration in this paper. pure::variants also support addition, product and mean aggregation functions, although approximation arithmetic equations are not supported, and thus, reasoning about the combination of several quality attributes is not possible. Neither optimal search operations are supported. SPL Conqueror supports addition, product and some equations and allows optimal search operations with maximums and ranges. FeatureIDE, QAMTool and HADAS do not provide any support for optimisation. Regarding the genetic algorithms, they approximate optimal configurations using sampling strategies and considering feature-wise QAs. They all support the addition aggregation function and the maximum, minimum and multi-objective search operations. They do not support range optimisation. CT framework supports all the quality-aware operations discussed in Table 5.1.

5.3 Data Integration with Algebraic Theories

As discussed in Section 2, a complete and tested solution for unified modelling and reasoning was nonexistent. Suppose we broaden the scope to non-SPL theories. In that case, we find *Set Theory* (ST), which, similarly to CT, is a branch of mathe-

matical logic that studies set, which informally are collections of objects [130]. ST lacks support for numerical equations, inequalities, and infinite datatypes. Similarly, *Higher-Order Logic*(HOL) deals just with declarative propositions, predicates and quantification (e.g., $\forall x$) [131]. *Codd Theory* is the first and only formalisation of relational algebra, which uses algebraic structures with well-founded semantics for modelling data and defining queries on it. While databases support various numerical components such as datatypes, counting, grouping, arithmetic, etc., they are programming workarounds outside Codd's Theory. In other words, it is unclear that Codd relational algebra should be extended above a pure Boolean domain [132]. Pseudo-Boolean (i.e. [0,1]) reasoners are based on *Arithmetic* [132] – the study of numbers and their operations. While they are promising, if not considering the complexity and overload of model-transforming HOL, their performance in current SAT competitions is often quite poor [133]. It should be pointed out that all of the theories mentioned above (ST, HOL, Codd Algebra) are well-formalised categories in CT.

Chapter 6

Conclusions and Future Work

This chapter presents the conclusions of the thesis and future work.

6.1 Conclusions

Our research has focused on covering all the modelling properties and reasoning techniques required to develop energy-efficient CPS and IoT devices. We started by following a traditional SPL approach but end-up welcoming exotic techniques from other fields like abstract algebra. Concretely, what we needed to integrate and support that are not considered in a classical SPL approach were: (i) modelling and optimal reasoning of NFs and arithmetic constraints in the form of NFMs, (ii) measuring, modelling and reasoning of variant-wise complex QAs like energy efficiency which information is stored in independent third party databases, (iii) fast analyses of colossal solution spaces linked with energy-efficiency values, (iv) consider partially-unmeasured solution spaces with random, biased, duplicated and contradictory measurements alongside the respective meta-data, (iv) arithmetic constraints among variant-wise and aggregated feature-wise QAs, (v) complex quality-aware reasoning operations and multi-objective optimisation functions of measured solution spaces defined by a unified variability and quality model.

Our approach can be summarised by integrating current modelling standards into one, extending it with the discussed variability and quality requirements, and providing direct analysis and advanced reasoning support that generate fast and

rich insights into CPS's energy efficiency. To do it, we followed three main paths:

1. The design of a bit-blasting technique to transform extended NFMs into extended FMs. This allows direct use of the state-of-the-art (Boolean) solvers for counting and near-optimal search of colossal spaces.

2. To develop an index-based process to link the reasoning of NFM solvers and database managers hosting variant-wise QAs information. Due to the complexities of the energy consumption information stored in different databases, we integrated this solution into a sequence of 5 steps: solver-based sampling, kNN, statistical tests, TL, and a weighted sorting of noteworthy features and interactions affecting a QA.

3. To unify extended NFMs, variant-wise QA values, and QMs into the $\mathcal{QVM}$ category, as well as lambda, define and group 8 different types of quality-aware reasoning operations: report, satisfiability, count, filter, bound, random, aggregate and optimisation. Additionally, due to the nature of CT, these operations can be reused and compounded to form more complex ones (e.g., $\boldsymbol{URS} : \mathcal{QVM} \xrightarrow{Bound \circ Random} \mathcal{C}onfigurations$).

Besides the academic contributions, we develop 3 tools and 1 framework respectively: $Nemo_2$ to model NFMs and generate the respective FM in the common standards, $HADAS$ web-services that interconnect Clafer solver and MariaDB to provide graphical and statistic insights of the energy consumption of CPS devices, $SAVRUS$ that generates in seconds a ranking of up to pairwise influencing features of a colossal but randomly-unmeasured solution space, and $\mathcal{QVM}$ framework in CQL IDE to directly model and quality-aware reason over $\mathcal{QVM}$s without prior CT or functional programming knowledge.

Consequently, approach and tools provide (1) CPS and IoT developers with the necessary services and methods to directly uncover the most efficient alternatives of their applications regardless of the quality of the data and the development time available, (2) tools developers with micro-services to include reasoning in third-party applications like integrated development environments (i.e., IDEs), and (3) the SPL community a extended variability and quality modelling standard compatible with a new set of (categorical) solvers and reasoning algorithms. Finally,

we expect this thesis to increase the consciousness of energy-efficient alternatives and good practices while keeping the performance and costs of the current solutions. As a global conclusion, we desire that our tools will be adopted to reduce the energy footprints of the industry up to a 90% and help stop, and even reverse, the causes and effects of climate change.

6.2 Future Work

As immediate improvements of our approaches and tools, we look forward to integrating machine and transfer learning techniques into our $\mathcal{QVM}$ framework. Additionally, in the *SAVRUS* sequence, we expect to a) introduce a self-trained neural network, b) replace in the second step the kNN with the *Approximate Nearest Neighbours* (ANN) technique like *Locality-Sensitive Hashing* [134], and c) extend the pairwise to a T-wise analysis with the minimum computational overhead. For future research, we plan to explore 4 new directions:

- Continue increasing the number of modelling tools and solvers. Concretely, non-relational databases, algebraic tools like CoQ and Agda, and pure mathematical suites like Matlab and Octave.
- Explore other algebraic theories like Geometry, Topology and Calculus.
- Formalise exotic operations for newer quality-aware reasoning. For instance, Horner schemes to check satisfiability, Newton method for approximate counting, and even Non-Linear Programming.
- Uncover new (useful) properties of QVMs by defining a pure mathematical meta-model. This will allow us to perform (partial) derivations and integrals of the measured solution space or calculate their limits and determinants. This information can potentially improve and reduce the time of reasoning operations.

Finally, we expect to apply the acquired knowledge to other software engineering fields and reasoning theories like blockchain, artificial intelligence and algebraic topology.

Part II

Thesis Publications

Chapter 7

Uniform Random Sampling of Numerical Feature Models

Title:	Uniform Random Sampling Product Configurations of Feature Models That Have Numerical Features
Authors:	Daniel-Jesus Munoz, Jeho Oh, Mónica Pinto, Lidia Fuentes and Don Batory
Conference:	23th ACM International Systems and Software Product Line Conference (SPLC 2019)
Conference Rating:	GGS Class: 2 / GGS Rating: A- / CORE: B / LiveSHINE: A / MA: A-
Publication Date:	September 2019
DOI:	`https://doi.org/10.1145/3336294.3336297`

Abstract. Analyses of *Software Product Lines*(**SPL**s) rely on automated solvers to navigate complex dependencies among features and find legal configurations. Often these analyses do not support numerical features with constraints because propositional formulas use only Boolean variables. Some automated solvers can represent numerical features natively, but are limited in their ability to count and *Uniform Random Sample*(**URS**) configurations, which are key operations to derive unbiased statistics on configuration spaces.

Bit-blasting is a technique to encode numerical constraints as propositional

formulas. We use bit-blasting to encode Boolean and numerical constraints so that we can exploit existing #SAT solvers to count and URS configurations. Compared to state-of-art Satisfiability Modulo Theory and Constraint Programming solvers, our approach has two advantages: 1) faster and more scalable configuration counting and 2) reliable URS of SPL configurations. We also show that our work can be used to extend prior SAT-based SPL analyses to support numerical features and constraints.

Chapter 8

Transforming NFMs into PFs and the UVL

Title:	Transforming Numerical Feature Models into Propositional Formulas and the Universal Variability Language
Authors:	Daniel-Jesus Munoz, Mónica Pinto, Lidia Fuentes and Don Batory
Journal:	Journal of Systems and Software (JSS)
JCR Impact Factor:	3.514 (Q2)
Publication Date:	June 2023
DOI:	https://doi.org/10.1016/j.jss.2023.111770

Abstract. Real-world *Software Product Lines*(**SPLs**) need *Numerical Feature Models*(**NFMs**) whose features have not only boolean values that satisfy boolean constraints but also have numeric attributes that satisfy arithmetic constraints. An essential operation on NFMs finds near-optimal performing products, which requires counting the number of SPL products. Typical constraint satisfaction solvers perform poorly on counting and sampling.

Nemo (Numbers, features, models) is a tool that supports NFMs by *bit-blasting*, the technique that encodes arithmetic expressions as boolean clauses. The newest version, Nemo2, translates NFMs to propositional formulas and the *Universal*

Variability Language(**UVL**). By doing so, products can be counted efficiently by #SAT and Binary Decision Tree solvers, enabling finding near-optimal products. This article evaluates Nemo2 with a large set of synthetic and colossal real-world NFMs, including complex arithmetic constraints and counting and sampling experiments. We empirically demonstrate the viability of Nemo2 when counting and sampling large and complex SPLs.

Chapter 9

Finding Features Correlations with Energy and Performance

Title:	Finding Correlations of Features Affecting Energy Consumption and Performance of Web Servers Using the HADAS Eco-Assistant
Authors:	Daniel-Jesus Munoz, Mónica Pinto and Lidia Fuentes
Journal:	Computing
JCR Impact Factor:	2.063 (Q2)
Publication Date:	June 2018
DOI:	https://doi.org/10.1007/s00607-018-0632-7

Abstract. The impact of energy consumption on the environment and the economy is raising awareness of "green" software engineering. HADAS is an eco-assistant that makes developers aware of the influence of their designs and implementations on the energy consumption and performance of the final product. In this paper, we extend HADAS to better support the requirements of users: *researchers*, automatically dumping the energy-consumption of different software solutions; and *developers*, who want to perform a sustainability analysis of different software solutions. This analysis has been extended by adding Pearson's chi-squared differentials and Bootstrapping statistics, to automatically check the

significance of correlations of the energy consumption, or the execution time, with any other variable (e.g., the number of users) that can influence the selection of a particular eco-efficient configuration. We have evaluated our approach by performing a sustainability analysis of the most common web servers (i.e. PHP servers) using the time and energy data measured with the *Watts Up? Pro* tool previously dumped in HADAS. We show how HADAS helps web server providers to make a trade-off between energy consumption and execution time, allowing them to sell different server configurations with different costs without modifying the hardware.

