CEUR-WS.org/Vol-1708
urn:nbn:de:0074-1708-1

Copyright © 2016 for the individual papers by the papers' authors. Copying permitted for private and academic purposes. This volume is published and copyrighted by its editors.

MeGSuS 2016
Measurement and Metrics for Green and Sustainable Software Systems

Proceedings of the 3rd International Workshop on Measurement and Metrics for Green and Sustainable Software Systems co-located with 10th International Symposium on Empirical Software Engineering and Measurement (ESEM 2016)

Ciudad Real, Spain, September 7, 2016.

Edited by

Nelly Condori-Fernández, Vrije Universiteit Amsterdam, Netherlands
Giuseppe Procaccianti, Vrije Universiteit Amsterdam, Netherlands
Coral Calero, University of Castilla-La Mancha, Spain
Alessandra Bagnato, SOFTEAM, France

Table of Contents

- Preface from the organizers

- Green Indexes Used in CAST to Measure the Energy Consumption in Code (Invited Talk) 2-3
  Marco Bessi
- Indicators for Green in IT Audits: A Systematic Mapping Study 4-12
  J. David Patón-Romero, *Mario Piattini*
- A Learning based approach for Green Software Measurements 13-22
  Sarah Dahab, *Stephane Maag*, *Alessandra Bagnato*, *Marcos Aurélio Almeida Da Silva*
- An Effort Allocation Method to Optimal Code Sanitization for Quality-Aware Energy Efficiency Improvement 23-32
  Roberto Pietrantuono, *Gabriella Carrozza*, *Stefano Russo*, *Marco Bessi*
- Measuring Green Software Engineering In the MEASURE ITEA 3 Project 33-42
  Alessandra Bagnato, *Marcos Aurélio Almeida Da Silva*, *Antonin Abherve*, *Jerome Rocheteau*, *Claire-Lise Pihery*, *Pierre Mabit*
- How Sustainable are Model Software Artifacts in the Context of Model Driven Software Engineering 43-52
  Damiano Torre, *Coral Calero*

We offer a BibTeX file for citing papers of this workshop from LaTeX.

2016-10-14: submitted by Giuseppe Procaccianti, metadata incl. bibliographic data published under Creative Commons CC0
2016-10-15: published on CEUR-WS.org |valid HTML5|

How sustainable are model software artifacts in the context of Model Driven Software Engineering

Damiano Torre[§,¶], *Coral Calero*[§]

[¶] *Carleton University, Department of Systems and Computer Engineering,*
Software Quality Engineering Laboratory
1125 Colonel By Drive, Ottawa, ON K1S5B6, Canada

[§] *University of Castilla-La Mancha, Department of Technologies and Information Systems,*
ALARCOS Research Group,
Paseo de la Universidad, 4 13071 Ciudad Real, Spain

dctorre@sce.carleton.ca, coral.calero@uclm.es

Abstract*—Today, software is pervasive in our daily lives and its sustainability and environmental impact have become major factors in the development of software systems. Due to increasing of software complexity, it is widely acknowledged that it is impossible to test every aspect of system software. One method of increasing understanding between the software developer and the customer, of deepening the understanding of how software works, and ultimately of reducing the complexity of software, is through the use of models. Model driven software engineering (MDSE) is an established approach for developing complex software systems. This work presents a first attempt at reshaping existing basic dimensions for software sustainability in terms of MDSE. Moreover, we describe 8 quality aspects and their relationships with each other. We also present a set of guidelines to identify the 8 quality aspects, and the paths to measure sustainability in terms of MDSE. Finally, we describe an example with a set of hypothetical scenarios where we show how our quality aspects could be helpful to measure and analyze the degree of sustainability in the context of MDSE.*

Keywords— Sustainability, Green, Model Driven Software Engineering, MDSE, MDSE Sustainability, MDE, Model Sustainability, Green IN models.

I. INTRODUCTION

The latest measurement (June 2016) of the global concentration of CO^2 in the atmosphere (the primary driver of contemporary climate change) has reached 404.48 parts per million (ppm) (Fig. 1), a first in recorded history [3]. According to Charles Miller (Principal investigator of the Carbon in Arctic Reservoirs Vulnerability Experiment mission [4]) [5], current atmospheric CO^2 values are more than 100 ppm higher than at any time in the last one million years and maybe even higher than any time in the last 25 million years. These increases in atmospheric CO^2 are causing real, significant changes in the Earth system at this very moment, not in some distant future, and these will continue to be felt for centuries to come.

Unless serious actions are taken immediately, we risk the next threshold being the "point of no return", when no amount of cutbacks on greenhouse gas emissions will save us from potentially catastrophic global warming.

