Graph-Based Analysis of the Architectural Restructuring Impact on Energy Efficiency

Basma khil
Faculty of Mathematical, Physical and Natural Sciences of Tunis
Tunis, Tunisia
Email: basma.khil@fst.utm.tn

Adel Khalfallah
Higher Institute of Computer Science
Ariana, Tunisia
Email: adel.khalfallah@isi.utm.tn

Samir Ben Ahmed
Faculty of Mathematical, Physical and Natural Sciences of Tunis
Tunis, Tunisia
Email: samir.benahmed@fst.utm.tn

***Abstract*—Software design patterns and refactoring are widely used in software engineering to enhance maintainability, reuse and productivity. However, recent empirical studies revealed the high energy overhead in these patterns. Our approach consists of automatically applying refactoring techniques, detecting and injecting design patterns during design level for better energy efficiency without impacting existing coding practices. Regarding that, refactoring techniques could help to tackle these issues considering that it is a method of changing the internal design of the system while preserving the external behavior. In this paper, we propose a graph transformation for refactoring, design pattern injection and furthermore rules to compute the total energy consumption and perform an initial evaluation of the energy efficiency.**

Keywords–Energy-efficiency; Software Architectures; Graph transformation rules; Energy consumption.

I. INTRODUCTION

Energy consumption has emerged as an important design constraint in software engineering. Information and Communication Technology (ICT) [1] and the Internet of Things (IoT) yield a huge potential increase in energy demand. These kinds of systems are mostly imposed by a restrict power budget. This is a problem that now looks to exceed many challenges and has been enlarged into a mammoth task. It takes into account the effects of hardware, devices, networks, drivers, operating systems, and applications on energy consumption. In this paper, we focus on applications and, particularly, on how we experiment with the effect of applying transformations activities at a design level which can be optimized in terms of energy consumption.

In searching generic transformation units, we worked on the original definition of standardized transformations such as the refactoring catalog. Refactoring is proven to improve the quality of a system. Thus, it can be a potential solution to increase software maintainability and re-usability. It is proven that software engineering best practices can improve software maintainability [1][2]. Hence, investigating refactoring activities to optimize energy consumption seems more and more trendy.

Some tentative proved that software engineering best practices can improve energy efficiency [3][4][5]. Notably, [6][7][8][9] focused on the impact of refactoring activities on energy consumption. Nevertheless, the available evidences are tried and tested in a limited number of refactoring techniques. Typically, they are applied in a code artifact, Whereas the most common approach used in software engineering makes a great emphasis on the use of modeling artifact [1]. Therefore, it is desirable to master the modeling in software architecture by working at a relevant level of abstraction. That gives rise to the idea of managing energy consumption since the phase architectural design.

The scope of this work lies at the design level. We aim to explore the effect of transformation activities on energy consumption. In particular, we propose a combinatorial approach based on graph transformations. Namely, we use metamodeling to represent the architectural design artifact, as well as graph transformation rules to explore the different alternatives induced by the design decisions and transformations.

The remainder of the paper is organized as follows: Section 2 surveys recent literature to have an overview of how software engineering researchers are tackling energy consumption issues. In Section 3, the current proposal is explained by a suitable process and with an adequate architectural metamodel. Sections 4 and 5 state the graph transformation rules. The first kind of rule embodies transformation activities. the other kind surveys their effect on energy consumption. In Section 6, a motivating scenario is presented to illustrate the proposal with a concrete example. Finally, Section 7 concludes this paper and gives avenues for future work.

II. RELATED WORK

The literature on the energy efficiency topic shows divergences: According to some works, the energy consumption in network infrastructure is predominant, whereas, for others, it is prominent in the terminals. Many experimental approaches specifically deal with identifying the parts of an application that mostly affect the total energy consumption [10] and try to minimize its consumption. For this purpose, some trials [11][12] optimized code to minimize power consumption.

Luo [13] proposed an ant colony algorithm for task scheduling to optimizing the energy cost. Liu [14] explored the non-dominant sorting genetic algorithm to bridge the trade-off between energy consumption and delay in task scheduling. However, scheduling tasks using only offline power consumption information cannot generate efficient schedules on account of the dynamic variation in energy consumption. Thus,[15][16] proposed a real-time monitoring and management system for energy consumption. However, there is further evidence that changes in architectural design tend to have a greater impact on energy consumption [17].

Other attempts rely on the quantitative evaluation of energy consumption of software systems at higher levels and in early

 ISBN: 978-1-61208-752-8

stage in the software development process [18][19][20]. Some of these works proposed architecture description languages that support the analysis of power consumption at the design level [1][12][21]. Other implemented experimental solutions and tools to evaluate and monitor energy consumption. Seo has performed an energy consumption analysis for specific architectural styles [22] such as client-server. The works [23][24][25] propose an approach that predicts energy consumption using linear power models.

Other attempts aimed to evaluate the energy consumption of the Cloud Computing application and High-Performance Computing (HPC) systems. Previously discussed approaches focus on the energy consumption analysis at an architectural-level. Some of them do not take account of parametric dependencies between software components [24][25] while others percept the dependency between different energy concerns [26]. They specify the relationships between energy concerns under the modeled component. These relationships can then be used during the analysis phase to see how an energy concern (communication) can affect other energy concerns (for example, compression storage).

Recently, certain approaches take advantage of the positive effect of software engineering best practices on software quality [1][2]and on energy consumption to tackle trade-off between productivity and energy efficiency [3][4][6][7][8][9]. Some works prove that software engineering best practices can improve energy efficiency [3][4]. Others [6][7][8][9] focus on the impact of refactoring activities on energy consumption. The searchers experiment that idea in a code application such as Java, C and Android applications. However, these evidences are experimented in a limited number of refactoring techniques. Typically, they are empirical studies applied in a code artifact. However, it is desirable to work at a relevant level of abstraction and also manage the Global Software issues through modeling.

