# Towards platform specific energy estimation for executable domain-specific modeling languages

Thibault Béziers La Fosse, Massimo Tisi, Erwan Bousse, Jean-Marie Mottu, Gerson Sunyé

# Towards platform specific energy estimation for executable domain-specific modeling languages

Thibault Béziers la Fosse*, Massimo Tisi*, Erwan Bousse†, Jean-Marie Mottu†*, Gerson Sunyé†
LS2N, UMR CNRS 6004, *IMT Atlantique, †Université de Nantes {firstname.lastname}@ls2n.fr

***Abstract*—Energy consumption is becoming a major subject when designing, developing and running programs. Most developers code and run their applications in an energy oblivious manner, mostly because of a lack of energy-related knowledge about their system. This problem also exists in the realm of executable domain-specific modeling languages, where end-users create models conforming to a given meta-model and execute them with little knowledge about their operational semantic and related energy consumption. In this work, we propose a domain-specific language for decorating meta-models of executable languages with platform-specific energy estimation formulas. We also extend the GEMOC execution engine to dynamically perform energy estimations on any executable model conforming to the decorated meta-model. The energy estimation model defined can then be easily adapted to other models and platforms, without requiring any measurement tooling or knowledge from the end-user.**

***Index Terms*—Model-Driven Engineering, xDSMLs, Energy Estimation**

## I. Introduction

In 2007, Jed Scaramella stated that in the realm of data-centers, for each $1.00 spent on new hardware, an additional $0.50 is spent on power and cooling [23]. This same year, the estimated greenhouse emissions from the IT sector represented about 2% of the total human activity [25]. Several studies show that this amount keeps on growing, and is expected to grow even more in the future: up to 23% of the global greenhouse emissions of human activities, and as much as 51% of the global electricity in the worst-case scenarios [1].

Even if several hardware optimizations successfully manage to reduce the energy consumption of computers [20], [13], [26], a considerable amount of this energy is due to the software executed on it [6], [22]. Implementing energy-efficient applications should be the main focus for developers. Nevertheless, studies show that the majority of them do not consider energy efficiency when developing or do not know enough about the energy consumption of their programs for doing more efficient design choices [18], [19]. Having feedbacks about the energy consumption of their programs would help towards that purpose. However, deploying energy measurement tools is complex, and often requires knowledge about systems, software analysis, or hardware properties [17], [10].

This is especially true in the realm of executable domain-specific modeling languages (xDSMLs), where end-users do not necessarily know, or consider, the implementation of the operational semantics of their language. To help end-users improve the energy efficiency of the executable models they design, we propose in this paper a generic approach for estimating the energy consumption of their execution. This approach does not require any measurement tooling from the end-user, nor any knowledge about energy consumption, and can provide estimations about the energy consumption of any executed model, on the platform for which it has been designed.

Our contributions towards that purpose are the following: (1) A domain-specific language (DSL) for specifying energy-estimation formulas. This DSL can be attached to any xDSML in the GEMOC language workbench and can be used for estimating the energy consumption of all the executable models conforming to this meta-model, and for a specific platform. This DSL is meant to be used by qualified DSL engineers, or energy measurement specialists with knowledge of the xDMSL. (2) An extension of the GEMOC modeling workbench to estimate the energy consumption of running models using the modeled energy-estimation formulas. This work is the first step towards a fully generic, automated, and validated approach for estimating the energy consumption of any interpreted DSL.

This paper is organized as follow: Section II presents related work about energy estimation and measurement, Section III presents our approach for modeling and evaluating energy estimation formulas, Section IV discusses the approach, and Section V concludes the paper and proposes some future work.

## II. Related Work

In this section, we present some related work on energy estimation and measurements. We group this work according to the measurement level to which the approach is applied.

**Power meters.** Power-meters are external devices, usually plugged to the power supply of the computers they monitor [27], [15], [14]. Their main benefit is that they do not require any modification of the hardware, nor the software, to measure the power consumed by the system. Nonetheless, the metrics obtained are coarse-grained: they include the entire system's power consumption and require heavier analysis than the remaining of the approaches. They measure *power* at a fixed frequency: from $1\,\mathrm{Hz}$[1] to $500\,\mathrm{kHz}$, on an external device. This implies cumbersome and error-prone analysis, to correlate running programs with the measured power consumptions.