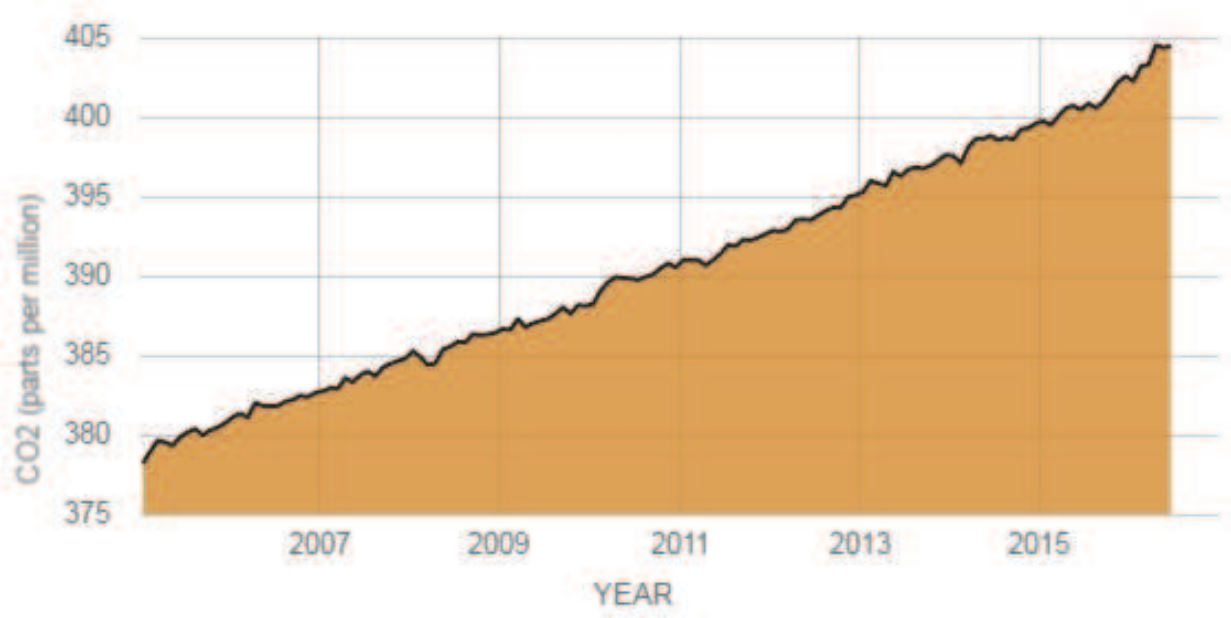

Fig. 1. Direct Measurements: 2005-Present. Credit: NOAA [2]

Having said that, it is not surprising that several initiatives across different fields that try to contribute to a more sustainable society, are gaining importance worldwide. In this paper we focus on the sustainability in the area of object oriented software engineering [6], and specifically in the context of the model driven software engineering [7].

Today, software is pervasive in our daily lives. Almost any technical item comprises a little computer with a corresponding software [8]. With the usage of software, the development of software has also become much more widespread. Today software development is taught in school as any other subject like mathematics or geography [8]. According to [9], in the future, everyone will be a programmer for 15 minutes. For this reason, the topic of software sustainability, although still in its early stages, is a very important research topic. In fact, as [10] remark, sustainability has recently started to receive recognition as a quality objective for software systems, whereas other important qualities like software security has been addressed from the 1980s onward and software safety since the early 1990s. The initial goal of this work is to raise awareness on the part of all those involved with software, such as the companies that develop software.

An organization may be considered sustainable depending, among other issues, on the sustainability of its business processes, services and information systems (IS). IS sustainability depends, in turn, on the sustainability of ICT (Information and Communications Technology) sustainability, which is composed of communications and IT (Information Technology) systems, whose sustainability is determined by the sustainability of their hardware and software components (see Fig. 2). A complete compilation of the definitions of the organization sustainability levels presented in Fig. 2 can be found in [1].

According to [11], when pursuing strategic software sustainability, the environmental impact of software is classified in two categories:

- Software that helps tackle environmental issues (using video conferences, software for car sharing, etc.), which is called green "BY" software;
- Software itself that are often responsible for major environmental degradation (amounts of energy consumed by software engineering processes used to manufacture the software), this is called green "IN" software.

In this paper we focus on the latter green “IN” software. In our case we focus on green “IN” models. This means to develop models which later will be transformed in software, in a way we can protect and reduce the depletion of natural resources.

According to [12], software is becoming increasingly complex, in fact, it is widely acknowledged that it is impossible to test every aspect of system software or application programs before release. One method of increasing understanding between the software developer and the customer, of deepening the understanding of how software works, and ultimately of reducing the complexity of software, is done through the use of models [13].

Model driven software engineering (MDSE) is an established approach for developing complex software systems and has been adopted successfully in automotive, aerospace and telecommunications industries, as well as in business information systems [14]. MDSE can be defined as a methodology for applying the advantages of modeling to software engineering activities [7] that includes various model-driven approaches to software development, including model-

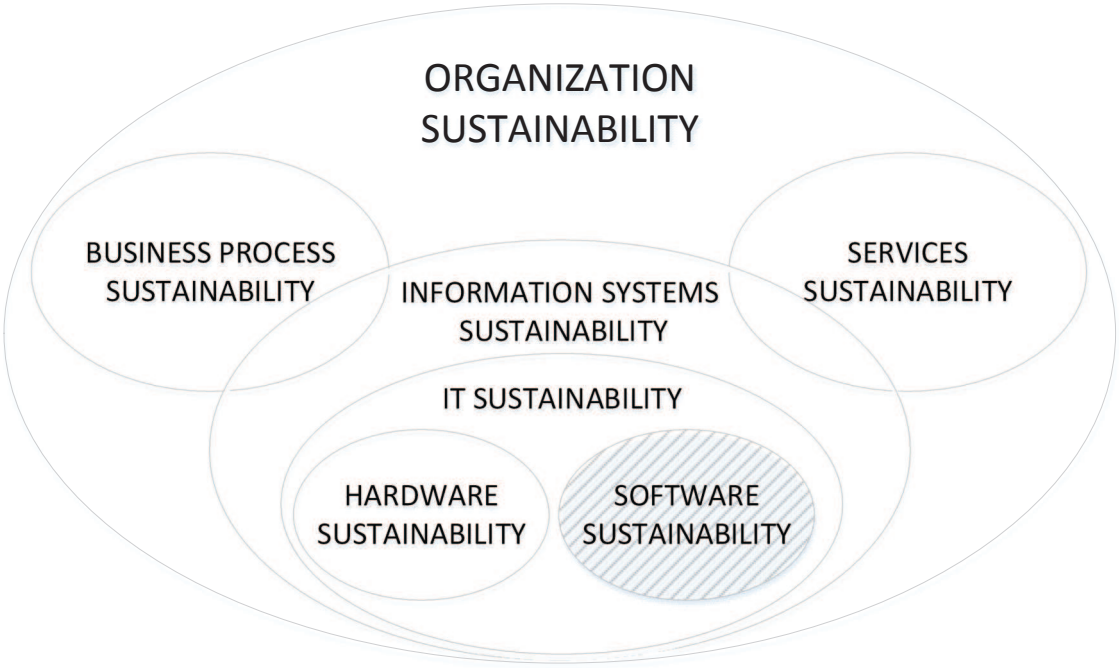

Fig. 2. Sustainability levels [1]