Our approach is involved in this area, it consists of applying refactoring activities by analyzing modeling artifacts and monitor the changes in energy consumption. That gives rise to the idea of managing energy consumption at architectural design. And follow the impact of refactoring on software consumption. Therefore Graph-based approaches seem very promising, due to their robust theoretical foundation.

III. Motivation

A major challenge that is currently faced in the design of applications concerns the optimal use of available energy resources. In particular, the IoT applications are imposed by the battery lifetime of the devices. The challenge is derived from the heterogeneity of the devices, in terms of their hardware and the provided functionalities. Several works in energy management are focusing their studies on the hardware side of computational systems. However, it is tempting to suppose that only hardware dissipates power, not software. Since energy consumption is the amount of energy used by devices or a built environment. The energy consumption varies according to the kind of device and the time that it remains in the operating modes. It is, therefore, necessary to think about saving energy, and that requires a careful choice of electrical appliances. Recently, the software engineering community started to carry out researches about estimations of energy consumption in software applications [12][17]. According to the authors,the software directs much of the activity of the hardware. Therefore, the software can have a substantial impact on the power dissipation of computational systems. They investigated the mixes of hardware-software designs to minimize energy consumption. However, these approaches were focused on low levels of software design, such as the number of execution cycles of a software, optimization of memory addresses. Other works analyze this issue from a different perspective [27] where energy management is discussed in higher levels of abstraction. Such levels are related to software requirement analysis, design and specification. We intend to approach this issue in design levels.

IV. Proposed Approach

The main contributions of this paper are defined as follows. First, it outlines an approach for modeling and injection of restructuring activities such as refactoring and ever more design patterns by analyzing UML diagrams. Second, a methodical analysis to assess the relative impact of that restructuring activity and expect the total energy consumption induced by the different components of an IoT application. Following that, we perform an analysis of the impact of refactoring activities on energy consumption and performance in software applications. We aim to take advantage of the formal foundations of graphs transformation.

A. The proposed method

In the thought of taking advantage of the formal foundation of graphs, we presented a graph-based transformation to introduce restructuring activities in the design of a new application or an existing one. Furthermore, we established sets of graph transformation rules to estimate the total energy consumed. We presented our graph-based system and then performed an exploratory analysis of the impact of design transformation on energy consumption and performance in software applications. In this area, graph-based approaches seem more promising due to their robust theoretical foundation. Consequently, graphs are well-known structures combining rigor with simplicity [28], which are beneficial in modeling design software systems.

B. Method process

In the exploratory study here reported we investigated the impact of software architecture restructuring with well-known transformation techniques on energy consumption. We consider design transformations activity as a set of graph transformations applied to a graph instance representing a given model. So, a given transformation recognition is provided by the mechanism of matching within graphs.

A graph transformation introducing a complex restructuring, such as a design Pattern, is composed of a sequence of graph transformation units. In searching for a generic transformation unit, we worked on the original definition of the refactoring catalog and the Elements Design Patterns (EDPs) defined by Smith [29]. They represent micro-transformations whose different combinations lead to the introduction of Design Patterns into models. So we built a library of the possible instantiations of each transformation such as Refactoring, EDP and intermediate pattern compositions. In practice, we used the toolset GROOVE to implement these transformations as graph transformation rules [11]. Also, we enrich our rule set by another kind of rules that compute the energy consumed

 ISBN: 978-1-61208-752-8

under each stage in the entire application tasks. We will explain later in this article how to investigate the effect of refactoring transformation in energy efficiency by analyzing UML diagrams (class diagram) and not only existing code. Software optimizations and energy computing, in this context, have been discussed at three levels of granularity (Figure 1): Type graph, graph transformation rules and instance graph.

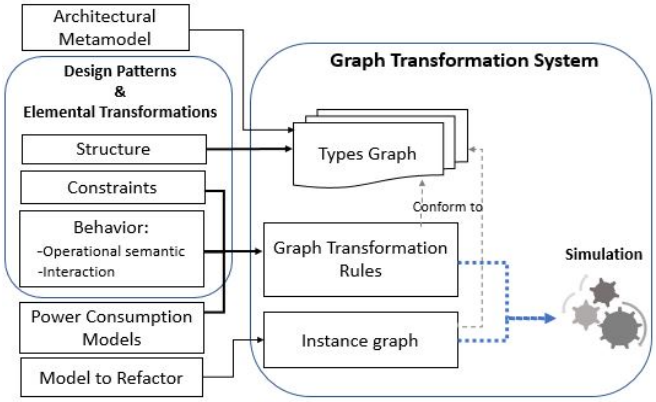

Figure 1. Graph transformation system.

We represent our architectural design as a graph, we identify and represent refactoring activity as graph transformation rules and then we formalize another cluster of rules to compute the energy consumption. The following section will detail every level individually.

V. META MODEL

To analyze the energy consumption of a software system, we propose the meta-model depicted in Figure 2. It includes the required artifacts, on the one hand, to describe the software architecture and on the other hand to assess the expected energy consumption.

Our model is based on the concepts of component-oriented architectures (CBSE) and service-oriented architectures (SOA). To analyze the expected energy consumption of so many alternative architectural solutions, we must define the elements of such a solution. As shown in Figure 2, the architectural aspect is represented by an architectural style element that includes a collection of homogeneous and heterogeneous components (elements and architectural constraints). An architectural element is composed, in turn, by other architectural elements, components that interact through connectors.