**Specialized systems for energy monitoring.** Specialized systems are designed and deployed to enable fine-grained

[1] 1 Hz corresponds to one cycle per second

energy measurements. The most popular ones are Atom LEAP [24] and Spartan FPGA [21]. Such systems embed multiple sensors, plugged to several parts of the system (e.g., CPU, disks, RAM), and can perform fine-grained measurements at high frequencies. The main drawback of such systems is that they require extremely specific tooling and operating system, and hence are not easily accessible to most developers and xDSMLs end-users.

**Application level monitoring tools.** Several software and middleware applications or libraries can be queried at the software level to measure or estimate the energy consumption of a given application. Their main benefits are that they do not require any hardware tweaking, can be used on popular exploitation systems, and are accurate as they can measure the energy consumption of a given running process. The most used ones are RAPL, an Intel feature easily accessible on Unix-based system [8]; JRAPL, a Java library enabling high-level access to RAPL [16]; and POWERAPI, a middleware solution providing process-level power estimations at custom frequencies [3]. The main drawback of this approach is that using these applications during the software development lifecycle requires knowledge about dynamic analysis, and applying it in the realm of xDSMLs requires tweaking the operational semantics of the language, or the execution engine. Furthermore, the values measured might be inaccurate and influenced by the state of the system, making the approach complicated to use for developers and xDSMLs end-users.

**Model-level monitoring.** Little work has been done in the realm of xDSMLs for enabling energy-aware design and modeling. Luca Berardinelli et. al., propose an extension of the Agilla Modeling Language (AML) with an instruction triggering energy measurements of nodes in Wireless Sensors Network (WSN) [2]. The energy measured is then used for predicting and optimizing the design of the WSN, towards a better energy-awareness. This approach is platform-specific as it requires a WSN running a specialized exploitation system and only focuses on AML.

Compared to the related work, our approach is meant to be used regardless of the operating system, does not require any hardware device, does not require any knowledge about dynamic analysis from the end-user, and can be used on any xDSMLs designed with the GEMOC language workbench.

## III. APPROACH

This section presents our approach for specifying generic energy estimations and dynamically evaluating them. An implementation is available on GitHub[2]. Figure 1 proposes an overview of the approach and its three actors implied in it: the DSL Engineer, who knows the meta-programming approach and can design an executable meta-model with its operational semantic, the end-user who knows the executable meta-model (or a DSL), creates executable models conforming to it and runs them using the execution engine, and finally the Energy Estimation (EE) Specialist, who knows the EE meta-model,

[2]https://github.com/atlanmod/energy-estimation-language

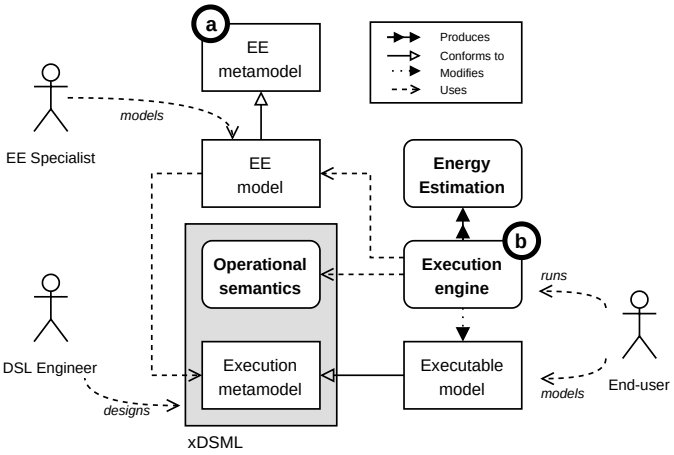


Fig. 1. Overview of the approach.

the operational semantics of the execution meta-model, and the platform on which the models are executed.

Whereas the DSL engineer and end-user actors are rather common in the realm of xDSMLs [5], the (EE) specialist is a novelty. She has enough knowledge about energy estimation and measurement to specify generic estimations and associate them with the xDSML.