driven development and model-driven architecture. Modeling languages represent one of the main ingredients of MDSE. They may consist of graphical representations (e.g. Unified Modeling Language (UML) [15]), textual specifications (e.g. the Backus Normal Form), or both [7].

With regard to the dimensions and quality aspects used to measure degrees of software sustainability, given the sudden burst of attention paid to the topic in recent years, the literature proves to be rather chaotic and sometimes inconsistent. In an attempt to improve the epistemology in the area of sustainability in the context of MDSE, we present a proposal that reshapes existing dimensions for software sustainability in terms of MDSE, and add 8 quality aspects and their relationships with each other.

Later, we present a set of guidelines to obtain the 8 quality aspects, and the paths to measure sustainability in terms of MDSE. Finally, we describe an example with a set of hypothetical scenarios where we show how our quality aspects could be helpful to measure and analyze the degree of sustainability in the context of MDSE.

The remainder of this document is structured as followed: Section 2 provides the summary of the related work; Section 3 describes the background along with the reshaping of the basic dimensions for MDSE sustainability; Section 4 presents the quality aspects for MDSE sustainability; Section 5 provides the guidelines to identify the quality aspects and the paths to measure MDSE sustainability. A small example of how the quality aspects could be used in the context of MDSE sustainability is presented in Section 6; finally, our conclusions and outlines of future work are shown in Section 7.

II. Related Work

We are not aware of any proposal that is specifically dedicated to the issue of describing sustainable dimensions and quality aspects in the context of MDSE (Green "IN" models). We are aware of the workshop entitled "1st Modelling For Sustainability Workshop" [16] which took place in 2016. This workshop and the proposals that were presented focused on different goals than ours. In fact, they used a case study to orient the workshop towards something more concrete to allow attendees to create actual models. The options for the case study considered by the workshop organizers were a) a smart city project related to transportation, b) an electric vehicles project, and c) a project related to sustainable agriculture.

The "1st Modelling For Sustainability Workshop" focused on Green "BY" models, which means to use models to help tackle environmental issues. Rather, in this paper we specifically focus on Green "IN" models.

III. Background

Sustainability is a widely-used term, several definitions of which can be found in literature [17]. The most frequently adopted is found in the United Nations (UN)'s Brundtland report, which defines sustainable development as the ability to "meet the needs of the present without compromising the ability of future generations to satisfy their own needs" [18]. According to the UN, sustainable development needs to satisfy the requirements of three dimensions, which are society, the economy, and the environment. With regards the UN's Brundtland report, economic development is afforded a priority over the environment. If the debate truly was about environmental and social sustainability, surely one would expect the relationship to be reversed, on the assumption that economic development proceeds within the constraints and limits of the biophysical environment, meaning the reshaping of markets and production processes to fit the logic of nature [19]. For this reason, in this paper, we decided to simply report the definition of the economic aspect without adopting it.

A. Reshaping basic dimensions for MDSE Sustainability

Some authors consider that the dimensions of the UN's Brundtland report are not sufficient when applied to Software Sustainability.

For example, [20] propose five dimensions of sustainability for software systems: Individual, Social, Economic, Environmental and Technical sustainability.

At the second GREENS workshop at ICSE 2013 [21], two different working groups, which were created during the workshop, proposed different sets of sustainability dimensions. One of them identified: social, environmental, economic, and technical sustainability. The other working group produced a model containing four views of sustainability: business, technical, environmental and social.

Recently, in [22], four dimensions were included in a framework for software-quality sustainability requirements: social, environmental, technical, and economic.

In Fig. 3, we merge and reshape, in terms of MDSE sustainability, the definitions cited above; in Fig. 3 we identify three main dimensions: green/environmental (that contain the sub-dimensions Green "IN" and Green "By"), human (that contain the sub-dimensions individual and community), and economic.

As presented in Fig. 3, the dimensions that describe MDSE sustainability are as followed:

a) Green/Environmental sustainability: how the development, maintenance and use of model software artifacts affect energy consumption and the usage of other resources. In other words, it is the process of developing model software artifacts with regard to protect and reduce the depletion of natural resources.

   a.1. Green "IN";

   a.2. Green "BY";

   The definitions for Green IN and BY are already presented in Section 1. As we already said, this paper is focused on Green "IN" models.

b) Human sustainability: how the development and maintenance affect the sociological and psychological aspects in the context of MDSE.

   b.1. Individual sustainability: refers to the human development (e.g. education), personal health and well-being in the context of MDSE. This can be seen

as the amount of working time that should be spent modeling, in a way that enables developers to be satisfied with their job over a long period of time.

b.2. Community sustainability, also called social sustainability [20, 21], in the context of MDSE, aims at developing model software artifacts promoting social equality, inclusiveness, and connectivity across the communities.

c) Economic sustainability: this aims to maintain assets. Assets include not only capital but also added value. This requires a definition of income as the "amount one can consume during a period and still be as well off at the end of the period, as it devolves on consuming added value (interest), rather than capital" [23].

