Software: Practice and Experience

WILEY

RESEARCH ARTICLE

On the Energy Consumption of ATL Transformations

Riccardo Rubei | Juri di Rocco | Davide di Ruscio

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica (DISIM), Università degli studi dell'Aquila, L'Aquila, Italy

Correspondence: Riccardo Rubei (riccardo.rubei@univaq.it)

Received: 28 May 2024 | **Revised:** 26 November 2024 | **Accepted:** 12 January 2025

Funding: This work is supported by the Ministero Università e Ricerca, EMELIOT project (PRIN 2020), Grant Number (2020W3A5FY), Ministero Università e Ricerca, TRex SE project (PRIN 2022), Grant Number (2022LKJWHC), Ministero Università e Ricerca, FRINGE project (PRIN 2022), Grant Number (P2022553SL), European Union NextGenerationEU, Matters Project, SERICS program (CUPJ33C22002810001).

Keywords: green computing | model-driven engineering | software engineering | software performances

ABSTRACT

Background: Model transformations play a crucial role in Model-Driven Engineering (MDE), with the ATLAS Transformation Language (ATL) being a powerful technology for developing model-to-model transformations.

Methods: This paper presents a comprehensive investigation into the energy consumption of ATL transformations, aiming to identify possible correlations among transformation rules, model size, and metamodel structural characteristics. We conducted experiments on 52 ATL transformations, analyzing power usage and extending our inquiry to understand the impact of mutations on both models and transformations.

Results: The experimental findings reveal relationships between the energy utilization of ATL transformations and the structural characteristics of metamodels. Furthermore, we establish a connection between energy consumption, model size, and the complexity of transformation processes.

Conclusion: The insights gained from this research lay the groundwork for devising future energy-efficient strategies while developing model transformations.

1 | Introduction

The concept of Model-to-Model (M2M) transformation holds a pivotal role within the context of Model-Driven Engineering (MDE). The widespread adoption of MDE techniques in various industries is well-documented [1–4]. As MDE continues to gain popularity, the accessibility of big data becomes increasingly prevalent, making it an unavoidable aspect of consideration [5]. In certain domains, the utilization of very large models (VLMs) [6], comprised of millions of elements, introduces unique challenges related to performance. Empirical estimates from industrial contexts highlight the potential threat and limitation posed by the limited support for handling large models, thereby hindering the widespread adoption of MDE in industrial processes [1]. Furthermore, the presence of large models can adversely impact runtime execution [7], leading to unacceptable delays in the case of model transformations [8]. To address these challenges, significant efforts have been invested in recent years. Numerous studies, notably those focusing on scalability [9, 10], have contributed to the development of solutions. Gremlin [11], for instance, is a framework specifically designed to tackle scalability issues. Additionally, a recent study [12] highlights the performance and scalability issues associated with model transformations, prompting researchers to explore directions for enhancing efficiency through static analysis [13]. Unfortunately, the factor of energy consumption has largely been overlooked in existing considerations. Recent studies project that by 2030, data centers alone will account for 10% of the planet's electricity consumption' [14]. The significance of energy-related concerns is increasingly recognized by both industry and academia [15].

In this paper, our objective is to contribute to filling the gap in the existing literature by thoroughly investigating the energy consumption associated with model-to-model transformations by focusing on ATL transformations. Our paper delves into an in-depth analysis of the energy consumption during the execution of model transformations. This analysis encompasses both energy consumption and artifact metrics. The energy consumption is quantified in Joules, while the artifact metrics include a well-established set tailored for ATL transformations and metamodels. More specifically, we aim to establish correlations between the energy consumption of a model transformation and the structural metrics of the transformation itself, as well as the structural metrics of its input and output metamodels. In particular, studying several aspects such as *input model navigation and searches*, *calls of rules and helpers*, *target object resolution*, and *target object creation and initialisation*, we aim to identify patterns and practices that contribute to energy consumption, thereby providing insights into understanding the energy demands of model transformations and their contributing factors. To achieve this, we selected 52 transformations from the Eclipse ATL Zoo[1] and measured the power consumption of their executions by applying them to models of different sizes. Our investigation seeks to elucidate the interplay between these measurements, the structural characteristics of meta-models, and the ATL transformations themselves.

We conducted our experiments in a controlled environment to establish a first analytical step toward a generalizable construct. The industrial transformations available are scarce, and the ones usable in the ATL Zoo are generally old and relatively simple. Therefore, it is difficult to predict to which extent the analysis could be mapped in a real-case scenario. Nonetheless, we employed several metrics and strategies to mitigate the biases related to the inner characteristics of the ATL Zoo. A crucial aspect under consideration involves examining the impact of input model complexity. To address this, we curated a test set comprising six ATL transformations, introducing mutations to the models associated with these transformations. The resulting sets manifest as three distinct groupings, each consisting of 100 mutants. To this end, we employed a model mutation tool named Wodel [16]. An expected result is the observable increase in execution time accompanying an augmented dataset size. However, it is noteworthy that despite the amplified dataset dimensions, the energy consumption remains modest for certain transformations. In addition, we have examined the change of Object Constraint Language (OCL) helpers that are integral to ATL transformations. Different studies discussed the importance of efficient OCL expressions [13, 17]. Complex transformations generally retain several OCL statements and helpers. Some of these expressions are conceived to navigate the collections and perform operations throughout the whole input model. Therefore, inefficient OCL statements can lead to a degradation of the transformation, mainly using collection operations. Changes of OCL helpers can have an impact on the transformation consumption and execution time. This may seem obvious, but it suggests an important idea: customized simplification of OCL helpers can result in a substantial improvement in power consumption metrics. Lastly, we have created a comprehensive online repository that contains the replication package of our experiments.[2]

The paper is structured as follows: Section 2 motivates the work and makes an overview of existing approaches focusing on the problem of measuring the energy consumption of software systems. Section 3 presents the methodology of this study. The results of the performed experiments are presented in Section 4. In Section 5, we report the threats to the validity of the study and the strategies we adopted to mitigate them. Section 6 concludes the paper and discusses possible future research directions.

2 | Background and Related Work

2.1 | ATL transformations

ATL is a model transformation language that incorporates both declarative and imperative elements for developing model-to-model transformations. It enables the definition of transformations between source and target metamodels, which serve as the abstract syntax definitions for the models being transformed. By specifying these transformations, source models, which conform to the source metamodels, are systematically transformed into target models, adhering to the target metamodels.

Listing 1 presents an excerpt of JavaSource2Table included in the ATL Zoo. This transformation generates a table indicating how many times each method in a given piece of Java code is invoked within any defined method. The transformation is outlined through a module specification, which includes a header section (lines 1 and 2), helper functions (lines 4-7), and transformation rules (lines 9–23). The header defines both the source and target models of the transformation, along with their corresponding meta-models. As a result, the `JavaSource2Table` module performs a one-to-one transformation that produces a target model adhering to a `Table` meta-model from a source `JavaSource` model (see line 2).

```
1  module JavaSource2Table;
2  create OUT : Table from IN : JavaSource;
3
4  helper context JavaSource!MethodDefinition
5  def : computeContent(col : JavaSource!MethodDefinition) :
         String =
6  self.invocations->select(i | i.method.name = col.name
7  and i.method.class.name = col.class.name)->size().
       toString();
8
9  rule Table {
10   from s : JavaSource!ClassDeclaration
11   to t : Table!Table ( rows <- s.methods )
12 }
13 rule MethodDefinition {
14   from m : JavaSource!MethodDefinition
15   to   row : Table!Row (
16   cells <- Sequence{JavaSource!MethodDefinition.
         allInstances()
17   ->collect(md | thisModule.DataCells(md, m))}
18  )
19 }
20 lazy rule DataCells {
21 from md: JavaSource!MethodDefinition, m: JavaSource!
       MethodDefinition
22 to   cell: Table!Cell(content <- m.computeContent(md))
23 }
```

LISTING 1: Fragment of a sample ATL transformation

Helpers and rules are key components of ATL that determine the transformation's behavior. The source pattern of rules (for instance, line 10) comprises types from the source meta-model.

Therefore, a rule is applied to any instance of the specified source types that meet the optional OCL rule guard. Additionally, rules outline a target pattern (e.g., line 11) that specifies the target objects generated by the rule and includes bindings to set their features (attributes and references). For example, the binding `rows ← s.methods` (line 11) sets the `rows` feature of the target type `Table` with elements created by the rules applied to the input elements denoted by `s.methods`.

The rule `MethodDefinition` (lines 13–19) generates a target `Row` for each source `MethodDefinition`. Within this rule, the binding assigns a sequence of elements to the reference `cells` using an OCL expression that selects all source `MethodDefinition` objects and applies the lazy rule `DataCells`. In contrast to *matched* rules (like `Table` and `MethodDefinition`), lazy rules are triggered only upon explicit invocation and utilize the arguments provided.

The `DataCell` rule takes two `MethodDefinition` objects as inputs and produces a target `Cell` containing a value computed using the helper functions defined in lines 4–7. Helpers serve as auxiliary operations that allow the creation of complex model queries with OCL. Specifically, the helper `computeContent` returns a string representing the number of occurrences of the given `MethodDefinition` object.

2.2 | Related Work

Energy consumption is a concern of paramount importance for its financial and environmental implications, and consequently, over the last years, the energy concern has gained the interest of academia and industry. Java is one of the most investigated programming languages from the energy consumption point of view [18]. In [19], the authors investigated the energy eagerness of Java data collections. The authors first created energy profiles, executing micro-benchmarks to collect information about energy behavior. Hence, they studied statically the target system. Combining energy profiles and system specifications allows the recommendation of alternative collection implementations. The evaluation conducted on controlled environments revealed energy savings of about 17%.

Kumar and Shi [20]. studied the energy effects of Java command-line options. The authors proved that the Oracle JDK is more energy efficient than OpenJDK. Furthermore, the Xint command-line option resulted as the worst in terms of energy efficiency with an average increase of 125%. The Xint flag prevents the Hotspot JVM from compiling methods to native code. Analogously, Pereira et al. [21] evaluated 27 popular programming languages across 10 benchmark problems. Their findings revealed key insights into the complex relationships between execution time, memory usage, and energy consumption, emphasizing that faster execution does not always equate to greater energy efficiency. Moreover, they introduced a multi-dimensional analysis by combining energy, time, and memory usage, offering a valuable tool for developers to optimize their choices based on limited resources. In contrast, our work focuses on evaluating energy consumption within a well-identified domain, specifically model transformations using the ATL language. Our study investigates potential factors that impact the energy consumption of executing model transformations, extending the conversation on energy efficiency to domain-specific tasks.

In [22] the authors investigated the energy performance of 27 Java I/O methods and explored the effects on software by substituting these I/O methods. They managed to achieve energy savings of even 30% in some real-case scenarios.

Georgiou, Kechagia and Spinellis [23]. compared the energy consumption of small tasks developed in different programming languages, including Java, C, C++, Go, and Python. The authors conducted an empirical study by analyzing tasks retrieved from the *Rosetta Code* repository[3] which is a repository of generic tasks implemented in different programming languages, and eventually tested 14 languages. Besides particular cases, the study demonstrated that compiled languages are more efficient than interpreted ones. Empirical works about the energy consumption of software systems have also been done in the domain of Android [24], IoS [25], or wireless sensor networks [26]. In the domain of Machine Learning different studies have been conducted with the aim of improving the energy efficiency of ML-based systems [27–29].

By focusing on investigations done in the MDE research field, the performance of model transformations has been studied from different perspectives. Cuadrado et al. [13]. developed a new compiler for ATL model transformation called A2L. The work aims at improving the performance and scalability of developed transformations. To this end, A2L can optimize OCL expressions and transformation rules and exploit a novel algorithm to enable the execution of parallel transformations.

In [30], the authors exploited the Ant World case study to compare the performance of manual and automated transformation optimizations. This case study was conceived as a contest organized by the 4th International Workshop on Graph-Based Tools[4] aimed to simulate an anthill through model transformations. The results reported an increment of 70% of execution time when manual optimizations are neglected.