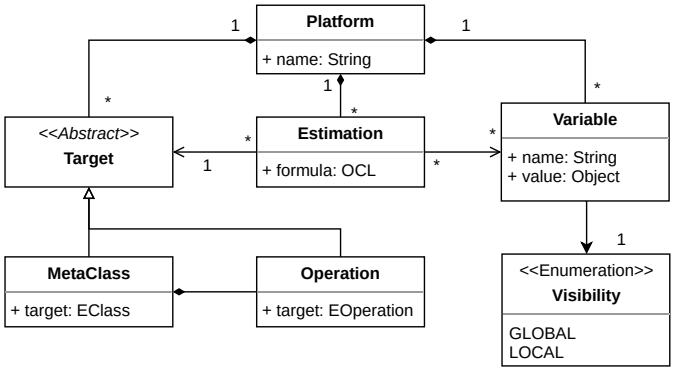


Fig. 2. Energy estimation meta-model.

**Energy estimation meta-model.** Figure 2 depicts the meta-model we propose for specifying energy calculations. The *Platform* entity is the root element of the meta-model and refers to a specific system. It contains *Variables* and *Estimations*. The former defines valued entities that are either defined once for the platform (i.e., Global) or once for each instance conforming to the targeted entity (i.e., Local). Variables can be used for defining local counters, or system attributes, according to the needs of the EE Specialist. The latter defines the formulas used for estimating the energy consumed by the executable model. These formulas are defined as OCL queries and return Double values. Each estimation refers to a *Target* entity, which is either a *MetaClass* in the execution meta-model, or an *Operation* in its operational semantic.

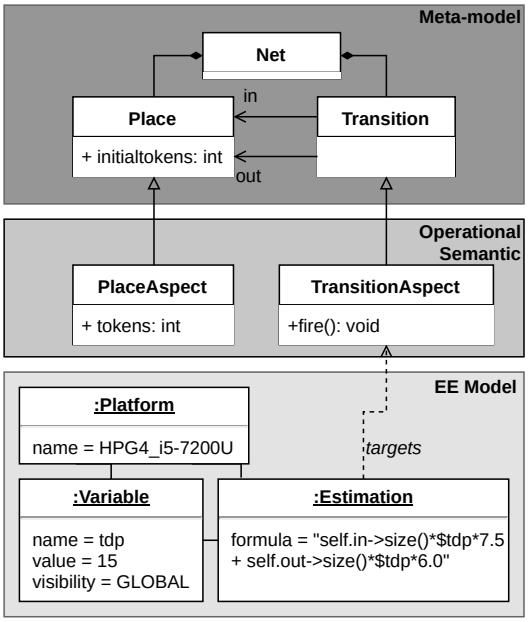


Fig. 3. Petri net xDSML with an EE model.

**Dynamic estimation.** To perform dynamic energy estimation for executable models, we extended the existing GEMOC modeling workbench [4], to accept EE models as additional input for the model execution. Furthermore, an execution listener is added to the execution engine. This execution listener is called every time a Step annotated operation [5] is executed in the operational semantics of the xDSML. There, a simple check is performed, verifying if any estimation in the EE model targets either the running operation, or a meta-class the running object is an instance of.

If an estimation targeting the running entity exists, its contained OCL Query is executed on that entity. Our current approach returns a double value, corresponding to an estimation of the energy consumed by the execution of that entity. This value can be then used for providing a graphical representation of the energy consumption of the model or stored for any future usage. Existing work on quantity specifications could be applied here, to return other types of energy estimations, such as probability distributions or measurement uncertainty [7].

**Example of an energy estimation.** To illustrate our approach, we propose a simple use-case, where a DSL Engineer designs a Petri Net meta-model and an EE Specialist defines an EE model for a specific platform (a laptop running on Ubuntu 18.04, with an Intel Core i5-7200U), as shown in Figure 3.

This EE model associates an EE formula to the `fire()` operation defined in the operational semantics of the Petri Net meta-model. This formula, defined in OCL, estimates the energy consumption of `fire()` according to the number of inputs and outputs of the Transition. A variable is defined in the EE model and used in this estimation. This variable is the Thermal Design Power (tdp) of the CPU, a system attribute used for estimating the power consumption [12], [3]. The execution engine replaces this variable by its value in the OCL formula before evaluating it, hence returning an estimation of the energy consumed by the execution of `fire()`.

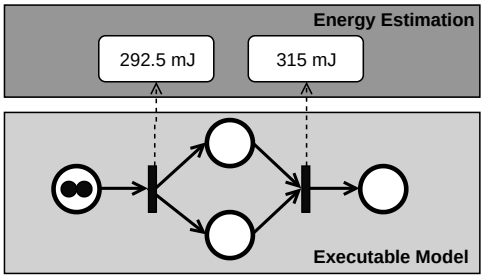


Fig. 4. Petri net model conforming to the execution meta-model.