A component is defined as a set of interacting tasks and services to fill a role and communicate with the environment via two interfaces. Typically, connectors define abstractions that encapsulate the mechanisms of communication, coordination, and conversion (type, number, frequency, and order of interactions) between components. The component is also defined by a predefined set of tasks or roles. A task is a semantic entity of the basic unit of work (activity or role). It can be a task of calculation or storage, etc. It can be extended by others under spots. Sometimes the execution of a task is heavily dependent on other tasks (for example, remote storage of data depends on the communication problem). As a result, this information is defined by a reflexive relationship that reflects this dependency [30].

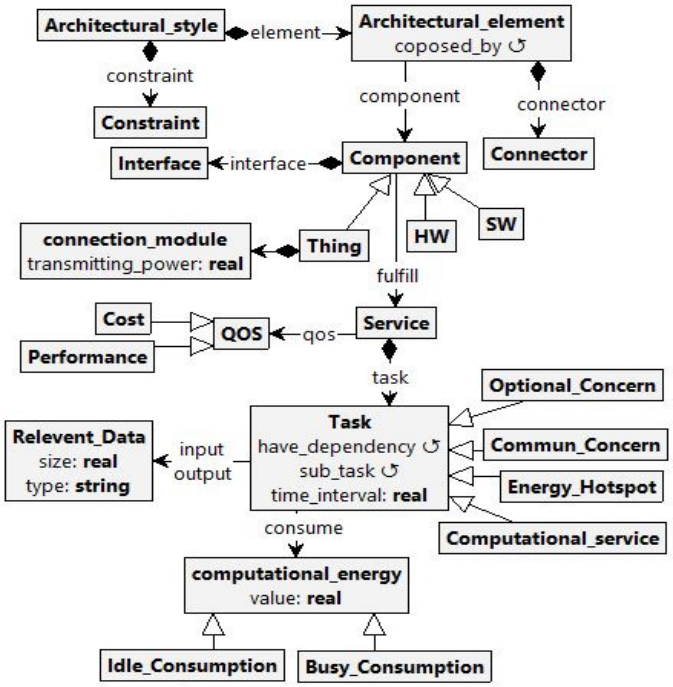

Figure 2. Component-oriented architecture as a graph.

To enlarge the scope of the current approach, in particular, tackling the artifacts involved with the IoT paradigm. the metamodel encompasses concepts related to a thing.

A Thing is organized into two categories (Figure 3):

- A physical thing is organized into a group of physical networked things, including devices, sensors, actuators, and even embedded devices.
- A virtual thing is organized into a virtual group of things in a network, including web services and programs.

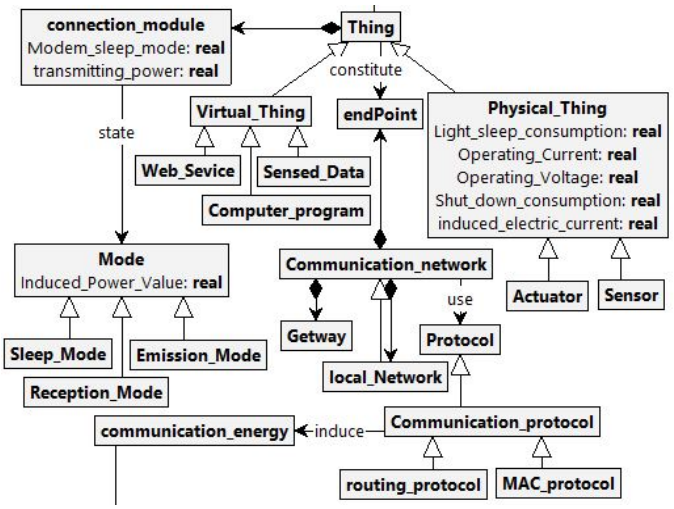

Figure 3. Meta-model of the entities types in IoT.

To analyze the expected energy consumption, it is required to discriminate the fields that affect the global consumption. They

are mainly due to several operations such as computational tasks and data communication.

The Energy consumed by the processing unit is divided into two parts: the energy induced by computational tasks in the busy state and the energy consumed in the idle state. The first one is determined by the supply voltage and the total capacity switched likewise in the hardware level (sensors and actuators, etc.) and the software level (by running software program, services, etc.). The second one corresponds to the energy consumed when the calculation unit does not carry out any treatment. The communication energy can be divided into the reception energy, the energy of the emission and the sleep mode. This energy is determined by the amount of data to be communicated and the transmission distance, as well as by the physical properties of the communication module.

Another part of our meta-model is dedicated to representing power distribution infrastructure (see Figure 4). Power distribution infrastructure can be organized in a hierarchical manner [1]. Power distribution units distribute power to racks which in turn provide power to the connected devices.

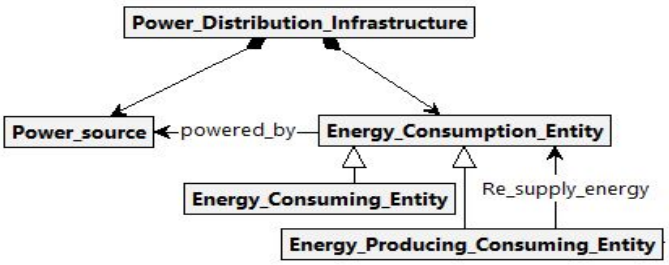

Figure 4. hierarchical of power distribution infrastructure.

In order to percept the dependency between different tasks, we add a reflexive link labeled as ”have dependency”. It specifies the relationships between energy concerns under the modeled component. Then, we classified tasks according to the well-known energy concerns hierarchy and activities more recurrent in a given application such as Store, Communication, Data Access, Data collect [31] (see Figure 5). That list will be augmented once there is new evidence about other energy hotspots.

There are many variabilities in how to design and implement concerns (e.g., the data could be stored locally or remotely, compress audio or video files). Additionally, these concerns could depend on each other. For instance, there are several concerns related to Communication, such as Data Access and Store. Due to that dependency, for every energy consumption concern cannot be analyzed on an isolated basis. Instead, a whole architecture should be analyzed taking into account these explicit dependencies modeled.