Chapter 10

Energy-Aware Environments for Greener Cyber-Physical Systems

Title:	Energy-Aware Environments for the Development of Green Applications for Cyber-Physical Systems
Authors:	Daniel-Jesus Munoz, José A. Montenegro, Mónica Pinto and Lidia Fuentes
Journal:	Future Generation Computing Systems
JCR Impact Factor:	6.125 (Q1)
Publication Date:	September 2018
DOI:	https://doi.org/10.1016/j.future.2018.09.006

Abstract. *Cyber-Physical Systems* are usually composed by a myriad of battery-powered devices. Therefore, developers should pay attention to the energy consumption of the global system so as not to compromise the system lifetime. There are plenty of experimental studies that give hints about how to reduce the energy consumption. However, this knowledge is not readily available for the software developers of cyber-physical systems. They normally use software development environments that do not provide useful advice about the energy consumption of the software solutions being implemented. In this paper, we propose a *Developer Eco-Assistant* to integrate the experimental results obtained by researchers into

the software development environments, so as to increase the energy-awareness of cyber-physical systems developers. In our solution, the energy information is obtained in real-time from a repository of energy consuming concerns, where researchers store their experimental measurements. Developers use the repository to perform sustainability analyses, which, in turn, will lead to greener design/implementation decisions. In this paper, we illustrate the use of our approach in the context of cyber-physical systems development using both open source environments (e.g. JetBrains IDEs) and proprietary environments (e.g. Waspmote development environment). We experimentally demonstrate that cyber-physical systems can reduce more than 40% of its energy consumption depending on the scenario, reaching approximately 90% in some certain cases.

Chapter 11

Learning Feature Influences to Quality Attributes in Incomplete Spaces

Title:	Detecting Feature Influences to Quality Attributes in Large and Partially Measured Spaces Using Smart Sampling and Dynamic Learning
Authors:	Daniel-Jesus Munoz, Mónica Pinto and Lidia Fuentes
Journal:	Knowledge-Based Systems
JCR Impact Factor:	8.139 (Q1)
Publication Date:	April 2023
DOI:	https://doi.org/10.1016/j.knosys.2023.110558

Abstract. Emergent application domains (e.g., Edge Computing/Cloud/B5G systems) are complex to be built manually. They are characterised by high variability and are modelled by large *Variability Models* (VMs), leading to large configuration spaces. Due to the high number of variants present in such systems, it is challenging to find the best-ranked product regarding particular *Quality Attributes* (QAs) in a short time. Moreover, measuring QAs sometimes is not trivial, requiring a lot of time and resources, as is the case of the energy footprint of soft-

ware systems – the focus of this paper. Hence, we need a mechanism to analyse how features and their interactions influence energy footprint, but without measuring all configurations. While practical, sampling and predictive techniques base their accuracy on uniform spaces or some initial domain knowledge, which are not always possible to achieve. Indeed, analysing the energy footprint of products in large configuration spaces raises specific requirements that we explore in this work. This paper presents **SAVRUS** (**S**mart **A**nalyser of **V**ariability **R**equirements in **U**nknown **S**paces), an approach for sampling and dynamic statistical learning without relying on initial domain knowledge of large and partially QA-measured spaces. SAVRUS reports the degree to which features and pairwise interactions influence a particular QA, like energy efficiency. We validate and evaluate SAVRUS with a selection of likewise systems, which define large searching spaces containing scattered measurements.

Chapter 12

Category Theory Framework for VMs with Non-functional Requirements

Title:	Category Theory Framework for Variability Models with Non-functional Requirements
Authors:	Daniel-Jesus Munoz, Dilian Gurov, Mónica Pinto and Lidia Fuentes
Conference:	33rd International Conference on Advanced Information Systems Engineering (CAiSE 2021)
Conference Rating:	GGS Class: 2 / GGS Rating: A / CORE: A / LiveSHINE: A / MA: A-
Publication Date:	June 2021
DOI:	https://doi.org/10.1007/978-3-030-79382-1_24

Abstract. In *Software Product Line* (SPL) engineering one uses *Variability Models* (VMs) as input to automated reasoners to generate optimal products according to certain *Quality Attributes* (QAs). Variability models, however, and more specifically those including numerical features (i.e., NVMs), do not natively support QAs, and consequently, neither do automated reasoners commonly used

for variability resolution. However, those satisfiability and optimisation problems have been covered and refined in other relational models such as databases.

Category Theory (CT) is an abstract mathematical theory typically used to capture the common aspects of seemingly dissimilar algebraic structures. We propose a unified relational modelling framework subsuming the structured objects of VMs and QAs and their relationships into algebraic categories. This abstraction allows a combination of automated reasoners over different domains to analyse SPLs. The solutions' optimisation can now be natively performed by a combination of automated theorem proving, hashing, balanced-trees and chasing algorithms. We validate this approach by means of the edge computing SPL tool HADAS.

Chapter 13

Analysis and Optimisation of Virtual Network Functions

Title:	Quality-aware Analysis and Optimisation of Virtual Network Functions
Authors:	Daniel-Jesus Munoz, Mónica Pinto and Lidia Fuentes
Conference:	26th ACM International Systems and Software Product Line Conference (SPLC 2022)
Conference Rating:	GGS Class: 2 / GGS Rating: A- / CORE: B / LiveSHINE: A / MA: A-
Publication Date:	September 2022
DOI:	https://doi.org/10.1145/3546932.3547007

Abstract. The softwarisation and virtualisation of network functionality is the last milestone in the networking industry. Software-Defined Networks (SDN) and Network Function Virtualization (NFV) offer the possibility of using software to manage computer and mobile networks and build novel *Virtual Network Functions* (VNFs) deployed in heterogeneous devices. To reason about the variability of network functions and especially about the quality of a software product defined as a set of VNFs instantiated as part of a service (i.e., Service Function Chaining), a variability model along with a quality model is required.

However, this domain imposes certain challenges to quality-aware reasoning of service function chains, such as numerical features or configuration-level *Quality*

Attributes (QAs) (e.g., energy consumption). Incorporating numerical reasoning with quality data into SPL analyses is challenging and tool support is rare. In this work, we present 3 groups of operations: model report, aggregate functions to dynamically convert QAs at the feature-level into the configuration-level, and quality-aware optimisation. Our objective is to test the most complete reasoning tools to exploit the extended variability with quality attributes needed for VNFs.

Part III

Appendices

Appendix A

Publications

This chapter presents all the publications done in the context of the thesis, and that naturally support its results and conclusions. They account for 18 publications in total along the 7 years of the PhD program (2016-2023) divided as follows: 4 JCR-indexed journals, 5 GGS-indexed international conferences, 1 tool demonstration papers in GGS-indexed international conferences, 2 workshops in GGS-indexed international conferences, 1 doctoral symposium in a GGS-indexed international conference, 1 international conference and 4 national conferences. Table A.1 lists the publications in journals, Table A.2 list the publications in any international conference, Table A.3 lists the rest of the publications except the national conferences ones that are listed in Table A.4. It is worth to highlight the award-winning best paper in an international conferences corresponding to the HADAS path of our approach. Additionally, 4 of the indexed publications are the result of two different research stays finishing in fruitful collaborations. Finally, this thesis has received two awards:

- Top 22 thesis of Universidad de Málaga in 2022 [1].
- Top 2 Excellence Young Scientist 2018 award of the Universidad de Sevilla and Universidad de Málaga by the Fundación IMFAHE (International Mentoring Foundation for the Advancement of Higher Education) [2].

[1] Thesis Talk UMA 2022: https://www.uma.es/media/tinyimages/file/

Table A.1: Journal Publications

JOURNALS	
Title:	Transforming Numerical Feature Models into Propositional Formulas and the Universal Variability Language (*Post Research Stay at the University of Texas in Austin, USA*)
Authors:	Daniel-Jesus Munoz,* Mónica Pinto, Lidia Fuentes and Don Batory
Journal:	Journal of Systems and Software (JSS)
JCR Impact Factor:	3.514 (Q2)
Publication Date:	June 2023
DOI:	https://doi.org/10.1016/j.jss.2023.111770
Title:	Detecting Feature Influences to Quality Attributes in Large and Partially Measured Spaces Using Smart Sampling and Dynamic Learning
Authors:	Daniel-Jesus Munoz,* Mónica Pinto and Lidia Fuentes
Journal:	Knowledge-Based Systems
JCR Impact Factor:	8.139 (Q1)
Publication Date:	April 2023
DOI:	https://doi.org/10.1016/j.knosys.2023.110558
Title:	Energy-Aware Environments for the Development of Green Applications for Cyber-Physical Systems
Authors:	Daniel-Jesus Munoz,* José A. Montenegro, Mónica Pinto and Lidia Fuentes
Journal:	Future Generation Computing Systems
JCR Impact Factor:	6.125 (Q1)
Publication Date:	September 2018
DOI:	https://doi.org/10.1016/j.future.2018.09.006
Title:	Finding Correlations of Features Affecting Energy Consumption and Performance of Web Servers Using the HADAS Eco-Assistant
Authors:	Daniel-Jesus Munoz,* Mónica Pinto and Lidia Fuentes
Journal:	Computing
JCR Impact Factor:	2.063 (Q2)
Publication Date:	June 2018
DOI:	https://doi.org/10.1007/s00607-018-0632-7

* Corresponding author.