We do not consider technical sustainability because it does not map directly onto any of the resources needed in the context of MDSE.

IV. Quality Aspects for MDSE Sustainability

According to the MDSE sustainability dimensions described in Section 3, in this section we present a set of quality aspects that can help to measure how sustainable are model software artifacts in the context of MDSE. In TABLE I. we introduce the quality aspects and their acronyms that will be explained in details in this section.

TABLE I. Quality Aspects

Acronym	Quality Aspects
EE	Energy efficiency
TE	Time effectiveness
PRR	Portability and Reuse-Recycling
ST	Standardability
MQC	Model Quality Checkability
RP	Reliability and Perdurability
TR	Transformability
SSD	Software System Dimension

The quality aspects for MDSE sustainability are as followed:

- Energy efficiency (EE): developing model software artifacts that require less energy. Improving EE may affect:
  - (a.1) because it would reduce natural resources such as electrical energy.
- Time effectiveness (TE): time spent to model is a valuable resource that can be minimized along the MDSE activities. Improving TE may affect:
  - (EE) because by spending less time modeling, we would require less electrical energy.
- Portability and Reuse-Recycling (PRR): model software artifacts where it is easy to import/export other model software artifacts. Having high degree of PRR may affect:

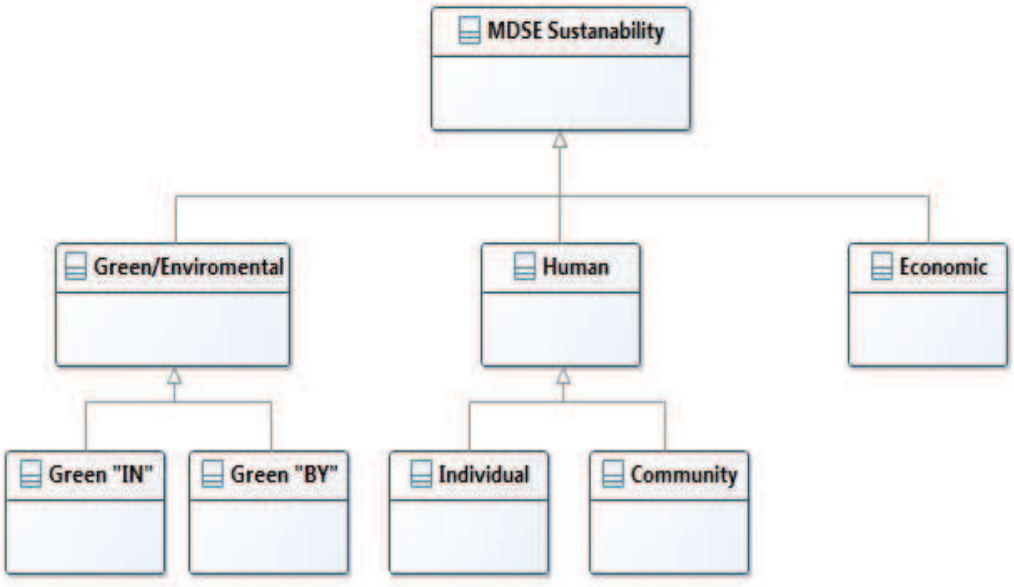

Fig. 3. *MDSE Sustainability Schema*

- (b.1) because it could help to reduce working time by reusing model software artifacts;
- (RP) because it allows that a model software artifact can be used over a long period.

- Standardability (ST): model software artifacts that rely to a standard specification (i.e. the UML metamodel [15]), and maximizes compatibility, consistency, correctness, interoperability, safety, repeatability, commoditization of the MDSE activities. Using a ST may affect:
  - (b.1) because it could reduce/increase working time;
  - (b.2) the presence of a standard could contribute the inclusiveness, and connectivity in a community that "speak" the same model language;
  - (EE) because by adopting or not a model standard, it could require less or more electrical energy.
  - (TE) because it could increase/reduce the time spent modelling;
  - (PRR) because it could improve the process of importing/exporting other model software artifacts;
  - (RP) because it could increase the degree to which a model software artifact can be used over a long period;
  - (TR) because it could help to make the transformation of the models to the code more/less easy;
  - (MQC) because it could help the process of checking the model quality.
- Model Quality Checkability (MQC): is the degree of how easy it is to check the qualities of a model software artifact. For model qualities we refer the model characteristics such as correctness, consistency, completeness, maintainability, etc. For instance, the case of model consistency: since various diagrams in a model typically describe different views of one, and only one, software system under development, they strongly depend on each other and hence need to be consistent with one another. Nevertheless, inconsistencies between different diagrams may arise. Such inconsistencies may be a source of faults in software systems down the road [24]. Having model software artifact where it is easy to check MQC may affect:
  - (TR) because a poor quality of models may be a source of faults in the software systems (code).
- Reliability and Perdurability (RP): is the degree to which a model software artifact can be used over a long period, being, therefore, easy to modify, adapt and reuse. Having model software artifact with a high degree of RP may affect:
  - (TE) because it could reduce the time spent to model.
- Transformability (TR): refers to how easy it is to transform (automatically, semi-automatically, or manually) a model software artifact into an executable code. Having model software artifact easy to transform to code (TR), may affect:

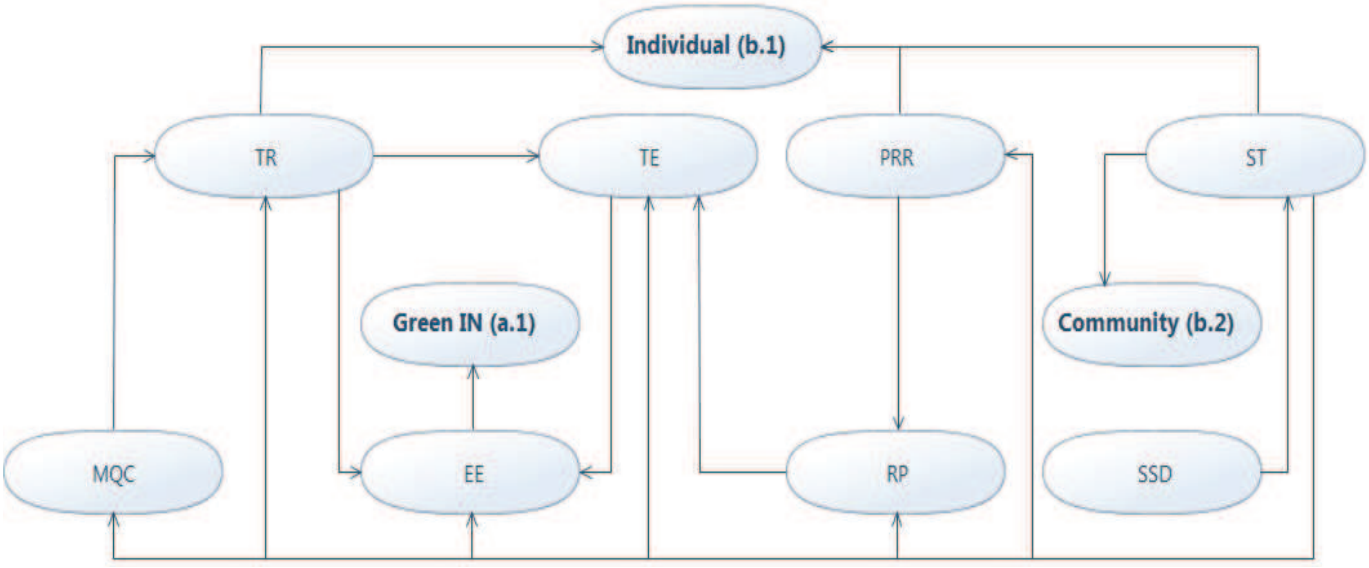

Fig. 4. Affect relationships of the q*uality aspects in MDSE Sustainability*

- (b.1) because it could help to reduce working time spent in the coding phase;
- (EE) because by spending less/more time coding, we would require less/more energy;
- (TE) because it could increase/reduce the time spent to code;

- Software System Dimension (SSD): refers to the magnitude of the model software artifact that we want to develop by using MDSE. SSD represent the number of views (e.g. diagrams such as classes, use case, associations and so on) that we want to develop. The SSD may affect:
  - (ST) in the decision regarding the use of a standard or not.

  SSD definitely affects the rest of the quality aspects, but we decided not to describe them because, we think, that these "affect relationships" between SSD and the other quality aspects would be all static. In other words, the increase or decrease of the SSD would probably affect in the same way the rest of the other quality aspects.

The definitions of EE, RP, and TE (from Resource effectiveness, where "time" is the only resource considered in this work) have been adapted from the context of software sustainability [1] to the MDSE one. PRR, ST, MQC and TR are quality aspects used in MDSE, and here were adapted to the sustainability context.

In Fig. 4 we present the interactability and interconnectivity of the quality aspects (presented above) with each other and with the dimensions (described in Section 3). The 'affect relationships' (a quality aspect that may affect another one) described between quality aspects and dimensions is expressed in Fig. 4 by a directional arrow from a quality aspect to another affected quality aspect or dimension. As already explained in the previous section, in this paper we chose to focus only on the environmental (in our context Green "IN" models) and human (divided in individual and community sustainability) dimensions.

V. Guidelines and Paths for MDSE Sustainability

In this section we present an initial set of guidelines obtained from the affect relationships between the quality aspects presented in Fig. 4.

The guidelines presented in TABLE II. could be helpful to identify quality aspects (qa) in the context of MDSE sustainability.

TABLE III. , 0, and TABLE V. show how, according to the affect relationships of the quality aspects presented in Fig. 4, we created 13 paths that can help to measure the degree of sustainability of this example.

TABLE II. MDSE Sustainability Guidelines

(qa)	Relationships between quality aspects
EE	EE can be identified by considering (not only) the degree of the TE, from how easy is to TR a model software artifact, and by the fact of using or not a model standard (ST).
TE	TE can be identified by considering (not only) the degree of the RP, from how easy is to TR a model software artifact, a and by the fact of using or not a model standard (ST).
TR	The degree of TR can be identified by considering (not only) how easy it is to check the MQC of a model software artifact, and by the fact of using or not a model standard (ST).
PRR	The degree of PRR can be identified by considering (not only) the fact of using or not a model standard (ST).
MQC	The degree of how easy it is to check the MQC can be identified by considering (not only) the fact of using or not a model standard (ST).
RP	The degree of RP can be identified by considering (not only) the fact of using or not a model standard (ST)., and the degree of the PRR.
ST	The fact of using or not a model standard can be decided by considering (not only) the SDD.
SSD	We did not find quality aspects that could affect the SSD.

The three tables below describe respectively the navigable paths to measure Green IN sustainability (a.1) TABLE III. , Individual sustainability (b.1) 0, and Community sustainability (b2) TABLE V.

With the character "#" we create a reference number that will be used later in later in TABLE VI. Each of the 13 paths presented in these three tables begin with "SSD→ST→" which represent 1) the dimensions of the software system that we want to design with models (SSD); 2) the choice of using or not a standard (ST). SSD and ST are the only two quality aspects assumed to be static in our example.