Amstel et al. [31]. studied the performance of ATL and QVT transformation languages. The authors correlate the size and complexity of input models to the execution time. The analysis is limited to two simple transformations and neglects the aspect of energy consumption.

The work by Piers [32]. provides a working profiler to find performance issues inside a transformation. Developers can employ two indicators: execution time and executed instructions. In [33], the authors provide a refactoring catalog for rule-based ATL transformation and study possible smells in rules and helpers by providing corresponding solutions. The authors also studied the performance effect of such refactoring by analyzing the CPU time.

Lano and Rahimi [34] defined a catalog of transformation design patterns aimed at improving the modularization, efficiency, and data storage requirements of model transformations. Their study provides a systematic classification of patterns across leading model transformation languages such as ATL, QVT, and GrGen.NET, and introduces a metamodel-based framework to define these patterns. An experiment has been conducted to guide pattern selections, accompanied by evaluating their effectiveness across a range of case studies.

Amstel and Brand [35]. discuss the quality of ATL transformations. In particular, the authors conducted a user evaluation to correlate the automatically generated metrics and the quality attributes. The qualitative analysis confirms the importance of the right usage of helpers as a mechanism to increase transformation performances. Tichy et al. explore the effects of bed smells on the performances of transformations defined in Henshin language. In particular, the study correlated the presence of bad smells in rules as a potential threat during transformations. Recently, an ad-hoc profiler for Henshin [36] has been released to provide information about the execution time at the transformation level.

Groner et al. [37] studied how to predict the execution times of ATL model transformations across different sets of model characteristics. They experimented with the prediction performance of various machine learning techniques on a large and diverse set of input models, specifically varying the size and characteristics of these models. Pinto, Soares-Neto and Filho [38]. investigated the state of the art concerning the task of improving energy efficiency using refactoring tools. In particular, the authors identified several domains, reporting a set of studies achieving refactoring and the related challenges.

As mentioned earlier, the performance of model transformations in MDE has been a subject of extensive research. Numerous approaches, including compiler optimization, performance benchmarking, and refactoring catalogs, have been investigated to enhance their scalability and efficiency. While prior studies have focused on metrics such as execution time, the number of executed instructions, and quality attributes, the aspect of energy consumption in model transformations remains largely unexplored.

3 | Experiment design

3.1 | Research Questions

In this study, our focus is to provide an initial investigation of the energy consumption associated with the execution of ATL transformations and assess the contribution of each structural component to this consumption. In this respect, we framed our experiments to answer the following research questions:

- **RQ$_1$**: *To what degree does the energy consumption needed for executing an ATL transformation correlate with the structural attributes of the source and target metamodels, as well as the inherent characteristics of the ATL transformation itself?* We explore the relationship between energy consumption and well-known metrics for metamodels (which can include factors such as the size and complexity) involved in the transformation [39], as well as transformation metrics [40].
- **RQ$_2$**: *To what extent does the size of input models impact the energy consumption of ATL transformations?* We produce different mutations of input models to examine their impact on the energy consumption of ATL transformations.
- **RQ$_3$**: *How do OCL rules impact energy consumption?* We modify OCL rules to evaluate their influence on the energy consumption of the ATL transformations under analysis. Since we cannot foresee the effects of complex modifications to the transformation semantics, we limit ourselves to simple helper changes.

3.2 | Methodology

The methodology utilized to conduct the experiments is illustrated in Figure 1. Specifically, we utilized JoularJX [41] for measuring the energy consumed by ATL transformations. JoularJX is a Java agent that relies on Intel RAPL [42] to read the power usage of the CPU. It can monitor the power consumption of methods at runtime and focus on a specific Java application, filtering out the consumption of the rest of the system. We chose JoularJX due to its specific ability to measure the energy consumption of running Java programs, such as JAR files. This tool uniquely provides method-level analysis, offering performance evaluations at intervals as short as 10 milliseconds. To the best of our knowledge, it is the only tool explicitly designed to measure the energy consumption of JAR files. Additionally, JoularJX is part of a family of monitoring tools that have been widely adopted in various research

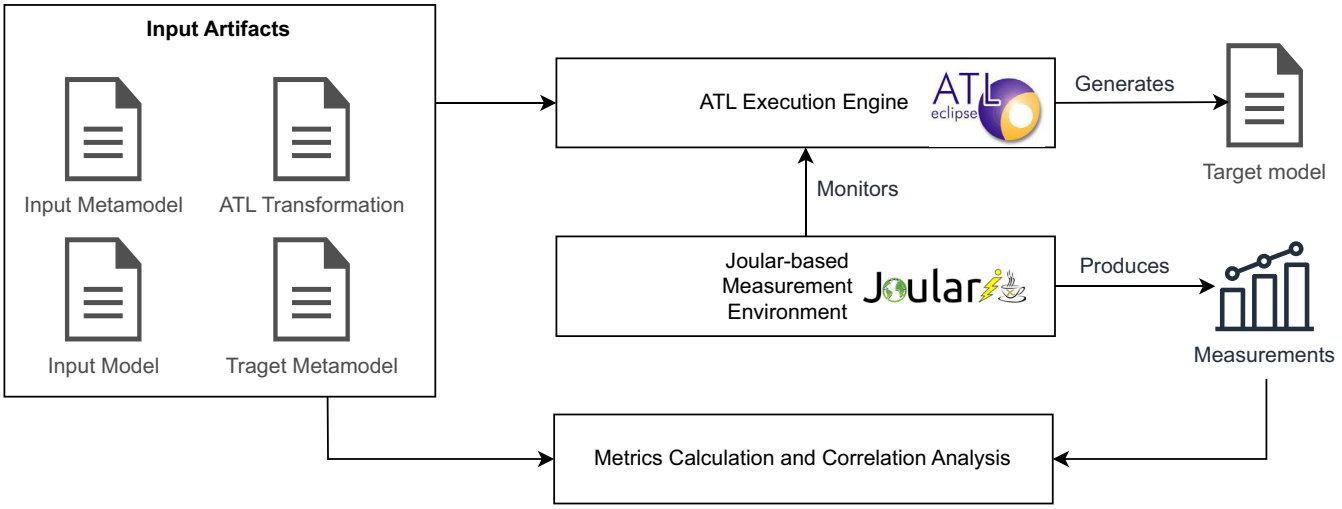

FIGURE 1 | Experimental execution procedure.

```
13/03/2024 04:22:58.199 - [INFO] - +--------------------------------+
13/03/2024 04:22:58.200 - [INFO] - | JoularJX Agent Version 2.0     |
13/03/2024 04:22:58.200 - [INFO] - +--------------------------------+
13/03/2024 04:22:58.224 - [INFO] - Results will be stored in joularjx-result/38983-1710343378214/
13/03/2024 04:22:58.239 - [INFO] - Please wait while initializing JoularJX...
13/03/2024 04:22:59.249 - [INFO] - Initialization finished
13/03/2024 04:22:59.250 - [INFO] - Started monitoring application with ID 38983
../Data/AtlTransformation/XML2SpreadSheetMLSimplified/XML2SpreadsheetMLSimplified.atl
XML2SpreadsheetMLSimplified executed in 0,07s.
Total Execution Time: 1.165
13/03/2024 04:23:01.438 - [INFO] - JoularJX finished monitoring application with ID 38983
13/03/2024 04:23:01.439 - [INFO] - Program consumed 70,55 joules
13/03/2024 04:23:01.468 - [INFO] - Energy consumption of methods and filtered methods written to files
```

FIGURE 2 | JoularJX at work on the *XML2SpreadsheetMLSimplified* transformation.

works (see e.g. [43–45]). These features made it the most suitable option for our study. ATL transformations are executed by an execution engine, packaged in a JAR file to enable stand-alone execution without requiring the entire Eclipse IDE to be running. We employed the ATL version 3.5.0 v2014 to perform all the transformations. Additionally, various metrics are computed on the input artifacts and correlated with the execution time and power consumption measured by JoularJX.

Figure 2 shows an explanatory execution of JoularJX on the input *XML2SpreadsheetMLSimplified* transformation. JoularJX reports the energy consumption, which is 70.55 J in this specific case, and the total execution time is equal to 1.165 s. We can also notice the time taken only by the transformation, which is 0.07 s. A remarkable difference exists between the total execution time and the one taken by the transformation only. The latter is related only to the application of the transformation under analysis. In contrast, the former regards all the mandatory operations to perform the transformation, i.e., model loading, parsing, transformation and saving of the generated model.

3.3 | Metrics

3.3.1 | Metrics for Energy Consumption and Execution Time

In our investigation, we wanted to quantify the energy consumption and execution time of ATL transformations. Thus, the following metrics were utilized:

- **Energy Consumption**: This metric quantifies the energy consumed during the execution of a given ATL transformation. It is measured in terms of power usage, providing a comprehensive understanding of the energy requirements of the transformation process. We calculate the energy consumption of the transformation using JoularJX. The consumption is expressed in *Joule* and is the energy necessary to read the models and metamodels plus the time to execute the transformation. Each energy consumption value that we report is calculated on average. As an example, according to the *#Tests* value for the RQ_1 shown in Table 5, the energy consumption of a given ATL transformation t is the average of 50 measurements as follows:

$$\overline{E}(t) = \frac{1}{50}\sum_{i=1}^{50} E(t_i)$$

  where E(t) represents the value calculated by JoularJX for the transformation t. Concerning RQ_2 and RQ_3, we tested each mutant twice. Therefore, the average is calculated as the following formula:

$$\overline{E}(t) = \frac{1}{200}\sum_{i=1}^{100}\sum_{k=1}^{2} E(t_{\text{mutant}_i})$$

  where we test all the 100 mutants_i of t twice, and finally we get the average value.

- **Execution Time**: The execution time metric measures the duration taken by a given ATL transformation to complete its execution. Thus, it is defined as the sum of the time to parse the input metamodel and model, execute the transformation, and save the output model.

3.3.2 | Metrics Related to Structural Characteristics of the Modeling Artifacts

To explore the relationship between energy consumption, model size, metamodel, and transformation structural characteristics, we employed a range of metrics focusing on the structural properties of models, metamodels, and transformations. These metrics facilitate the analysis of correlations among various components involved in the transformation process. Metrics are based on existing studies (see the metrics defined by Van Amstel and Van Den Brand [40] and Williams et al. [46]) and calculated by exploiting a model-based tool chain developed in a previous work [47].

Metrics related to metamodels' structural characteristics were utilized to gain insights into the complexity and intricacies of the metamodels employed in the transformation process. Metrics focusing on the structural properties of transformations were utilized to assess the complexity of transformation rules. These metrics aid in understanding the intricacies of the transformation and its impact on energy consumption and execution time. Table 1 lists the structural metrics we computed among the transformations and metamodels. Building on the structural metrics analyzed in our previous work [47], we introduce two additional metrics to further investigate metamodel complexity. Specifically, H_{MAX} measures the longest inheritance path within the metamodel, whereas G_C quantifies the graph complexity, treating classes as nodes and references as edges [48].

TABLE 1 | Metrics of modeling artifact.

	Metric	Acronym
Transformation	Number of bindings	$\#\,B$
	Number of Bindings Involving Object Resolution	$\#\,BOR$
	Number of Input Patterns	$\#\,IP$
	Number of Output Patterns	$\#\,OP$
	Number of Matched Rule	$\#\,MR$
	Number of Rules With Using	$\#\,RWU$
	Number of Helpers	$\#\,H$
	Number of Attribute Helpers	$\#\,AH$
	Number of Attribute Helpers With Context	$\#\,AHWC$
	Number of Operational Helpers	$\#\,OH$
	Number of Lines Of Code	LOC
Metamodel	Number of Meta Class	$\#\,MC$
	Number of Abstract MetaClass	$\#\,AMC$
	Number of Structural Feature	$\#\,SF$
	Number of Structural Feature with inherited	$\#\,SF_{ALL}$
	Number of Attribute	$\#\,A$
	Number of Attribute with inherited	$\#\,A_{ALL}$
	Number of Reference	$\#\,R$
	Number of Reference with inherited	$\#\,R_{ALL}$
	Max Hierarchy Length	H_{MAX}
	Max Graph Complexity	G_C

3.3.3 | Metrics Correlation Analysis

In addition to individual metrics, correlation analysis was performed to identify relationships among transformation rules, model size, metamodel structural characteristics, and energy consumption. Correlation is a commonly employed statistical technique for identifying associations and evaluating connections among observable data. Specifically, by examining correlations, we answered RQ_1 to determine the specific structural properties of ATL transformation, as well as the input and output metamodels, that impact energy consumption.