VI. Expected energy consumption based-graph transformation rules

Commonly, energy calculation is straightforward. The electrical energy is (in kilowatt-hour, kWh), found by multiplying the power use (in kilowatts, kW) by the number of hours during which the power is consumed.
Accurate the expected energy consumption characterization

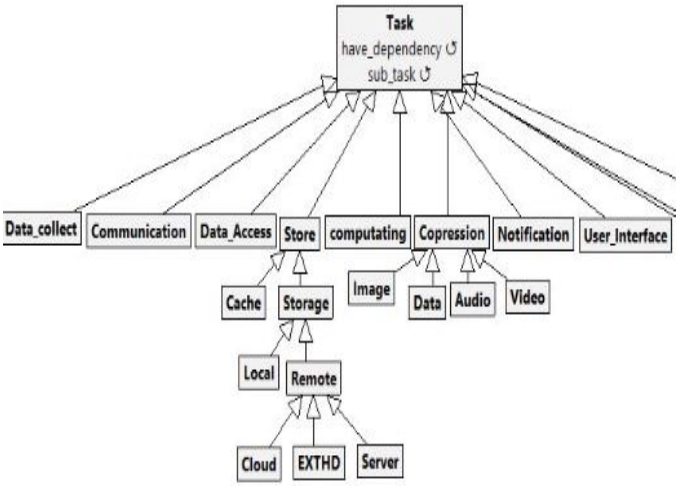

Figure 5. Relevant concerns in IoT applications.

is important in computing platforms, notably IoT based applications. To extend our approach on a large scale of IoT applications, we adopt an incremental scheme to quantify total energy consumption. We consider that the total consumption is evaluated by the sum of the energy consumption induced by the different tasks from the collection of the data, sending via the network until the processing of this data. The total spot is estimated and summarized over the period of real-time which is the typical IoT application architecture. Thus, the period T is outlined by three layers as follow:

- The first is the perception layer: It is defined by physical objects and sensor devices that collect and acquire data from the physical world. It consists of two parts: the detection devices and the wireless sensor network. The first includes sensor nodes and the Radio Frequency Identification (RFID) tag. The latter is a self-organized wireless network consisting of numerous sensor nodes distributed over a large area. These devices coordinate to monitor the state of the physical environment (M2M terminal and a sensor gateway). These devices monitor in a coordinated manner the state of the physical world. The collected information is then passed on to the transport layer.
- Transport layer is an intermediate layer: It enables the transfer of data received from the perception layer to the application layer through different networks as wireless or cable networks. There are various technologies used include infrared, 2G, 3G and Bluetooth, depending on the sensor devices. The collected data will be transferred across long distances and over a large area through different kinds of networks that employ different technologies and protocols.
- Application Layer: this part focuses on data processing and providing services for all user types. The transmitted data will be treated and managed by suitable management systems, Then various services will be provided to the target users.

Figure 6 displays how energy consumption at the global process of an IoT application is estimated. It is the sum

 ISBN: 978-1-61208-752-8

of several contributions represented by different areas, called layers. A layer, Li, corresponds to the consumption of part of the system for task i. The energy consumption of the entire system is the addition of the energy consumption of each task. It essentially consists of applying estimation for each task of the system with the order of layers.

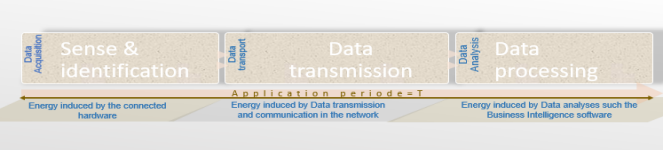

Figure 6. Energy consumption induced within IoT architecture.

Based on this assumption the power required can be broken down into three main blocks: power for data acquisition appointed as Pacq, power for data processing Pprc and power for data networking Pnet. Additionally, a tiny fraction is intended for system management tasks such as rising the system at periodic wake-ups or running a real-time operating system. The needs of these management tasks are gathered in this Psys contribution. The general formula (1) is expressed by the contribution of these elements together.

$$Ptot = Pacq + Pprc + Pnet + Psys \quad (1)$$

In order to estimate the power consumption, it is required to avail Power models. These models correlate energy consumption with measurable metrics. A wide variety of power models exist [20][32][33]. We establish a set of graph transformation rules implementing power consumption models as mentioned in Figure 1. Those rules enable computing energy consumption in the different layers (Figure 7). For instance, Figure 8 introduce the formula (1) as graph transformation rule.

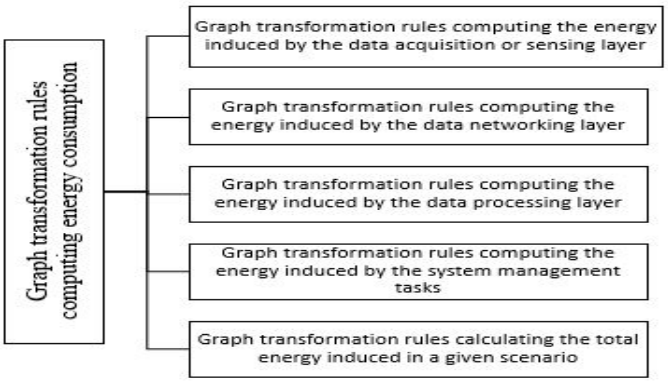

Figure 7. kinds of graph transformation rules computing energy consumption.

We start with exploring the energy consumed by the connected devices (sensor, actuator and computer program). Although software systems do not consume energy themselves, they affect the use of the hardware resulting in indirect energy consumption. Namely, it is required to inquire into the given software under execution, hardware platform and during a given time. The energy consumption E is an accumulation of power dissipation P over time (formula (2)). The energy E is measured in watts and power P is measured in joules, i.e.

$$E = P * t \quad (2)$$

For example, if a given task takes 15 seconds to be achieved and dissipate 5 watts, it consumes 75 joules of energy.

