Towards Green Metrics Integration in the MEASURE Platform

Alessandra Bagnato
Softeam Research & Development Division
Paris, France
alessandra.bagnato@softeam.fr,

Jérôme Rocheteau
Institut Catholique d'Arts et Métiers
35 av. du champ de Manœuvres, Carquefou, France
jerome.rocheteau@icam.fr,

***Abstract*—Current approaches to measure software energy consumption rely on different measurement tools that are not integrated with the software development life-cycle phases. We develop a model-based design methodology in order to cope with this problem. This methodology is tooled by a processing platform and a development environment based on OMG's Structured Metrics Metamodel models. We apply the approach to three specific measures that correspond to green metrics and we explain how they fit with different software development phases.**

I. Introduction

Measuring software engineering during the software development life-cycle phases involves different tools that are used by different teams within the same company or in different ones. It lacks of global view, unified collect platform and deep analysis that could allow cross decision making in software development that may engender costs loss due to a management decision longer time.

OMG's Structured Metrics Metamodel (SMM) specification [1] defines an extensible meta-model for representing and exchanging software-related measurement information concerning software assets (designs, implementations, operations, etc). The MEASURE platform consists of a web application that allows to deploy, configure, collect, compute, store, combine and visualize measurements by execution of software measures that may be defined according to the SMM specification. Such a usage has been described in [2].

We apply the previous tooling to green computing through out green metrics design and implementation based on the EMIT power measurement tool. Three SMM measures are modelled and deployed onto the MEASURE platform.

This short paper is structured as follows: The section II presents the MEASURE platform. The section III presents the deployed tooling for embedding and processing the previous green metrics within the MEASURE platform. The section IV illustrates the methodology for reusing green metrics across software development projects that are drawn out from open source project unit test suites.

II. MEASURE platform

The MEASURE Platform [3], [4], [5] is a tool dedicated to measure, analyse, and visualise the metrics to extract and show information of the software engineering processes. Implement the tools for automatically measure software engineering processes during the whole software lifecycle by executing measures defined in SMM standard and extracted from a catalogue of formal and platform-independent measurements.

1) Provide methodologies and tools which allow measure tools provider to develop a catalogue of formal and platform-independent measure.
2) Implement storage solution dedicated to measurements resulting of measure execution in big data context.
3) Implement visualization tools to expose the extracted results in an easy-readable fashion, so allowing a quick understanding of the situation and the possible actions that can be taken to improve the diverse stages of the software lifecycle.
4) Implement an extended API which allows external analysis tools to access to collected measurement and to contextual data related the measurement collect process itself.

The platform activity is organised around its ability to collect measurement by executing measures defined by the SMM standard. SMM measures are auto-executable component, implemented externally, which can be interrogated by the platform to collect measurements. The Measure platform provides services to host, configure and collect measures, storing measurement, present and analyse them. These measures are first defined in SMM standard using the Modelio modelling tool [6] and its extension dedicated to SMM modelling [7]. They are packaged under an executable format as Measure Definition. Next, measures are registered and stored on Measure platform using the dedicated REST service or the Web user interface. In order to initiate the collect of measurement, the next step consists on defining instance of measure based on measure definitions. A measure represents a generic data collection algorithm that has to be instantiated and configured to be applied on a specific context. For example, a measure which collects data related to an SonarQube repository must be configured by the URL of this repository. Next the Measure Platform can start collecting measurement (data resulting of the execution of an instantiated measure). Direct measures collect data in physical world while the Derived Measures are calculated using previously collected measurement as input. Collected measurements are stored on a No SQL designed to be able to process a very large amount of data. To collect

measurements, the direct measures can delegate the collect work to existing Measure tool. Finally, stored measurements are presented directly to the end user following a business structured way by the Decision-making platform, a web application which allows organising measures based on projects / software development phases and display its under various forms of charts. The measurements can also be processed by analysis tools to present consolidated results [3], [4], [5]. The Measure platform [3], [4], [5] provides services to host, configure and collect measures, storing measurement, present and analyse them. These measures are first defined in SMM standard using the Modelio modelling tool and its extension dedicated to SMM modelling. They are packaged under an executable format as Measure Definition . Next, measures are registered and stored on Measure platform using the dedicated REST service or the Web user interface. In order to initiate the collect of measurement, the next step consists on defining instance of measure based on measure definitions. A measure represents a generic data collection algorithm that has to be instantiated and configured to be applied on a specific context. For example, a measure which collects data related to an SVN repository must be configured by the URL of this repository. Challenges for MEASURE:

1) Define green metrics and develop methods and tools for automated, precise, and unbiased measurement.
2) Deploy and evaluate a set of dedicated metrics,
3) Deploy an infrastructure able to processing a large volume of Measures provided by big models repository,
4) Integrate the Measure platform with existing quality and project management tools already deploied and used in existing development environment,
5) Integrating green metric tools into development environments and processes.
6) Offer a synthetic view resulting of the analysis several measures allowing a manager to extract key information related to several development process.
7) Used advanced analysis techniques like forecasting and automated recommendation system to help us to exploit the monitoring data and define mitigations actions to issues detected with the help of the Measure platform.

A. Modelio

Modelio [6] is a Modeling Tool developed by Softeam since January 2009 supporting and integrating all the latest major modelling or methodology standards. Modelio is first and foremost a UML modelling environment, supporting a wide range of models and diagrams, and providing model assistance and consistency checking features. BPMN support is integrated with UML with Modelio combining Business Process Modelling and Notation (BPMN) and UML support in one tool, with dedicated diagrams to support business process modelling. The Modelio SMM Module [4] is the first building block of the MEASURE project tool chain. The Modelio modeling tool, enabled with the SMM Module is based on Modelio's open source distribution to allow the specification of all the MEASURE's metrics for measuring activities in different phases of software engineering. Within the MEASURE project the metrics modelled are structured according to five software engineering phases: specification, design, implementation, testing and production. These metrics define measures then deployed on the MEASURE platform for measurement collect and analysis. The figure 2 illustrates the Modelio Environement and the MEASURE library covering all the phases of the software development lifecycle used in the MEASURE Project [3], [4]. We concentrate in this paper on the software engineering production phase and in particular on green metrics.

III. Green Metrics in the MEASURE Platform

Three metrics are integrated into the MEASURE platform about software energy consumption. The first metrics corresponds to the energy efficiency of a software artifact with respect to another one. It is defined as a SMM ratio measure [1, §11.8] between two other measures and consists of a positive version of the energy waste rate in [8]. This metrics is defined on the top of the two other metrics. The second one corresponds to the energy consumption of a software artifact. it is defined as a SMM collective measure [1, §11.2] with a custom accumulator that corresponds to the trapezoidal rule. The figure 3 illustrates the Energy measure. In fact, energy is computed from power measurements during a given time-span. Indeed, energy measures are related to a list of power measures which are defined as SMM direct measures [1, §11.5] i.e. whose measurements are directly provided by external tools. Every SMM power measures are defined by a float value that corresponds to the power. The figure 4 illustrates the Power and TimeStamp measure. Such measures are also time-stamped such that it enables to compute the trapezoidal rule of an list of power measures ordered by their time-stamps. The SMM specification makes possible to relate measures to software artifact by the means of the measurand attribute of SMM measures [1, §7.4]. It corresponds to any element of a MOF model [9] and then allows a model-based approach for measuring software and software engineering features.

The tool chain that provides support to the previous SMM measures consists in several collaborative tools around the MEASURE platform:

1) It starts with power-meters that broadcast measurements for electric features – such as current (intensity), voltage (tension), power, power factor, etc – to corresponding topics on a messaging-queue broker.
2) The broker forwards such messages to a data processing system called EMIT that stores such messages and exposes them throughout HTTP web services.
3) The SMM power measure integrated in the MEASURE platform continuouslyretrieves and stores messages that only corresponds to power measurements of a given power-meter thanks to the previous services. Instances of this SMM power measure can be configured in setting EMIT server location and in selecting the corresponding topics. It retrieves all the messages from the last retrieval time-stamp to the current time-stamp.