3.4 | Dataset

We employed the ATL Zoo as our principal repository of transformations and related models and metamodels. These transformations are predominantly linked with various research works, embodying a broad spectrum of complexities. They range from straightforward to complex transformations, reflecting the diverse challenges and structures encountered in the practical applications of ATL. The developers of these transformations form at least two groups of users, including MDE experts, such as the creator of ATL, and beginners, comprising students and MDE newcomers. Table 2 reports detailed information about the transformations we analyzed in our study. In particular, the columns report common metrics employed in different studies [49, 50] to elicit meta-data about the transformations. Table 1 introduces the acronyms of the measured metrics, and some of those are shown in Table 2 to characterize the analyzed transformations. We analyzed the transformations also to identify the use of deprecated features as outlined in the official ATL documentation.[5] Transformations containing at least one deprecated statement are indicated in **bold** in Table 2. We chose to retain these transformations to avoid further reducing the dataset. Additionally, the energy consumption of transformations with deprecated features aligns with the dataset's average, and none of these transformations are among the identified outliers, highlighted in *italic* in Table 2.

3.4.1 | Model Transformations

The ATL Zoo contains a collection of 103 model transformations. It has established itself as a benchmarking dataset in model transformations, having been employed in numerous empirical studies over the past years (see e.g., [17, 37, 51, 52]).

To answer RQ_1 we selected 52 transformations together with their input models and metamodels based on the following *inclusion* criteria:

- *Transformations processing single input metamodel*: We adopt this rationale because RQ_1 aims to evaluate the relationship between energy consumption and structural attributes of source and target metamodels. Considering multi input transformations complicates conducting a homogeneous correlation analysis because these transformations do not maintain a direct and singular relationship between input and output elements. This lack of one-to-one correspondence introduces variability that can obscure or distort the relationships under investigation, making it challenging to measure and interpret correlations accurately.
- *Transformations that do not incorporate external modules in their process*: Because I/O operations can negatively impact energy consumption [53], we decided to filter out transformations that require external modules.
- *Transformations that do not involve UML as either input or output metamodel*: Because of the different versions of UML, we could not identify the corresponding metamodel versions involved in the transformations as input or output metamodel. Moreover, the projects with ATL transformations working on UML do not include the corresponding metamodel nor the unique metamodel identifier, i.e., *nsURI*. For this reason, we could not run the transformations involving UML.
- *Transformations that do not raise exceptions during their execution*. Unfortunately, some transformations fail when given a different model from the primary use case as input. Since our assessment of energy consumption is empirical, we excluded transformations that raised exceptions during their execution.

Figure 3 shows two examples of transformation headers, i.e., `UML2AnyLogic` and `Families2Persons`. The former has been excluded from our investigation because it is not a one-to-one transformation; it uses an external module, i.e.,

TABLE 2 | Structural metrics of the considered transformations (in **bold**, transformations with deprecated features; in *italic*, outlier transformations).

Trafo Name	# B	# BOR	# IP	# OP	# MR	# RWU	# H	# AH	# AHWC	# OH	LOC
KM32CONFATL.atl	591	221	5	287	5	2	1	1	1	0	1057
KM32Metrics.atl	27	0	2	9	2	0	15	8	3	7	304
KM32OWL.atl	74	34	10	49	10	0	4	1	1	3	307
KM32Problem.atl	54	0	18	18	18	0	6	2	2	4	471
SpreadsheetMLSimplified2XML.atl	46	3	12	20	12	1	1	0	0	1	248
XML2SpreadsheetMLSimplified.atl	42	6	11	17	11	1	10	0	0	10	380
Replace.atl	46	8	9	11	9	0	0	0	0	0	181
WithContext/ForeignKey.atl	39	8	8	9	8	0	0	0	0	0	150
WithoutContext/ForeignKey.atl	25	6	6	6	6	0	0	0	0	0	100
WithContext/Removing.atl	87	27	10	15	9	0	2	2	1	0	259
WithoutContext/Removing.atl	69	21	6	11	5	0	2	2	1	0	201
WithContext/partial2totalRole.atl	59	16	10	12	10	0	0	0	0	0	182
WithoutContext/partial2totalRole.atl	34	10	5	7	5	0	0	0	0	0	109
WithContext/PartialRolesTotalB.atl	78	32	13	13	9	0	1	1	0	0	248
WithoutContext/PartialRolesTotalB.atl	60	26	9	9	5	0	1	1	0	0	192
WithContextPrimaryKey.atl	41	11	8	9	8	0	0	0	0	0	150
WithoutContext/PrimaryKey.atl	17	4	4	5	4	0	0	0	0	0	72
Families2Persons.atl	2	0	2	2	2	0	2	1	0	1	49
SpreadsheetMLSimplified2XML.atl	46	3	12	20	12	1	1	0	0	1	248
Table2SpreadsheetMLSimplified.atl	12	6	3	9	3	2	1	0	0	1	117
XML2Ant.atl	81	0	29	29	29	0	7	0	0	7	459
BibTeX2DocBook.atl	25	2	9	20	9	0	4	3	3	1	261
XML2Book.atl	5	0	2	2	2	0	1	0	0	1	30
Book2Publication.atl	3	0	1	1	1	0	3	0	0	3	50
Class2Relational.atl	22	2	6	11	6	0	1	1	1	0	113
Grafcet2PetriNet.atl	24	14	5	5	5	0	0	0	0	0	89
PetriNet2PNML.atl	29	13	4	15	4	0	0	0	0	0	110
PetriNet2Grafcet.atl	26	14	5	5	5	0	0	0	0	0	92
IEEE1471_2_MoDAF.atl	54	18	14	14	13	0	1	1	1	0	229
JavaSource2Table.atl	7	0	2	7	2	0	2	1	1	1	115
A2B.atl	3	0	2	3	2	0	0	0	0	0	30
Make2Ant.atl	16	4	5	6	5	0	0	0	0	0	88
TT2BDD.atl	11	3	6	6	6	1	5	0	0	5	257
MySQL2KM3.atl	117	12	11	15	11	0	17	6	4	11	611
PathExp2PetriNet.atl	16	8	3	5	3	0	1	1	1	0	105
PetriNet2PathExp.atl	8	2	3	3	3	0	0	0	0	0	71
PathExp2TextualPathExp.atl	11	3	5	7	5	1	10	1	1	9	337
Port/TypeA2TypeB_v1.atl	4	2	1	3	1	0	0	0	0	0	20
SideEffect/TypeA2TypeB_v1.atl	4	3	1	3	1	0	1	0	0	1	29
TypeA2TypeB_v2.atl	3	1	2	3	2	0	1	1	1	0	64
TypeA2TypeB_v3_firstStep.atl	2	0	1	2	1	0	0	0	0	0	29
SimpleClass2SimpleRDBMS.atl	12	2	1	5	1	1	6	6	0	0	233
SoftwareQualityControl2Bugzilla.atl	32	1	2	3	2	0	1	0	0	1	107
SoftwareQualityControl2Mantis.atl	40	9	2	11	2	1	2	0	0	2	131
SimpleSBVR2SimpleUML.atl	24	2	8	10	8	0	2	0	0	2	187
Syntax2SimpleSBVR.atl	28	4	17	25	17	0	3	0	0	3	333
XSLT2XQuery.atl	33	15	7	21	7	0	0	0	0	0	169
KM32ATL_KM22MM.atl	64	21	5	29	5	1	5	3	3	2	239
DSL2KM3.atl	50	9	9	12	9	1	3	0	0	3	227
KM32DSL.atl	82	11	8	12	8	0	18	5	5	13	374
ATOM2RSS.atl	21	3	3	4	3	0	0	0	0	0	71
KM32ATLCopier.atl	40	13	2	17	2	1	4	1	1	3	144

```
1 -- @atlcompiler atl2006
2 module UML2AnyLogic; -- Module Template
3
4
5 create OUT : AnyLogic from IN : UML, IN_DI : DI;
6
7
8 uses strings;
9
```

(a) Header of an example of filtered-out transformation

```
1 -- @path Families=/Families2Persons/Families.ecore
2 -- @path Persons=/Families2Persons/Persons.ecore
3
4 module Families2Persons;
5 create OUT : Persons from IN : Families;
6
```

(b) Header of an example of considered transformation.

FIGURE 3 | Including and excluding criteria at work, (a) Header of an example of filtered-out transformation, (b) Header of an example of considered transformation.

`string`, and involves `UML` as an input metamodel. Contrariwise, the latter is the header of a transformation, which has been instead considered in our investigation because it satisfies all the considered inclusion criteria.

To address RQ_2 and RQ_3, we selected 6 model transformations from the 52 transformations collected from the ATL Zoo based on the previous constraints. In particular, the following additional inclusion criteria have been applied:

- *Transformations with the highest number of rules*: this criterion pertains to the transformations that incorporate a substantial number of transformation rules. Transformation rules are the entry points for the transformation process and are paramount for initiating pattern matching over source models.
- *Transformations that convert a model into a target metamodel with the highest number of metaclasses*: This criteria focuses on transformations where the target metamodels are considerably larger than the source ones. Such transformations are particularly challenging as they often require the creation of substantial numbers of model elements in the target models, thereby necessitating intricate transformation mappings.

3.4.2 | Input Models

To address RQ_1, we executed the chosen 52 transformations on the input models sourced from the ATL Zoo, maintaining their original configurations within the respective projects. We aim to identify potential correlations between the structural characteristics of the involved artifacts and the energy consumption of the executed transformations. Consequently, we opted to utilize the default configuration without altering any elements.

To give a comprehensive answer to RQ_2 and RQ_3, instead, we needed the availability of several input models. In a recent work [37], the authors, to gain enough data, exploited available datasets of real case models. Furthermore, the authors manually crawled GitHub to complete the datasets. The main issue with this approach is that we would be forced to rely on the existing datasets, which support only a few transformations. Moreover, the datasets are dimensionally sparse. Thus, we decided to generate mutants of different sizes. To this end, we employ Wodel,[6] a well-known mutation framework [16, 54]. In particular, Wodel provides users with a domain-specific language for the specification and generation of model mutants. Section 3.5 gives details on the employed mutation process.

In the experimental setup detailed in Table 5, the number of tests conducted varies across the configurations: Configuration C1 involved an extensive evaluation with 50 tests, providing a comprehensive dataset. In contrast, Configurations C2 and C3 each underwent a limited set of only 2 tests, likely aimed at assessing specific aspects or variations under controlled conditions.

3.5 | Model Mutations

To give a comprehensive answer to RQ_2 and RQ_3, we needed the availability of different input models. Thus, we decided to generate mutants of different sizes. Therefore, we defined three mutation sizes (i.e., small, medium, and large) in which we differently increase the number of elements that are added to mutants. To this end we employed Wodel, which provides users with a domain-specific language to specify the wanted mutations and includes a mechanism to identify and avoid the generation of duplicated mutants. Wodel is meta-model independent and introduces a range of essential functionalities, including the *definition of mutation actions* (such as creation, deletion, and reference reversal), *selection of items* using different strategies (like random, specific, or all), and *orchestration of mutation compositions*.

The typical Wodel workflow involves the user initially providing a set of seed models adhering to a meta-model. Subsequently, Wodel is employed to articulate the desired mutation operators and their execution specifics, such as determining the quantity of each mutation type per mutant or establishing the sequence of execution. Each Wodel program requires a declaration of the considered model's meta-model. This enables thorough type-checking of the program to ensure validity, restricting references to only valid meta-model types and properties and validating the resultant mutation.