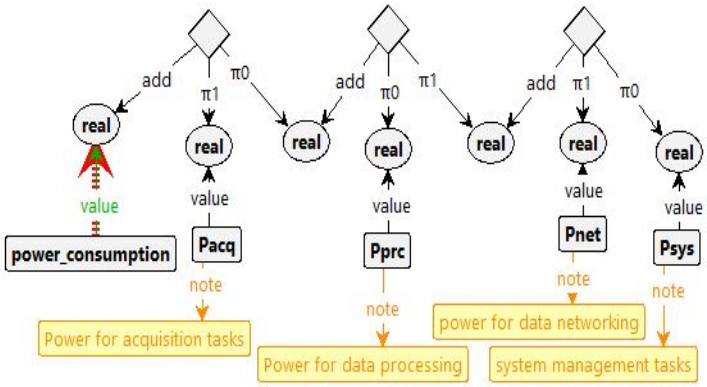

Figure 8. Computing total power consumption.

The display view of the corresponding rule (Figure 9) is composed of different kinds of nodes: the node depicted by a diamond stand for triple of data values. It states a multiplication operation for a pair of real values 0 and 1 which correspond respectively to power and time interval. The edge labeled as “mul” specifies the data node representing the result of the performed operation. Then the result will be attributed to a node typed as power consumption which is an element of the adopted meta-model. Note that the ellipses, typed as real, allow to handle unknown values and the values will only be established when matching the rule. It ensures the applicability of that rule in all cases of value. This rule calculates energy consumption by multiplying the power by the estimated time for such a task. After computing the energy consumption in each task alone, it is required to elaborate a rule that encompasses the total energy induced by a set of tasks, which collaborate to achieve a particular intent or service.

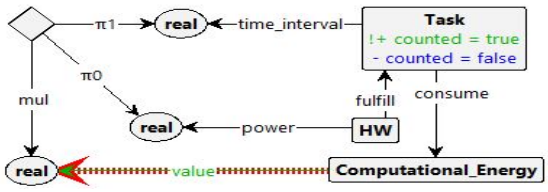

Figure 9. Graph transformation rule calculating energy induced by a set of tasks.

Figure 10 and Figure 11 depict graph transformation rules modeling Data Acquisition Energy.
Commonly, monitoring applications could be classified into two categories: regular sensing with a fixed acquisition time interval, and event-driven sensing characterized by stochastic distribution. In event-driven sensing, a random event triggers the acquisition of a collection of samples from the sensor. Thus, the energy consumption of the acquisition component

can be established as follow (4)[33].

$$E_{acq} = \begin{cases} E_{smp} * N & (Regular) \\ E_{smp} * N' * Pr(e) & (Event) \end{cases} \quad (3)$$

E_{smp} is the energy required to acquire one sample and N is the number of samples taken during one regular sensing interval. For event-driven sensing, Pr(e) is the probability of an event occurring in one sensing interval, and N' is the number of samples taken following the occurrence of an event.

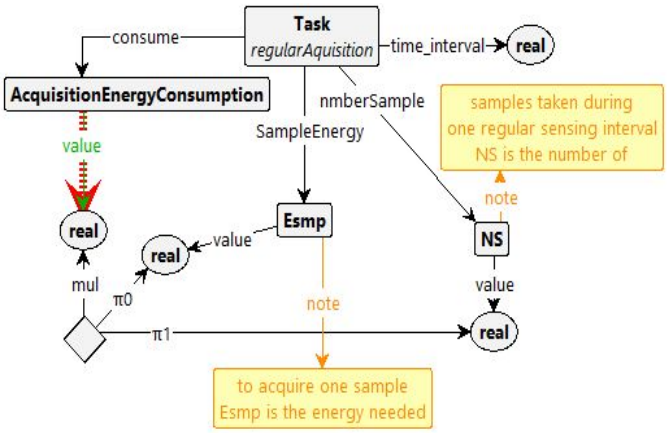

Figure 10. Graph transformation rule computing energy consumption of regular sensing

In our framework, the calculation is carried with graph transformation rules. For instance, the graph transformation rule (shown in Figure 10) calculates the energy consumed by the acquisition component during one regular sensing interval.

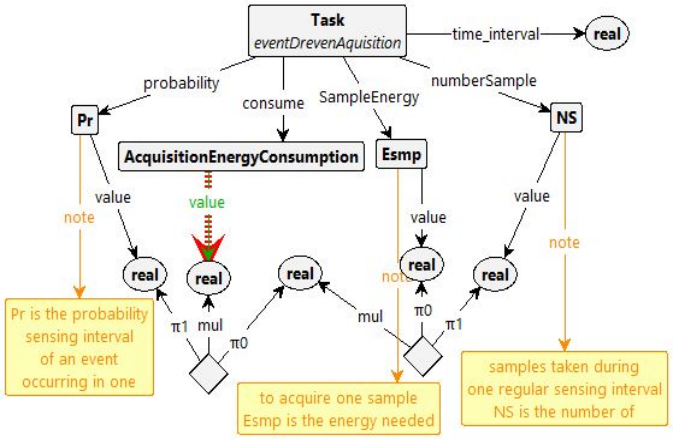

Figure 11. Graph transformation rule computing energy consumption of event-driven sensing

Figure 11 computes the energy consumed by the acquisition component in event-driven sensing.

VII. Elementary Transformations Based Graph Transformation Rules

In order to implement our approach, we build a library of combined graph transformation rules that incorporate restructuring activities and refactoring [34]. We take advantage of the formal specifications of refactoring techniques presented in the literature such as the refactoring catalog, EDPs catalog. Entire EDPs can be found in [29] with full definition and original explication. Please refer to that base document if necessary. Some refactoring operations are represented in TABLE I.