ThesisTalk_ParticipantesSeleccionados.pdf

[2]IMFAHE AWARD 2018: https://www.imfahe.org/wp-content/uploads/2022/08/IMFAHE-Fellowship-Winners-.pdf

Table A.2: International Conference Publications

INTERNATIONAL CONFERENCE	
Title:	Quality-aware Analysis and Optimisation of Virtual Network Functions
Authors:	Daniel-Jesus Munoz,* Mónica Pinto and Lidia Fuentes
Conference:	26th ACM International Systems and Software Product Line Conference (SPLC 2022)
Conference Rating:	GGS Class: 2 / GGS Rating: A- / CORE: B / LiveSHINE: A / MA: A-
Publication Date:	September 2022
DOI:	https://doi.org/10.1145/3546932.3547007
Title:	Nemo: A Tool to Transform Feature Models with Numerical Features and Arithmetic Constraints (*Post Research Stay at the University of Texas in Austin, USA*)
Authors:	Daniel-Jesus Munoz,* Jeho Oh, Mónica Pinto, Lidia Fuentes and Don Batory
Conference:	20th International Conference on Software and Systems Reuse (ICSR 2022)
Conference Rating:	GGS: 3 / CORE: B / LiveSHINE: B / MA: B
Publication Date:	June 2022
DOI:	https://doi.org/10.1007/978-3-031-08129-3_4
Title:	Category Theory Framework for Variability Models with Non-functional Requirements
Authors:	Daniel-Jesus Munoz,* Dilian Gurov, Mónica Pinto and Lidia Fuentes (*Post Research Stay at the KTH Royal Institute of Technology in Stockholm, Sweden*)
Conference:	33rd International Conference on Advanced Information Systems Engineering (CAiSE 2021)
Conference Rating:	GGS Class: 2 / GGS Rating: A / CORE: A / LiveSHINE: A / MA: A-
Publication Date:	June 2021
DOI:	https://doi.org/10.1007/978-3-030-79382-1_24
Title:	Uniform Random Sampling Product Configurations of Feature Models That Have Numerical Features (*Post Research Stay at the University of Texas in Austin, USA*)
Authors:	Daniel-Jesus Munoz,* Jeho Oh, Mónica Pinto, Lidia Fuentes and Don Batory
Conference:	23th ACM International Systems and Software Product Line Conference (SPLC 2019)
Conference Rating:	GGS Class: 2 / GGS Rating: A- / CORE: B / LiveSHINE: A / MA: A-
Publication Date:	September 2019
DOI:	https://doi.org/10.1145/3336294.3336297
Title:	Green Security Plugin for Pervasive Computing Using the HADAS Toolkit
Authors:	Daniel-Jesus Munoz,* José A. Montenegro, Mónica Pinto and Lidia Fuentes
Conference:	IEEE International Symposium on Dependable, Autonomic and Secure Computing (DASC 2018)
Conference Rating:	GGS Class: W / GGS Rating: Work in Progress / CORE: C / LiveSHINE: B / MA: -
Publication Date:	April 2018
DOI:	https://doi.org/10.1109/DASC-PICom-DataCom-CyberSciTec.2017.136
Title:	HADAS and Web Services: Eco-Efficiency Assistant and Repository Use Case Evaluation (***Best Student Paper Award***)
Authors:	Daniel-Jesus Munoz,* Mónica Pinto and Lidia Fuentes
Conference:	2017 International Conference in Energy and Sustainability in Small Developing Economies (ES2DE 2017)
Publication Date:	August 2017
DOI:	https://doi.org/10.1109/ES2DE.2017.8015334

* Corresponding author.

Table A.3: Demonstration Tool, Workshop, and Doctoral Symposium Publications

DEMONSTRATION TOOL, WORKSHOP AND DOCTORAL SYMPOSIUM	
Title:	Defining Categorical Reasoning of Numerical Feature Models with Feature-Wise and Variant-Wise Quality Attributes
Authors:	Daniel-Jesus Munoz,* Mónica Pinto, Dilian Gurov and Lidia Fuentes
Workshop:	5th International Workshop on Languages for Modelling Variability - 26th ACM International Systems and Software Product Line Conference (SPLC 2022)
Conference Rating:	GGS Class: 2 / GGS Rating: A- / CORE: B / LiveSHINE: A / MA: A-
Publication Date:	September 2022
DOI:	https://doi.org/10.1145/3503229.3547057
Title:	HADAS: Analysing Quality Attributes of Software Configurations
Authors:	Daniel-Jesus Munoz,* Mónica Pinto and Lidia Fuentes
Workshop:	Demonstration Tool Track - 23th ACM International Systems and Software Product Line Conference (SPLC 2019)
Conference Rating:	GGS Class: 2 / GGS Rating: A- / CORE: B / LiveSHINE: A / MA: A-
Publication Date:	September 2019
DOI:	https://doi.org/10.1145/3307630.3342385
Title:	Green Software Development and Research with the HADAS Toolkit
Authors:	Daniel-Jesus Munoz,* Mónica Pinto and Lidia Fuentes
Workshop:	Third Workshop on Sustainable Architecture: Global Collaboration, Requirements, Analysis (SAGRA 2017) - 11th European Conference on Software Architecture (ECSA 2017)
Conference Rating:	GGS Class: 3 / GGS Rating: B / CORE: A / LiveSHINE: B / MA: C
Publication Date:	September 2017
DOI:	https://doi.org/10.1145/3129790.3129818
Title:	Achieving Energy Efficiency Using a Software Product Line Approach
Authors:	Daniel-Jesus Munoz*
Workshop:	Doctoral Symposium - 21st ACM International Systems and Software Product Line Conference (SPLC 2017)
Conference Rating:	GGS Class: 2 / GGS Rating: A- / CORE: B / LiveSHINE: A / MA: A-
Publication Date:	September 2017
DOI:	https://doi.org/10.1145/3109729.3109744

* Corresponding author.

Table A.4: National Conference Publications

NATIONAL CONFERENCES	
Title:	Composición Categórica de Análisis Automáticos para Líneas de Productos Extendidas
Authors:	Daniel-Jesus Munoz,* Mónica Pinto, Lidia Fuentes
Conference:	XXVI Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2022)
Publication Date:	September 2022
DOI:	http://hdl.handle.net/11705/JISBD/2022/8601
Title:	Teoría de Categorías Aplicada a Variabilidad
Authors:	Daniel-Jesus Munoz,* Mónica Pinto, Lidia Fuentes
Conference:	XXV Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2021)
Publication Date:	September 2021
DOI:	http://hdl.handle.net/11705/JISBD/2021/057
Title:	Aplicabilidad de la Caracterización de Benchmarks a Modelos de Variabilidad
Authors:	Daniel-Jesus Munoz,* Lidia Fuentes
Conference:	XXIII Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2018)
Publication Date:	September 2018
DOI:	http://hdl.handle.net/11705/JISBD/2018/061
Title:	HADAS: Asistente de eco-eficiencia con repositorio de consumo energético
Authors:	Daniel-Jesus Munoz,* Mónica Pinto, Lidia Fuentes
Conference:	XXII Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2017)
Publication Date:	September 2017
DOI:	http://hdl.handle.net/11705/JISBD/2017/005

* Corresponding author.

Appendix B

Resumen en Español

B.1 Introducción

La OCDE afirma que el consumo de energía es un combustible para el crecimiento económico, pero está estrechamente relacionado con el cambio climático global a través de las emisiones de gases de efecto invernadero [1]. Los sistemas de software no consumen energía por sí mismos, pero afectan a la utilización del hardware, lo que conlleva un consumo indirecto de energía. Reducir el consumo energético de los SPI exige un estudio exhaustivo de los distintos agentes, componentes y entorno implicados. Según la AIE, sólo los centros de datos y las redes en funcionamiento consumen el 1% de la electricidad mundial [3]. El diseño de software con conciencia energética puede reducir el consumo total de energía entre un 30% y un 90% [6].

No obstante, las distintas formas de medir el consumo de energía en tiempo real son muy complejas. Los medidores de energía son dispositivos que se conectan a la toma de corriente y miden la cantidad de electricidad que circula por ella, mientras que los modelos energéticos de hardware con herramientas de software de predicción pueden mostrar cuánta energía utiliza cualquier componente para un proceso específico. Las lecturas de energía se proporcionan como el consumo total de energía en julios o la tasa de consumo de energía en vatios. Para los dispositivos alimentados por batería, los julios por tarea son una métrica más interesante, mientras que los vatios por tarea se usan más en dispositivos conectados directamente a corriente [6]. El enfoque habitual para modelar y almacenar las lecturas

de consumo energético es describirlas como una característica de componentes individuales, como el coste monetario de un componente hardware.

Sin embargo, los valores de consumo energético presentan muchas interacciones entre componentes, lo que dificulta su descripción mediante valores energéticos individuales y estáticos. En su lugar, podemos almacenar y completar con información energética bases de datos de forma colaborativa. La AIE y Datarade ofrecen bases de datos gratuitas con datos de consumo energético para el análisis de la eficiencia energética y la sostenibilidad. Sin embargo, las bases de datos no son escalables para sistemas altamente configurables debido a la *maldición de la dimensión*. Nuestro trabajo se centra en la Industria 4.0, concretamente en los *Sistemas Ciberfísicos* (acrónimo inglés **CPS**), los cuales se caracterizan por su alta configurabilidad y adaptabilidad, presentando un gran número de alternativas y un número colosal de sistemas diferentes en funcionamiento. Esto se conoce como el espacio de búsqueda/solución, y cuyo tamaño es el conjunto de todos los puntos posibles que satisfacen un problema de optimización.

Los espacios de solución parcialmente conocidos son un problema habitual en muchos campos, como la ingeniería informática, el aprendizaje automático, la inteligencia artificial y la optimización orientada a objetivos. Los *Problemas de Satisfacción de Restricciones* (acrónimo inglés **CSP**) son problemas matemáticos definidos como un conjunto de objetos cuyo estado debe satisfacer una serie de restricciones [11]. Los Modelos de Variabilidad (acrónimo inglés **VMs**) son estructuras arborescentes que se utilizan para representar los elementos comunes y las diferencias de un CPS. Las Características Numéricas (acrónimo inglés **NFs**) pueden utilizarse en los VMs para representar propiedades cuantitativas del sistema, pero la mayoría de las herramientas no admiten NFs. Además, las NFs aumentan el tamaño del espacio de soluciones multiplicándolo por su tamaño de dominio, lo que convierte los espacios de soluciones grandes en colosales.