TABLE III. Relationships of the quality aspects for Green IN Sustainability

#	GREEN IN (a.1)
1	SSD→ST→EE→(a.1)
2	SSD→ST→MQC→TR→EE→(a.1)
3	SSD→ST→MQC→TR→TE→EE→(a.1)
4	SSD→ST→TR→EE→(a.1)
5	SSD→ST→TR→TE→EE→(a.1)
6	SSD→ST→TE→EE→(a.1)
7	SSD→ST→RP→TE→EE→(a.1)
8	SSD→ST→PRR→RP→TE→EE→(a.1)

TABLE IV. RELATIONSHIPS OF THE QUALITY ASPECTS FOR INDIVIDUAL SUSTAINABILITY

#	INDIVIDUAL (b.1)
9	SSD→ST→MQC→TR→(b.1)
10	SSD→ST→TR→(b.1)
11	SSD→ST→PRR →(b.1)
12	SSD→ST→ (b.1)

TABLE V. RELATIONSHIPS OF THE QUALITY ASPECTS FOR COMMUNITY SUSTAINABILITY

#	COMMUNITY (b.2)
13	SSD→ST→ (b.2)

VI. EXAMPLE OF HOW THE QUALITY ASPECTS COULD BE USED IN THE CONTEXT OF MDSE SUSTAINABILITY

In this sub-section we present a small example of how the sustainability quality aspects and the paths presented above could be used in the context of MDSE.

In the example, we intend to develop a software system using a MDSE methodology where we want to design a view of the software describing 100 classes and their relationships (this would be our SSD). We consider two hypothetical case solutions to design this model software artifact:

Case 1) It is decided to use Model Driven Architecture (MDA) [25] as standard of MDSE methodology. UML will be used as modeling language, and the 100 class and their relationships will be designed with a UML class diagram.

Case 2) It is decided to not use any standard, the 100 class and their relationships will be designed manually (sketches by hand).

In TABLE VI. we present a set of hypothetical scenarios of how the decision of using MDA in Case 1, and not using any standard in Case 2, could affect the degree of sustainability in this example.

TABLE VI. HYPOTETICAL SCENARIOS FOR MDSE SUSTAINABILITY

#	Case 1	Case 2
1	The fact that we use UML (ST), will require more energy for the process of model software artifacts development (EE) (than in the Case 2).	The fact that we represent manually (not using any specific ST) the 100 classes (SSD), will require no electrical energy for the design of the models (made by hand) (EE).
2	The fact that we use UML (ST), it could help the process of checking (e.g. by Object Constraint Language (OCL) in UML [26]) the quality (MQC) of the 100 classes (SSD). The improvement of the quality of models could also help the process of the transformation (TR) of the models to the code (that could be done automatically or semi-automatically). All these factors could result in reducing the time spent modeling and coding (TE) which could require less energy for the process of model software artifacts development (EE). This could finally lead to reduce the depletion of electrical energy (a.1).	The fact that we represent manually (not using any specific ST) the 100 classes (SSD), it could affect negatively the process of checking the quality (MQC) of the software model artifact under development. The possible decrease of the quality of models could also negatively affect the process of the transformation (TR) of the models to the code (that has to be done manually). All these factors could result in increasing the time spent modeling and coding (TE), which, on one hand, will require no electrical energy for the design of the models (made by hand); on the other hand, due to the missing of the benefits related to using a ST, the MQC, and the TR, will probably require more electrical energy (than in the Case 1) for the coding phase (EE and then a.1).
3		
4	The fact that we use UML (ST) to represent the 100 classes (SSD), it could help the process of the transformation (TR) of the models to the code (that could be done automatically or semi-automatically). These factors could result in reducing the time spent modeling and coding (TE) which could require less energy for the process of model software artifacts development (EE). This could finally lead to reduce the depletion of electrical energy (a.1).	The fact that we represent manually (not using any specific ST) the 100 classes (SSD), it could affect negatively the process of the transformation (TR) of the models to the code (that has to be done manually). These factors could result in increasing the time spent modeling and coding (TE), which, on one hand, will require no electrical energy for the design of the models (made by hand); on the other hand, due to the missing of the benefits related to to using a ST, and the TR, will probably require more electrical energy (than in the example 1) for the coding phase (EE and then a.1).
5		
6	The fact that we use UML (ST) to represent the 100 classes (SSD), it could reduce the time spent modeling and coding (TE) which could require less energy for the process of model software artifacts development (EE). This could finally lead to reduce the depletion of electrical energy (a.1).	The fact that we represent manually (not using any specific ST) the 100 classes (SSD), it could increase the time spent modeling and coding (TE), which, on one hand, will require no electrical energy for the design of the models (made by hand); on the other hand, due to the missing of the benefits related to using a ST, will probably require more electrical energy (than in the Case 1) for the coding phase (EE and then a.1).