TABLE I. ELEMENTARY TRANSFORMATIONS INCLUDING THEIR ACTORS AND ROLES.

Elementary transformations	*tackled artifacts*	*Role*
Extract class	class	source class, new class
	field	moved fields
	method	moved methods
Extract interface	class	source classes, new interface
	field	moved fields
	method	moved methods
Inline class	class	source class, target class
Move field	class	source class, target class
	field	moved field
Move method	class	source class, target class
	method	moved method
Push down field	class	superclass, subclasses
	field	moved field
Push down method	class	superclass, subclasses
	method	moved method
Pull up field	class	subclasses, superclass
	field	moved field
Pull up method	class	subclasses, superclass
	method	moved method
Move class	package	source package, target package
	class	moved class

Every restructuring activity is associated with a graph transformation rule implemented using GROOVE, with the same name and the corresponding components. This part will depict some of the techniques formalized as graph transformation rules. Figure 12 represents an elemental transformation as graph transformation rule. It aims to create a class hierarchy once two classes have two attributes with the same names and the same types, and then pull up that attribute to the superclass. Figure 13 represents another elemental transformation as a graph transformation rule. It leads to pull up the other attributes to the superclass.

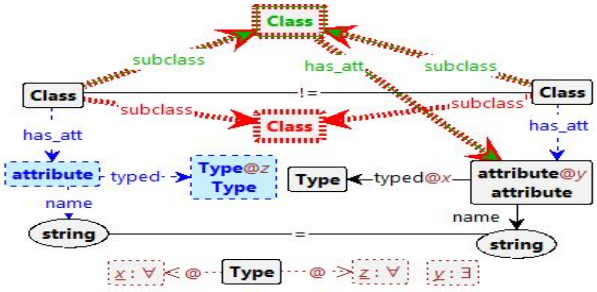

Figure 12. The transformation"pull up field" represented as graph transformation rule

Applying some of the elemental transformation techniques can impact energy consumption whether they can contribute to the design pattern injection. In particular when applying the technique 'Inline Method' can enhance performance and support power reduction for a specific application [35]. In the embedded systems, the 'Inline Method' may be useful (if it is small) since it can yield less code than the method call preamble and return. Extract Method and Extract Class can increase the energy consumption of mobile devices due to the increase of message traffic between the objects [6].

 ISBN: 978-1-61208-752-8

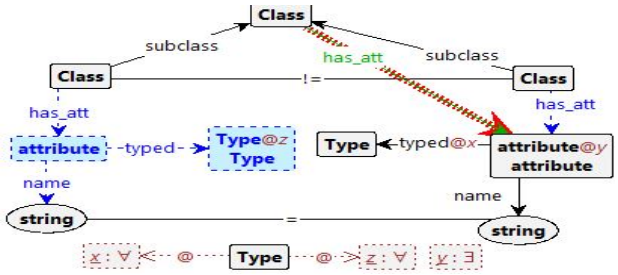

Figure 13. The transformation"Move field" represented as a graph transformation rule

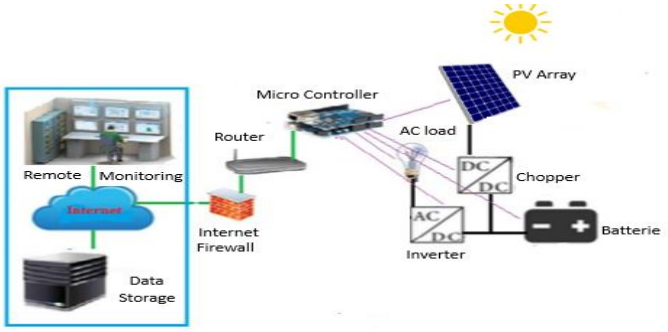

Figure 14. Architecture photovoltaic system based on IoT technology.

Additionally, applying a sequence of elemental transformations in particular orders could contribute to the injection of design patterns [36]. Thus, they can enhance energy efficiency corresponding that software engineering best practices can improve energy efficiency [3][4][5].

VIII. Motivating Scenario

We are investigating the validity of our approach in motivating scenarios in the scope of IoT. A focus is made on the use of IoT in the monitoring and remote control in the solar photovoltaic system accurately on the off-grid system installing. It is customary that this kind of electricity-management system includes a combination of a photovoltaic module, electric power converters and Storage devises to handle the intermittency of power output presented by renewables [37]. Besides, it contains power-conditioning equipment, including devices to limit current and voltage to maximize power output and convert direct-current to alternating current.

Availing the IoT technique, additional smart components enable to achieve energy efficiency in PV systems. That technology is used at all levels of the network such as production, distribution and consumption. It allows to:

- Real-time flow control: System-wide sensors instantly show electrical flows and consumption levels. Operators can then redirect energy flows according to demand.
- The integration of different types of renewable energies.
- More responsible management of consumption resources (scheduling): They provide useful information for the scheduling of household electricity supplies during the day in case of lack of energy.

IoT based photovoltaic system architecture can be established by three different layers as clearly depict in Figure 14. The PV system layer, gateway linkage layer and the remote control and monitoring layer. Figure 14 clearly depicted the IoT architecture for photovoltaic systems.

Although the performance and cost of each component of the PV System are important parameters to be considered before the design process, it is required to carry out optimizing in the software held in the system. thus, software design choice effects heavily global cost and performances. For the sake of enhancing performances and cost reduction for a prospective mini-grid architect, we undertake an analysis of various architectural solutions.

Our investigation is ongoing for identifying eligible tasks that constitute hotspot and undertake transformations and refactoring techniques. We are availing quality metrics measurement to access the impacts of applying restructuring activities and to make informed trade-off decisions between costs and QoS of offered services.