Los *Modelos de Calidad* (acrónimo inglés **QMs**) son estructuras en forma de árbol que se utilizan para determinar qué *Atributos de Calidad* (acrónimo inglés **QA**s) como la eficiencia energética se tendrán en cuenta al evaluar un sistema. ISO/IEC 25010 es la formalización de QM más popular, que agrupa las QAs en ocho tipos distintos. El razonamiento automatizado es la automatización del razonamiento lógico formal para calcular distintos tipos de información sobre modelos

de sistemas. Algunos ejemplos son proporcionar un VM o QM a una herramienta de razonamiento, y calcular el tamaño del espacio de soluciones, comprobar la satisfabilidad del modelo o generar únicamente sistemas óptimos basados en funciones objetivo basadas en uno o varios QAs.

Esta tesis pretende encontrar un enfoque nativo de modelado y razonamiento para un *Modelo* unificado *de Variabilidad y Calidad* (acrónimo inglés **QVM**). Se trata de desarrollar un enfoque que apoye el modelado y el razonamiento orientado a la optimización de características numéricas, un marco algebraico para QVMs unificados, un ecoasistente en línea para optimizar un espacio de soluciones restringido por el usuario y medido para la calidad, y un algoritmo y una herramienta web para aprender las influencias de la energía y las características de espacios de soluciones restringidos por el usuario, desconocidos por el dominio y parcialmente medidos.

B.2 Antecedentes

Esta tesis presenta los antecedentes de las *Líneas de Productos de Software* ((acrónimo inglés **SPLs**)), la *Teoría de Categorías* (acrónimo inglés **CT**) y los análisis estadísticos. Las SPL son métodos, herramientas y técnicas de ingeniería de software para crear una colección de sistemas de software similares a partir de un conjunto compartido de activos de software utilizando un medio de producción común. Este enfoque concilia la producción y personalización masiva, y ayuda a reducir los costes de desarrollo y el tiempo de comercialización, al tiempo que mejora la calidad del producto y la satisfacción del cliente. Los VMs son un medio común para la representación de aspectos comunes y específicos de los artefactos de software, y pueden utilizarse para generar otros activos como estadísticas, documentos, definición de arquitectura e incluso trozos de código. La comunidad SPL considera los *Modelos de Características* (acrónimo inglés **FMs**) como el estándar de VMs.

FODA fue la primera formalización del modelado y razonamiento de la variabilidad [28]. Las *Fórmulas Proposicionales* (acrónimo inglés **PFs**) define la variabilidad SPL mediante características con valores y restricciones Booleanos [34]. Un *Modelo con Características Numéricas* (acrónimo inglés **NFM**) admite car-

acterísticas numéricas y mezclas de restricciones aritméticas y de lógica proposicional. Los QAs son características de un sistema para evaluar su rendimiento desde la perspectiva del usuario final. La norma ISO/IEC 25010 es la más popular, y comprende ocho supertipos de calidad: funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad, portabilidad, compatibilidad y seguridad. Existen dos tipos de relaciones entre los valores de calidad y la variabilidad de un sistema: a nivel de característica y a nivel de variante [33].

Por características se entienden los valores respectivos de calidad de las características individuales junto a una fórmula de agregación, mientras que por variantes se entiende el valor absoluto para un sistema concreto, denominado configuración o producto final. El consumo de energía es una QA de variantes.

Los CSPs son cuestiones matemáticas definidas como un conjunto de objetos cuyo estado debe satisfacer una serie de restricciones. Los tipos de solucionadores más populares son SAT, SMT y CP. SAT resuelve el problema de la satisfacción Booleana, SMT generaliza el problema de la satisfacción Booleana a fórmulas más complejas y CP resuelve problemas Booleanos, numéricos y lineales mediante retroceso y propagación de restricciones.

Por otro lado, la CT es una teoría algebraica de estructuras matemáticas que permite captar y relacionar aspectos similares de las estructuras abstrayéndose al mismo tiempo de las particularidades individuales de sus disimilitudes [37]. Los principales componentes son la categoría (i.e., meta-modelo), los objetos (i.e., componentes), las flechas (i.e., relaciones), las funciones (i.e., operaciones), las rutas (i.e., composición de flechas o funciones), los elemento (i.e., variables de un objeto) y las instancia (i.e., modelos).

Finalmente, formas avanzadas de razonamiento incluyen análisis estadísticos y aprendizaje máquina. Éstos pueden utilizarse para predecir el resultado de un modelo o el comportamiento de algunas de sus características. Los más importantes en la literatura son el muestreo en la selección y medición de un subconjunto del espacio de búsqueda formado por pares de configuración/valor de QA. También se utilizan pruebas no paramétricas para determinar si dos conjuntos son significativamente diferentes entre sí. También se utilizan técnicas de aprendizaje automático para mejorar la escalabilidad reduciendo el número de muestras solicitadas. Uno de los usados en este trabajo es el *Aprendizaje por Transferencia*

(acrónimo inglés **TL**).

B.3 Motivación

Esta tesis propone un enfoque SPL para automatizar diferentes análisis de la eficiencia energética de dominios emergentes como los CPSs y las redes B5G. El recuento de configuraciones de modelos permite un rápido muestreo aleatorio insesgado de grandes espacios de productos, pero los solucionadores #SAT no soportan NFMs [50], los CP y SMT enumeran en lugar de realizar un recuento de configuraciones, y todavía no existen herramientas #CP o #SMT. La primera pregunta clave (acrónimo inglés **RQ1**) es **¿Cómo podemos ampliar los solucionadores (booleanos) más avanzados para que admitan automáticamente los NFMs sin degradar el rendimiento?**.

Hay tres componentes necesarios para automatizar la construcción de un sistema y la medición de un QA: la generación de determinadas configuraciones de un espacio de soluciones específico con un solucionador compatible con los NFMs, el modelado de los QA de un QM, y los datos de mediciones de configuraciones almacenados en centros de datos. La falta de consenso en la comunidad ha dado lugar a diferentes ideas que finalmente no se materializan en una herramienta de análisis sólida para QAs de variantes. Falta la interconexión entre configuraciones de un espacio de soluciones específico y un espacio de soluciones medidas incompleto dado por un QM y un repositorio de datos. Se necesitan enfoques agnósticos del dominio para identificar y analizar características significativas y características interactivas para los QAs de variantes cuando muchas mediciones de configuración no están disponibles y no pueden solicitarse sobre la marcha. La **RQ2** es **¿Cómo podemos proporcionar automáticamente información sobre la eficiencia energética cuando nos enfrentamos a espacios de soluciones colosales, parcialmente desconocidos y parcialmente medidos basados en NFMs y QAs por variantes?**

Responder a RQ1 y RQ2 permitiría obtener información sobre la eficiencia energética, pero son menos escalable que una solución nativa y presentaría problemas de mantenimiento. El proceso de interconexión está limitado por las escasas operaciones de razonamiento que admiten los razonadores NFM y QM, que no

tienen en cuenta los QAs. Una solución completa permitirá modelar y razonar cualquier tipo de NFM ampliado junto con QA variables. Esto es factible de implementar en un IDE que soporte un NFM y un QM extendidos unificados, y es más escalable e intuitivo que interconectar dos razonadores distantes. La **RQ3** es **¿Cómo podemos unificar adecuadamente los NFMS extendidos y los QM al tiempo que contamos con el soporte nativo de las herramientas de razonamiento automatizado?**

B.4 Propuesta

Esta sección presenta la visión general de los 3 enfoques incrementales seguidos para responder a las RQs[1-3].

La primera línea se centra en la transformación de dominios numéricos en dominios booleanos mediante vectores de bits y operaciones aritméticas. La principal propiedad de los vectores de bits es su anchura, que define los límites de valor mínimo y máximo de las variables numéricas, y si el vector es sin signo o con signo. También utiliza la representación Big-Endian, en la que el primer bit codifica el signo como positivo (0) o negativo (1). Hemos desarrollado $Nemo_2$, una herramienta que automatiza el proceso de aplicación de Bit-Blasting a aritmética con grandes vectores de bits.

Para $Nemo_2$ hemos definido un lenguaje de NFMs completo, aunque también admite estándares de entrada y salida compatibles con distintos solucionadores estado-del-arte. También presenta optimizaciones del modelo para generarlos pequeños y rápidos de razonar en cualquiera de los formatos admitidos. Los detalles más importantes es que podemos usar los modelos resultantes en las mejores estrategias de muestreo como las que se basan en técnicas insesgadas y aleatorias: Muestreo Aleatorio Uniforme (acrónimo inglés **URS**) y la Búsqueda Estadística Recursiva (acrónimo inglés **SRS**) [111]. Luego podemos encontrar las configuraciones más eficientes realizando un análisis de prueba T-Student frente a otro conjunto de muestras, calculando así la confianza. La distancia media entre la configuración calculada y la configuración con mejores resultados es tiende a ser $1\% \pm 1\%$ de media.

La segunda línea presenta dos enfoques complementarios en forma de eco-

asistentes que ayuden a responder la RQ2. El primer enfoque consiste en interconectar un solucionador NFM con un gestor de bases de datos integrando los datos reales de QAs de tipo variante obtenidos de diferentes bases de datos. Esto implica tener en cuenta todas las métricas y valores en los razonamientos, así como informar de metadatos interesantes. Nuestro enfoque propone un proceso que vincula externamente las configuraciones y sus valores de QAS de tipo variante para proporcionar información valiosa sobre la calidad de cada configuración real.

Para ellos, creamos enlaces simbólicos basados en índices entre configuraciones válidas y sus valores de QA de tipo variante. El método es asignar identificadores estáticos a las características de las hojas. Esto nos permite realizar análisis estadísticos del espacio de soluciones medido. Hemos implementamos este enfoque en nuestra segunda herramienta denominada *HADAS*, la cual brinda razonamientos de NFMs basados en solucionadores, una base de datos colaborativa, un proceso en modo servidor, una interfaz gráfica e interactiva para los usuarios, un análisis estadístico sobre la influencia de características, un microservicio para la interacción con aplicaciones de terceros y un plugin para IntelliJ IDEA. La arquitectura y funcionalidad de este sistema se basa en Clafer Modelling Language, Clafer-solver, MariaDB, análisis de diferenciales de Pearson chi-cuadrado, remuestreo aleatorio, resultados gráficos y estadísticos. En el caso del plugin para IntelliJ IDEA, la interfaz se sustituye por la del IDE, mientras que los datos se intercambian con *HADAS* en formato JSON.