Figure 4 shows explanatory mutation rules we employed in the performed experiments. According to the given specification, 100 mutants conforming to the input meta-model `XML.ecore` are required (see lines 1–4). The first command that gets executed is `Select One Root` (line 7), to *select* a random (*one*) object of type `Root` from the input model as a starting element for further modification. The keyword `create` (see lines 9, 15, 21) is used to create objects of the class indicated by the type reference. Furthermore, two attributes are added to the created object,

```
generate 100 mutants
in "data/out/"
from "data/model/"
metamodel "/XML2Ant/data/model/XML.ecore"

with commands {
    select_root = select one Root

    create Element with{
        name = random-string(5,20),
        value = random-string(1,2),
        parent = one Root
    }[250..250]

    create Text with{
        name = random-string(5,20),
        value = random-string(1,2),
        parent = one Element
    }[250..250]

    create Attribute with{
        name = random-string(5,20),
        value = random-string(1,2),
        parent = one Element
    }[250..250]
}
```

FIGURE 4 | Example of Wodel mutation rules.

TABLE 3 | Summary of performed model mutations.

ID	Transformation	# create statements	Mutation size-small	Mutation size-medium	Mutation size-large
T1	Families2Persons	8	5	20	100
T2	Ieee2MoDaf	7	5	25	100
T3	MySQL2KM3	3	5–25	20–100	50–250
T4	Spreadsheet2XML	5	5	25	100–250
T5	Table2MSOffice	3	5	20–100	50–250
T6	XML2ANT	3	5	25	250

i.e., *name* and *value*. The reference *parent* is also added by picking one object of type `Root` previously selected. The cardinality of the objects to be created is also specified (see lines 13, 19, and 25). In particular, 250 objects will be created for each specified `create` rule. We varied these cardinalities to generate the datasets as reported in Table 3. For example, the meaning of 50–250 in the case of MySQL2KM3 is that some objects were created 50 times and others 250, according to their complexity (i.e., the number of attributes and references). The decision to employ varying cardinalities is influenced by the complexity of the considered metamodels. Specifically, metamodels with a limited number of classes require the mutator to generate more instances within these classes than those with a broader set of metaclasses. In scenarios involving more metaclasses, the generation of new instances is distributed more widely across them. This strategy ensures that mutations within models conforming to simpler metamodels are of comparable size to those within models adhering to more intricate metamodels. This balanced approach facilitates a uniform analysis and comparison across different model complexities.

Once mutation rules are defined, the `Wodel` execution engine generates mutants. As previously discussed, for our experiments, we generated 100 mutants for each transformation across three distinct size dimensions—small, medium, and large—resulting in a total of 300 mutants per transformation. This approach aligns with the methodology of similar studies, such as the personalized approach detailed in Barriga, Rutle and Heldal [55], and ensures a comprehensive and varied sample size for our analysis. In Table 3 we show the performed mutations. For each transformation, we depict the results of the mutation procedure on the input model, in particular:

- `# create statements`: Represents the number of create statements; as an example in Figure 4, we mutate three classes of the XML2ANT transformation.
- `Mutation size (Small/Medium/Large)`: Indicates the number of required invocations of each *create* mutation operator (as for instance in lines 13, 19, and 25 in Figure 4).

3.6 | Changes of OCL Helpers

As previously discussed, with RQ_3 we want to understand to which extent changing an ATL transformation reflects on the energy consumption. An ATL transformation consists of two main components: transformation rules and helpers. A transformation rule describes how a part of the input model should contribute to generating part of the target model. The helper is a support function that simplifies the structure of a rule. Cuadrado et al. proposed *BeautyOCL* [56], a framework designed to simplify OCL statements. While *BeautyOCL* aims to simplify and enhance the performance of OCL expressions, it does not specifically target the improvement of energy consumption during transformation executions. Thus, we decided to apply manual changes on transformation helpers.

Figure 5 shows an explanatory example of changes operated on ATL transformation helpers. In particular, we commented out lines (50–54) and added line 49. Through this change, every time a rule invokes the *getAttribute* helper, the string *"Artificial Value"* is provided. We applied similar changes only once for each of the six transformations we analyzed. To decide which helper to modify, we employed the tool proposed by Götz and Tichy [57]. In their work, the authors investigated the complexity of ATL transformations. Specifically, we used *Syntactic Complexity*, a method for defining the complexity of a transformation specification based on the intricacy of its expressions and activities. This methodology posits that the complexity of each construct is a composite measure, encompassing a baseline static value for the construct itself, augmented by the cumulative complexities of its constituent elements, including the complexity imparted by each helper function involved. The column *Helper Complexity* in Table 4 shows the results obtained by applying the tool presented in [57] and the Maximum OCL Expressions Length (MEL) [33] values to the six analyzed transformations. For each transformation, the corresponding helpers and their respective complexities are shown. Notably, both metrics identified the same most complex helpers across all transformations. Based on these values, we selected and modified the most complex helpers for each transformation in one set of experiments.

In another set of experiments, we focused on the frequency of helper usage. While executing the transformations on the generated mutants, we identified the most frequently fired helpers, as

```
48 helper context XML!Element def:getAttribute(name : String):String =
49     'Artificial Value';
50 --  if (self.testAttribute(name))
51 --      then self.getAttrVal(name)
52 --      else ''
53 --      endif
54 --;
```

FIGURE 5 | Example of operated ATL helper changes.

TABLE 4 | Analysis of helpers (most fired helpers are in **bold**, and most complex helpers are in *italic*).

Transformation	Helper Name	MEL	Helper Complexity	# Fired Helper Original Model	# Fired Helper Small Mutants	# Fired Helper Medium Mutants	# Fired Helper Large Mutants
T1	*familyName*	8	44	9	28	88	208
T1	**isFemale**	9	16	18	56	176	416
T2	*rationale*	11	24	1	1	1	1
T3	dataBaseElt	4	9	1	1	1	1
T3	**isStringType**	3	4	243	924	2,695	5,517
T3	isIntegerType	9	7	195	434	655	1,234
T3	isDoubleType	9	7	74	230	299	594
T3	isUnsupportedType	15	13	57	223	658	1,168
T3	*km3TypeExistsIn*	20	74	36	86	236	536
T3	isForeignKey	4	9	58	108	258	558
T3	isDefinedIn	6	18	54	54	54	54
T3	isEquivalentTo	11	28	28	30	29	28
T3	enumExistsIn	4	15	5	5	5	5
T3	enumSet	8	20	1	1	1	1
T3	dbTypeSet	9	19	1	1	1	1
T3	km3TypeSet	6	20	1	1	1	1
T3	getTableNameRec	9	25	132	132	132	132
T3	getTableName	1	5	17	17	17	17
T3	getReferredTable	10	28	17	17	17	17
T3	getKM3TypeName	7	14	138	501	1,401	3,201
T4	*getDateTimeStringValue*	18	44	0	0	0	0
T5	*isNumber*	29	62	82	58	218	518
T6	getList	7	14	222	252	522	1,122
T6	*getListAux*	13	57	925	1,050	2,175	4,675
T6	getAttrVal	8	23	2,472	2,697	4,722	9,222
T6	**testAttribute**	9	24	4,624	5,659	19,474	79,174
T6	getAttribute	3	13	4,476	5,491	19,126	78,426
T6	testElement	9	24	1	1	1	1
T6	getText	13	38	1	1	1	1

depicted in Table 4. We then changed the most fired OCL helpers with respect to the different mutant sizes. By including this frequency dimension, we aimed to better investigate the contribution of the helpers.

Through the two sets of experiments, we investigated the effects of helper *complexity* and *frequency* dimensions on the energy consumption of the analyzed ATL transformations.

3.7 | Configurations

To address the three research questions, we applied the execution process shown in Figure 1 by varying the input artifacts to emphasize and explore the aspects relevant to each of the three research questions. To mitigate possible biases concerning the measurements, we repeated each transformation multiple times [27, 29]. In particular, for the RQ_1, we executed each transformation 50 times. For RQ_2 and RQ_3, we tested each mutant twice for a final counting of 200 times. Furthermore, to avoid possible power tail states, we forced a pause of 10 s after each transformation execution [27, 58]. The details of the configurations that have been defined to perform the experiments are shown in Table 5 and described below:

- C_1: To address RQ_1, we examined the 52 original transformations obtained from the ATL Zoo. Each transformation

TABLE 5 | Experiment Configurations.

	C_1	C_2	C_3
Number of Input Models	1 (Original Dataset)	3 × 100 (Mutated Models)	3 × 100 (Mutated Models)
Number of Input Transformation	52 (Original Dataset)	6	6 (Simplified Transformation)
Number of Tests	50	2	2
Employed Metrics (Common)	Energy Consumption, Execution Time		
Employed Metrics (Specific)	Correlation of Artifact Metrics	Transformation Complexity, Helper Complexity	

TABLE 6 | Average values for energy consumption and execution time with serialization (S) and without (−S).

	Average^{S}	Average^{-S}	Standard deviation^{S}	Standard deviation^{-S}	Max^{S}	Max^{-S}	Min^{S}	Min^{-S}
Energy consumption (J)	84.829	85.262	10.186	9.776	133.9	132.55	67.13	66.97
Execution time (s)	0.731	0.724	0.110	0.111	1.673	1.423	0.599	0.594

was executed using the original model provided in the corresponding project.

- C_2: For RQ_2, we selected six ATL transformations, each of which was executed on 100 mutants generated from the original input models across three distinct size dimensions (see Section 3.5).
- C_3: To explore RQ_3, we maintained the same number of transformations and input models as in RQ_2. However, we modified the six transformations by simplifying the *most complex and fired helpers* in the transformation (see Section 3.6) to investigate their potential impact on energy consumption.

We run our experiments on a laptop equipped with an AMD Ryzen7 6850U, which featured a clock speed of 2.7 GHz[7], an SSD M.2 2280 PCIe 4.0×4 NVMe Opal 2.0 and 32 GB of RAM. The operating system used was Linux Manjaro 23.0.4 (kernel version: 5.15.167). The same conditions were maintained throughout the experiments: the laptop was connected to a power source, and the battery was fully charged before starting the study.

Various methodologies and statistical instruments exist for the identification and quantification of correlations. In this study, we opt for Pearson's coefficients [59] that have been examined to quantify the correlations among previously introduced metrics. Pearson's correlation index assumes values in the range of −1.00 (perfect negative correlation) and +1.00 (perfect positive correlation). A correlation of 0 indicates no correlation between two variables.

4 | Experiment Results

4.1 | RQ1

In Table 6, we present summary statistics derived from the execution of the 52 ATL transformations with respect to configuration C1 shown in Table 5. The average energy consumption is approximately 85 J, calculated as the mean consumption across all transformations. The standard deviation indicates minimal variation in energy consumption. Specifically, only five outliers exceeded 95 J, with one reaching 133.9 J. In particular, the outlier transformations are (KM32CONFATL; KM32OWL; XML2SpreadhseetMLSimplified; XML2Ant; MySQL2KM3). The outlier transformations evidence some common peculiarities. In particular, we highlight with *italic* font the transformations in Table 2, and we can see that they exhibit more bindings (*# B* and *# BOR*), more output patterns (*# OP*), and are generally longer (*LOC*). The transformation time, as defined in Section 3, represents the average duration required to complete a transformation. Furthermore, we studied the effect of the serialization of the output models. In particular, we removed the method devoted to the serialization. Table 6 presents a comparison of energy consumption and execution time, highlighting the effects of including (^{S}) or omitting ($^{-S}$) output serialization. The difference in terms of energy and execution time is negligible. The explanation is that the laptop is equipped with an SSD hard drive, and the size of the input models is relatively modest.

Table 7 presents a correlation analysis between energy consumption and the model's structural features. The table specifically shows the correlation between the model and transformation's structural metrics and the energy consumption during transformation execution, with and without output serialization. In this analysis, we applied the Pearson correlation coefficient to paired observations of two variables, e.g., energy consumption and one of the studied metrics. For instance, for n distinct transformations, we have corresponding values $\{e_1, e_2, \ldots, e_n\}$ for *average energy consumption*, and $\{b_1, b_2, \ldots, b_n\}$ for *numbers of bindings*. The correlation metrics, along with their associated p-values, are computed to quantify the linear relationship between each structural feature metric and the average energy consumption.