IX. Conclusion and Future Work

Energy-aware software development is a growing trend in computing. Indeed, the software developer community is paying more and more attention to energy-efficiency concerns. Refactoring can be a potential solution to many of the discussed challenges as architectural choices and design quality; it is proven to improve the quality of a system. However, the impact of design refactoring on energy efficiency has been scarcely investigated. In the exploratory study here reported, a graph-based approach is proposed to investigate the impact of refactoring on energy consumption on the design level, focusing on how we experiment with the effect of applying transformations activities at a design level, which can be optimized in terms of energy consumption. According to the literature, though refactoring is involved in the area of code re-engineering successfully there is huge potential for refactoring at the architectural level.

In further work, it is intended to develop a concept of following detailed refactoring techniques which include methods to identify architecture smells and to evaluate its effect in the consumption energy, apply suitable refactoring and test applied refactoring to guarantee less energy consumption of the system.

References

[1] G. Procaccianti, H. Fernández, and P. Lago, "Empirical evaluation of two best practices for energy-efficient software development," Journal of Systems and Software, vol. 117, 2016, pp. 185–198, ISSN: 0164-1212.

[2] F. A. Moghaddam, G. Procaccianti, G. A. Lewis, and P. Lago, "Empirical validation of cyber-foraging architectural tactics for surrogate provisioning," Journal of Systems and Software, vol. 138, 2018, pp. 37–51, ISSN: 0164-1212.

[3] A. Hindle, "Green mining: a methodology of relating software change and configuration to power consumption," Empirical Software Engineering, vol. 20, no. 2, 2015, pp. 374–409, ISSN: 1382-3256.

[4] A. R. Tonini, L. M. Fischer, J. C. B. de Mattos, and L. B. de Brisolara, "Analysis and evaluation of the android best practices impact on the efficiency of mobile applications," in Proceedings of the 3rd Brazilian Symposium on Computing Systems Engineering (SBESC) December 4–8, 2013, Niteroi, Rio De Janeiro, Brazil. IEEE, Dec. 2013, pp. 157–158, ISBN: 978-1-4799-3890-2.

[5] Linares-Vásquez et al., "Mining energy-greedy api usage patterns in android apps: an empirical study," in Proceedings of the 11th Working Conference on Mining Software Repositories(MSR) May 31 – June 01, 2014, Hyderabad, India. ACM, May 2014, pp. 2–11, ISSN: 978-1-4503-2863-0.

[6] R. Pérez-Castillo and M. Piattini, "Analyzing the harmful effect of god class refactoring on power consumption," IEEE software, vol. 31, no. 3, 2014, pp. 48–54, ISSN: 0740-7459.

[7] A. Vetrò, L. Ardito, G. Procaccianti, and M. Morisio, "Definition, implementation and validation of energy code smells: an exploratory study on an embedded system," in Proceedings of the 4th international conference on Future energy systems (e-Energy) May 21 – 24, 2013, Berkeley, California, USA. ThinkMind, May 2013, pp. 34–39, ISSN: 978-1-4503-2052-8.

[8] M. Gottschalk, J. Jelschen, and A. Winter, "Saving energy on mobile devices by refactoring." in Proceedings of the 28th International Conference on Informatics for Environmental Protection: ICT for Energy Effieciency, (EnviroInfo) September 10–12, 2014, Oldenburg, Germany,. BIS-Verlag, Sep. 2014, pp. 437–444, ISBN: 978-3-8142-2317-9.

[9] A. Rodriguez, M. Longo, and A. Zunino, "Using bad smell-driven code refactorings in mobile applications to reduce battery usage," in Simposio Argentino de Ingeniería de Software (ASSE) September 3–4, 2015, Rosario, Santa Fé, Argentina, Sep. 2015, pp. 56–68, ISSN: 2451-7593.

[10] G. Mouzon and M. B. Yildirim, "A framework to minimise total energy consumption and total tardiness on a single machine," International Journal of Sustainable Engineering.

[11] S. Hasan et al., "Energy profiles of java collections classes," in Proceedings of the 38th International Conference on Software Engineering (ICSE) May 14 – 22, 2016, Austin, Texas. ACM, May 2016, pp. 225–236, ISBN: 978-1-4503-3900-1.

[12] D. Li, S. Hao, J. Gui, and W. G. Halfond, "An empirical study of the energy consumption of android applications," in 2014 IEEE International Conference on Software Maintenance and Evolution (ICSME) Sep 28 – Oct 3, 2014, Victoria, British Columbia, Canada. IEEE, Sep. 2014, pp. 121–130, ISBN: 978-1-4799-6146-7.

[13] H. Luo, B. Du, G. Q. Huang, H. Chen, and X. Li, "Hybrid flow shop scheduling considering machine electricity consumption cost," International Journal of Production Economics, vol. 146, 2013, pp. 423–439, ISSN: 0925-5273.

[14] Y. Liu, H. Dong, N. Lohse, S. Petrovic, and N. Gindy, "An investigation into minimising total energy consumption and total weighted tardiness in job shops," Journal of Cleaner Production, vol. 65, 2014, pp. 87–96, ISSN: 0959-6526.

[15] M. Trejo-Perea et al., "Development of a real time energy monitoring platform user-friendly for buildings," Procedia Technology, vol. 7, 2013, pp. 238–247, ISSN: 1877-7058.

[16] R. Bayindir, E. Irmak, I. Colak, and A. Bektas, "Development of a real time energy monitoring platform," International Journal of Electrical Power & Energy Systems, vol. 33, no. 1, 2011, pp. 137–146, ISSN: 0142-0615.

[17] K. Grosskop and J. Visser, "Identification of application-level energy optimizations," Proceeding of ICT for Sustainability (ICT4S), vol. A4, 2013, pp. 101–107, ISBN: 978-3-906031-24-8.