El segundo enfoque de la segunda línea viene representado por *SAVRUS*, una estrategia modular que navega a través de espacios grandes y difíciles de medir con valores QA parcialmente desconocidos, trabaja con los datos disponibles y proporciona una rápida identificación y clasificación por importancia de las características y sus interacciones por pares. Comprende cinco pasos secuenciales, los cuales se integran en un prototipo de aplicación web con las siguientes opciones de técnicas para cada paso.

Para el primer paso de muestreo hemos implementado URS y *Muestreo Diversificado Basado en la Distancia* (acrónimo inglés **DDbS**). El segundo paso es aprendizaje perezoso, para el que hemos implementado los k Vecinos Cercanos (acrónimo inglés kNN) como técnica de regresión. El tercer paso es el test estadístico y para lo cual hemos seleccionado la prueba de la U de Mann-Whitney (acrónimo inglés

MWU). El cuarto paso es la detección de anomalías con selección de instancias no supervisada para lo que implementos el método de TL *Local Outlier Factor* (LOF) midiendo la desviación local de una muestra desconocida. Finalmente, se realiza una actualización de pesos para clasificar las características y sus combinaciones en pares. Los desarrolladores pueden optimizar los sistemas sustituyendo las características notables y blindando las más energéticamente eficientes.

La tercera línea es en la que unificamos los NFMs extendidos con cualquier tipo de QAs relacionados en una categoría de SPLs. La categoría unificadora $\mathcal{QVM}$ comprende tres objetos categóricos de tipo datos y cinco objetos categóricos estructurados. Se definen operaciones de razonamiento de calidad sobre configuraciones SPL, que permiten a los desarrolladores realizar directamente análisis conscientes de la calidad de los ahora unidos espacios de características, soluciones y mediciones de QAs. Adicionalmente, hemos formalizado el razonamiento **nativo** basado en calidad y agrupado en 8 operaciones sobre $\mathcal{QVM}$: generar un informe, satisfacibilidad, recontar, filtrar, limitar, aleatorizar, agregar y optimizar. Estas operaciones son functores categóricos entre $\mathcal{QVM}$ y una categoría resultante.

CQL IDE es el estado del arte de las herramientas categóricas, ya que admite la definición de categorías y el razonamiento basado en functores. Procesa operaciones con su conjunto de razonadores, y la composición de operaciones es una operación natural en álgebras y teorías formales. CQL IDE está desarrollado para soportar futuras extensiones de SPL, proporcionando una solución unificada y escalable para modelar NFMs extendidos con relaciones y razonamientos QAs.

B.5 Discusión de Resultados

$Nemo_2$ es una herramienta para modelar, ampliar, optimizar y transformar NFMs a los formatos más comunes de FM clásicos. Puede representar fórmulas complejas de hasta 12 bits de ancho. También permite transformar NFMs reales de tamaños colosales sin sobrecarga para casi todas las combinaciones de operaciones booleanas y aritméticas en menos de 15 minutos. Los modelos resultantes son idóneos para buscar configuraciones (casi) óptimas en espacios de soluciones de tamaño 10^{45} con URS o SRS en menos de 10 segundos con un solucionador BDD. También podemos realizar un reconteo colosal de tamaño 10^{248} en menos de 5

horas con un solucionador #SAT. Por tanto, nuestra **respuesta a RQ1 es que podemos codificar NFs y restricciones aritméticas en vectores de bits y PFs. En consecuencia, con** $Nemo_2$ **extendemos automáticamente los FM con ramas NFMs, ejecutamos NFMs en solucionadores (booleanos) clásicos y realizamos búsquedas de configuraciones (casi) óptimas con una degradación mínima del rendimiento (i.e., segundos)**.

La herramienta web *HADAS* y sus plugins para JetBrains IDE proporcionan ricos análisis de razonamiento en forma de percepciones gráficas de eficiencia energética, correlaciones estadísticas y consejos contextuales de código de CPSs. El enfoque *SAVRUS* permite crear una clasificación de características e interacciones influyentes (es decir, preocupaciones) navegando rápidamente por grandes espacios medidos que contienen muchas configuraciones desconocidas aleatoriamente. Con SAVRUS hemos conseguido analizar un espacio de $\sim \mathbf{5.3 * 10^8}$ configuraciones en donde tan solo habían 132.500 medidas aleatorias, y todo ello en menos de un 1 minuto con cobertura y exactitud de aproximadamente el 80%. Por tanto, nuestra **respuesta a RQ2 es que podemos ofrecer una visión completa de QAs de tipo variante con un proceso basado en índices como *HADAS*, interconectando un razonamiento del solucionador NFM con gestores de bases de datos que albergan mediciones como los valores de eficiencia energética. Para los casos en los que el espacio medido es grande, sesgado e incompleto, podemos proporcionar una clasificación de las preocupaciones que consumen energía siguiendo la secuencia de técnicas *SAVRUS*: muestreo basado en solucionadores, aprendizaje perezoso, pruebas estadísticas, aprendizaje de transferencia y clasificación ponderada.**

Por último, hemos evaluado empíricamente este enfoque en CQL IDE transformando en un único $\mathcal{QVM}$ el NFM y el modelo de base de datos ya existente en HADAS. También los hemos comprobado empíricamente con 5 NFMs reales y comparando su ejecución con el estado-del-arte para hasta 20 operaciones de razonamiento basadas en calidad. Nuestra implementación en CQL IDE due el único solucionador compatible con los modelos completos y las 20 operaciones, y en promedio también fue la solución más rápida. Por tanto, nuestra **respuesta a RQ3 es que con un enfoque CT, podemos unificar el NFM extendido y**

QM como la categoría $\mathcal{QVM}$, así como definir cualquier operación basada en calidad para un solucionador categórico como CQL IDE permitiendo razonar nativamente sobre espacios de solución medidos por la calidad.

B.6 Conclusiones y Trabajo Futuro

Nuestra investigación se ha centrado en cubrir todas las propiedades de modelado y técnicas de razonamiento que requiere el desarrollo de CPSs y dispositivos IoT energéticamente eficientes. Esto incluye el modelado y el razonamiento óptimo de NFs y restricciones aritméticas, la medición, el modelado y el razonamiento de QA complejos en función de las variantes, el análisis rápido de espacios de soluciones colosales, la consideración de espacios de soluciones parcialmente no medidos, las restricciones aritméticas entre QA en función de las variantes y agregadas en función de las características, las operaciones de razonamiento complejas conscientes de la calidad y las funciones de optimización multiobjetivo de espacios de soluciones medidos definidos por un modelo unificado de variabilidad y calidad.

Desarrollamos 3 herramientas y 1 framework para modelar NFMs: el transformador de NFMs a FMs $Nemo_2$, los servicios web *HADAS*, la clasificación de *SAVRUS* y el framework unificador $\mathcal{QVM}$ para CQL IDE. Estas herramientas proporcionan a los desarrolladores de CPS e IoT servicios y métodos para descubrir las alternativas más eficientes, a los desarrolladores de herramientas micro-servicios para incluir el razonamiento en aplicaciones de terceros, y a la comunidad SPL un estándar ampliado de modelado de variabilidad y calidad.

Como trabajo futuro, planeamos integrar técnicas de aprendizaje automático y de transferencia en el framework $\mathcal{QVM}$, introducir redes neuronales autoformadas, sustituir kNN por *Approximate Nearest Neighbours* (ANN) [134] y ampliar el análisis por pares a T-wise. También tenemos previsto explorar 5 nuevas direcciones: aumentar las herramientas de modelado y los solucionadores, explorar otras teorías algebraicas, formalizar operaciones exóticas, descubrir propiedades útiles de los $\mathcal{QVM}s$ y aplicar los conocimientos a otros campos de la ingeniería de software.

References

[1] OECD, *Energy*. OECD Library, 2012. [Online]. Available: https://www.oecd-ilibrary.org/content/publication/9789264115118-en 3, 77

[2] M. F. Wehner, J. R. Arnold, T. Knutson, K. E. Kunkel, and A. N. LeGrande, "Droughts, floods, and wildfires," *Climate science special report: fourth national climate assessment*, vol. 1, no. GSFC-E-DAA-TN49033, 2017. 3

[3] J. Kim, J. Moon, and Y. G. Kim, "Thematic section: Green transition and national efforts towards net-zero target," *ematic Section: Green Transition and National E orts towards Net-Zero Target*, p. 43, 2023. 3, 77

[4] K. E. Rosendahl and D. R. Rubiano, "How effective is lithium recycling as a remedy for resource scarcity?" *Environmental and Resource Economics*, vol. 74, pp. 985–1010, 2019. 4

[5] I. Research, "Data Centres and Data Transmission Networks," https://www.iea.org/energy-system/buildings/data-centres-and-data-transmission-networks, 2020. 4

[6] H. Kanso, A. Noureddine, and E. Exposito, "A review of energy aware cyber-physical systems," *Cyber-Physical Systems*, pp. 1–42, 2023. 4, 5, 77

[7] I. Research, "World Energy Outlook 2022 Free Dataset," https://www.iea.org/data-and-statistics/data-product/world-energy-outlook-2022-free-dataset, 2022. 5

[8] D. AI, "Energy Data: Best Energy Datasets and Databases," https://datarade.ai/data-categories/energy-data, 2023. 5

[9] M. A. Schuh, T. Wylie, and R. A. Angryk, "Mitigating the curse of dimensionality for exact knn retrieval," in *The Twenty-Seventh International Flairs Conference*, 2014. 6

[10] J. M. Horcas, J. A. Galindo, M. Pinto, L. Fuentes, and D. Benavides, "Fm fact label: A configurable and interactive visualization of feature model characterizations," in *Proceedings of the 26th ACM International Systems and Software Product Line Conference - Volume B*, ser. SPLC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 42–45. [Online]. Available: https://doi.org/10.1145/3503229.3547025 6

[11] F. Rossi, P. Van Beek, and T. Walsh, *Handbook of constraint programming*. Elsevier, 2006. 6, 15, 78

[12] P. Van Beek, "Backtracking search algorithms," in *Foundations of artificial intelligence*. Elsevier, 2006, vol. 2, pp. 85–134. 7

[13] S. Apel, D. Batory, C. Kästner, and G. Saake, *Feature-oriented software product lines*. Springer, 2016. 7, 12, 13

[14] J.-M. Horcas, M. Pinto, and L. Fuentes, "Software product line engineering: A practical experience," in *Proceedings of the 23rd International Systems and Software Product Line Conference - Volume A*, ser. SPLC '19. New York, New York, USA: ACM, 2019, p. 164–176. 7, 9, 35

[15] D.-J. Munoz, J. Oh, M. Pinto, L. Fuentes, and D. Batory, ": A tool to transform feature models with numerical features and arithmetic constraints," in *Reuse and Software Quality*, G. Perrouin, N. Moha, and A.-D. Seriai, Eds. Cham: Springer International Publishing, 2022, pp. 59–75. 7, 37

[16] J. Estdale and E. Georgiadou, "Applying the iso/iec 25010 quality models to software product," in *Systems, Software and Services Process Improvement*, X. Larrucea, I. Santamaria, R. V. O'Connor, and R. Messnarz, Eds. Cham: Springer International Publishing, 2018, pp. 492–503. 8

[17] D. Benavides, P. Trinidad, and A. Ruiz-Cortés, *Automated Reasoning on Feature Models*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp.