Correlation coefficients span from −1 to 1, wherein values nearing 1 signify a robust positive correlation, implying that as the feature count increases, so does energy consumption. Conversely,

TABLE 7 | Correlation and p-value for artifact metrics and transformation energy consumption with serialization (S) and without serialization (−S).

	Metric	Correlation^{S}	p-value^{S}	Correlation^{-S}	p-value^{-S}
Transformation	# *B*	0.708	0.000e+00	0.672	0.000e+00
	# *OP*	0.664	0.000e+00	0.626	1.536e−282
	# *BOR*	0.610	1.547e−264	0.572	7.239e−226
	# *IP*	0.550	1.357e−205	0.565	2.805e−219
	# *MR*	0.538	8.298e−195	0.551	1.402e−206
	# *RWC*	0.457	1.431e−134	0.445	1.102e−126
	# *H*	0.422	1.264e−112	0.445	2.024e−126
	# *AH*	0.316	2.543e−61	0.339	9.649e−71
	# *AHWC*	0.329	9.228e−67	0.347	1.555e−74
	# *OH*	0.395	1.312e−97	0.413	1.379e−107
MM_{in}	# *R*	0.610	1.185e−264	0.604	1.174e−258
	# *A*	0.509	3.199e−171	0.509	2.427e−171
	# H_{MAX}	0.496	2.955e−30	0.505	5.050e−01
	# *MC*	0.350	1.086e−75	0.360	1.619e−80
	# SF_{ALL}	0.314	1.037e−60	0.320	7.072e−63
	# A_{ALL}	0.237	2.154e−34	0.245	7.511e−37
	# *SF*	0.224	4.851e−31	0.231	6.057e−33
	# *AMC*	0.221	2.955e−30	0.234	8.828e−34
	# R_{ALL}	0.136	3.085e−12	0.139	9.427e−13
	inMaxG	−0.172	−1.725e−01	−0.172	−1.719e−01
MM_{out}	# *MC*	0.521	7.491e−181	0.513	8.667e−175
	H_{MAX}	0.497	4.923e−162	0.501	1.524e−165
	# *R*	0.479	1.612e−149	0.467	7.202e−141
	# A_{ALL}	0.473	4.372e−145	0.464	4.653e−139
	# *SF*	0.468	1.549e−141	0.460	2.826e−136
	# *AMC*	0.463	3.687e−138	0.452	7.612e−131
	# *A*	0.462	2.204e−137	0.450	1.461e−129
	# SF_{ALL}	0.426	3.950e−115	0.415	9.958e−109
	# R_{ALL}	0.266	1.880e−43	0.263	1.888e−42
	G_C	0.066	7.748e−04	0.064	1.106e−03

coefficients approaching −1 indicate a strong negative correlation, suggesting that higher feature counts correspond to lower energy consumption. Coefficients approximating 0 denote negligible linear correlation. To assess the significance of these correlations, p-values are employed, with a conventional threshold of 0.05 indicating statistical significance. Table 7 presents the correlation coefficients between various structural metrics of the transformation and metamodels with the energy consumption of executing the transformation, both with and without output serialization. The data reveal that correlation values are generally higher when output serialization is included, a trend that is especially pronounced in metrics such as # *B*, # *BOR*, # *OP*, and # *R*. This observation suggests that, in the absence of serialization, these structural metrics have a stronger association with energy consumption during transformation execution.

Although the correlation values with serialization are generally lower, the analysis reveals a consistent trend across both conditions. Specifically, the influence of structural metrics on energy consumption remains significant, with similar patterns appearing in the presence of serialization. For instance, # *B* and # *OP* continue to show high correlations with energy consumption in both cases, with values of 0.672 and 0.626, respectively, when serialization is included. This consistency indicates that, while serialization reduces the magnitude of correlation, it does not alter the underlying relationship between these structural metrics and energy consumption.

These results imply that the number of bindings, along with input and output patterns, significantly influences energy usage in model transformations. In ATL transformations, bindings can either involve direct assignments or object resolution, with the latter presumed to require more computational effort. After computing the number of bindings involving object resolution, we found that, on average, bindings involving object resolution occur with a frequency of 0.24. Moreover, a Pearson correlation analysis

revealed a strong correlation between object resolution bindings and energy consumption, with a correlation coefficient of 0.610 and a p-value of 1.547E−264 without serialization and a slightly reduced correlation of 0.572 and a p-value of 7.239E−226 with serialization. This suggests that object resolution bindings are indeed a significant factor influencing energy consumption, and their impact is evident in both cases, though slightly diminished when output serialization is included. A similar observation applies to the input and output metamodel (MM_{in} and MM_{out}) metrics. Metrics such as $\#\ A$, $\#\ R$, and $\#\ SF$ maintain comparable correlation patterns across both conditions, despite a slight reduction in correlation values with serialization. For example, $\#\ A$ in MM_{in} shows a correlation of 0.509 in both cases, while $\#\ R$ in MM_{out} changes only marginally from 0.479 without serialization to 0.467 with serialization. H_{MAX}, which measures the longest inheritance path, demonstrates moderate correlations with energy consumption. In contrast, G_C, representing the graph complexity of the metamodel, shows a weaker correlation, suggesting that it has a minimal impact on energy consumption compared to other structural metrics. Such metrics exhibit a different trends to other metrics when comparing scenarios with and without serialization, indicating weaker correlations when serialization is considered. However, the magnitude of the correlations remains consistent.

In summary, the higher correlation values observed without serialization coincide with the metrics most strongly related to energy consumption. Even though correlation values with serialization are generally lower, the analysis demonstrates similar trends in both scenarios. These findings suggest that output serialization acts as a smoothing factor, reducing the impact of structural metrics on energy consumption but not altering the overall relationships. This insight is valuable for optimizing energy-efficient transformations, as it highlights the role of serialization in modulating the energy impact of structural characteristics.

The MM_{in} and MM_{out} features, which share identical metrics, highlight structural attributes, whether inherited or not, and also demonstrate a statistically significant correlation with energy consumption. Particularly, the $\#SF_{ALL}$ consistently displays a moderate positive correlation across both MM_{in} and MM_{out} metamodels, emphasizing the role of inherited attributes in shaping the energy footprint of the transformation process.

Answer to RQ$_1$. Our investigation reveals an average energy consumption of 85 J across the 52 ATL transformations of the considered dataset, with minimal variation observed, except for three outliers. Correlation analysis underscores the substantial impact of metrics such as $\#\ B$ and $\#\ OP$ on energy consumption, emphasizing their pivotal role in transformation complexity. Furthermore, it is worth noting that also $\#\ MC$ and $\#SF_{ALL}$ play a relevant role. These insights shed light on the relationship between transformation intricacy and energy usage, suggesting potential research directions for optimization in transformation development. Moreover, output serialization slightly augments the impact of structural metrics on energy consumption, acting as a smoothing factor without altering the overall relationship. This suggests that the effect of serialization on energy usage is marginal.

4.2 | RQ2

In this research question, we investigate the impact of input model size by employing configuration C2 as shown in Table 5. As detailed in Section 3, we classify mutants into three groups based on their size, i.e., small, medium, and large. Similar to the analysis done in the RQ1, we study the effect of the serialization of the output models. In particular, we removed the method devoted to the serialization.

Figure 6 shows the average energy consumption of original models with that of corresponding mutants. The x-axis represents transformations based on the mutant sets, while the y-axis denotes energy consumption. The Figure 6a depicts the consumption of the transformation considering the serialization, whereas the Figure 6b shows the consumption without the serialization.

The initial finding indicates that larger models incur higher energy consumption. For instance, transformations T3 and T6 exhibit energy increases of 54% and 43%, respectively, when applied to larger models. An interesting observation pertains to the energy variations in other transformations. Notably, the energy increment is marginal. For instance, transformations T4 and T5 demonstrate a linear progression, with energy consumption rising from 91.1 J to 92.8 J and from 82.0 J to 90.9 J, respectively. Similarly, transformations T1 and T2 follow a comparable trend, albeit with more substantial gains observed with larger datasets. For T1 there is an increase of 14% in energy, whereas 10% for T2 when applied to the larger dataset. Concerning the difference between employing the serialization, we can notice almost no improvements. The improvement is less than 1% in the case of T3, which required the writing of the largest output model (about 112 kb).

In Figure 7, the impact of mutations on *total execution time* is illustrated. Furthermore, we report the measurements concerning the serialization. From a temporal perspective, the findings related to the serialized transformations are more foreseeable, as visible in Figure 7a. It is observed that larger models necessitate more execution time. Specifically, transformations T3 and T6 incur the most substantial time overhead. T3, for instance, requires approximately 102% more time, while T6 consumes over 91.6% additional time. In absolute terms, T3 takes around one second with the original model, whereas with the larger model, it extends to 1.7 s. Similarly, for T6, the time escalates from 0.992 s to 1.901 s. Furthermore, a consistent trend is discernible for transformation T2, with a gain of approximately 18%, equivalent to 0.132 s. An important observation worth highlighting is that while increasing model size leads to longer execution times, the corresponding rise in energy consumption is negligible for certain transformations. Taking transformation T2 as an example, we observe that with larger models, the execution time increases by 18%, whereas the energy consumption rises by only 9.9%. One possible explanation for this phenomenon is that the transformation engine is sufficiently optimized to handle such larger models efficiently. This fact is relevant since we can derive some clues to generate models that, although vast, do not require excessive energy. In Figure 7b, we illustrate the time required by the transformations not considering the serialization. The results

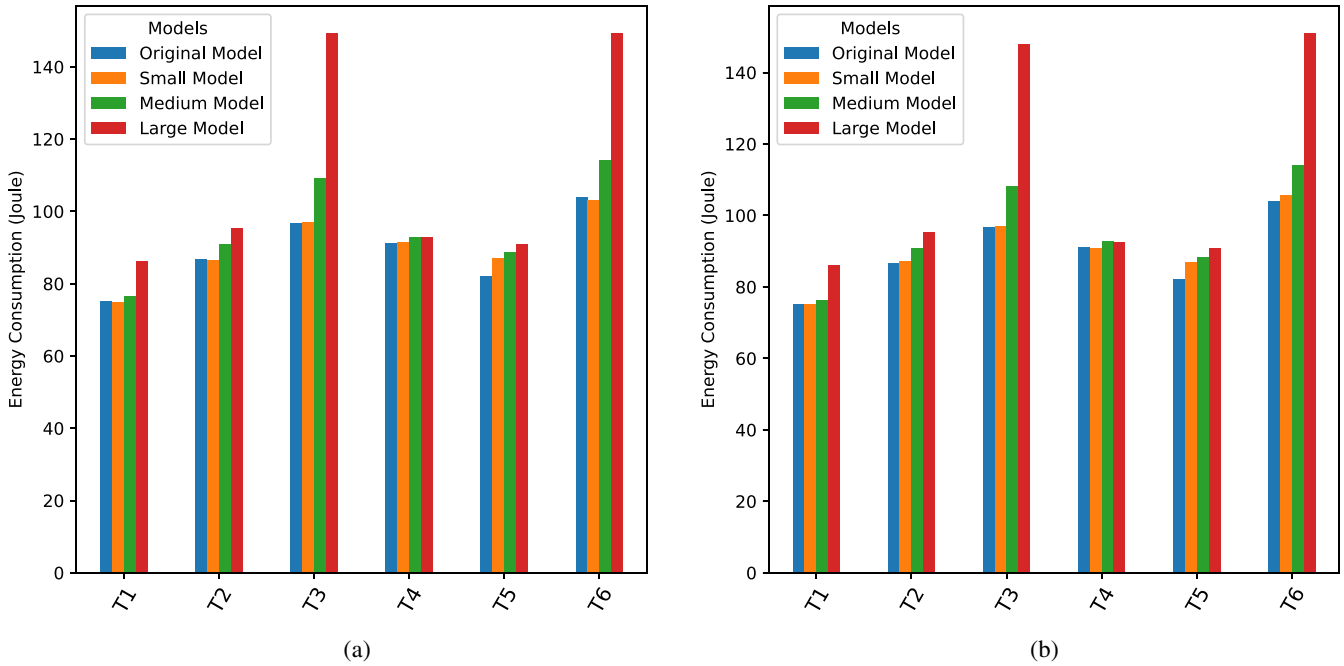

FIGURE 6 | Analysis of the energy consumption for the six transformations with serialization (a) and without serialization (b). (a) Energy consumption with serialization. (b) Energy consumption without serialization.