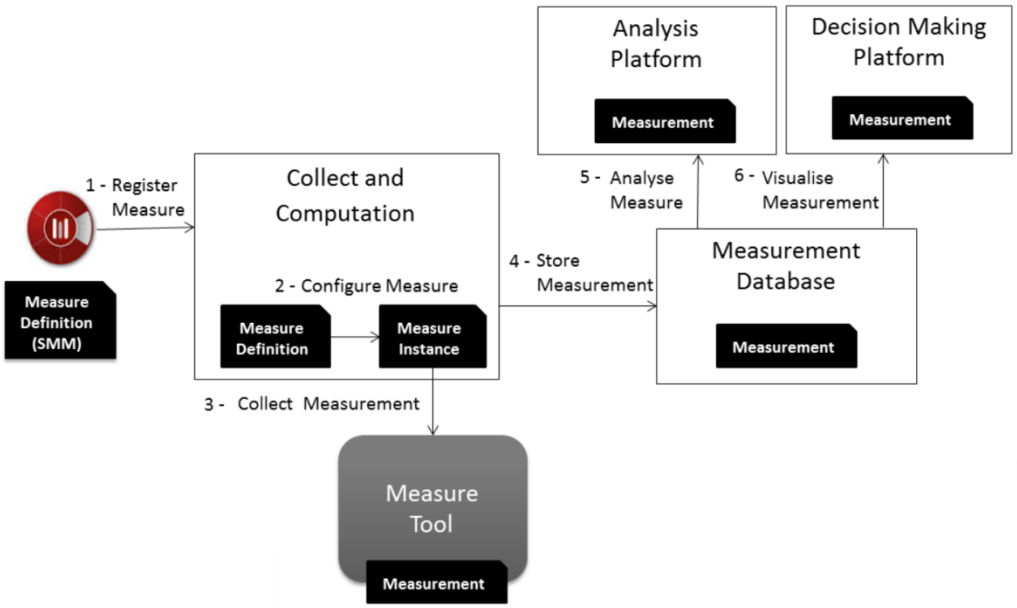

Fig. 1. MEASURE Platform Measurements and Analysis Tools

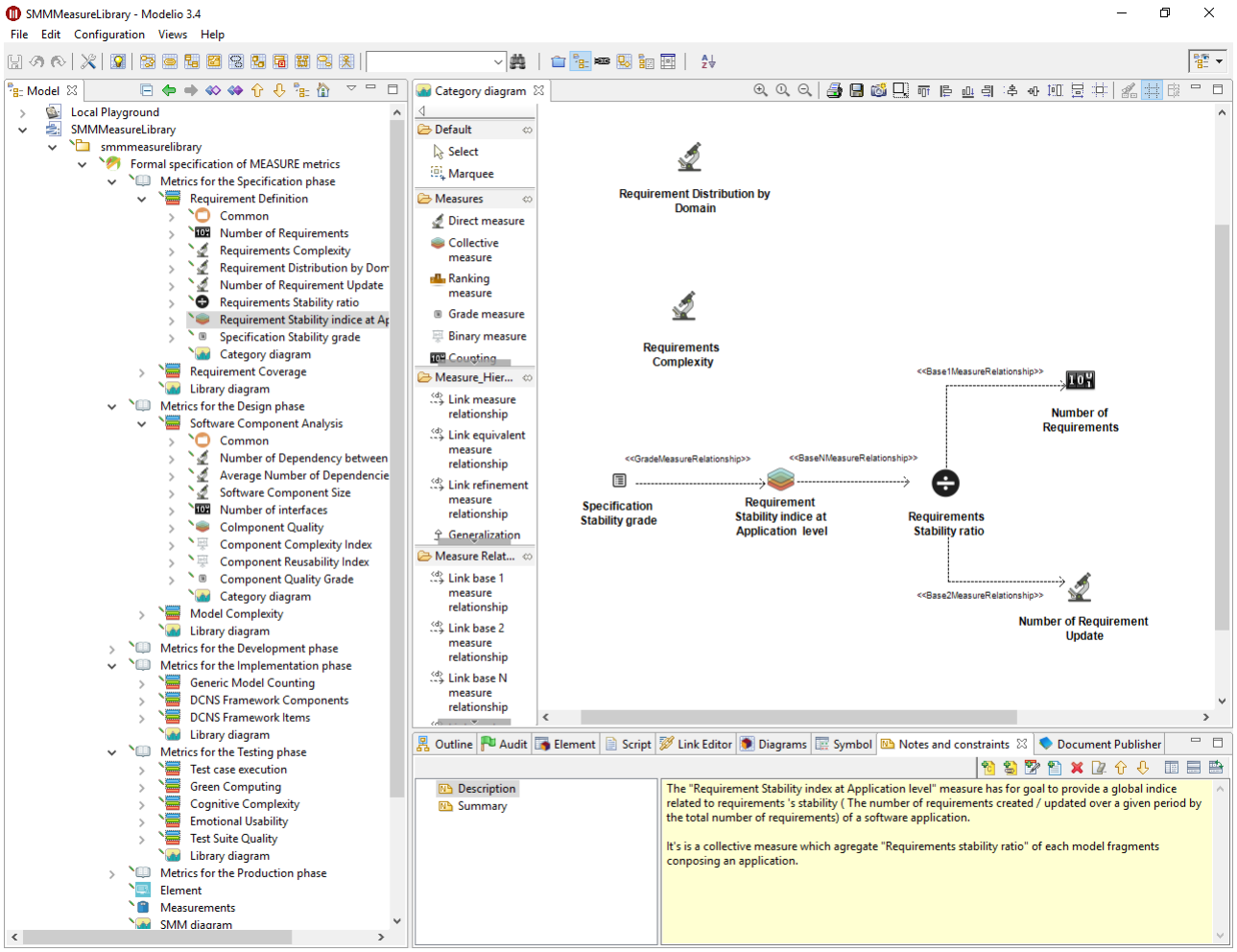

Fig. 2. Modelio SMM Modeling Environment

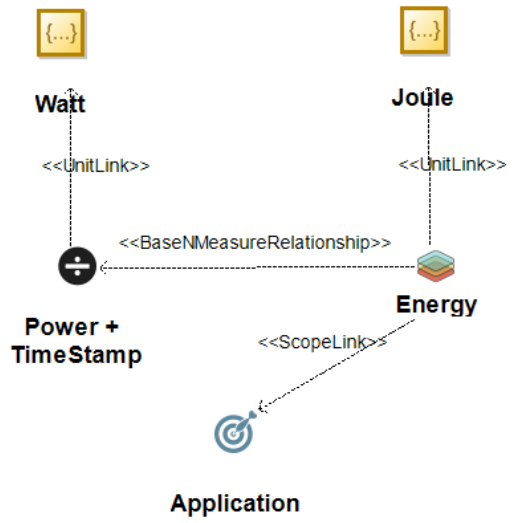

Fig. 3. Modelio's Measure Diagram Power TimeStamp and Energy Measure

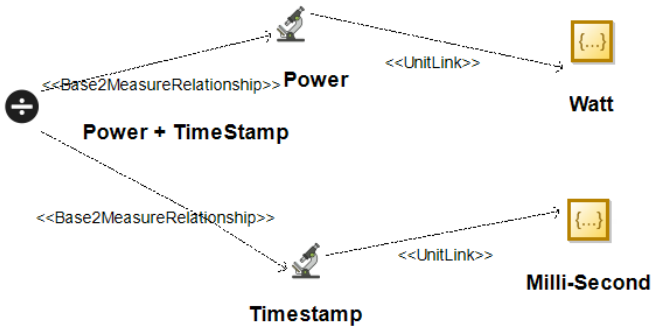

Fig. 4. Diagram Power & TimeStamp Measure Diagram in SMM

4) The SMM energy measure integrated in the MEASURE platform continuously computes the energy consumption from power measurements during a given time-span. Instances of this SMM energy measure can be configured in setting the duration in milliseconds for computing the energy consumption and in selecting the underlying SMM power measure.
5) The SMM energy-efficiency measure integrated in the MEASURE platform continuously relates energy measures whose measurands are the same MOF element. Instances of this SMM energy-efficiency measure can be configured in setting the qualified name of this element.

The figure 5 illustrates the data-flow between all these instruments, tools and platforms.

IV. Model-Based Methodology

This set of tools around the MEASURE platform enables energy efficiency to be taken into account during several software development phases. For instance, a basic use case consists in monitoring energy consumption and/or power peaks at run-time during software production phase. A second use case consists in comparing energy consumption of similar software components (same functionality same interface, same contract, etc) while testing i.e. in evaluating their energy efficiency one to each other during the validation phase. A third use case would consist in building a knowledge base about software energy consumption in order to help developers at earlier phases (design or implementation). Such a knowledge base would make possible to build static analysis methods on source code or models such that developers could improve their software without feedback from testing or monitoring.