361–373. [Online]. Available: https://doi.org/10.1007/978-3-642-36926-1_29 8, 14, 44

[18] J.-M. Horcas, M. Pinto, and L. Fuentes, "Towards the dynamic reconfiguration of quality attributes," in *Companion Proceedings of the 15th International Conference on Modularity*, ser. MODULARITY Companion 2016. New York, NY, USA: Association for Computing Machinery, 2016, p. 131–136. [Online]. Available: https://doi.org/10.1145/2892664.2892686 8

[19] J. Harrison, *Handbook of practical logic and automated reasoning*. Cambridge University Press, 2009. 8

[20] D.-J. Munoz, J. Oh, M. Pinto, L. Fuentes, and D. Batory, "Uniform random sampling product configurations of feature models that have numerical features," in *Proceedings of the 23rd International Systems and Software Product Line Conference - Volume A*. New York, New York, USA: ACM, 2019, p. 289–301. 9, 13, 35, 37

[21] D.-J. Munoz, M. Pinto, L. Fuentes, and D. Batory, "Transforming numerical feature models into propositional formulas and the universal variability language," *Journal of Systems and Software*, p. 111770, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121223001656 9, 37

[22] D.-J. Munoz, D. Gurov, M. Pinto, and L. Fuentes, "Category theory framework for variability models with non-functional requirements," in *Advanced Information Systems Engineering*, M. La Rosa, S. Sadiq, and E. Teniente, Eds. Cham: Springer International Publishing, 2021, pp. 397–413. 9, 39, 44

[23] D.-J. Munoz, M. Pinto, and L. Fuentes, "Quality-aware analysis and optimisation of virtual network functions," in *Proceedings of the 26th ACM International Systems and Software Product Line Conference - Volume A*, ser. SPLC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 210–221. [Online]. Available: https://doi.org/10.1145/3546932.3547007 9, 40

[24] ——, "Finding correlations of features affecting energy consumption and performance of web servers using the hadas eco-assistant," *Computing*, vol. 100, no. 11, p. 1155–1173, nov 2018. 9, 38, 43, 44

[25] D.-J. Munoz, J. A. Montenegro, M. Pinto, and L. Fuentes, "Energy-aware environments for the development of green applications for cyber–physical systems," *Future Generation Computer Systems*, vol. 91, pp. 536–554, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X18307295 9, 38

[26] D.-J. Munoz, M. Pinto, and L. Fuentes, "Detecting feature influences to quality attributes in large and partially measured spaces using smart sampling and dynamic learning," *Knowledge-Based Systems*, vol. 270, p. 110558, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705123003088 9, 39

[27] P. Clements and L. Northrop, *Software product lines*. Addison-Wesley Boston, 2002. 11

[28] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-oriented domain analysis (foda) feasibility study," Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, Tech. Rep., 1990. 13, 79

[29] R. Heradio, D. Fernandez-Amoros, J. Galindo, D. Benavides, D. Batory, "Uniform and Scalable Sampling of Highly Configurable Systems," *Empirical Software Engineering*, Jan. 2022. 13, 16

[30] R. Bryant, "Binary Decision Diagrams," in *Handbook of Model Checking*, E. Clarke, T. Henzinger, H. Veith, and R. Bloem, Eds. Springer, 2018, pp. 1–2. 13, 43

[31] D. Batory, *Automated Software Design, Vol. 1*. Lulu.com, 2021. 13

[32] J. Oh, P. Gazillo, D. Batory, M. Heule, and M. Myers, "Uniform Sampling from Kconfig Feature Models," University of Texas at Austin, Department of Computer Science, Tech. Rep. TR-19-02, 2019. 13, 43

[33] V. Myllärniemi, J. Savolainen, and T. Männistö, "Performance variability in software product lines: A case study in the telecommunication domain," in *Proceedings of the 17th International Software Product Line Conference*, ser. SPLC '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 32–41. [Online]. Available: https://doi.org/10.1145/2491627.2491631 14, 80

[34] D. Batory, "Feature models, grammars, and propositional formulas," in *Software Product Lines*, H. Obbink and K. Pohl, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 7–20. 15, 79

[35] C. Barrett and C. Tinelli, *Satisfiability Modulo Theories*. Cham: Springer International Publishing, 2018, pp. 305–343. [Online]. Available: https://doi.org/10.1007/978-3-319-10575-8_11 15

[36] M. Thurley, "sharpsat-counting models with advanced component caching and implicit bcp," *SAT*, vol. 4121, pp. 424–429, 2006. 16, 43

[37] M. Barr and C. Wells, *Category theory for computing science*. Hoboken, New Jersey, USA: Prentice Hall, 1990. 16, 80

[38] K. S. Brown, D. I. Spivak, and R. Wisnesky, "Categorical data integration for computational science," *Computational Materials Science*, vol. 164, pp. 127–132, 2019. 17

[39] T. Pett, T. Thüm, T. Runge, S. Krieter, M. Lochau, and I. Schaefer, "Product sampling for product lines: The scalability challenge," in *Proceedings of the 23rd International Systems and Software Product Line Conference-Volume A*. Paris, France: Association for Computing Machinery, 2019, pp. 78–83. 17, 21, 22, 23

[40] M. Varshosaz, M. Al-Hajjaji, T. Thüm, T. Runge, M. R. Mousavi, and I. Schaefer, "A classification of product sampling for software product lines," in *Proceedings of the 22nd International Systems and Software Product Line Conference-Volume 1*. Gothenburg Sweden: ACM, 2018, pp. 1–13. 17

[41] I. Etikan and K. Bala, "Sampling and sampling methods," *Biometrics & Biostatistics International Journal*, vol. 5, no. 6, p. 00149, 2017. 17

[42] F. Medeiros, C. Kästner, M. Ribeiro, R. Gheyi, and S. Apel, "A comparison of 10 sampling algorithms for configurable systems," in *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, IEEE. Austin, USA: IEEE, 2016, pp. 643–654. 17, 45

[43] G. D. Ruxton, "The unequal variance t-test is an underused alternative to Student's t-test and the Mann–Whitney U test," *Behavioral Ecology*, vol. 17, no. 4, pp. 688–690, 05 2006. [Online]. Available: https://doi.org/10.1093/beheco/ark016 18

[44] E. Alpaydin, *Introduction to machine learning*. Massachusetts, USA: MIT press, 2020. 18, 22

[45] C. Kaltenecker, A. Grebhahn, N. Siegmund, and S. Apel, "The interplay of sampling and machine learning for software performance prediction," *IEEE Software*, vol. 37, no. 4, pp. 58–66, 2020. 18

[46] J. Krüger, S. Nielebock, S. Krieter, C. Diedrich, T. Leich, G. Saake, S. Zug, and F. Ortmeier, "Beyond software product lines: Variability modeling in cyber-physical systems," in *Proceedings of the 21st International Systems and Software Product Line Conference - Volume A*, ser. SPLC '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 237–241. [Online]. Available: https://doi.org/10.1145/3106195.3106217 21

[47] C. Jiang, T. Fan, H. Gao, W. Shi, L. Liu, C. Cérin, and J. Wan, "Energy aware edge computing: A survey," *Computer Communications*, vol. 151, pp. 556–580, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S014036641930831X 21

[48] J. Liang, V. Ganesh, K. Czarnecki, and V. Raman, "SAT-based Analysis of Large Real-world Feature Models Is Easy," in *SPLC*. NJ, USA: IEEE/ACM, 2015, pp. 2–8. 21

[49] J. Oh, D. Batory, M. Myers, and N. Siegmund, "Finding Near-optimal Configurations in Product Lines by Random Sampling," in *ESEC/FSE*, 2017, pp. 3–9. 21, 22, 28

[50] C. Sundermann, M. Nieke, P. M. Bittner, T. Heß, T. Thüm, and I. Schaefer, "Applications of #SAT Solvers on Feature Models," in *VaMoS*. NY, USA: ACM, 2021, pp. 3–8. 21, 81

[51] J. Oh, D. Batory, and R. Heradio, "Finding near-optimal configurations in colossal spaces with statistical guarantees," *ACM TOSEM*, to appear 2024. 21, 28

[52] M. Borges, Q.-S. Phan, A. Filieri, and C. S. Păsăreanu, "Model-counting approaches for nonlinear numerical constraints," in *NASA Formal Methods*, C. Barrett, M. Davies, and T. Kahsai, Eds. Cham: Springer International Publishing, 2017, pp. 131–138. 22

[53] L. Zhou, J. Li, F. Li, Q. Meng, J. Li, and X. Xu, "Energy consumption model and energy efficiency of machine tools: a comprehensive literature review," *Journal of Cleaner Production*, vol. 112, pp. 3721–3734, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0959652615006617 22

[54] A. E. Chacón-Luna, A. M. Gutiérrez, J. A. Galindo, and D. Benavides, "Empirical software product line engineering: A systematic literature review," *Information and Software Technology*, vol. 128, p. 106389, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584920301555 22