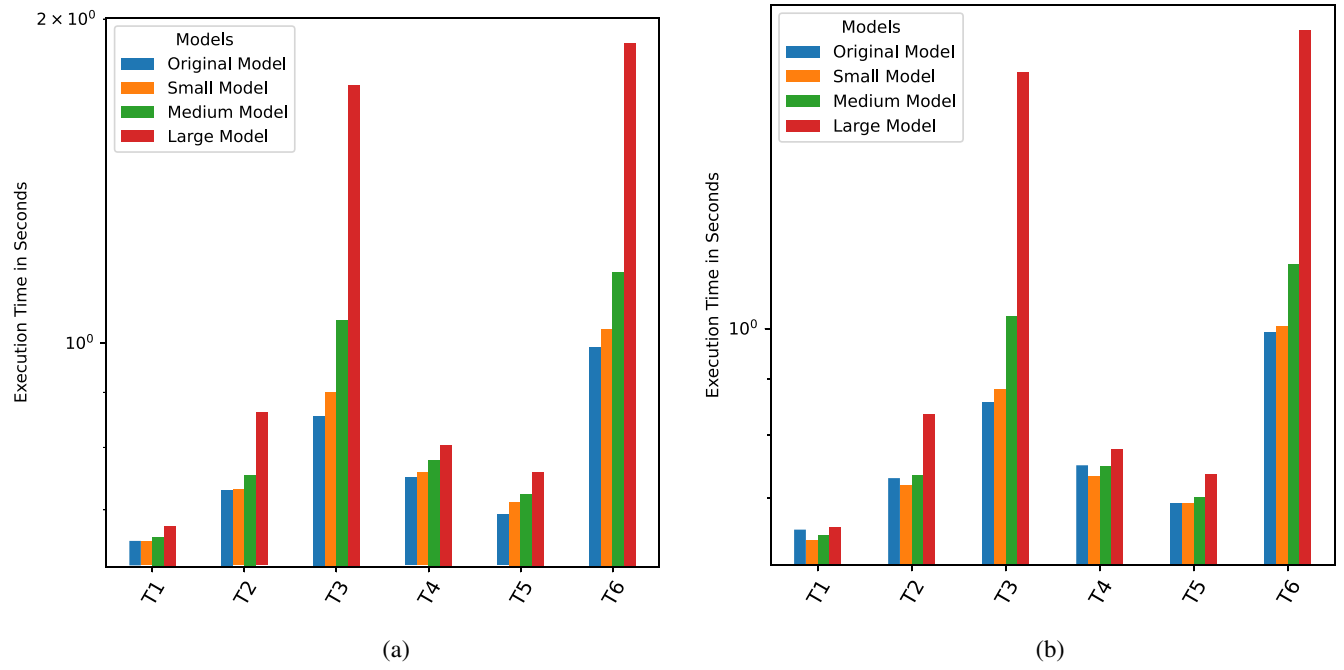

FIGURE 7 | Analysis of the energy consumption for the six transformations with serialization (a) and without serialization (b). (a) Total execution time comparison on mutants with serialization. (b) Total execution time comparison without serialization.

demonstrate a general yet modest improvement of the performances. In particular, the transformations T1 and T2 reduced the overhead for the large models from 3.2% to 0.6% and 18% to 14.3%, respectively.

A more in-depth study using K-means cluster [60] analysis has been undertaken to delve deeper into the factors influencing the energy and time consumption for two particular transformations, T3 and T6, notably impacted by the size of the input models. This technique employs Euclidean distance and specifies the number of clusters (K) as 2. We leverage ATL metrics, detailed in Table 8, which have been computed for the selected transformations as vector features to streamline the clustering analysis. The clustering analysis reveals that T3 and T6 are classified into one cluster

TABLE 8 | Metrics of the analyzed ATL transformations.

	T1	T2	T3	T4	T5	T6
# B	2	54	117	46	12	81
# IP	2	14	11	12	3	29
# OP	2	14	15	20	9	29
# MR	2	13	11	12	3	29
# AH	1	1	6	0	0	0
# AHWC	0	1	4	0	0	0
# OH	1	0	11	1	1	7

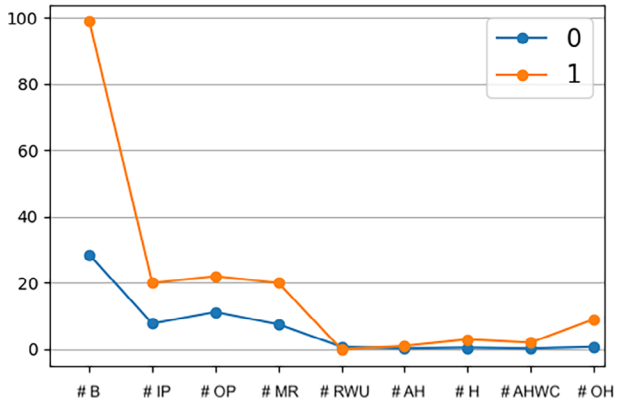

FIGURE 8 | The cluster analysis.

(labeled as *0*), while the remaining transformations are grouped into another cluster labeled as *1*.

Figure 8 shows the cluster analysis to visualize the characteristics of cluster centers within a multidimensional dataset. Each cluster is represented by a line, with points along the line corresponding to the average value of the cluster's members on that feature. The plot aids in discerning how clusters vary from each other across the features. Features where the lines diverge significantly indicate important features for distinguishing between clusters. This type of visualization is particularly useful for identifying the defining characteristics of each cluster, helping to interpret the clustering results in a multidimensional space, and hypothesizing about the nature of the clustered data. In particular, the plot depicts the first cluster (line *0*) exhibiting significantly higher values for the *# B* compared to the second cluster (line *1*). This suggests that *# B* is the most influential metric for clustering transformation using their metrics. It is worth noting that in RQ_1 we also identified that *# B* metrics are one of the most related metrics to energy consumption.

Both clusters appear to exhibit similar trends for other metrics, albeit at varying levels. This suggests that although the absolute values differ, the proportions of metrics such as matched rules and helpers within context, relative to each other, remain relatively consistent between the clusters.

Answer to RQ_2. The experiments revealed that, in general, an augmentation in model size corresponds to an increase in both energy usage and execution time. This trend is particularly pronounced in the cases of *MySQL2KM3* (T3) and *XML2ANT* (T6). The former witnessed a consumption surge from 96.8 J to 149.4 J, accompanied by a 102% rise in execution time. Similarly, the latter experienced a consumption hike of 43.7%, with the execution time extending from 0.99 s to 1.90 s. The comparison of serialization vs. no serialization showed minimal differences. This result is interesting since, with the experimental configuration, the improvement in removing the serialization was negligible. Therefore, this result suggests concentrating the possible optimization efforts during the definition of the transformation rules. The cluster analysis underscores the significant role of the number of bindings in determining cluster membership, which in turn may be linked to differences in energy consumption and execution time between the two clusters.

4.3 | RQ3

In this section, we illustrate the effects of OCL changes operated to ATL transformations. We aim at investigating the effects of changing the most complex OCL helpers, as proposed by Götz and Tichy [57], and the most frequently activated ones by employing configurations C3 and C4 shown in Table 5.

Table 9 compares the energy consumption of the original transformation with the changed ones. For the majority of transformations, the impact of the changes is minimal. Although T1 and T5 show slight improvements, the effects are not significant. However, it is noteworthy that T3 exhibits a worsening performance. Overall, the modifications resulted in an increase in consumption of approximately 4%. In Table 10, we present the effects of the modifications on the execution time. Similar to the case of energy consumption, the variations are minimal for most transformations, with T3 exhibiting poor performance once again. Based on these findings, it can be concluded that there is not a strong correlation between energy consumption, execution time, and the complexity of the helpers for the 6 ATL transformations that have been selected and analyzed. To further investigate these insights, we conducted a more detailed examination of transformation T3.

TABLE 9 | Comparison of energy consumption values (expressed in Joules) for the original transformations and the modified ones.

	Original ATL			Modified most complex helper			Modified most fired helper		
Transformation	Small model	Medium model	Large model	Small model	Medium model	Large model	Small model	Medium model	Large model
T1	74.94	76.51	86.13	73.77 (−1.59)	75.94 (−0.74)	85.18 (−1.11)	74.46 (−0.65)	77.22 (0.91)	85.25 (−1.02)
T2	86.51	91.02	95.24	88.15 (1.86)	91.93 (0.99)	97.17 (1.97)		Only one helper	
T3	97.04	109.29	149.40	96.77 (−0.27)	113.74 (3.91)	155.53 (3.94)	94.72 (−2.45)	113.79 (3.91)	153.74 (3.94)
T3*				97.16 (0.12)	109.58 (0.27)	147.41 (−1.34)	97.61 (−0.5)	110.33 (0.93)	148.98 (−0.27)
T4	91.52	92.89	92.84	92.44 (0.99)	91.08 (−1.98)	93.58 (0.79)		Only one helper	
T5	87.06	88.70	90.91	82.82 (−5.12)	89.416 (0.80)	92.65 (1.87)		Only one helper	
T6	103.21	114.19	149.38	109.72 (5.93)	114.79 (0.52)	152.44 (2.00)	98.97 (−4.27)	102.32 (−11.6)	112.79 (−32.44)

TABLE 10 | Comparison of total execution time (expressed in seconds) for the original transformations and the modified ones.

	Original ATL			Modified most complex helper			Modified most fired helper		
Transformation	Small model	Medium model	Large model	Small model	Medium model	Large model	Small model	Medium model	Large model
T1	0.654	0.66	0.676	0.664 (1.8)	0.666 (3.08)	0.681 (0.73)	0.658 (0.6)	0.667 (1.04)	0.679 (0.44)
T2	0.731	0.754	0.862	0.746 (2)	0.764 (1.3)	0.883 (2.38)		Only one helper	
T3	0.901	1.051	1.737	0.913 (1.31)	1.096 (4.1)	1.817 (4.4)	0.913 (1.3)	1.101 (4.5)	1.794 (3.18)
T3*				0.898 (−0.3)	1.04 −(1.05)	1.694 (−2.5)	0.906 (0.55)	1.055 (0.38)	1.719 (−1.04)
T4	0.758	0.779	0.804	0.756 (−0.26)	0.772 (−0.9)	0.801 (−0.37)		Only one helper	
T5	0.712	0.724	0.759	0.699 (−1.86)	0.724 (0)	0.763 (0.52)		Only one helper	
T6	1.03	1.164	1.901	1.032 (0.19)	1.157 (−0.6)	1.902 (0.05)	0.937 (−9.9)	0.97 (−20)	1.094 (−73.7)

Specifically, we focused on the modified helper, which was initially a boolean. In our initial round of experiments, we simplified the helper by replacing it with the value **False**. Subsequently, we conducted another round of experiments, modifying the helper with the value **True**, resulting in a transformation identified as T3* in Table 9 and Table 10. Notice that now in the tables some values are missing since we reported only the new estimation of T3*. As we can see, changing the value from true to false altered the outcome. Notably, we observed a slight decline in performance only with the Large models. This demonstrates that the semantics of simplification can significantly affect energy consumption and execution time. Such changes can potentially have a substantial impact on the overall transformation process, as a boolean guard directs the process to alternative branches that may vary in energy consumption.