[18] A. Noureddine and A. Rajan, "Optimising energy consumption of design patterns," in Proceedings of the 37th International Conference on Software Engineering-Volume 2 (ICSE) May 16 – 24, 2015,Florence, Italy. IEEE Press, May 2015, pp. 623–626, ISSN: 1558-1225.

[19] G. Procaccianti, P. Lago, and G. A. Lewis, "Green architectural tactics for the cloud," in Proceedings of the 2014 IEEE/IFIP Conference on Software Architecture (WICSA) April 7–11, 2014, Sydney, NSW, Australia. IEEE, Apr. 2014, pp. 41–44, ISBN: 978-1-4799-3412-6.

[20] C. Stier, A. Koziolek, H. Groenda, and R. Reussner, "Model-based energy efficiency analysis of software architectures," in Proceedings of the 9th European conference on software architecture (ECSA) September 7 – 11, 2015, Dubrovnik/Cavtat, Croatia. Springer, Sep. 2015, pp. 221–238, ISBN: 978-3-319-23727-5.

[21] V. De Maio, R. Prodan, S. Benedict, and G. Kecskemeti, "Modelling energy consumption of network transfers and virtual machine migration," Future Generation Computer Systems, vol. 56, 2016, pp. 388–406, ISSN: 0167-739X.

[22] C. Seo, G. Edwards, S. Malek, and N. Medvidovic, "A framework for estimating the impact of a distributed software system's architectural style on its energy consumption," in Proceedings of the 7th Working IEEE/IFIP Conference on Software Architecture (WICSA) February 18 – 21, 2008, Vancouver, BC, Canada. IEEE, Feb. 2008, pp. 277–280, ISBN: 978-0-7695-3092-5.

[23] A. Brunnert, K. Wischer, and H. Krcmar, "Using architecture-level performance models as resource profiles for enterprise applications," in Proceedings of the 10th international ACM Sigsoft conference on Quality of software architectures (QoSA) June 30 - July 04, 2014, Marcq-en-Bareul, France. ACM, Jul. 2014, pp. 53–62, ISBN: 978-1-4503-2576-9.

[24] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Software: Practice and experience, vol. 41, 2011, pp. 23–50, ISBN: 978-1-60750-073-5.

[25] K. Kurowski et al., "Dcworms–a tool for simulation of energy efficiency in distributed computing infrastructures," Simulation Modelling Practice and Theory, vol. 39, 2013, pp. 135–151, ISBN: 978-3-642-40516-7.

[26] A. Memari, J. Vornberger, J. M. Gómez, and W. Nebel, "A data center simulation framework based on an ontological foundation," in Proceedings of the 28th International Conference on Informatics for Environmental Protection: (ICT) for Energy Effieciency, September 10-12, 2014, Oldenburg, Germany. BIS-Verlag, Sep. 2014, pp. 461–468, ISBN: 978-3-8142-2317-9.

[27] N. Amsel, Z. Ibrahim, A. Malik, and B. Tomlinson, "Toward sustainable software engineering: Nier track," in Proceedings of the 33RD international conference on software engineering (ICSE) May 21 – 28, 2011, Honolulu, Hawaii, USA. IEEE, May 2011, pp. 976–979, ISBN: 978-1-4503-0445-0.

[28] E. Biermann et al., "Emf model refactoring based on graph transformation concepts," Electronic Communications of the EASST, vol. 3, 2006, ISSN: 1863-2122.

[29] J. M. Smith, Elemental design patterns. Addison-Wesley, Apr. 2012, ISBN: 978-0321711922.

[30] M. Kim, H. Ahn, and K. P. Kim, "Process-aware internet of things: A conceptual extension of the internet of things framework and architecture," KSII Transactions on Internet and Information Systems (TIIS), vol. 10, 2016, pp. 4008–4022, ISSN: 1976-7277.

[31] N. Gamez, M. Pinto, and L. Fuentes, "Hadas green assistant: designing energy-efficient applications," arXiv preprint arXiv:1612.08095, vol. abs/1612.08095, 2016.

[32] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in Proceedings of the 34th annual international symposium on Computer architecture (ISCA) June 9–13, 2007, San Diego, California, USA, vol. 35. ACM, Jun. 2007, pp. 13–23, ISSN: 0360–5442.

[33] B. Martinez, M. Monton, I. Vilajosana, and J. D. Prades, "The power of models: Modeling power consumption for iot devices," IEEE Sensors Journal, vol. 15, 2015, pp. 5777–5789, ISSN: 1558-1748.

[34] B. Babic, N. Nesic, and Z. Miljkovic, "A review of automated feature recognition with rule-based pattern recognition," Computers in industry, vol. 59, no. 4, 2008, pp. 321–337, ISSN: 0166-3615.

[35] W. G. da Silva, L. Brisolara, U. B. Corrêa, and L. Carro, "Evaluation of the impact of code refactoring on embedded software efficiency," in Proceedings of the 1st Workshop de Sistemas Embarcados (WESE) October 28 – 28, 2010, Scottsdale, Arizona, Oct. 2010, pp. 145–150, ISBN: 978-1-4503-0521-1.

[36] N. Zoubeir, A. Khalfallah, and S. Ahmed, "Graph-based decomposition of design patterns," International Journal of Software Engineering and Its Applications (IJSEIA), vol. 8, 2014, pp. 391–408, ISSN: 1738-9984.

[37] N. M. Kumar, K. Atluri, and S. Palaparthi, "Internet of things (iot) in photovoltaic systems," in Proceedings of the National Power Engineering Conference (NPEC) March 9–10, 2018, Madurai, India. IEEE, Mar. 2018, pp. 1–4, ISBN: 978-1-5386-3804-0.

 ISBN: 978-1-61208-752-8