On the one hand, the last use case is the most difficult to realize according to the lack of knowledge in the domain of software energy consumption and the reliability of power measurements over programming language primitive statements. On the other hand, the first use case is the most easy to realize as many tools already provide measurements and analysis results that fulfill it. We are focusing on the second use case as it corresponds to the combination of the MEASURE platform (model-based and metrics-oriented software engineering) and the set of EMIT tools (power measurements and energy analysis). In fact, it allows developers to improve their source code throughout their knowledge of the differences between two revisions of a same unit test. For example, if the same unit test consumes more (resp. less) energy from a revision $r1$ to a revision $r2$, this means that the source code involved in this unit test is responsible for this increase (resp. decrease) of energy consumption. In fact, we assume that the same unit test in different revisions assess the same functionality uniformly.

Such an approach about unit test energy consumption and efficiency requires to deploy a twofold continuous integration server both on the system under test and on another computer. The system under test will compile the source code and execute the unit tests whereas the other server will, on the one hand, command the system under test to execute a given unit test and, on the other hand, retrieve from the EMIT tool the time-stamps of the unit test execution beginning and end. Then energy efficiency can be computed by the means of the energy analysis tool and integrated backwards into the MEASURE platform in order to provide reports. Such an architecture and a process closely correspond to those presented in [10, §4].

V. Conclusion and Future Work

Software energy consumption has in fact mainly been studied with respect to factors such as execution duration, CPU load, RAM usage, network bandwidth, disk access, etc. The MEASURE platform makes possible to easily relate software energy consumption with other software metrics thanks to its SMM measure integration support. However, the underlying tooling architecture that makes such an integration that easy are itself rather complex. In fact, this architecture relies on dispatched tools and services that provide, collect and analyze measurements. In the future months we plan to run experiments over software source code repositories. The planned experiments will aim at providing examples of the energy efficiency measure between the same test units from different source code revisions.

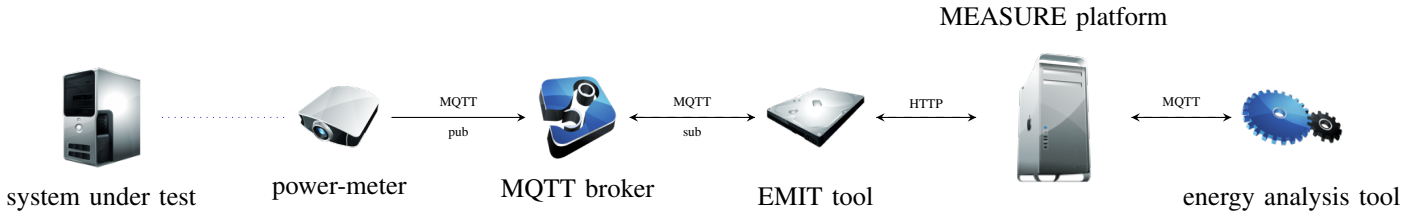

Fig. 5. Tooling based on the MEASURE Platform from Power Measurements to Energy Analysis

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Measure name="EmitPowerMeasure" type="DIRECT" category="Emit" provider="Measure.org">
    <description>Power in watt provided by EMIT powermeters.</description>
    <scopeProperties name="hostname" type="STRING" defaultValue="app.icam.fr"></scopeProperties>
    <scopeProperties name="port" type="INTEGER" defaultValue="80"></scopeProperties>
    <scopeProperties name="protocol" type="STRING" defaultValue="http"></scopeProperties>
    <scopeProperties name="toolname" type="STRING" defaultValue="emit"></scopeProperties>
    <scopeProperties name="username" type="STRING" defaultValue="measure@emit.icam.fr"></scopeProperties>
    <scopeProperties name="password" type="STRING" defaultValue="**********"></scopeProperties>
    <scopeProperties name="topic" type="STRING" defaultValue="peaktech/power"></scopeProperties>
    <unit name="EmitPowerMeasurement">
        <fields fieldName="topic" fieldType="u_text"/>
        <fields fieldName="issued" fieldType="u_long"/>
        <fields fieldName="value" fieldType="u_float"/>
    </unit>
</Measure>
```

Fig. 6. Power Measure XML Declaration