We decided to conduct additional investigations by examining the number of activations for each helper. For each transformation, we calculated the activations of all helpers. As previously discussed, Table 4 presents the results, showing that the most complex helper is not necessarily the most frequently invoked one. For instance, in transformation T1, the most complex helper is *familyName*, while the most frequently triggered is *isFemale*. In this case, one helper is called exactly twice as often as the other. In other transformations, the difference is more significant. In T6, *testAttribute* is invoked 79,174 times, approximately 17 times more than the most complex helper, *getListAux*. We then selected the most frequently fired helpers and changed them. Subsequently, we tested the transformations to measure the new energy consumption and execution time values. The results are shown in Table 9. Notably, the operated OCL change produced improvements in both energy consumption and execution times across all cases. For example, in T6, energy consumption decreased from 149.38 J to 112.79 J for large mutations. A similar trend was observed for execution times as shown in Table 10.

Answer to RQ$_3$. The experimental findings suggest that for the analyzed ATL transformations, there is not a strict correlation between energy consumption, execution time, and helper complexity. However, a quantitative analysis of the helpers reveals a stronger correlation. Specifically, it is observed that some helpers that are heavily activated lead to a significant increase in both energy consumption and execution time. For instance, the simplification of the helper *testAttribute* in the XML2ANT transformation resulted in a saving of 39 J, corresponding to a 32% reduction. In conclusion, complexity and activations should be considered together to assess the effects of rule changes.

5 | Threats to Validity

This section discusses the threats that may hamper the validity of the presented study results, distinguishing between internal and external threats to validity. Internal Validity pertains to any adversary factor that may have an influence on our results. In this work it is mainly related to the precision and reliability of the energy measurements. External validity refers to the generalizability of the obtained results and findings. In this work it is bound to the considered dataset.

Internal Validity: One significant threat is measurement reliability, which is affected by various factors such as interference

from noise, voltage spikes, and background processes. These factors can significantly impact the precision of our energy measurements. To mitigate this threat, we implemented several strategies. We utilized JoularJX 2.0, a tool specifically designed for measuring process energy consumption, to ensure accurate readings. Additionally, we adopted iterative testing methodologies: for RQ1, we conducted 50 tests of each ATL transformation, while for RQ2 and RQ3, we performed 200 tests, recording their average values. This approach helps to smooth out anomalies and provide more reliable data. We also implemented a 10-s sleep command after each test to reduce the impact of potential energy spikes. Another potential threat is the variability in the experimental environment. Conducting all experiments within a Linux environment, we selectively halted non-essential services and daemons to minimize operating system overhead. This setup aims to create a consistent and controlled environment, reducing the influence of extraneous factors on our results. Prior works, such as those by Ournani et al. [18, 22], have validated this approach, demonstrating its effectiveness in similar studies.

External Validity: Regarding the generalizability of the experiments, we acknowledge that the employed rules may not comprehensively mirror real-world models. ATL Zoo is a widely exploited dataset [61, 62] containing old and relatively simple transformations. Therefore, we could not test industrial-case transformations. Nonetheless, our approach involved a gradual increase in mutant size to encompass a wide array of scenarios with the attempt to mimic real-size industrial models, capping the maximum size at 250 elements due to exponential increments in mutant generation time. Additionally, the simplification of OCL helpers might not assure semantic transformation correctness, constituting a potential limitation. Nevertheless, our focus was directed towards investigating the impact of this simplification. The limited number of simplified transformations could introduce a bias. To address this, we selected heterogeneous transformations, as reported in Section 3.4, ensuring a decent level of representativeness of ATL applications. The results are strictly bound to the hardware configuration. Replicating the experiments on different hardware may lead to significant differences. We mitigated this threat by focusing on the correlation and variance of energy consumption and execution time in relation to model size and OCL rule change.

6 | Conclusion and Future Work

In this paper, we focused on evaluating the energy consumption of ATL transformations to gain insights into potential correlations between consumption and various (meta)model and transformation characteristics. Our experiments revealed a correlation between energy consumption and ATL structural features. Additionally, we investigated the effects of model mutations, discovering a strong correlation between transformations and energy consumption in some instances. Furthermore, we explored the impact of simplifying OCL rules, finding that balancing rule complexity and activation frequency is crucial for achieving power reduction.

Our study established correlations between several structural elements of transformations and their impact on energy consumption. Specifically, elements such as the number of bindings, output patterns, number of metaclasses, and structural features inherited appear to play a significant role. Therefore, refactoring efforts could focus on reducing the complexity or frequency of these structural elements to optimize energy efficiency. For instance, rewriting transformation rules to minimize the number of bindings or consolidating bindings where possible could mitigate their energy impact. Regarding helper functions, while their overall correlation with energy consumption was marginal across most scenarios, we observed that in some specific cases (e.g., XML2ANT), helpers contributed significantly to energy use. This suggests that certain optimizations—such as caching frequently used helper function results to prevent repetitive calculations—could be effective in specific contexts. Future work could explore these caching strategies in more detail to determine whether they yield meaningful reductions in energy consumption across a broader set of transformations. As for the serialization process, our experiments indicated that its effects on energy consumption were negligible, particularly when performed on solid-state drives. Therefore, model serialization might not need to be a primary target during optimization refactoring efforts, allowing focus to shift to more impactful elements like bindings and helpers. We recognize that the present study is primarily analytical, and we do not yet provide fully fledged solutions for energy optimization. However, we believe that these insights could serve as a starting point for identifying energy-efficient refactoring strategies in future research.

For future research, there are numerous opportunities to enhance the contributions of our analytical study. We recognize certain limitations that could pave the way for new investigations into energy consumption during model-to-model transformation. In particular, we aim to conduct a comparative study to evaluate the differences between ATL transformations and other transformation languages, such as Henshin and YAMTL, or Java [50]. Additionally, we want to evaluate the possible differences between the tested version of the ATL engine (3.5.0) and the more up-to-date version (4.8.0). This could provide valuable insights into the energy efficiency of different transformation languages. Additionally, RQ2 and RQ3 focused on specific transformations selected based on certain criteria, as reported in Section 3.4.1. We aim to expand these criteria to include a broader range of transformations, guided in part by OCL complexity, and provide a more comprehensive analysis. Furthermore, we plan to investigate how the use of typical ATL caching constructs, e.g., lazy unique rules, attribute helpers, and extracting global helpers [33] impacts the energy consumption of a model transformation and further validate their potential to improve its efficiency. Most importantly, to assess our study's generalizability, we seek to validate our findings by examining real-world and industrial transformations, which could significantly enhance the applicability of our research. Finally, we aspire to utilize this knowledge to develop a recommender system capable of providing energy-aware suggestions during the development of ATL transformations.

Author Contributions

The author takes full responsibility for this article.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data that support the findings of this study are openly available in SPE_2024 at https://github.com/riccardoRubei/SPE_2024.

Endnotes

[1] https://eclipse.dev/atl/atlTransformations/.

[2] https://github.com/riccardoRubei/SPE_2024.

[3] https://rosettacode.org/wiki/Rosetta_Code.

[4] http://grabats2010.inf.mit.bme.hu/.

[5] https://wiki.eclipse.org/ATL/User_Guide_-_The_ATL_Language#Iterative_target_pattern_element.

[6] https://gomezabajo.github.io/Wodel/.

[7] CPU specification https://www.amd.com/en/products/apu/amd-ryzen-7-pro-6850u.

References

1. J. Hutchinson, J. Whittle, and M. Rouncefield, "Model-Driven Engineering Practices in Industry: Social, Organizational and Managerial Factors That Lead to Success or Failure," *Science of Computer Programming* 89 (2014): 144–161.

2. J. Whittle, J. E. Hutchinson, and M. Rouncefield, "The State of Practice in Model-Driven Engineering," *IEEE Software* 31, no. 3 (2014): 79–85, https://doi.org/10.1109/MS.2013.65.

3. D. D. Ruscio, R. F. Paige, and A. Pierantonio, "Guest Editorial to the Special Issue on Success Stories in Model Driven Engineering," *Science of Computer Programming* 89 (2014): 69–70, https://doi.org/10.1016/J.SCICO.2013.12.006.

4. P. Muñoz, S. Zschaler, and R. F. Paige, "Preface to the Special Issue on Success Stories in Model Driven Engineering," *Science of Computer Programming* 233 (2024): 103072, https://doi.org/10.1016/j.scico.2023.103072.

5. N. Huijboom, and V. D. T. Broek, "Open Data: An International Comparison of Strategies," *European Journal of ePractice* 12, no. 1 (2011): 4–16.

6. M. Tisi, S. M. Perez, and H. Choura, "Parallel Execution of ATL Transformation Rules," in *Lecture Notes in Computer Science*, vol. 8107 (Springer, 2013), 656–672, https://doi.org/10.1007/978-3-642-41533-3_40.

7. H. Giese and R. Wagner, "Incremental Model Synchronization With Triple Graph Grammars," in *Model Driven Engineering Languages and Systems, 9th International Conference, MoDELS 2006, Genova, Italy, October 1–6, 2006*. Proceedings. 4199 of Lecture Notes in Computer Science, eds. O. Nierstrasz, J. Whittle, D. Harel, and G. Reggio (Springer, 2006), 543–557.

8. E. R. Luna, J. M. Rivero, M. Urbieta, and J. Cabot, "Improving the Scalability of Web Applications With Runtime Transformations," in *Web Engineering, 14th International Conference, ICWE 2014, Toulouse, France, July 1–4, 2014. Proceedings. 8541 of Lecture Notes in Computer Science*, eds. S. Casteleyn, G. Rossi, and M. Winckler (Springer, 2014), 430–439.

9. U. A. Q. Ali, D. S. Kolovos, and K. Barmpis, "Efficiently Querying Large-Scale Heterogeneous Models" (2020): 73:1–73:5. https://doi.org/10.1145/3417990.3420207.

10. D. S. Kolovos, L. M. Rose, N. D. Matragkas, et al., "A Research Roadmap Towards Achieving Scalability in Model Driven Engineering" (2013): 2, https://doi.org/10.1145/2487766.2487768.

11. G. Daniel, F. Jouault, G. Sunyé, and J. Cabot, "Gremlin-ATL: a Scalable Model Transformation Framework" (2017): 462–472, https://doi.org/10.1109/ASE.2017.8115658.

12. L. Burgueño, J. Cabot, and S. Gérard, "The Future of Model Transformation Languages: An Open Community," *Journal of Object Technology* 7 (2019): 1–11, https://doi.org/10.5381/jot.2019.18.3.a7.

13. J. S. Cuadrado, L. Burgueño, M. Wimmer, and A. Vallecillo, "Efficient Execution of ATL Model Transformations Using Static Analysis and Parallelism," *IEEE Transactions on Software Engineering* 48, no. 4 (2022): 1097–1114, https://doi.org/10.1109/TSE.2020.3011388.

14. R. Verdecchia, P. Lago, C. Ebert, and D. C. Vries, "Green IT and Green Software," *IEEE Software* 38, no. 6 (2021): 7–15, https://doi.org/10.1109/MS.2021.3102254.

15. A. Fonseca, R. Kazman, and P. Lago, "A Manifesto for Energy-Aware Software," *IEEE Software* 36, no. 6 (2019): 79–82, https://doi.org/10.1109/MS.2019.2924498.

16. P. Gómez-Abajo, E. Guerra, and d. J. Lara, "Wodel: A Domain-Specific Language for Model Mutation," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing, Pisa, Italy, April 4–8, 2016*, ed. S. Ossowski (ACM, 2016), 1968–1973.

17. J. S. Cuadrado, E. Guerra, and d. J. Lara, "AnATLyzer: An Advanced IDE for ATL Model Transformations," in *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings, ICSE 2018, Gothenburg, Sweden, May 27–June 03, 2018*, eds. M. Chaudron, I. Crnkovic, M. Chechik, and M. Harman (ACM, 2018), 85–88.

18. Z. Ournani, R. Rouvoy, P. Rust, and J. Penhoat, "On Reducing the Energy Consumption of Software: From Hurdles to Requirements," in *ESEM '20* (Association for Computing Machinery, 2020).