[55] A. Yrjönen and J. Merilinna, "Extending the nfr framework with measurable non-functional requirements," in *Proceedings of the 2nd International Workshop on Non-functional System Properties in Domain Specific Modeling Languages*, M. Boškoviæ, D. Gaševiæ, C. Pahl , and B. Schätz, Eds. New York, New York, USA: ACM, 2009, pp. 0–14, 2nd International Workshop on Non-functional System Properties in Domain Specific Modeling Lan-

guages, NFPinDSML2009, NFPinDSML2009 ; Conference date: 04-10-2009 Through 04-10-2009. 23, 44

[56] J. Meinicke, T. Thüm, R. Schröter, F. Benduhn, T. Leich, and G. Saake, *Quality Assurance for Feature Models and Configurations*. Cham: Springer International Publishing, 2017, ch. 8, pp. 81–94. 23, 44

[57] R. Bashroush, M. Garba, R. Rabiser, I. Groher, and G. Botterweck, "Case tool support for variability management in spls," *ACM Comput. Surv.*, vol. 50, no. 1, Mar. 2017. 23

[58] S. Fischer, L. Linsbauer, A. Egyed, and R. E. Lopez-Herrejon, "Predicting higher order structural feature interactions in variable systems," in *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. CA, USA: IEEE, 2018, pp. 252–263. 23

[59] A. Knüppel, T. Thüm, S. Mennicke, J. Meinicke, and I. Schaefer, "Is there a mismatch between real-world feature models and product-line research?" in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2017. New York, NY, USA: Association for Computing Machinery, 2017, p. 291–302. [Online]. Available: https://doi.org/10.1145/3106237.3106252 26

[60] C. Sundermann, K. Feichtinger, D. Engelhardt, R. Rabiser, and T. Thüm, "Yet another textual variability language? a community effort towards a unified language," in *Proceedings of the 25th ACM International Systems and Software Product Line Conference - Volume A*, ser. SPLC '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 136–147. [Online]. Available: https://doi.org/10.1145/3461001.3471145 27

[61] C. Pang, A. Hindle, B. Adams, and A. E. Hassan, "What do programmers know about software energy consumption?" *IEEE Software*, vol. 33, no. 3, pp. 83–89, 2016. 28

[62] O. Laayati, M. Bouzi, and A. Chebak, "Smart energy management: Energy consumption metering, monitoring and prediction for mining industry," in

2020 IEEE 2nd International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS), 2020, pp. 1–5. 28

[63] Z. Yi and P. H. Bauer, "Effects of environmental factors on electric vehicle energy consumption: a sensitivity analysis," *IET Electrical Systems in Transportation*, vol. 7, no. 1, pp. 3–13, 2017. 29

[64] D. An and P. Meenan, "Why marketers should care about mobile page speed," 2016. 31

[65] A. Bacchelli and C. Bird, "Expectations, outcomes, and challenges of modern code review," in *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 2013, pp. 712–721. 31

[66] S. Vilkomir, O. Starov, and R. Bhambroo, "Evaluation of t-wise approach for testing logical expressions in software," in *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops*, IEEE. Luxembourg, Luxembourg: IEEE, 2013, pp. 249–256. 31

[67] C. Kaltenecker, A. Grebhahn, N. Siegmund, J. Guo, and S. Apel, "Distance-based sampling of software configuration spaces," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, IEEE. Montreal, Canada: IEEE, 2019, pp. 1084–1094. 32

[68] G. Baldini and D. Geneiatakis, "A performance evaluation on distance measures in knn for mobile malware detection," in *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2019, pp. 193–198. 32

[69] L. Yang, H. Zhang, H. Shen, X. Huang, X. Zhou, G. Rong, and D. Shao, "Quality assessment in systematic literature reviews: A software engineering perspective," *Information and Software Technology*, vol. 130, p. 106397, 2021. 35

[70] J. A. Pereira, M. Acher, H. Martin, J.-M. Jézéquel, G. Botterweck, and A. Ventresque, "Learning software configuration spaces: A systematic liter-

ature review," *Journal of Systems and Software*, vol. 182, p. 111044, 2021. 35

[71] D.-J. Munoz, "Achieving energy efficiency using a software product line approach," in *Proceedings of the 21st International Systems and Software Product Line Conference - Volume B*, ser. SPLC '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 131–138. [Online]. Available: https://doi.org/10.1145/3109729.3109744 38

[72] D. J. M. Guerra, M. P. Alarcon, and L. F. Fernandez, "HADAS: Asistente de eco-eficiencia con repositorio de consumo energético," in *JISBD2017*. SISTEDES, 2017. [Online]. Available: http://hdl.handle.net/11705/JISBD/2017/005 38

[73] . D. J. Munoz Guerra and . L. Fuentes, "Aplicabilidad de la Caracterización de Benchmarks a Modelos de Variabilidad," in *Actas de las XXIII Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2018)*. Sistedes, 2018. [Online]. Available: https://hdl.handle.net/11705/JISBD/2018/061 38

[74] D.-J. Munoz, J. A. Montenegro, M. Pinto, and L. Fuentes, "Green security plugin for pervasive computing using the hadas toolkit," in *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, 2017, pp. 796–803. 38

[75] D.-J. Munoz, M. Pinto, and L. Fuentes, "Green software development and research with the hadas toolkit," in *Proceedings of the 11th European Conference on Software Architecture: Companion Proceedings*, ser. ECSA '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 205–211. [Online]. Available: https://doi.org/10.1145/3129790.3129818 38

[76] ——, "Hadas and web services: Eco-efficiency assistant and repository use case evaluation," in *2017 International Conference in Energy and Sustainability in Small Developing Economies (ES2DE)*, 2017, pp. 1–6. 38

[77] ——, "Hadas: Analysing quality attributes of software configurations," in *Proceedings of the 23rd International Systems and Software Product Line Conference - Volume B*, ser. SPLC '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 13–16. [Online]. Available: https://doi.org/10.1145/3307630.3342385 38

[78] . D. Munoz, . M. Pinto, and . L. Fuentes, "Teoría de Categorías Aplicada a Variabilidad," in *Actas de las XXV Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2021)*. Sistedes, 2021. [Online]. Available: https://hdl.handle.net/11705/JISBD/2021/057 39

[79] C. Henard, M. Papadakis, M. Harman, and Y. Le Traon, "Combining multi-objective search and constraint solving for configuring large software product lines," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1, 2015, pp. 517–528. 40

[80] D.-J. Munoz, M. Pinto, D. Gurov, and L. Fuentes, "Defining categorical reasoning of numerical feature models with feature-wise and variant-wise quality attributes," in *Proceedings of the 26th ACM International Systems and Software Product Line Conference - Volume B*, ser. SPLC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 132–139. [Online]. Available: https://doi.org/10.1145/3503229.3547057 40

[81] . D. Munoz, . M. Pinto, and . L. Fuentes, "Composición Categórica de Análisis Automáticos para Líneas de Productos Extendidas," in *Actas de las XXVI Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2022)*. Sistedes, 2022. [Online]. Available: https://hdl.handle.net/11705/JISBD/2022/8601 40

[82] J. Shi, J. Wan, H. Yan, and H. Suo, "A survey of cyber-physical systems," in *Wireless Communications and Signal Processing (WCSP), 2011 International Conference on*. IEEE, 2011, pp. 1–6. 41

[83] D. Zeng, L. Gu, and H. Yao, "Towards energy efficient service composition in green energy powered cyber–physical fog systems," *Future Generation Computer Systems*, 2018. 41

[84] Y. Liu, A. Liu, S. Guo, Z. Li, Y.-J. Choi, and H. Sekiya, "Context-aware collect data with energy efficient in cyber–physical cloud systems," *Future Generation Computer Systems*, 2017. 41

[85] P. Palensky, E. Widl, and A. Elsheikh, "Simulating cyber-physical energy systems: Challenges, tools and methods," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 3, pp. 318–326, 2014. 41

[86] S. A. Murtza, A. Ahmad, M. Y. Qadri, N. N. Qadri, and J. Ahmed, "Optimizing energy and throughput for mpsocs: an integer particle swarm optimization approach," *Computing*, pp. 1–18, 2017. 41

[87] S. Parvin, F. K. Hussain, O. K. Hussain, T. Thein, and J. S. Park, "Multi-cyber framework for availability enhancement of cyber physical systems," *Computing*, vol. 95, no. 10-11, pp. 927–948, 2013. 41

[88] G. Wang, Y. Guan, Y. Wang, and Z. Shao, "Energy-aware assignment and scheduling for hybrid main memory in embedded systems," *Computing*, vol. 98, no. 3, pp. 279–301, 2016. 43

[89] L. Marchezan, E. Rodrigues, W. K. G. Assunção, M. Bernardino, F. P. Basso, and J. Carbonell, "Software product line scoping: A systematic literature review," *Journal of Systems and Software*, vol. 186, 2022. 43

[90] T. Berger, S. She, R. Lotufo, A. Wąsowski, and K. Czarnecki, "A Study of Variability Models and Languages in the Systems Software Domain," *IEEE TSE*, vol. 39, no. 12, 2013. 43

[91] V. Döller and D. Karagiannis, "Formalizing Conceptual Modeling Methods with MetaMorph," in *Enterprise, Business-Process and Information Systems Modeling*. Springer: Springer, 2021, pp. 4–14. 43

[92] C. Kästner *et al.*, "Variability-aware Parsing in the Presence of Lexical Macros and Conditional Compilation," in *OOPSLA*, vol. 46. NJ, USA: IEEE/ACM, 2011, pp. 5–18. 43

[93] K. Shi, "Combining Evolutionary Algorithms with Constraint Solving for Configuration Optimization," in *ICSME*. IEEE/ACM, sep 2017, pp. 3–4. 43

[94] D. Benavides, S. Segura, and A. Ruiz-Cortés, "Automated Analysis of Feature Models 20 Years Later: A Literature Review," *Information Systems*, vol. 35, no. 6, 2010. 43, 46

[95] Q. Phan, "Model Counting Modulo Theories," Ph.D. dissertation, Queen Mary University of London, apr 2015. 43