#	Case 1	Case 2
7	The fact that we use UML (ST) to represent the 100 classes (SSD), it could increase the degree of how easy is to import/export other model software artifacts (PRR), and consequently the degree to which a model software artifact can be used over a long period (RP). All these factors could result in reducing the time spent modeling and coding (TE) which could require less energy for the process of model software artifacts development (EE). This could finally lead to reduce the depletion of electrical energy (a.1).	The fact that we represent manually (not using any specific ST) the 100 classes (SSD), it could decrease the degree of how easy is to import/export other model software artifacts (PRR), and then also the degree to which a model software artifact can be used over a long period (RP). All these factors could result in increasing the time spent modeling and coding (TE), which, on one hand, will require no electrical energy for the design of the models (made by hand); on the other hand, due to the missing of the benefits related to using a ST, the PRR, and the RP, will probably require more electrical energy (than in the Case 1) for the coding phase (EE and then a.1).
8		
9	The fact that we use UML (ST), it could help the process of checking (e.g. by Object Constraint Language (OCL) in UML) the quality (MQC) of the 100 classes (SSD). The improvement of the quality of models could also help the process of the transformation (TR) of the models to the code (that could be done automatically or semi-automatically). All these factors could help to reduce the working time spent modelling and coding (b.1).	The fact that we represent manually (not using any specific ST) the 100 classes (SSD), it could affect negatively the process of checking the quality (MQC) of the software model artifact under development. The possible decrease of the quality of models could also negatively affect the process of the transformation (TR) of the models to the code (that has to be done manually). All these factors could increase the working time spent modelling and coding
10		
11	The fact that we use UML (ST) to represent the 100 classes (SSD), it could increase the degree of how easy is to import/export other model software artifacts (PRR), which could help to reduce working time by reusing model software artifacts (b.1).	The fact that we represent manually (not using any specific ST) the 100 classes (SSD), it could decrease the degree of how easy is to import/export other model software artifacts (PRR) which could increase the working time spent modelling (b.1).
12	The fact that we use UML (ST) to represent the 100 classes (SSD), it could increase the the working time spent learning the UML standard, and then also the time spent modelling (b.1).	The fact that we represent manually (not using any specific ST) the 100 classes (SSD), it could decrease the the working time spent learning the model standard (in this case UML), and then also the time spent modelling (b.1).
13	The fact that we use UML (ST) to represent the 100 classes (SSD), it could contribute to improve the inclusiveness and connectivity in a community (b.2) that "speak" the same (model) language.	The fact that we represent manually (not using any specific ST) the 100 classes (SSD), it could affect negatively the inclusiveness and connectivity (b.2) in a community that do not "speak" the same (model) language.

Some of the paths presented in TABLE VI. were represented within only one scenario. We decided to merge 2 different paths because they present similar situations. For example, between the following two paths:

#2 (SSD→ST→MQC→TR→EE→(a.1));

#3 (SSD→ST→MQC→TR→TE→EE→(a.1));

They only difference is the presence of the TE in path #3. These two paths were merged in TABLE VI. (see scenario #2-3) where we described a scenario that covers both of the two paths. The same procedure was used to merge the paths #4 with #5, #7 with #8, and #9 with #10.

The example presented in TABLE VI. represents a first attempt of describing possible scenarios where, the quality aspects and the paths presented in section 4, could be helpful to measure and analyze the degree of sustainability in the context of MDSE.

In the three tables below (TABLE VII. , 0, and TABLE IX.), we show the summary of the scenarios presented in TABLE VI. where for each path(s) and Case solution adopted (1 or 2), we assign the symbol "+" or "-". The symbol "+" represent the hypothetical "positive affect relationship" of the quality aspects described in a given scenario in regards to a specific MDSE sustainability dimension; on the other hand, the symbol "-" mean the hypothetical "negative affect relationship" of the quality aspects described in a given scenario in regards to a specific MDSE sustainability dimension.

TABLE VII. SUMMARY OF GREEN IN SUSTAINABILITY

Scenarios #	Case 1	Case 2
	Green IN (a.1)	
1	-	+
2-3	+	-
4-5	+	-
6	+	-
7-8	+	-

TABLE VIII. SUMMARY OF INDIVIDUAL SUSTAINABILITY

Scenarios #	Case 1	Case 2
	Individual (b.1)	
9-10	+	-
11	+	-
12	-	+

TABLE IX. SUMMARY OF COMMUNITY SUSTAINABILITY

Scenarios #	Case 1	Case 2
	Community (b.2)	
13	+	-

For example, according to the scenarios described in TABLE VI. for the dimension "Green IN (a.1)" sustainability (TABLE VII.), for the scenario 1, the Case 2 (+) may be considered more sustainable than Case 1 (-); on the contrary, for the scenarios 2-3, 4-5, 6, and 7-8, the Case 1 (+) may be considered more sustainable than Case 2 (-).

The three tables TABLE VII. , 0, and TABLE IX. may be useful to measure how sustainable are model software artifacts in the context of MDSE because, by depending to the quality aspects (Section 4) on which we want to focus on, we could see how these quality aspects may affect the MDSE sustainability dimensions.

Regarding our example, if we know that the system under development will not need to be implemented in an actual software, we would not be interested in considering quality aspects such as PRR, MQC, RP and TR; in this context, the Case 2 solution may be considered more sustainable than Case 1 for the reason explained in scenario #1 and #12 of the Case 2 in the TABLE VI. On the other hand, if we focus on all the quality aspects presented in Section 4, the Case 1 solution may be considered (for the majority of the quality aspects) more sustainable than the Case 2 solution.

VII. CONCLUSION

In recent years, great attention has been paid to the topic of Software Sustainability. However, no previous study has, to the best of our knowledge, described sustainable dimensions and quality aspects in the context of Model Driven Software Engineering (MDSE) (Green "IN" models).

In this work we presented a first attempt of reshaping existing dimensions for software sustainability in terms of MDSE along with 8 quality aspects and their relationships with each other. Later, we present a set of guidelines to obtain the 8 quality aspects, and the paths to measure sustainability in terms of MDSE. Finally, we described an example with a set of hypothetical scenarios where we showed how the 8 quality aspects (Section 4) and the paths (Section 5) are used to measure and analyze the degree of sustainability in the context of MDSE. According to [27], the software development life cycle and related development tools and methodologies rarely, if ever, consider energy efficiency as an objective. In this paper, we tried to put particular attention on the Energy Efficiency (EE) by describing quality aspects that may affect it.

We believe that this first attempt of work in the context of MDSE sustainability could be used as a starting point in the future by researchers in this field.