Finally, a Petri Net is modeled by an end-user, as represented in Figure 4. The EE model and this Petri Net are given as inputs to the execution engine. The engine thus runs the model and estimates the energy consumption of the `fire()` operation for each Transition executed. The OCL formula in the EE model would evaluate the first transition energy consumption to $292.5\,\mathrm{mJ}$ and the second transition energy consumption to $315\,\mathrm{mJ}$.

## IV. Discussion

In this section, we discuss the accuracy of our approach for defining energy estimation formulas, how EE specialists can define them, and the interest of dynamic analysis.

**Estimation accuracy.** First of all, the accuracy of the estimation calculated depends on the formula specified by the EE specialist. To evaluate the accuracy of such formula, measurements with specialized tools should be performed on the running model and compared with the estimation. We consider a formula to be accurate if, with enough model executions, the energy estimation calculated converges towards the average energy consumptions measured. This can be defined as:

$$E(p) \approx \frac{1}{n} \sum_{x=0}^{n-1} m_x(p)$$

where $E(p)$ is the estimated energy consumption of entity $p$, $m_i(p)$ is the i-nth measure of the energy consumption of $p$, and $n$ the number of measurements to perform until the values obtained are stable.

**Defining energy estimation formulas.** The EE specialists have to define accurate energy estimation formulas to attach to the meta-elements of an execution meta-model. To define such formulas, the specialist needs several executable models conforming to the execution meta-model, to cover the majority of the execution scenarios. Multiple measured model executions have to be performed to obtain accurate energy measurements. EE formulas can then be defined by performing a thorough study of the operational semantics of the xDSML, and analyzing the energy measurements obtained.

This approach is similar to *eCalc*, where the energy consumption of each byte-code instructions (execution meta-elements) in the language is measured, to estimate the energy

consumption of all programs (executable models) compiling to bytecode [11].

**Static vs. Dynamic energy estimation.** Our current approach focuses on a dynamic analysis: the estimations are performed at runtime, and directly on the running entities. However, considering the Petri Net used as an example in Figure 4, the same energy estimation could have been performed statically. Performing a dynamic analysis ensures that the estimated entities are really executed. A static analysis would be either less precise (e.g., providing EE of non-executed entities) or slower (e.g., considering all the possible execution scenarios) [9]. Nonetheless, statically estimating the energy consumption of the executable models at design time could effectively help end-user taking more energy-efficient decisions when designing the executable models. We will consider this in future work.

Finally, our current EE meta-model cannot estimate some specific xDSMLs accurately. This includes model executions depending on external inputs or events and non-deterministic operational semantics. In a nutshell, all the executable models where the execution duration (and, *de facto*, energy consumption), cannot be estimated at the xDSML level. Future work will also focus on those cases.

## V. Conclusion and Future work

This paper is the first step towards a generic approach for estimating the energy consumption of any executable model. We propose a meta-model for specifying energy estimation, along with an extension of the GEMOC modeling workbench enabling the dynamic energy estimation of executable models. Our future work will empirically evaluate our approach by comparing the energy estimations of several executable models, on multiple platforms, to energy measurements obtained with specialized tooling. Furthermore, a dedicated language will be proposed to specify the measurements. Finally, more research will be performed to automatize the generation of EE formulas.

## References


[1] Anders Andrae and Tomas Edler. On global electricity usage of communication technology: trends to 2030. *Challenges*, 2015.

[2] Luca Berardinelli, Antinisca Di Marco, Stefano Pace, Luigi Pomante, and Walter Tiberti. Energy consumption analysis and design of energy-aware wsn agents in fuml. In *European Conference on Modelling Foundations and Applications*, pages 1–17. Springer, 2015.

[3] Aurélien Bourdon, Adel Noureddine, Romain Rouvoy, and Lionel Seinturier. Powerapi: A software library to monitor the energy consumed at the process-level. *ERCIM News*, 2013(92), 2013.

[4] Erwan Bousse, Thomas Degueule, Didier Vojtisek, Tanja Mayerhofer, Julien Deantoni, and Benoit Combemale. Execution framework of the gemoc studio (tool demo). In *Proceedings of the 2016 ACM SIGPLAN International Conference on Software Language Engineering*, 2016.