[96] D. Chistikov, R. Dimitrova, and R. Majumdar, "Approximate Counting in SMT and Value Estimation for Probabilistic Programs," *Acta Informatica*, vol. 54, no. 8, 2017. 43

[97] L. De Moura and N. Bjørner, "Z3: An efficient smt solver," in *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, Springer. NY, USA: Springer, 2008, pp. 337–340. 43

[98] B. Dutertre, "Yices 2.2," in *International Conference on Computer Aided Verification*, Springer. NY, USA: Springer, 2014, pp. 737–744. 43

[99] V. Ganesh and D. Dill, "The Simple Theorem Prover (STP) solver," https://stp.github.io/, 2006. 43

[100] N. Sorensson and N. Een, "Minisat V1. 13-a Sat Solver with Conflict-clause Minimization," *SAT*, vol. 2005, no. 53, 2005. 43

[101] R. Olaechea, S. Stewart, K. Czarnecki, and D. Rayside, "Modelling and multi-objective optimization of quality attributes in variability-rich software," in *Proceedings of the Fourth International Workshop on Nonfunctional System Properties in Domain Specific Modeling Languages*. New York, New York, USA: ACM, 2012. 44

[102] N. Siegmund, M. Rosenmüller, M. Kuhlemann, C. Kästner, S. Apel, and G. Saake, "Spl conqueror: Toward optimization of non-functional properties in software product lines," *Software Quality Journal*, vol. 20, no. 3–4, p. 487–517, sep 2012. 44

[103] P. Trinidad, A. Ruiz-Cortés, and D. Benavides, *Automated Analysis of Stateful Feature Models*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 375–380. [Online]. Available: https://doi.org/10.1007/978-3-642-36926-1_30 44

[104] G. Zhang, H. Ye, and Y. Lin, "Quality attribute modeling and quality aware product configuration in software product lines," *Softw. Qual. J.*, vol. 22, no. 3, pp. 365–401, 2014. 44

[105] C. Henard, M. Papadakis, M. Harman, and Y. Le Traon, "Combining multi-objective search and constraint solving for configuring large software product lines," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1. New York, NY, USA: Association for Computing Machinery, 2015, pp. 517–528. 44

[106] T. Saber, D. Brevet, G. Botterweck, and A. Ventresque, "Milpibea: Algorithm for multi-objective features selection in (evolving) software product lines," in *Evolutionary Computation in Combinatorial Optimization - 20th European Conference, EvoCOP 2020, Held as Part of EvoStar 2020, Seville, Spain, April 15-17, 2020, Proceedings*, vol. 12102. Springer, 2020, pp. 164–179. [Online]. Available: https://doi.org/10.1007/978-3-030-43680-3_11 44

[107] G. G. Pascual, R. E. Lopez-Herrejon, M. Pinto, L. Fuentes, and A. Egyed, "Applying multiobjective evolutionary algorithms to dynamic software product lines for reconfiguring mobile applications," *J. Syst. Softw.*, vol. 103, pp. 392–411, 2015. [Online]. Available: https://doi.org/10.1016/j.jss.2014.12.041 44

[108] R. Olaechea, D. Rayside, J. Guo, and K. Czarnecki, "Comparison of exact and approximate multi-objective optimization for software product lines," in *Proceedings of the 18th International Software Product Line Conference - Volume 1*, ser. SPLC '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 92–101. [Online]. Available: https://doi.org/10.1145/2648511.2648521 44

[109] J. Guo, K. Czarnecki, S. Apel, N. Siegmund, and A. Wasowski, "Variability-aware performance prediction: A statistical learning approach," in *Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE'13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 301–311. [Online]. Available: https://doi.org/10.1109/ASE.2013.6693089 45

[110] A. Sarkar, J. Guo, N. Siegmund, S. Apel, and K. Czarnecki, "Cost-efficient sampling for performance prediction of configurable systems (t)," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, IEEE. Piscataway, NJ, USA: IEEE/ACM, 2015, pp. 342–352. 45

[111] J. Oh, D. Batory, M. Myers, and N. Siegmund, "Finding near-optimal configurations in product lines by random sampling," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ACM. Piscataway, NJ, USA: IEEE/ACM, 2017, pp. 61–71. 45, 82

[112] S. El-Sharkawy, A. Krafczyk, and K. Schmid, "An empirical study of configuration mismatches in linux," in *Proceedings of the 21st International Systems and Software Product Line Conference-Volume A*, ACM. New York, NY, USA: ACM, 2017, pp. 19–28. 45

[113] J. Guo, J. H. Liang, K. Shi, D. Yang, J. Zhang, K. Czarnecki, V. Ganesh, and H. Yu, "Smtibea: a hybrid multi-objective optimization algorithm for configuring large constrained software product lines," *Software & Systems Modeling*, vol. 0, pp. 1–20, 2017. 45

[114] C. Henard, M. Papadakis, M. Harman, and Y. Le Traon, "Combining multi-objective search and constraint solving for configuring large software product lines," in *Proceedings of the 37th International Conference on Software Engineering-Volume 1*, IEEE Press. Piscataway, NJ, USA: IEEE/ACM, 2015, pp. 517–528. 45

[115] A. S. Sayyad, J. Ingram, T. Menzies, and H. Ammar, "Scalable product line configuration: A straw to break the camel's back," in *2013 28th IEEE/ACM*

International Conference on Automated Software Engineering (ASE). ASE: IEEE/ACM, Nov 2013, pp. 465–474. 45

[116] J. Chen, V. Nair, R. Krishna, and T. Menzies, "" sampling" as a baseline optimizer for search-based software engineering," *IEEE Transactions on Software Engineering*, vol. 0, 2018. 45

[117] R. Tartler, D. Lohmann, C. Dietrich, C. Egger, and J. Sincero, "Configuration coverage in the analysis of large-scale system software," in *Proceedings of the 6th Workshop on Programming Languages and Operating Systems*, ser. PLOS '11. New York, NY, USA: ACM, 2011, pp. 2:1–2:5. [Online]. Available: http://doi.acm.org/10.1145/2039239.2039242 45

[118] M. Al-Hajjaji, S. Krieter, T. Thüm, M. Lochau, and G. Saake, "Incling: efficient product-line testing using incremental pairwise sampling," in *ACM SIGPLAN Notices*, vol. 52, ACM. ACM: ACM, 2016, pp. 144–155. 45

[119] A. K. Sandhu and R. S. Batth, "Integration of artificial intelligence into software reuse: An overview of software intelligence," in *2021 2nd International Conference on Computation, Automation and Knowledge Management (ICCAKM)*, 2021, pp. 357–362. 45

[120] D. Tan, M. Suvarna, Y. Shee Tan, J. Li, and X. Wang, "A three-step machine learning framework for energy profiling, activity state prediction and production estimation in smart process manufacturing," *Applied Energy*, vol. 291, p. 116808, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S030626192100310X 45

[121] T. Mangels, A. Murarasu, F. Oden, A. Fishkin, and D. Becker, "Efficient analysis at edge," in *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion*. L'Aquila, Italy: IEEE, 2017, pp. 85–90. 45

[122] C. Feng, Y. Wang, Z. Zhao, T. Q. S. Quek, and M. Peng, "Joint optimization of data sampling and user selection for federated learning in the mobile

edge computing systems," in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*. Dublin, Ireland: IEEE, 2020, pp. 1–6. 45

[123] F. Roos-Frantz, D. Benavides, A. Ruiz-Cortés, A. Heuer, and K. Lauenroth, "Quality-aware analysis in product line engineering with the orthogonal variability model," *Software Quality Journal*, vol. 20, no. 3, pp. 519–565, 2012. 45

[124] L. Lesoil, M. Acher, A. Blouin, and J.-M. Jézéquel, "Deep software variability: Towards handling cross-layer configuration," in *15th International Working Conference on Variability Modelling of Software-Intensive Systems*, ser. VaMoS'21. New York, NY, USA: Association for Computing Machinery, 2021, p. 8. [Online]. Available: https://doi.org/10.1145/3442391.3442402 45

[125] M. H. Alsafrjalani and T. Adegbija, "Tasat: Thermal-aware scheduling and tuning algorithm for heterogeneous and configurable embedded systems," in *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, ser. GLSVLSI '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 75–80. [Online]. Available: https://doi.org/10.1145/3194554.3194576 45

[126] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. S. Shen, "Toffee: Task offloading and frequency scaling for energy efficiency of mobile devices in mobile edge computing," *IEEE Transactions on Cloud Computing*, vol. 1, pp. 1–1, 2019. 45

[127] A. Gamatié, G. Devic, G. Sassatelli, S. Bernabovi, P. Naudin, and M. Chapman, "Towards energy-efficient heterogeneous multicore architectures for edge computing," *IEEE Access*, vol. 7, pp. 49 474–49 491, 2019. 45

[128] M. Couto, P. Borba, J. Cunha, J. a. P. Fernandes, R. Pereira, and J. a. Saraiva, "Products go green: Worst-case energy consumption in software product lines," in *Proceedings of the 21st International Systems and Software Product Line Conference - Volume A*, ser. SPLC '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 84–93. [Online]. Available: https://doi.org/10.1145/3106195.3106214 45

[129] S. Chowdhury, S. Borle, S. Romansky, and A. Hindle, "Greenscaler: Training software energy models with automatic test generation," *Empirical Softw. Engg.*, vol. 24, no. 4, p. 1649–1692, Aug. 2019. [Online]. Available: https://doi.org/10.1007/s10664-018-9640-7 46

[130] A. A. Fraenkel, Y. Bar-Hillel, and A. Levy, *Foundations of set theory*. Elsevier, 1973. 47

[131] J. Lambek and P. J. Scott, *Introduction to higher-order categorical logic*. Cambridge University Press, 1988, vol. 7. 47

[132] A. Kızıltoprak and N. Y. Köse, "Relational thinking: The bridge between arithmetic and algebra," *International Journal of Elementary Education*, vol. 10, no. 1, pp. 131–145, 2017. 47

[133] J. Elffers, J. Giráldez-Cru, J. Nordström, and M. Vinyals, "Using combinatorial benchmarks to probe the reasoning power of pseudo-boolean solvers," in *Int. Conference on Theory and Applications of Satisfiability Testing*. Springer, 2018. 47

[134] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, 2006, pp. 459–468. 51, 86