As future work we are planning to develop a survey in the MDSE community about this topic to see how models are seen in terms of the sustainability dimensions and quality aspects identified in this paper. The final and more future goal will be to develop a consolidated and as much as possible valid set of guidelines in the context of MDSE sustainability.

ACKNOWLEDGMENT

Yvan Labiche (Carleton University, Ottawa, Canada), Marcela Genero, and Mario Piattini (University of Castilla-La Mancha, Ciudad Real, Spain). This work has been funded by the GINSENG project (TIN2015-70259-C2-1-R) (Ministerio de Economía y Competitividad and Fondo Europeo de Desarrollo Regional FEDER).

REFERENCES

[1] C. Calero and M. Piattini, *Green in Software Engineering*: Springer, 2015

[2] NOAA. (2016). *National Oceanic and Atmospheric Administration. June marks 14 consecutive months of record heat for the globe.* Available: http://www.noaa.gov/

[3] NASA. (2016, last access on June 2016). *Carbon Dioxide, Direct mesaurament 2005-Present*. Available: http://climate.nasa.gov/vital-signs/carbon-dioxide/

[4] CARVE. (2012, last access on June 2016). *NASA - Carbon in Arctic Reservoirs Vulnerability Experiment*. Available: http://science.nasa.gov/missions/carve/

[5] C. Miller. (2016, last access on June 2016). *NASA scientists react to 400 ppm carbon milestone*. Available: http://climate.nasa.gov/400ppmquotes/

[6] B. Bruegge and A. Dutoit., "Object-Oriented Software Engineering Using Uml, Patterns, and Java (3rd ed.)," Upper Saddle River, NJ, USA2009

[7] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice*, 1st ed., 2012.

[8] R. Gutbrod and C. Wiele, *The Software Dilemma Balancing Creativity and Control on the Path to Sustainable Software* Springer, 2012.

[9] F. Hermans, "Leaders of Tomorrow on the Future of Software Engineering: A Roundtable," *IEEE Software,* vol. 33, pp. 99-104, March 2016 2016.

[10] B. Penzenstadler, A. Raturi, D. Richardson, and T. B, "Safety, Security, Now Sustainability: The Nonfunctional Requirement for the 21st Century,," *IEEE Software,* vol. 31 pp. 40-47, 2014.

[11] W. Y. Du, S. L. Pan, and M. Y. Zuo, "How to balance sustainability and profitability in technology organizations: an ambidextrous perspective " *IEEE Transactions on Engineering Management* vol. 60, pp. 366–385, 2013.

[12] M. Genero, A. M. Fernández-Saez, H. J. Nelson, G. Poels, and M. Piattini, "A Systematic Literature Review on the Quality of UML Models," *Journal of Database Management,* vol. 22, pp. 46-70, 2011.

[13] D. Thomas, "MDA: Revenge of the modelers or UML utopia?," *IEEE Software,* vol. 21, pp. 15–17, 2004.

[14] G. Mussbacher, D. Amyot, R. Breu, J.-M. Bruel, B. H. C. Cheng, P. Collet*, et al.*, "The relevance of model-driven engineering thirty years from now,"in Proceedings of the 17th International Conference Model-driven engineering languages and systems (MODELS'2014), 2014.

[15] OMG, "OMG Unified Modeling LanguageTM - Superstructure Version 2.5," 2015.

[16] (2016, last access on June 2016). *1st Modelling For Sustainability Workshop at Bellairs'16*. Available: http://cs.mcgill.ca/~joerg/SEL/Sustainability_Bellairs_2016.html/
[17] B. C. K. Choi and A. W. P. Pak, "Multidisciplinarity, interdisciplinarity, and transdisciplinarity in health research, services, education and policy: 1," *Definitions, objectives, and evidence of effectiveness, Clin. Invest. Med.,* vol. 29 pp. 351-364, 2006.
[18] UN, "United Nations World Commission on Environment and Development. Report of the World Commission on Environment and Development: Our Common Future," 1987.
[19] S. B. Banerjee, "Who Sustains Whose Development? Sustainable Development and the Reinvention of Nature," *Organization Studies* vol. 24, pp. 143-180, 2003.
[20] B. Penzenstadler and H. Femmer, "A generic model for sustainability with process- and product-specific instances, ,"in Proceedings of the 2013 workshop on Green in/by software engineering Fukuoka, Japan, 2013
[21] P. Lago, N. Meyer, M. Morisio, H. A. Müller, and G. Scanniello, "Leveraging "energy efficiency to software users": summary of the second GREENS workshop, at ICSE 2013," *ACM SIGSOFT Software Engineering Notes,* vol. 39, pp. 36-38, 2014.
[22] P. Lago, S. Akinli, I. Crnkovic, and B. Penzenstadler, "Framing sustainability as a property of software quality," *Commun. ACM,* vol. 58, pp. 70-78, 2015.
[23] B. Penzesdtadler, "Towards a definition of sustainability in and for software engineering,"in Proceedings of the 28th Annual ACM Symposium on Applied Computing (SAC '13), 2013.
[24] D. Torre, Y. Labiche, and M. Genero, "UML consistency rules: a systematic mapping study,"in Proceedings of 18th International Conference on Evaluation and Assessment in Software Engineering (EASE 2014), London, UK, 2014.
[25] J. Mukerji and J. Miller, "Overview and guide to OMG's architecture," Object Management Group. 2003. Available: http://www.omg.org/mda/
[26] OMG. (2016, last access on June 2016). *Object Management Group - Object Constraint Language (OCL)*. Available: http://www.omg.org/spec/OCL/
[27] C. Calero, M. F. Bertoa, and M. A. Moraga, "A systematic literature review for software sustainability measures "in Proceedings of the 2nd International workshop on green and sustainable software (GREENS 2013), 2013