```java
    @Override
    public List<IMeasurement> getMeasurement() throws Exception {
            EmitClient client = this.getClient();
            client.setUp();
            String topic = this.getTopic();
            Long issued = this.getIssued();
            List<EmitPowerMessage> messages = client.getMessages(topic, issued);
            List<IMeasurement> measurements = new ArrayList<IMeasurement>(messages.size());
            for (EmitPowerMessage message : messages) {
                    EmitPowerMeasurement measurement = new EmitPowerMeasurement(message);
                    measurements.add(measurement);
                    if (last < message.getIssued()) {
                            last = message.getIssued();
                    }
            }
            client.tearDown();
            return measurements;
    }
```

Fig. 7. Power Measure Java Implementation

VI. Acknowledgment

The research presented in this extended abstract is funded by the ITEA3 Project no. 14009 called MEASURE (1st December 2015 and running till 31st August 2019).

References

[1] *Structured Metrics Metamodel*, Object Management Group, mar 2018. [Online]. Available: https://www.omg.org/spec/SMM/1.2/PDF

[2] A. Bagnato, M. A. Almeida Da Silva, A. Abhervé, J. Rocheteau, C.-L. Pihery, and P. Mabit, "Measuring Green Software Engineering in the MEASURE ITEA3 Project," in *MeGSuS 2016 - Proceedings of the 3rd International Workshop on Measurement and Metrics for Green and Sustainable Software Systems, Ciudad Real, Spain, 7 September, 2016*, 2016, pp. 33–42.

[3] Softeam R&D, "MEASURE project website," http://measure.softeam-rd.eu/, Oct. 2017, last accessed on 2018-11-01.

[4] A. Abherve, A. Bagnato, A. Stefanescu, and A. Baars, "Github project for the MEASURE platform," https://github.com/ITEA3-Measure/MeasurePlatform/graphs/contributors, Sep. 2017, last accessed on 2018-11-01.

[5] A. Abherve, "Github project for the SMM Measure API library," https://github.com/ITEA3-Measure/SMMMeasureApi, Aug. 2017, last accessed on 2018-11-01.

[6] Softeam , "Modelio project website," https://www.modelio.org/, Mar. 2017, last accessed on 2018-11-01.

[7] Object Management Group, "The Software Metrics Meta-Model Specification 1.1.1," http://www.omg.org/spec/SMM/1.1.1/, Apr. 2016, last accessed on 2018-11-01.

[8] J. Rocheteau, "Energy Wasting Rate as a Metrics for Green Computing and Static Analysis," in *MeGSuS 2015 - Proceedings of the 2nd International Workshop on Measurement and Metrics for Green and Sustainable Software Systems, Krakow, Poland, 5 October, 2015*, 2015.

[9] *Meta Object Facility*, Object Management Group, nov 2016. [Online]. Available: https://www.omg.org/spec/MOF/2.5.1/PDF

[10] E. Kern, L. M. Hilty, A. Guldner, Y. V. Maksimov, A. Filler, J. Gröger, and S. Naumann, "Sustainable software products—towards assessment criteria for resource and energy efficiency," *Future Generation Computer Systems*, vol. 86, pp. 199–210, 2018.

[11] J. Carver, B. Penzenstadler, and A. Serebrenik, "Software Analysis, Evolution, and Reengineering, and ICT Sustainability," *IEEE Software*, vol. 35, no. 4, 2018.

[12] M. Rodriguez-Cancio, B. Combemale, and B. Baudry, "Automatic Microbenchmark Generation to Prevent Dead Code Elimination and Constant Folding," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, 2016, pp. 132–143.

[13] M. Rodriguez-Cancio, B. Combemale, and B. Baudry, Eds., *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*. ACM, 2016.

[14] C. Thompson, D. C. Schmidt, H. A. Turner, and J. White, "Analyzing mobile application software power consumption via model-driven engineering," in *PECCS 2011 - Proceedings of the 1st International Conference on Pervasive and Embedded Computing and Communication Systems, Vilamoura, Algarve, Portugal, 5-7 March, 2011*, 2011, pp. 101–113.

[15] C. Benavente-Peces and J. Filipe, Eds., *PECCS 2011 - Proceedings of the 1st International Conference on Pervasive and Embedded Computing and Communication Systems, Vilamoura, Algarve, Portugal, 5-7 March, 2011*. SciTePress, 2011.

[16] J. Rocheteau, V. Gaillard, and L. Belhaj, "How Green Are Java Best Coding Practices?" in *SMARTGREENS 2014 - Proceedings of the 3rd International Conference on Smart Grids and Green IT Systems, Barcelona, Spain, 3-4 April, 2014*, 2014, pp. 235–246.