[5] Erwan Bousse, Dorian Leroy, Benoit Combemale, Manuel Wimmer, and Benoit Baudry. Omniscient debugging for executable dsls. *Journal of Systems and Software*, 137:261–288, 2018.

[6] David J. Brown and Charles Reams. Toward energy-efficient computing. *Queue*, 8(2):30:30–30:43, February 2010.

[7] Loli Burgueño, Tanja Mayerhofer, Manuel Wimmer, and Antonio Vallecillo. Specifying quantities in software models. *Information and Software Technology*, 113:82–97, 2019.

[8] Howard David, Eugene Gorbatov, Ulf R Hanebutte, Rahul Khanna, and Christian Le. Rapl: memory power estimation and capping. In *2010 ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED)*, pages 189–194. IEEE, 2010.

[9] Michael D Ernst. Static and dynamic analysis: Synergy and duality. In *WODA 2003: ICSE Workshop on Dynamic Analysis*, pages 24–27. New Mexico State University Portland, OR, 2003.

[10] Taher Ahmed Ghaleb. Software energy measurement at different levels of granularity. In *2019 International Conference on Computer and Information Sciences (ICCIS)*, pages 1–6. IEEE, 2019.

[11] Shuai Hao, Ding Li, William GJ Halfond, and Ramesh Govindan. Estimating android applications' cpu energy usage via bytecode profiling. In *Proceedings of the First International Workshop on Green and Sustainable Software*, pages 1–7. IEEE Press, 2012.

[12] John L Hennessy and David A Patterson. *Computer architecture: a quantitative approach*. Elsevier, 2011.

[13] Sebastian Herbert and Diana Marculescu. Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In *Proceedings of the 2007 international symposium on Low power electronics and design (ISLPED'07)*, pages 38–43. IEEE, 2007.

[14] Abram Hindle, Alex Wilson, Kent Rasmussen, E Jed Barlow, Joshua Charles Campbell, and Stephen Romansky. Greenminer: A hardware based mining software repositories software energy consumption framework. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 12–21. ACM, 2014.

[15] Jason M Hirst, Jonathan R Miller, Brent A Kaplan, and Derek D Reed. Watts up? pro ac power meter for automated energy recording, 2013.

[16] Kenan Liu, Gustavo Pinto, and Yu David Liu. Data-oriented characterization of application-level energy optimization. In *International Conference on Fundamental Approaches to Software Engineering*, pages 316–331. Springer, 2015.

[17] Adel Noureddine, Romain Rouvoy, and Lionel Seinturier. A review of energy measurement approaches. *ACM SIGOPS Operating Systems Review*, 47(3):42–49, 2013.

[18] Candy Pang, Abram Hindle, Bram Adams, and Ahmed E Hassan. What do programmers know about software energy consumption? *IEEE Software*, 33(3):83–89, 2015.

[19] Gustavo Pinto, Fernando Castor, and Yu David Liu. Mining questions about software energy consumption. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 22–31. ACM, 2014.

[20] Efraim Rotem, Alon Naveh, Avinash Ananthakrishnan, Eliezer Weissmann, and Doron Rajwan. Power-management architecture of the intel microarchitecture code-named sandy bridge. *Ieee micro*, 2012.

[21] Cagri Sahin, Furkan Cayci, Irene Lizeth Manotas Gutiérrez, James Clause, Fouad Kiamilev, Lori Pollock, and Kristina Winbladh. Initial explorations on design pattern energy usage. In *2012 First International Workshop on Green and Sustainable Software (GREENS)*. IEEE, 2012.

[22] Eric Saxe. Power-efficient software. *Queue*, 8(1):10:10–10:17, 2010.

[23] Jed Scaramella and Matthew Eastwood. Solutions for the datacenter's thermal challenges. *IDC, January*, 2007.

[24] Digvijay Singh and William J Kaiser. The atom leap platform for energy-efficient embedded computing. 2010.

[25] Molly Webb et al. Smart 2020: Enabling the low carbon economy in the information age. *The Climate Group. London*, 1(1):1–1, 2008.

[26] Qing Wu, Massound Pedram, and Xunwei Wu. Clock-gating and its application to low power design of sequential circuits. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 47(3):415–420, 2000.

[27] Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert P Dick, Zhuoqing Morley Mao, and Lei Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, 2010.