19. W. Oliveira, R. Oliveira, F. Castor, B. Fernandes, and G. Pinto, "Recommending Energy-Efficient Java Collections," in *MSR '19* (IEEE Press, 2019), 160–170.

20. M. Kumar and W. Shi, "Energy Consumption Analysis of Java Command-Line Options," in *2019 Tenth International Green and Sustainable Computing Conference (IGSC)* (IEEE Computer Society, 2019), 1–8.

21. R. Pereira, M. Couto, F. Ribeiro, et al., "Energy Efficiency Across Programming Languages: How Do Energy, Time and Memory Relate?," in *SLE '17* (ACM, 2017), 256–267.

22. Z. Ournani, R. Rouvoy, P. Rust, and J. Penhoat, "Comparing the Energy Consumption of Java I/O Libraries and Methods," in *Proceedings of the 37th International Conference on Software Maintenance and Evolution (ICSME)* (Luxembourg / Virtual, 2021).

23. S. Georgiou, M. Kechagia, and D. Spinellis, "Analyzing Programming Languages' Energy Consumption: An Empirical Study," in *PCI '17* (Association for Computing Machinery, 2017).

24. I. Malavolta, E. M. Grua, C. Y. Lam, et al., "A Framework for the Automatic Execution of Measurement-Based Experiments on Android Devices" (2021): 61–66, https://doi.org/10.1145/3417113.3422184.

25. A. A. Bangash, D. Tiganov, K. Ali, and A. Hindle, "Energy Efficient Guidelines for iOS Core Location Framework" (2021): 320–331, https://doi.org/10.1109/ICSME52107.2021.00035.

26. J. Amutha, S. Sharma, and J. Nagar, "WSN Strategies Based on Sensors, Deployment, Sensing Models, Coverage and Energy Efficiency: Review, Approaches and Open Issues," *Wireless Personal Communications* 111 (2020): 1089–1115.

27. S. Shanbhag and S. Chimalakonda, "An Exploratory Study on Energy Consumption of Dataframe Processing Libraries," in *IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)* (May 15 2023), 284–295, https://doi.org/10.1109/MSR59073.2023.00048.

28. E. García-Martín, C. F. Rodrigues, G. Riley, and H. Grahn, "Estimation of Energy Consumption in Machine Learning," *Journal of Parallel and Distributed Computing* 134 (2019): 75–88.

29. S. Georgiou, M. Kechagia, T. Sharma, F. Sarro, and Y. Zou, "Green AI: Do Deep Learning Frameworks Have Different Costs?," in *Proceedings of the 44th International Conference on Software Engineering* (2022), 1082–1094, https://doi.org/10.1145/3510003.3510221.

30. T. Mészáros, G. Mezei, T. Levendovszky, and M. Asztalos, "Manual and Automated Performance Optimization of Model Transformation Systems," *STTT* 12 (2010): 231–243, https://doi.org/10.1007/s10009-010-0151-0.

31. v M. Amstel, S. Bosems, I. Ivanov, and L. Ferreira Pires, "Performance in Model Transformations: Experiments With ATL and QVT," in *4th International Conference on Theory and Practice of Model Transformations, ICMT 2011*. Lecture Notes in Computer Science, eds. J. Cabot and E. Visser (Springer, 2011), 198–212.

32. W. Piers, "ATL 3. 1 – Industrialization Improvements," 2010.

33. M. Wimmer, S. M. Perez, F. Jouault, and J. Cabot, "A Catalogue of Refactorings for Model-To-Model Transformations," *Journal of Object Technology* 11, no. 2 (2012): 2: 1–2: 40, https://doi.org/10.5381/JOT.2012.11.2.A2.

34. K. Lano and S. K. Rahimi, "Model-Transformation Design Patterns," *IEEE Transactions on Software Engineering* 40, no. 12 (2014): 1224–1259, https://doi.org/10.1109/TSE.2014.2354344.

35. V. M. F. Amstel and V. Brand, "Using Metrics for Assessing the Quality of ATL Model Transformations," in *Proceedings of the 3rd International Workshop on Model Transformation With ATL, MtATL@TOOLS 2011, Zürich, Switzerland, July 1st, 2011. 742 of CEUR Workshop Proceedings*, eds. I. Kurtev, M. Tisi, and D. Wagelaar (CEUR-WS.org, 2011), 20–34.

36. R. Groner, S. Gylstorff, and M. Tichy, "A Profiler for the Matching Process of Henshin," in *Models '20* (Springer. Association for Computing Machinery, 2020).

37. R. Groner, P. Bellmann, S. Höppner, P. Thiam, F. Schwenker, and M. Tichy, "Predicting the Performance of ATL Model Transformations," in *ICPE '23* (Association for Computing Machinery, 2023), 77–89.

38. G. Pinto, F. Soares-Neto, and F. C. Filho, "Refactoring for Energy Efficiency: A Reflection on the State of the Art," in *4th IEEE/ACM International Workshop on Green and Sustainable Software, GREENS 2015, Florence, Italy, May 18, 2015*, eds. P. Lago and H. A. Müller (IEEE Computer Society, 2015), 29–35.

39. J. Di Rocco, D. Di Ruscio, L. Iovino, and A. Pierantonio, "Mining Metrics for Understanding Metamodel Characteristics," in *MiSE 2014* (Association for Computing Machinery, 2014), 55–60.

40. v M F. Amstel and M. Van Den Brand, "Quality Assessment of ATL Model Transformations Using Metrics," in *Proceedings of the 2nd International Workshop on Model Transformation With ATL (MtATL 2010)* (Citeseer, 2010), 115.

41. A. Noureddine, "PowerJoular and JoularJX: Multi-Platform Software Power Monitoring Tools," in *18th International Conference on Intelligent Environments* (*IE*) (IEEE, 2022), 1–4, https://doi.org/10.1109/IE54923.2022.9826760.

42. Z. Zhang, S. Liang, F. Yao, and X. Gao, "Red Alert for Power Leakage: Exploiting Intel RAPL-Induced Side Channels," in *Asia Ccs '21* (Association for Computing Machinery, 2021), 162–175.

43. S. Liu, Y. Zhang, K. Tang, and X. Yao, "How Good Is Neural Combinatorial Optimization? A Systematic Evaluation on the Traveling Salesman Problem," *IEEE Computational Intelligence Magazine* 18, no. 3 (2023): 14–28, https://doi.org/10.1109/MCI.2023.3277768.

44. L. Wagner, M. Mayer, A. Marino, A. Soldani Nezhad, H. Zwaan, and I. Malavolta, "On the Energy Consumption and Performance of WebAssembly Binaries Across Programming Languages and Runtimes in IoT," in *Ease '23* (Springer. Association for Computing Machinery, 2023), 72–82.

45. A. K. Yadav, S. Wijethilaka, A. Braeken, M. Misra, and M. Liyanage, "An Enhanced Cross-Network-Slice Authentication Protocol for 5G," *IEEE Transactions on Sustainable Computing* 8 (2023): 1–18, https://doi.org/10.1109/TSUSC.2023.3283615.

46. J. R. Williams, A. Zolotas, N. D. Matragkas, et al., "What Do Metamodels Really Look Like?," in *Proceedings of the 3rd International Workshop on Experiences and Empirical Studies in Software Modeling co-Located With 16th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2013), Miami, USA, October 1, 2013*. 1078 of *CEUR Workshop Proceedings*, eds. C. MRV, M. Genero, S. Abrahão, and L. Pareto (CEUR-WS.org, 2013), 55–60.

47. J. D. Rocco, D. D. Ruscio, L. Iovino, and A. Pierantonio, "Mining Correlations of ATL Model Transformation and Metamodel Metrics," in *7th IEEE/ACM International Workshop on Modeling in Software Engineering, MiSE 2015, Florence, Italy, May 16–17, 2015*, eds. J. Gray, M. Chechik, V. Kulkarni, and R. F. Paige (IEEE Computer Society, 2015), 54–59.

48. D. B. West, *Introduction to Graph Theory. 2* (Prentice hall, 2001).

49. K. Lano, S. K. Rahimi, M. Sharbaf, and H. Alfraihi, "Technical Debt in Model Transformation Specifications," in *Theory and Practice of Model Transformation: 11th International Conference, ICMT 2018, Held as Part of STAF 2018, Toulouse, France, June 25-26, 2018, Proceedings 11* (Springer International Publishing, 2018), 127–141, https://doi.org/10.1007/978-3-319-93317-7_6.

50. S. Höppner, T. Kehrer, and M. Tichy, "Contrasting Dedicated Model Transformation Languages Versus General Purpose Languages: A Historical Perspective on ATL Versus Java Based on Complexity and Size," *Software and Systems Modeling* 21, no. 2 (2022): 805–837, https://doi.org/10.1007/S10270-021-00937-3.

51. J. S. Cuadrado, E. Guerra, and d J. Lara, "Reverse Engineering of Model Transformations for Reusability," in *Theory and Practice of Model Transformations-7th International Conference, ICMT@STAF 2014, York, UK, July 21–22, 2014. Proceedings. 8568 of Lecture Notes in Computer Science*, eds. D. D. Ruscio and D. Varró (Springer, 2014), 186–201.

52. J. D. Rocco, D. D. Ruscio, J. Härtel, L. Iovino, R. Lämmel, and A. Pierantonio, "Understanding MDE Projects: Megamodels to the Rescue for Architecture Recovery," *Software and Systems Modeling* 19, no. 2 (2019): 401–423, https://doi.org/10.1007/s10270-019-00748-7.

53. V. K. Singh, K. Dutta, and D. E. VanderMeer, "Estimating the Energy Consumption of Executing Software Processes," in *IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing* (IEEE, 2013), 94–101, https://doi.org/10.1109/GREENCOM-ITHINGS-CPSCOM.2013.40.

54. R. Hierons, M. Gazda, P. Gómez-Abajo, R. Lefticaru, and M. Merayo, *Mutation Testing for RoboChart* (New York: Springer International Publishing, 2020), 345–375.

55. A. Barriga, A. Rutle, and R. Heldal, "Personalized and Automatic Model Repairing Using Reinforcement Learning," in *ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)* (IEEE, 2019), 175–181, https://doi.org/10.1109/MODELS-C.2019.00030.

56. J. S. Cuadrado and O. C. L. Optimising, "Synthesized Code," in *Modelling Foundations and Applications-14th European Conference, ECMFA@STAF 2018, Toulouse, France, June 26–28, 2018, Proceedings. 10890 of Lecture Notes in Computer Science*, eds. A. Pierantonio and S. Trujillo (Springer, 2018), 28–45.

57. S. Götz and M. Tichy, "Investigating the Origins of Complexity and Expressiveness in ATL Transformations," *Journal of Object Technology* 19, no. 2 (2020): 11–12.

58. J. Bornholt, T. Mytkowicz, and K. S. McKinley, "The Model is not Enough: Understanding Energy Consumption in Mobile Devices," in *IEEE Hot Chips 24 Symposium (HCS)* (IEEE, 2012). 17–3, https://doi.org/10.1109/HOTCHIPS.2012.7476509.

59. J. Lee Rodgers and W. A. Nicewander, "Thirteen Ways to Look at the Correlation Coefficient," *American Statistician* 42, no. 1 (1988): 59–66.

60. J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-Means Clustering Algorithm," *Journal of the Royal Statistical Society: Series C: Applied Statistics* 28, no. 1 (1979): 100–108.

61. Z. VaraminyBahnemiry, J. Galasso, B. Oakes, and H. Sahraoui, "Improving Repair of Semantic ATL Errors Using a Social Diversity Metric," *Software and Systems Modeling* 1–22 (2024): 1547–1568.

62. B. J. Oakes, J. Troya, J. Galasso, and M. Wimmer, "Fault Localization in DSLTrans Model Transformations by Combining Symbolic Execution and Spectrum-Based Analysis," *Software and Systems Modeling* 23, no. 3 (2024): 737–763, https://doi.org/10.1007/S10270-023-01123-3.